Guidelines for DDR configuration on STM32MP2 MPUs

Introduction

This application note describes how to configure the DDR subsystem (DDRSS) of the STM32MP25x MPU lines.

Configuring the DDR subsystem is done by programming multiple settings to the DDR controller (DDRCTRL), the PHY (DDRPHYC), and SDRAM mode registers.

These settings are determined according to system parameters such as DDR type, density, topology, frequency, and SDRAM JEDEC parameters.

The parameters are programmed during the initialization sequence. The PHY is trained by an embedded processor in the DDRPHYC PUB loaded with a protocol-specific firmware.

The STM32MP25xx STM32CubeMX DDR tool [7.] enables an easy initialization process thanks to intuitive panels and menus. Therefore, it requires only few inputs from the user to set up the DDR subsystem.

Initialization and training of the DDR subsystem are launched on every cold reset during the boot sequence. Training settings can be saved and restored for a fast wake-up.

During the system bring-up phase, the user should run extensive tests provided by DDRFW-UTIL. The tests can be launched from STM32CubeMX to verify the robustness of the DDR configuration.

Related documents and references

1. STM32MP25xx advanced Arm®-based 32/64-bit MPUs reference manual (RM0457)
2. Guidelines for DDR memory routing on STM32MP2 MPUs application note (AN5724)
3. Getting started with STM32MP25xx lines hardware development application note (AN5489)
4. JEDEC JESD79-3F DDR3 SDRAM standard
5. JEDEC JESD79-4B DDR4 SDRAM standard
6. JEDEC JESD209-4D LPDDR4 SDRAM standard
7. STM32CubeMX DDR tool for STM32MP25xx
8. DDR-PHY interface (DFI 4.0): see http://www.ddr-phy.org

1 General information

This document applies to the STM32MP25x lines, Arm®-based MPUs.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

arm

Table 1. Acronyms

Acronym	Description
ACSM	Address-command state machine, drive address/command during training
ANIB	4 lanes address/command unit (aka ACX4X)
BDL	Bit delay line that is used to deskew DQ and DBI signals. Each DBYTE has nine BDL
CSR	PUB configuration registers
DDC	DRAM drift compensation that is applicable to LPDDR4 for DQ to DQS retraining according to write tDQS2DQ and read tDQSCK timings drifts
DBYTE	Byte lane unit that is composed of 8 DQ lanes (2 nibbles), DQS/DQS, DMI (data mask and inversion control)
DDRCTRL	DDR controller
DDRPHYC	DDR PHY top level that is composed of a structured PHY (MASTER, DBYTES, ACX4) and PUB
DDRSS	DDR subsystem
DFI	DFI interface between controller (DDRCTRL) and PHY (DDRPHYC), it is DFI 4.0 version
DFICLK	The input clock to the DDRSS, it is the reference clock for the DDRCTRL and the DFI interface
DRTUB	PUB subblock and register regions that are used by a PHY firewall
DTSM	Data training state machine that is used by PHY firewall
INITENG	PUB subblock and register regions that are used by a PHY firewall
LCDL	Local calibrated delay line supporting fractional UI in 1/32 steps that are used at ANIB and DBYTE for compensated delays
MASTER	PUB subblock and register regions: MASTER includes PLL, impedance calibration engine, V_{REF} generator, reset controller
MEMCLK	The actual memory clock, it is twice DFICLK
MPC	Multipurpose command, LPDDR4 only
PIE	PHY initialization engine is a programmable micro-sequencer triggered on active edges of dfi_init_start to support state transitions
PMU	Programmable microcontroller unit used by a training firmware
PPGC	PRBS pattern generator checker that is used by a PHY firewall
PUB	Physical utility block that is composed of CSR and various subblocks and state machines to control PHY: ACSM, PPCG, DTSM
XPI	AXI interface block in DDRCTRL

2 Overview

General information about initialization and configuration of the DDR subsystem are detailed in this section.

STM32MP25x DDR subsystem is shown in Figure 1.

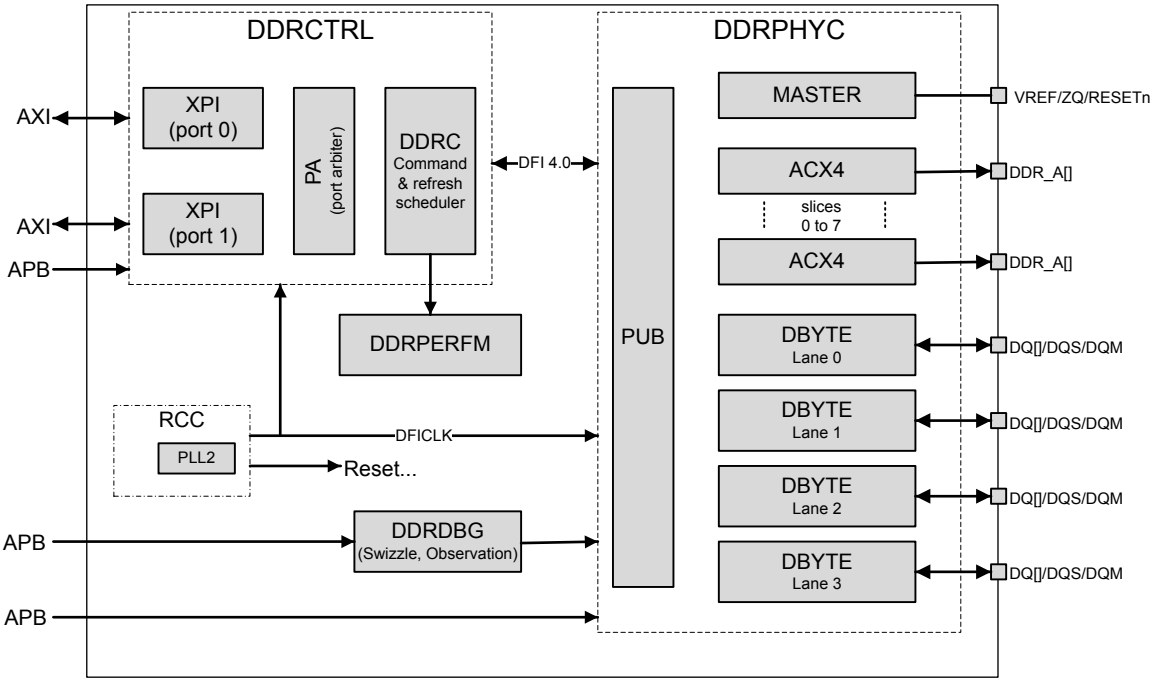
It includes:

- DDRCTRL (also named DDR controller)
- DDRPHYC (also named DDR PHY)
- Related components (DDRPERFM, DDRDBG)

These components are described with more details in [1.].

DDRCTRL supports the DDR command and refresh scheduled during normal operations. A normal operation is also called mission mode.

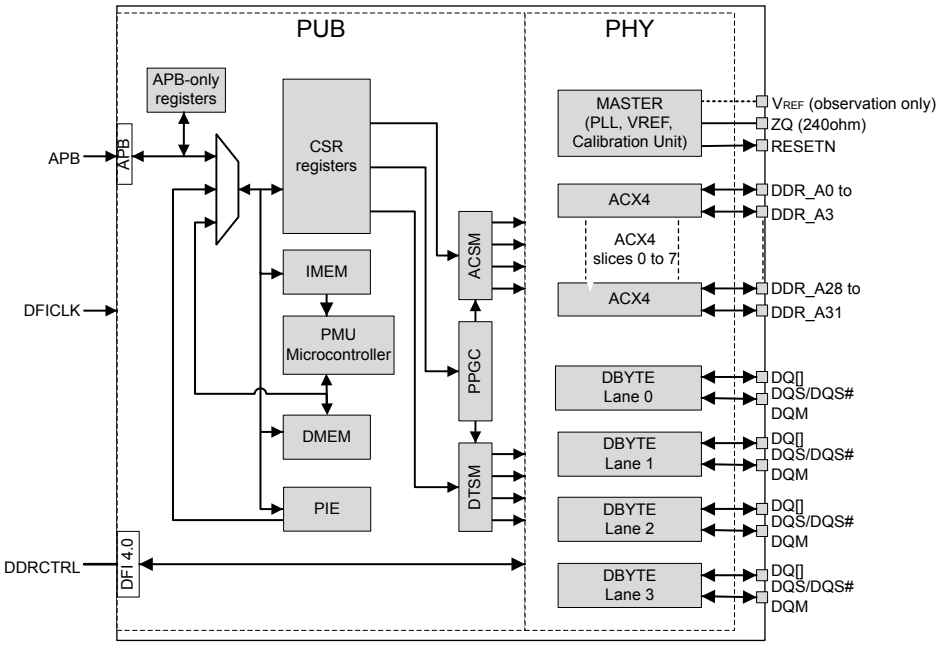
Figure 1. DDR subsystem



DT71618V1

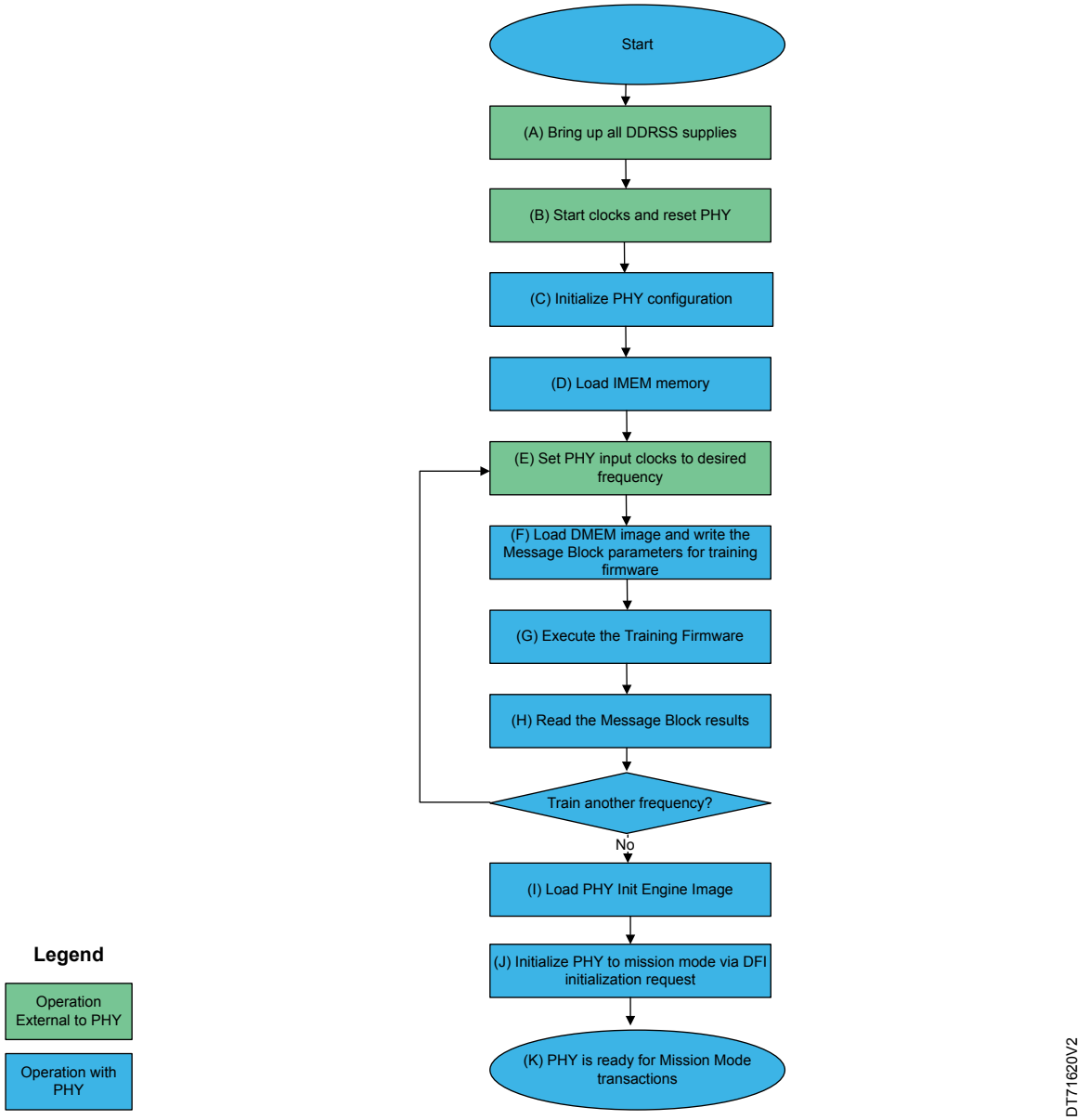
DDRPHYC is a multiprotocol DRAM PHY depicted in Figure 2. It features a lane-based architecture and a PHY utility block (PUB) to support autonomously DRAM initialization and interface training using an embedded processor with firmware.

Figure 2. DDRPHYC overview



The PHY initialization and training sequence are shown in Figure 3. The sequence is controlled by the DDR driver.

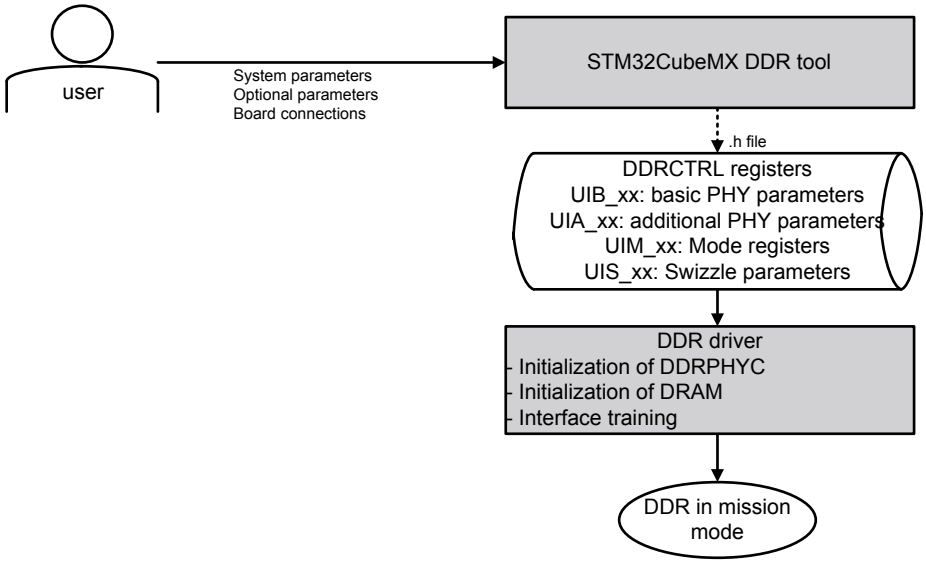
Figure 3. PHY initialization and training sequence



The configuration flow is depicted in Figure 4. The user has only to provide the system parameters in a STM32CubeMX DDR tool to generate a .h file for a generic DDR driver. The DDR driver is then used either during boot sequence (TF-A), or by HAL to support PHY and DRAM initialization and trainings.

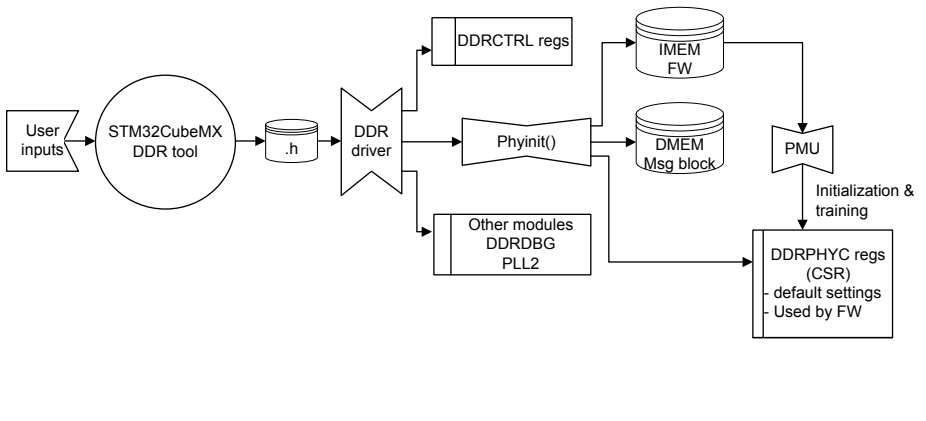
DDRSS configuration can be saved and restored to support the LP3 low-power mode with I/O retention (system Standby with DDR in self-refresh). In this case, the initialization with skip training is used.

Figure 4. **Configuration and initialization flow diagram**



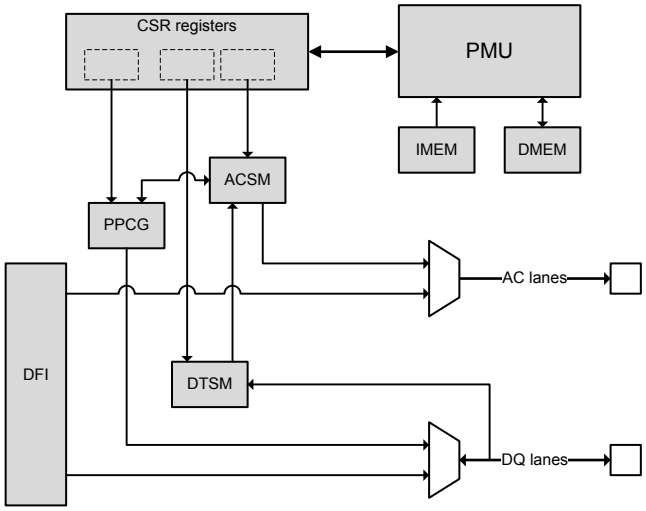
The software architecture that enables to support DDRSS initialization and training is depicted in Figure 5.

Figure 5. **DDR configuration software architecture**



A simplified overview of the training hardware in the PUB is shown in Figure 6.

Figure 6. **Simplified diagram of the training hardware**



DDRCTRL registers

Most DDRCTRL registers are static and loaded at reset by the DDR driver. Quasi-dynamic registers can be reprogrammed using specific sequence. For more information, refer to the reference manual dealing with PHY initialization and training completion. DDRCTRL is enabled into mission mode with SDRAM in self-refresh.

DDR configuration

Configuration parameters are described according to their different types in Section 3.

DDR initialization and training

For detailed information about DDR initialization and training, refer to Section 3.2.2.

DDR testing

DDRSS configuration robustness can be tested by running intensive sequences of tests that are launched from the STM32CubeMX tool. It is important to run all tests proposed by the STM32CubeMX tool during the system bring-up phase, and this, before using the DDR. DDR testing and test flow with failure diagnostic and corrections are described in Section 9. DDR subsystem bring-up is complete after DDR configuration, tuning and successful stress testing. The configuration parameters are saved for normal run mode initialization and DDR mission mode.

3 Description of configuration parameters

3.1 System parameters

These parameters show the settings needed for the DDR.

- The DDR protocol includes DDR3L, DDR4, LPDDR4.
- Interface width is 32-bit when it is full-populated, or 16-bit when it is half-populated.
- The DDR density is, according to the JEDEC protocol, menu selectable in Gbit per device (DDR3/4), or Gbit per channel (LPDDR4).
- The DDR frequency is the actual memory clock frequency. It is twice the DFICLK generated by RCC PLL2. The frequency range is limited according to protocol. The input frequency is quantified in MHz and used to determine timing parameters.
- DDR topology options are compliant with the JEDEC protocol and they are described in the JEDEC protocol specific sections.
- Other parameters include several DRAM secondary parameters (for example DQ impedances and terminations), voltage reference, preamble, and postamble timings. They are configurable with predefined values. The predefined values should not be modified in most cases. The total density in Gbit is determined according to system parameters defined by the user and reported by the DDR tool.

Note: Some parameters depend on the DDR protocol. They are described in the following sections: DDR3L configuration, DDR4 configuration and LPDDR4 configuration.

3.2 Mission mode parameters

These parameters are used by DDRCTRL for performance optimization. Some predefined values are proposed for:

- Low-power mode settings
- Flexible address mapping
- Quality of service (QoS) and scheduling

Low-power mode settings

They are automatic self-refresh entries. For more information, refer to [1.].

Flexible address mapping

DDRCTRL features a flexible address mapper to convert AXI bus address to DRAM row, banks, and columns.

STM32CubeMX DDR tool offers two predefined configurations:

- Banks/rows/columns (B/R/C) with lowest power
- Rows/banks/columns (R/B/C) which is the most conventional configuration. It should be used by default.

Note: DDRCTRL flexible address mapper enables the mapping of individual R/B/C bits into almost any sequence. The determination of corresponding ADDRMAP is nontrivial. For more information, refer to the DDRCTRL section in [1.].

Quality of Service (QoS) and scheduling

DDRCTRL schedules DDR commands to maximize their use.

It is done by:

- grouping read/write commands
- scheduling bank activations from incoming commands posted in a 32-entry read command buffer CAM and a 32-entry write command buffer.
  These buffers are made of content addressable memory (CAM).

DDRCTRL uses an advanced multitier arbitration scheme for optimal DDR use and low latency according to programmed settings.

DDRCTRL features a rich set or arbitration policies and optimization techniques:

- Page open/close policy
- Traffic mapping into classes and timeouts
- Scheduler and port arbitration control

Command arbitration is determined according to traffic classes and timeout settings for some traffic classes.

DDRCTRL maps incoming AXI port transactions into traffic classes with priorities and/or timeout, programmable for each AXI port 0 and port 1.

For **Read**, three classes are possible:

- HPR that corresponds to the allocated store with high priority. It is suitable for latency sensitive traffic.
- VPR that corresponds to the highest priority on timeout expiration. It is suitable for latency critical traffic.
- LPR that corresponds to the lower priority. It is suitable for best effort traffic.

Note: The current DDRCTRL configuration uses a single read address queue:

- *AXI port 0 supports only two traffic classes (HPR and VPR).*
- *AXI port 1 supports only two traffic classes (VPR and LPR).*

For **Write**, two classes are possible:

- VPW that corresponds to the highest priority on timeout expiration. It is suitable for latency critical traffic.
- NPW that corresponds to normal priority. It is suitable for best effort traffic.

Note: With the current DDRCTRL configuration, both AXI ports support the two traffic classes: VPW and NPW.

STM32CubeMX DDR tool offers predefined configurations aligned with the port allocation and the QoS settings of the AXI interconnect.

For more information, refer to [1.].

Fixed QoS values per master are used.

By default, the STM32MP25xx AXI interconnect matrix (AXIM) maps all masters to port 0, port 1 and port QoS according to Table 2. DDRCTRL QoS registers are listed in Table 3.

Table 2. Bus master port and QoS settings table

Master	DDRCTRL (port 0)	DDRCTRL (port 1)	Default QoS	Class (read)	Class (write)
CM33	S/NS	-	15	HPR	VPW
CA35	S/NS	-	14	HPR	VPW
SDMMC1	S/NS	-	12	VPR	VPW
SDMMC2	S/NS	-	12	VPR	VPW
SDMMC3	S/NS	-	12	VPR	VPW
HPDMA1	S	NS	11	VPR	VPW
HPDMA2	S	NS	11	VPR	VPW
HPDMA3	S	NS	11	VPR	VPW
AXI-AP [1]	S	NS	10	VPR	VPW
USBH_OHCI	S/NS	-	10	VPR	VPW
USBH_EHCI	S/NS	-	10	VPR	VPW
LTDC	S	NS	9	VPR	-
DCMIPP	-	S/NS	8	-	VPW
ETH1	-	S/NS	7	LPR	NPW
ETH2	-	S/NS	7	LPR	NPW
PCIE	-	S/NS	6	LPR	NPW
USB3DR	-	S/NS	5	LPR	NPW
ETR [2]	S/NS	-	3	VPR	NPW
VENC	-	S/NS	4	LPR	NPW

Master	DDRCTRL (port 0)	DDRCTRL (port 1)	Default QoS	Class (read)	Class (write)
VDEC	-	S/NS	4	LPR	NPW
GPU	-	S/NS	4	LPR	NPW

The QoS model depends on the specified classes (read/write):

- HPR (high-priority read) is allocated to the high-priority read queue and is suitable for latency sensitive traffic with bounded bandwidth.
- VPR/VPW (variable-priority read/variable-priority write) correspond to variable priorities and are suitable for latency critical traffic. VPR or VPW timeout expiration is preemptive.
- LPR/NPW (low-priority read/normal-priority write) are best effort traffic.
- Read is preferred to write.
- Read CAM has an allocation for HPR (three entries) and VPR/LPR (13 entries).
- DDRCTRL features some antistarvation controlled by max_starve / run_length for each queue. HPR and LPR are for read and WR is for write.
- VPR/VPW timeout can be specified per port and per QoS region according to r/wqos_map_timeout parameters with value in the clock cycle coded in hexadecimal. The lower the value, the faster VPR/VPW goes into ‘expired-VPR/VPW’ state.
- Port aging may be used to prevent port starvation in case head of line is blocking at AXI level. As, this is not expected, port aging is not used with the proposed configuration.
- Intelligent precharge policy may be used instead of open page policy for power saving.

Note: Most QoS settings and scheduling control parameters are static and set during configuration.

Caution: This description of QoS settings is generic. Current settings may be modified according to product release or tool updates.

3.2.1 DDR refresh controller

DDRCTRL refresh controller is programmed according to clock frequency and DRAM refresh requirement (JEDEC). Refresh parameters are determined by STM32CubeMX DDR tool according to the JEDEC timings (t_{REFI}, t_{RFC}) and they are programmed through DDRCTRL.RFSHTMG.

Periodic auto-refresh is used by default. However, DDRCTRL supports additional features such as:

- Burst refresh with speculative burst refresh timeout (optional)
- Per-bank refresh for LPDDR4 (optional)
- Refresh command enabled by the software (optional)

For more information, refer to [1.].

Note:

- *LPDDR4 temperature derating with mode register 4 (MR4) periodic polling for auto-refresh rate adjustment is enabled by default.*
- *ASR and SRT modes can be set during DDR3L configuration according to T range options.*

3.2.2 PHY training

The PHY optimized delays parameters are determined by the PHY firewall. The objective is to position accurately the interface and control signals for optimal timing margins. The PHY training consists in:

- Command bus training (CBT) only for LPDDR4
  - The SDRAM is put into a mode where the command/address lines are sampled and fed back on DQ lines during the CA training. Series of bit patterns are sent with varying delay to best align CA to CLK falling edges for each ANIB.
  - The CBT uses a slower frequency clock with unterminated CA. The DDRPHYC has dedicated circuits to generate a divided down clock according to memory clock.
- Read gate training
  - The purpose of read gate training (RxEn training) is to determine the round-trip latency, including CAS latency, between the DDRPHYC and the DRAM. The round-trip latency must be known so that the DDRPHYC receivers are activated by the read gate at the right time in order to capture the burst of read data.
  - All DBYTEs in a rank are trained in parallel, but ranks are trained independently.
  - This step compensates the delay for the read command to reach each DRAM, and to have the DQS pulse for the read return to the DDRPHYC. This delay includes all DRAM DQS related parameters such as t_{DQSCK}, but does not include the read CAS latency, which is known ahead of time. A delay is calculated for each nibble of each DBYTE to each timing group by finding the time of the first rising edge of DQS for each timing group, and changing the delay of the RxEnDly to place it as close to the middle of the DQS preamble as possible.
- Write leveling fine phase training
  - The write leveling fine grain phase is performed first. The goal of write leveling fine phase training is to align DQS rising edge with the MEMCLK rising edge at each DRAM device.
  - The write leveling fine phase training is required to compensate for any analog phase differences between MEMCLK and DQS distribution. This fine training ensures that the edges are aligned correctly, but not necessarily at the correct WCAS latency. The coarse part of the write leveling training calculates the delay to ensure the edges aligned with the correct WCAS latency. This step compensates for the phase difference between the CK signal and the DQS at each SDRAM. A delay is calculated for each DBYTE to each timing group by using the write leveling mode of SDRAMs to place the DQS rising edge as close as possible to the CK rising edge.
- Read deskew training
  - The read DQ deskew training compensates for the delay differences, primarily caused by board routing and SDRAM DQ output skew, among the DQ lanes during reads. The read deskew must not be skipped in silicon, even on systems without much skew between lanes. A read deskew failure can cause part of the read-eye to be missed by later stages. As a result, it can lead to nonoptimal results.
  - The read deskew training is implemented by using per lane delays to align the start of the passing regions for each nibble. Because the read enable and write leveling have been completed, the read deskew can send simple writes and reads to the devices to collect data on the skews, such as using MPR pattern as 0xB2B2.
  - The firmware sets up the chosen pattern, forces the receive clock delay to 0, and then runs an in-phase left search in order to find the largest passing per-lane delays for every lane.
  - Any nibble, which lanes all succeed in finding a left edge, has all per-lane delays recorded while the failing lanes continue to try incrementally a larger receive clock delay, extending the search range, until every nibble finds a working set of per-lane delays. This step compensates the delay difference between DQ signals from each SDRAM, and ensures the full receive-eye exists at positive rxclkdly. A delay is calculated for each signal in each DBYTE from each timing group to align all the DQ signals with each other as close as possible.

- Read 1D (SI friendly) training
  - The read 1D fine training is done so that reads can be trained to work and perform further training to properly train the write operations. The result of this training is not guaranteed to be optimal but is guaranteed to work well enough to perform the write training. Once the write training is complete, the read DQ/DQS timing can be optimized further with longer and more complex patterns to a final trained position.
  - The firmware implements this step of training by programming a pattern into an MPR, then running an automatic eye-search using the read clock delay. The results of the automatic eye-search are then used to write the read clock delay values to their eye midpoints.
  - This step compensates for the delay difference between the DQ and the DQS signals from each SDRAM device. It also moves the sampling edge of DQS to the center of the DQ eye. A delay is calculated for the upper and lower nibble of each DBYTE separately by calculating the compound eye for all signals in each nibble and centering the sampling edge of DQS in this eye.
- Write leveling coarse delay training
  - The Tx DQS is phase-aligned to MEMCLK during the write leveling fine phase training, but the Tx DQS latency may not match the CAS write latency. As the Tx DQS delay is already phase-aligned, it can be incorrect by some integer multiple of MEMCLK periods. During initialization, the coarse search calculates an educated guess for the delay between the transmitted DQS and DQ signals. This delay works for writes but is not delay margin optimized. The coarse search writes a relatively slow square wave to the rank being trained, and then reads back the values.
  - The DDRPHYC generates extra DQS pulses to ensure that DRAMs see enough DQS pulses during the time of the write even if the timing is off. Any error in the comparisons between what is written to the SDRAM and what is read out can be attributed to a coarse offset between DQS and MEMCLK signals.
  - The firmware adds to the DQS delay currently set, and reruns the square wave write/read sequence to check if the new delay has fixed the errors. The firmware continues to loop, adding a delay, until the data come back without issue. If all reasonable delay settings fail to return the data as it was written, the firmware terminates the training. The delays start to be unreasonable around the 20 UI mark. Such a large delay means that something has gone wrong. This step compensates for the delay difference between the CK signal and the DQS at each DRAM. A delay is calculated for each DBYTE to each timing group by using the write leveling mode of the SDRAMs to place the rising edge of DQS as close as possible to the rising edge of CK.
- Write 1D training
  - The user can safely read back arbitrary data from DRAM, even if the read training is not perfectly centered on the read eye. The objective of this training is to center the write DQ timing in the write DQS eye (write DQS timing is set in the write leveling steps).
  - The DDRPHYC has per-lane write DQ adjustment, allowing all the lanes to simultaneously train to their own value. The write DQ/DQS delay center training is implemented by an automatic eye-search that varies the delay on the write DQ signals. Once the automatic eye-search is complete, the resulting eye midpoints are written on the DQ delay DDRPHYC register.
  - This step compensates for the delay difference between DQ and DQS signals to each DRAM. This step also moves the center of each DQ eye to sampling edge of DQS.
- Read 1D training (optimized delay centering)
  - This training is similar to the read 1D (SI friendly). The delay-centering is using longer and more complex PRBS patterns to obtain a more accurate estimation of the mission mode eye and to optimize the DQS position for stressful patterns. Like the write DQ/DQS delay-centering, all lanes are trained in parallel, but each nibble has a common read delay setting. If read DBI is enabled, the DBI lanes are trained at the same time as the DQ lanes.
  - This step compensates for the delay difference between DQ and DQS signals from each SDRAM. This step also moves the sampling edge of DQS to the center of the DQ eye. A delay is calculated for the upper and lower nibble of each DBYTE separately by calculating the compound eye for all signals in each nibble and centering the sampling edge of DQS in this eye.
- Maximum read latency training
  - When reading data from SDRAMs, the received data are buffered into receive FIFO. After a short period, called the read latency, the data in the FIFO are guaranteed to be valid.
  - The read latency training finds the smallest read latency that works for all FIFOs in the DDRPHYC to optimize the system latency.

4 Low-power features

The DDR subsystem supports multiple low-power features. The power reduction features are applicable to:

- SDRAM with power down and self-refresh modes
- DDRPHYC with low-power modes (LP1, LP2/LP3, LP3 I/O retention)
- DDRCTRL with clock gating

The light low-power features can be programmed according to configuration. This is applicable to the automatic self-refresh (ASR) and automatic power down (APD). In these modes, DDRPHYC can be LP1, but must be clocked. A specific software sequence, supported by the driver, controls the lowest power modes.

They are mainly:

- LP2/LP3 that are with DDR in self-refresh and clock removal.
- LP3 I/O retention with DDR in self-refresh with controller power removal. This mode needs to save and restore the DDRSS configuration. SDRAM is preserved into self-refresh.

5 Configuration parameters

Starting from user inputs, the STM32CubeMX DDR tool calculates the configuration parameters. They are exported in a '.h' file used by the DDR driver at step C of the sequence in Figure 3. The parameters are grouped into categories.

- DDRCTRL static parameters
- PHY basic and advanced parameters
  They are used by the driver to set up DDRPHYC and to control DDR initialization and training. These parameters are prefixed with 'UIB_' and 'UAI_' in the .h file.
- SDRAM mode parameters
  They are passed to the initialization firewall to set up SDRAM mode registers. These parameters are prefixed with 'UIM_' in the .h file.
- Swizzle control parameters
  They are used to correct the DFI to SDRAM signal paths (functional and training) after the PCB and device signal swaps. These parameters are prefixed with 'UIS_' in the .h file.

The values and significance of parameters may depend on the DDR protocol and some typical configuration examples are provided in dedicated Section 6.2, Section 7.2 and Section 8.2.

5.1 DDRCTRL parameters

DDRCTRL features 97 read/write registers. The majority of registers are statically programmed during DDR initialization at step C (see Figure 3).

Table 3 lists DDRCTRL parameters according to:

- System configuration
- Timing
- Address mapping
- Quality of Service (QoS)

For more details, refer to the DDRCTRL section in [1.].

Table 3. DDRCTRL parameters

Register	Category	Register	Category	Register	Category	Register	Category
MSTR	system configuration	DIMMCTL	system configuration	DFITMG0	timing	SCHED	QoS
MRCTRL0	system configuration	RANKCTL	system configuration	DFITMG1	timing	SCHED1	QoS
MRCTRL1	system configuration	DRAMTMG0	timing	DFILPCFG0	timing	PERFHPR1	QoS
MRCTRL2	system configuration	DRAMTMG1	timing	DFILPCFG1	timing	PERFLPR1	QoS
DERATEEN	system configuration	DRAMTMG2	timing	DFIUPD0	timing	PERFWR1	QoS
DERATEINT	system configuration	DRAMTMG3	timing	DFIUPD1	timing	DBG0	system configuration
DERATECTL	system configuration	DRAMTMG4	timing	DFIUPD2	timing	DBG1	system configuration
PWRCTL	system configuration	DRAMTMG5	timing	DFIMISC	timing	DBGCMD	system configuration
PWRTMG	system configuration	DRAMTMG6	timing	DFITMG2	timing	SWCTL	system configuration
HWLPCTL	system configuration	DRAMTMG7	timing	DFITMG3	timing	POISONCFG	system configuration
RFSHCTL0	timing	DRAMTMG8	timing	DBICTL	system config	PCCFG	system configuration

Register	Category	Register	Category	Register	Category	Register	Category
RFSHCTL1	timing	DRAMTMG9	timing	DFIPHYMSTR	system config	PCFGR_0	QoS
RFSHCTL3	timing	DRAMTMG10	timing	ADDRMAP0	address mapping	PCFGW_0	QoS
RFSHTMG	timing	DRAMTMG11	timing	ADDRMAP1	address mapping	PCTRL_0	QoS
RFSHTMG1	timing	DRAMTMG12	timing	ADDRMAP2	address mapping	PCFGQOS0_0	QoS
CRCPARCTL0	system configuration	DRAMTMG13	timing	ADDRMAP3	address mapping	PCFGQOS1_0	QoS
CRCPARCTL1	system configuration	DRAMTMG14	timing	ADDRMAP4	address mapping	PCFGWQOS0_0	QoS
INIT0	timing	DRAMTMG15	timing	ADDRMAP5	address mapping	PCFGWQOS1_0	QoS
INIT1	timing	ZQCTL0	system configuration	ADDRMAP6	address mapping	PCFGR_1	QoS
INIT2	timing	ZQCTL1	system configuration	ADDRMAP7	address mapping	PCFGW_1	QoS
INIT3	timing	ZQCTL2	system configuration	ADDRMAP8	address mapping	PCTRL_1	QoS
INIT4	timing	ODTCFG	system configuration	ADDRMAP9	address mapping	PCFGQOS0_1	QoS
INIT5	timing	ODTMAP	system configuration	ADDRMAP10	address mapping	PCFGQOS1_1	QoS
INIT6	timing	-	-	ADDRMAP11	address mapping	PCFGWQOS0_1	QoS
INIT7	timing	-	-	-	-	PCFGWQOS1_1	QoS

5.2 PHY parameters

The parameters deal with UIB and UIA groups.

The basic and advanced parameters are listed in Table 4 and Table 5. These two structures are used by the PHY driver to initialize DDRPHYC CSR registers and to control the execution of the PHY firewall training at step G, preceded by the PHY firewall load into IMEM by the driver at step D; parameters are passed between the PHY driver and the PHY firewall using a mail box into DMEM at step F. More details about the sequence and the steps named here are shown in Figure 3.

Table 4. Basic PHY parameters

DDRPHYC basic	DDR3	DDR4	LPDDR4
UIB_DRAMTYPE	y	y	y
UIB_DIMMTYPE	y	y	y
UIB_LP4XMODE	-	-	-
UIB_NUMDBYTE	y	y	y
UIB_NUMACTIVEDBYTEDFI0	y	y	y
UIB_NUMACTIVEDBYTEDFI1	y	y	y
UIB_NUMANIB	-	-	-
UIB_NUMRANK_DFI0	y	y	y
UIB_NUMRANK_DFI1	y	y	y
UIB_DRAMDATAWIDTH	y	y	y

DDRPHYC basic	DDR3	DDR4	LPDDR4
UIB_NUMPSTATES	-	-	-
UIB_FREQUENCY_0	y	y	y
UIB_PLLBYPASS_0	y	y	y
UIB_DFIFREQRATIO_0	-	-	-
UIB_DFI1EXISTS	y	y	y
UIB8TRAIN2D	-	-	-
UIB_HARDMACROVER	-	-	-
UIB_READDBIENABLE_0	y	y	y
UIB_DFIMODE	-	-	-

Table 5. Advanced PHY parameters

DDRPHYC advanced	DDR3	DDR4	LPDDR4
UIA_LP4RXPREAMBLEMODE_0	-	-	y
UIA_LP4POSTAMBLEEXT_0	-	-	y
UIA_D4RXPREAMBLELENGHT_0	-	y	-
UIA_D4TXPREAMBLELENGTH_0	-	y	-
UIA_EXTCALRESVAL	-	-	-
UIA_IS2TTIMING_0	-	-	-
UIA_ODTIMPEDANCE_0	y	y	y
UIA_TXIMPEDANCE_0	y	y	y
UIA_ATXIMPEDANCE	y	y	y
UIA_MEMALERTEN	-	-	-
UIA_MEMALERTPUIMP	-	-	-
UIA_MEMALERTVREFLEVEL	-	-	-
UIA_MEMALERTSYNCBYPASS	-	-	-
UIA_DISDYNADRTRI_0	-	-	-
UIA_PHYMSTRTRAININTERVAL_0	-	-	-
UIA_PHYMSTRMAXREQTOACK_0	-	-	-
UIA_WDQSEXT	-	-	-
UIA_CALINTERVAL	-	-	-
UIA_CALONCE	-	-	-
UIA_LP4RL_0	-	-	y
UIA_LP4WL_0	-	-	y
UIA_LP4WLS_0	-	-	y
UIA_LP4DBIRD_0	-	-	y
UIA_LP4DBIWR_0	-	-	y
UIA_LP4NWR_0	-	-	y
UIA_LP4LOWPOWERDRV	-	-	-
UIA_DRAMBYTESWAP	-	-	-
UIA_RXENBACKOFF	y	y	y
UIA_TRAINSEQUENCECTRL	-	-	-

DDRPHYC advanced	DDR3	DDR4	LPDDR4
UIA_SNPSUMCTLOPT	-	-	-
UIA_SNPSUMCTLF0RC5X_0	-	-	-
UIA_TXSLEWRISEDQ_0	-	-	-
UIA_TXSLEWFALLDQ_0	-	-	-
UIA_TXSLEWRISEAC	-	-	-
UIA_TXSLEWFALLAC	-	-	-
UIA_DISABLERETRAINING	-	-	-
UIA_DISABLEPHYUPDATE	-	-	-
UIA_ENABLEHIGHCLKSKEWFIX	-	-	-
UIA_DISABLEUNUSEDADDRLNS	-	-	-
UIA_PHYINITSEQUENCENUM	-	-	-
UIA_ENABLEDFICSPOLARITYFIX	-	-	-

5.3 SDRAM mode parameters

The parameters deal with the UIM group.

The SDRAM mode registers are specific to each DDR type and passed to the driver. The PHY firewall may access mode registers during training (MPC or MPR). Several parameters are set during configuration: Impedance, ODT, V_{REF} from user input or calculated if frequency dependant, such as CL/CWL or RL/WL. Typical mode register settings are described in dedicated sections for DDR3L, DDR4 and LPDDR4.

5.4 Swizzle control parameters

The swizzle control parameters deal with the UIS group.

Signal swapping can be used to facilitate PCB routing. The swizzle control group is programmed to restore a one-to-one mapping from DFI to SDRAM during training and mission modes.

DDR3 and DDR4

The address and command signals can be swapped to ease PCB routing. The HwtSwizzle registers (for training signal path) and DDRDBG_DDR34_AC_SWIZZLE (for mission mode signal path) need to be programmed accordingly. The CLK, CKE, CSN, and ODT signals in the first two ACX4 in Table 7 cannot be swapped. DQ bits within a byte lane and DQ byte lanes are invariant in a device, therefore they can be swapped accordingly. However, the DqxLnSel registers must be kept to their default values.

LPDDR4

The CAA[] and CAB[] groups in Table 7 can be swapped within a channel to ease PCB routing. The MAPCAATODFI and MAPCABTODFI registers need to be programmed accordingly. Because of the narrow C/A per channel, the one-to-one mapping is usually implemented. DQ signals can be swapped within each byte lane to ease PCB routing. The DQXLNSEL registers need to be programmed adequately regards to the internal device swaps in Table 8. Byte swapping in a channel and channel swapping are not supported by the STM32CubeMX DDR tool.

The swizzle control registers depending on the protocol are listed in Table 6. The register values are determined by the STM32CubeMX DDR tool from PCB connections. These are provided by the user into a configuration panel. Some examples are shown in Table 13, Table 18, and Table 22.

Note: The mapping of the UIS_Swizzle group from the .h to the corresponding swizzle registers shown in Section 6.2, Section 7.2, and Section 8.2 correspond to validation boards. They are provided as reference design examples.

Table 6. Swizzle parameters

Lane	Register	DDR3	DDR4	LPDDR4
Byte 0	DDRPHYC_DBYTE0_DQ0LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ1LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ2LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ3LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ4LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ5LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ6LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ7LNSEL	-	-	y
Byte 1	DDRPHYC_DBYTE0_DQ0LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ1LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ2LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ3LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ4LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ5LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ6LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ7LNSEL	-	-	y
Byte 2	DDRPHYC_DBYTE0_DQ0LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ1LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ2LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ3LNSEL	-	-	y

Lane	Register	DDR3	DDR4	LPDDR4
-	DDRPHYC_DBYTE0_DQ4LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ5LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ6LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ7LNSEL	-	-	y
Byte 3	DDRPHYC_DBYTE0_DQ0LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ1LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ2LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ3LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ4LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ5LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ6LNSEL	-	-	y
-	DDRPHYC_DBYTE0_DQ7LNSEL	-	-	y
AC	DDRPHYC_MASTER0_MAPCAA0TODFI	-	-	y
-	DDRPHYC_MASTER0_MAPCAA1TODFI	-	-	y
-	DDRPHYC_MASTER0_MAPCAA2TODFI	-	-	y
-	DDRPHYC_MASTER0_MAPCAA3TODFI	-	-	y
-	DDRPHYC_MASTER0_MAPCAA4TODFI	-	-	y
-	DDRPHYC_MASTER0_MAPCAA5TODFI	-	-	y
-	DDRPHYC_MASTER0_MAPCAB0TODFI	-	-	y
-	DDRPHYC_MASTER0_MAPCAB1TODFI	-	-	y
-	DDRPHYC_MASTER0_MAPCAB2TODFI	-	-	y
-	DDRPHYC_MASTER0_MAPCAB3TODFI	-	-	y
-	DDRPHYC_MASTER0_MAPCAB4TODFI	-	-	y
-	DDRPHYC_MASTER0_MAPCAB5TODFI	-	-	y
AC	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS0	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS1	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS2	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS3	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS4	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS5	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS6	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS7	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS8	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS9	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS10	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS11	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS12	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS13	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS14	y	-	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS15	y	-	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTADDRESS17	-	-	-

Lane	Register	DDR3	DDR4	LPDDR4
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTACTN	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK0	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK1	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTBANK2	y	-	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTBG0	-	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTBG1	-	-	-
-	DDRPHYC_MASTER0_HwtSwizzleHwtCasN	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTRASN	y	y	-
-	DDRPHYC_MASTER0_HWTSWIZZLEHWTWEN	y	y	-
AC	DDRDBG_DDR34_AC_SWIZZLE_ADD3_0	y	y	-
-	DDRDBG_DDR34_AC_SWIZZLE_ADD7_4	y	y	-
-	DDRDBG_DDR34_AC_SWIZZLE_ADD11_8	y	y	-
-	DDRDBG_DDR34_AC_SWIZZLE_ADD15_12	y	y	-
-	DDRDBG_DDR34_AC_SWIZZLE_BK2_0_ACTN	y	y	-
-	DDRDBG_DDR34_AC_SWIZZLE_RASN_CASN_BG1_0	y	y	-
-	DDRDBG_DDR34_AC_SWIZZLE_WEN	y	y	-

The device pins DDR_Ax are used to output the address/command bus; Table 7 is listing the default swizzle mapping for each supported DDR protocol.

Table 7. AC pin mapping by device versus protocol (internal signal swap)

Device pin	LPDDR4	SDRAM DDR4	DDR3L	Notes
DDR_A0	CKEA0	CKE0	CKE0	-
DDR_A1	CKEA1	CKE1	CKE1	-
DDR_A2	CAA[0]	CS_N0	CS_N0	-
DDR_A3	CAA[1]	ODT0	ODT0	-
DDR_A4	CLKA_T	CLK0_T	CLK0_T	-
DDR_A5	CLKA_C	CLK0_C	CLK0_C	-
DDR_A6	CSA0	CS_N1	CS_N1	-
DDR_A7	CSA1	ODT1	ODT1	-
DDR_A8	CAA[2]	A[9]	A[9]	-
DDR_A9	CAA[3]	A[12]	A[12]	-
DDR_A10	CAA[4]	A[11]	A[11]	-
DDR_A11	CAA[5]	A[7]	A[7]	-
DDR_A12	CKEB0	A[8]	A[8]	-
DDR_A13	CKEB1	A[6]	A[6]	-
DDR_A14	CAB[0]	A[5]	A[5]	-
DDR_A15	CAB[1]	A[4]	A[4]	-
DDR_A16	CLKB_T	BG[1]	BA[2]	-
DDR_A17	CLKB_C	ACT_N	A[14]	-
DDR_A18	CSB0	BG[0]	A[15]	-

Device pin	LPDDR4	SDRAM DDR4	DDR3L	Notes
DDR_A19	CSB1	PAR	PAR	PAR is not used.
DDR_A20	CAB[2]	A[3]	A[3]	-
DDR_A21	CAB[3]	A[2]	A[2]	-
DDR_A22	CAB[4]	A[1]	A[1]	-
DDR_A23	CAB[5]	BA[1]	BA[1]	-
DDR_A24	-	-	-	PIN is not used.
DDR_A25	-	A[13]	A[13]	-
DDR_A26	-	BA[0]	BA[0]	-
DDR_A27	-	A[10]	A[10]	-
DDR_A28	-	A[0]	A[0]	-
DDR_A29	-	CAS_N/A[15]	CAS_N	-
DDR_A30	-	WE_N/A[14]	WE_N	-
DDR_A31	-	RAS_N/A[16]	RAS_N	-

The device pin DQs are internally swapped in SoC according to Table 8. This table corresponds to the default DQ swizzle settings. The device pin is a signal at the DRAM interface, while the DDRSS signal is an internal signal at the DFI interface. The internal swap from the DFI to DDR interface is shown in Table 8.

Table 8. DQ pin mapping by device (internal signal swap)

Device pin	DDRSS signal	Device pin	DDRSS signal	Device pin	DDRSS signal	Device pin	DDRSS signal
DQ0	DQ2	DQ8	DQ12	DQ16	DQ20	DQ24	DQ31
DQ1	DQ3	DQ9	DQ15	DQ17	DQ21	DQ25	DQ30
DQ2	DQ1	DQ10	DQ14	DQ18	DQ23	DQ26	DQ28
DQ3	DQ0	DQ11	DQ13	DQ19	DQ22	DQ27	DQ29
DQ4	DQ7	DQ12	DQ9	DQ20	DQ19	DQ28	DQ25
DQ5	DQ6	DQ13	DQ8	DQ21	DQ16	DQ29	DQ26
DQ6	DQ4	DQ14	DQ11	DQ22	DQ18	DQ30	DQ27
DQ7	DQ5	DQ15	DQ10	DQ23	DQ17	DQ31	DQ24

To illustrate the swizzle programing, let us take two examples.

- DQ bit for LPDDR4
  Assuming one-to-one PCB connections, according to Table 8, SDRAM DQ0 corresponds to the signal DQ2. Therefore, DBYTE0_DqLnSel2 has to be programmed for the end-to-end connection of DQ2.
- AC bit for DDR3/4
  Assuming DDR_A8 is connected to DDR4 pin A3 on the PCB, then according to Table 7, SDRAM pin A3 corresponds to the A9 signal. Therefore, HWTSWIZZLEADDRESS9 has to be positioned to A3 (for example, code 0x4) for end-to-end connection of A3. This code is also applicable to DDRDBG_DDR34AC_SWIZZLE_ADD11_8-bit field for the functional signal path.

6 DDR3L configuration

6.1 General considerations about DDR3L

Below general information about the DDR mode support: DDR3L

For more information, refer to [4.], [2.], and [7.].

Supplies and references

- The power supplies are the following:
  V_{DDQ} = 1.35 V
  V_{REF} = 0.675 V
- The voltage references are:
  External V_{REF} and V_{TT} are used at SDRAM.
  Device side V_{REF} is generated internally.

Density

DDR3L is available in BGA with 16-bit interface.

DDR3L supported packages are:

- 16 bits: single BGA in P2P and density is from 1 to 8 Gbit.
- 32 bits: two BGA in fly-by topology and density is from 1 to 8 Gbit per die (max = 2 Gbytes in total).

Note: 8 Gbit DDR3 defined by JEDEC is uncommon. Practically, DDR3L density is limited to 4 Gbits (max = 1 Gbyte in total). Single rank only is required.

Frequency

DDR3L frequency can be set with the following constraints:

- The frequency must be set between 300 MHz and 1066 MHz. 300 MHz is DDR3L lower limit with DLL on.
- DDR3L with DLL off when the frequency is equal to 125 MHz or less is not validated.

DDR3L speed grade and bins table are not described in this document, only the use of the ‘worse’ bin according to frequency grade is proposed.

Impedance

R_{ON} is set to 40 Ω on device according to address Tx impedance control register (ATxImpedance) and Tx impedance control register (TxImpedance).

R_{ON} is set to 40 Ω at DDR3L by MR1.

Terminations

ODT is set to 53 Ω at device according to on-die termination impedance (ODtImpedance).

ODT is set to 60 Ω at DDR3L by MR (RTT_WR), RTT_NOM is disabled.

Extended temperature range support

MR2 ASR and SRT modes are used to control the self-refresh in order to support extended temperature range. In case of extended temperature, use ASR = 1 when DDR3L supports this optional feature.

If ASR is not supported, then SRT can be used to indicate operating temperature for subsequent self-refresh intervals.

The proposed address mapping is fixed to Row > BA2 / BA1/ BA0 > columns and DDRCTRL ADDRMAP registers are determined accordingly.

There are 10 column bits (2-Kbyte page for a 16-bit device).

6.2 DDR3L configuration example

Below is an example of DDR3L configuration using the following parameters:

- Frequency = 933 MHz
- DDR3L-1866M 13-13-13 configured with ‘worse’ settings full populated
- 4 Gbits/channel (1 Gbyte in total)
  The total density is reported by the DDR tool as 8 Gbit.
- RBC addressing
- Modified QoS
- The Command/Address according to DDR3 validation board PCB connections for the A/C group is shown in Table 9.

Note: To simplify the speed grade and bin selection, the ‘worse’ setting is selected by the STM32CubeMX DDR tool. The objective is to provide the most conservative timings that have the little impact on performance.

Note: The following table is provided as a typical configuration example. Refer to actual values provided by the STM32CubeMX DDR tool.

Table 9. DDRCTRL parameters

Registers	Values	Registers	Values	Registers	Values	Registers	Values
MSTR	0x01040001	DIMMCTL	0x00000000	DFITMG0	0x038A8206	SCHED	0x00001B00
MRCTRL0	0x00000030	RANKCTL	0x0000066F	DFITMG1	0x00080303	SCHED1	0x00000000
MRCTRL1	0x00000000	DRAMTMG0	0x0E181F12	DFILPCFG0	0x07F04111	PERFHPR1	0x04000200
MRCTRL2	0x00000000	DRAMTMG1	0x00040419	DFILPCFG1	0x000000F0	PERFLPR1	0x08000080
DERATEEN	0x00000000	DRAMTMG2	0x0507050B	DFIUPD0	0xC0300018	PERFWR1	0x08000400
DERATEINT	0x00E3C880	DRAMTMG3	0x00502007	DFIUPD1	0x005700B4	DBG0	0x00000000
DERATECTL	0x00000000	DRAMTMG4	0x07020508	DFIUPD2	0x80000000	DBG1	0x00000000
PWRCTL	0x00000008	DRAMTMG5	0x05050403	DFIMISC	0x00000041	DBGCMD	0x00000000
PWRTMG	0x000F0001	DRAMTMG6	0x02020005	DFITMG2	0x00000A06	SWCTL	0x00000000
HWLPCTL	0x00000002	DRAMTMG7	0x00000505	DFITMG3	0x00000000	POISONCFG	0x00000000
RFSHCTL0	0x00210010	DRAMTMG8	0x03030804	DBICTL	0x00000001	PCCFG	0x00000000
RFSHCTL1	0x00000000	DRAMTMG9	0x0004040D	DFIPHYMSTR	0x80000000	PCFGR_0	0x07004100
RFSHCTL3	0x00000000	DRAMTMG10	0x001C180A	ADDRMAP0	0x0000001F	PCFGW_0	0x00004100
RFSHTMG	0x0071007A	DRAMTMG11	0x440C021C	ADDRMAP1	0x00080808	PCTRL_0	0x00000000
RFSHTMG1	0x008C0000	DRAMTMG12	0x1A020010	ADDRMAP2	0x00000000	PCFGQOS0_0	0x0021000C
CRCPARCTL0	0x00000000	DRAMTMG13	0x1C200004	ADDRMAP3	0x00000000	PCFGQOS1_0	0x01000080
CRCPARCTL1	0x00001000	DRAMTMG14	0x000000A0	ADDRMAP4	0x00001F1F	PCFGWQOS0_0	0x01100C07
INIT0	0xC002004E	DRAMTMG15	0x00000000	ADDRMAP5	0x070F0707	PCFGWQOS1_0	0x04000200
INIT1	0x00000000	ZQCTL0	0x00960026	ADDRMAP6	0x0F070707	PCFGR_1	0x07004100
INIT2	0x00000D00	ZQCTL1	0x12B038F2	ADDRMAP7	0x00000F0F	PCFGW_1	0x00004100
INIT3	0x1F140000	ZQCTL2	0x00000000	ADDRMAP8	0x00003F3F	PCTRL_1	0x00000000
INIT4	0x02200000	ODTCFG	0x06000610	ADDRMAP9	0x07070707	PCFGQOS0_1	0x00100007
INIT5	0x00130004	ODTMAP	0x00000001	ADDRMAP10	0x07070707	PCFGQOS1_1	0x01000080
INIT6	0x00000000	-	-	ADDRMAP11	0x00000007	PCFGWQOS0_1	0x01100C07
INIT7	0x00000000	-	-	-	-	PCFGWQOS1_1	0x04000200

Table 10. PHY basic and advanced parameters

Basic parameters: UIB_		Advanced parameters: UIA_			
UIB_DRAMTYPE	0x00000001	UIA_LP4RXPREAMBLEMODE_0	0x00000000	UIA_LP4RL_0	0x00000000
UIB_DIMMTYPE	0x00000004	UIA_LP4POSTAMBLEEXT_0	0x00000000	UIA_LP4WL_0	0x00000000
UIB_LP4XMODE	0x00000000	UIA_D4RXPREAMBLELENGTH_0	0x00000001	UIA_LP4WLS_0	0x00000000
UIB_NUMDBYTE	0x00000004	UIA_D4TXPREAMBLELENGTH_0	0x00000000	UIA_LP4DBIRD_0	0x00000000
UIB_NUMACTIVEDBYTEDFI0	0x00000004	UIA_EXTCALRESVAL	0x00000000	UIA_LP4DBIWR_0	0x00000000
UIB_NUMACTIVEDBYTEDFI1	0x00000000	UIA_IS2TTIMING_0	0x00000000	UIA_LP4NWR_0	0x00000000
UIB_NUMANIB	0x00000008	UIA_ODTIMPEDANCE_0	0x00000035	UIA_LP4LOWPOWERDRV	0x00000000
UIB_NUMRANK_DFI0	0x00000001	UIA_TXIMPEDANCE_0	0x00000028	UIA_DRAMBYTESWAP	0x00000000
UIB_NUMRANK_DFI1	0x00000001	UIA_ATXIMPEDANCE	0x00000028	UIA_RXENBACKOFF	0x00000000
UIB_DRAMDATAWIDTH	0x00000010	UIA_MEMALERTEN	0x00000000	UIA_TRAINSEQUENCECTRL	0x00000000
UIB_NUMPSTATES	0x00000001	UIA_MEMALERTPUIMP	0x00000000	UIA_SNPSUMCTLOPT	0x00000000
UIB_FREQUENCY_0	0x000003A5	UIA_MEMALERTVREFLEVEL	0x00000000	UIA_SNPSUMCTLF0RC5X_0	0x00000000
UIB_PLLBYPASS_0	0x00000000	UIA_MEMALERTSYNCBYPASS	0x00000000	UIA_TXSLEWRISEDQ_0	0x0000000F
UIB_DFIFREQRATIO_0	0x00000001	UIA_DISDYNADRTRI_0	0x00000001	UIA_TXSLEWFALLDQ_0	0x0000000F
UIB_DFI1EXISTS	0x00000001	UIA_PHYMSTRTRAININTERVAL_0	0x00000000	UIA_TXSLEWRISEAC	0x0000000F
UIB_TRAIN2D	0x00000000	UIA_PHYMSTRMAXREQTOACK_0	0x00000000	UIA_TXSLEWFALLAC	0x0000000F
UIB_HARDMACROVER	0x00000003	UIA_WDQSEXT	0x00000000	UIA_DISABLERETRAINING	0x00000001
UIB_READDBIENABLE_0	0x00000000	UIA_CALINTERVAL	0x00000009	UIA_DISABLEPHYUPDATE	0x00000000
UIB_DFIMODE	0x00000000	UIA_CALONCE	0x00000000	UIA_ENABLEHIGHCLKSKEWFIX	0x00000000
-	-	UIA_PHYVREF	0x00000040	UIA_DISABLEUNUSEDADDRLNS	0x00000001
-	-	UIA_SEQUENCECTRL_0	0x0000031F	UIA_PHYINITSEQUENCENUM	0x00000000
-	-	-	-	UIA_ENABLEDFICSPOLARITYFIX	0x00000000

Table 11. Mode registers

MRs	
MR0_0	0x00001F14
MR1_0	0x00000000
MR2_0	0x00000220
MR3_0	0x00000000

Table 12. DDR3 validation board PCB connections

Device Ball	SDRAM
DDR_A8	A1
DDR_A9	A10
DDR_A10	A15
DDR_A11	RASN
DDR_A12	BA0
DDR_A13	A3
DDR_A14	WEN
DDR_A15	A0
DDR_A16	BA2
DDR_A17	A14
DDR_A18	A9
DDR_A20	A7
DDR_A21	A2
DDR_A22	A13
DDR_A23	A5
DDR_A25	A8
DDR_A26	A11
DDR_A27	BA1
DDR_A28	CASN
DDR_A29	A4
DDR_A30	A12
DDR_A31	A6

Note: DDR_A0 to DDR_A7 are primary signals that cannot be swapped. The PCB connections are used as an example to illustrate the swizzle configuration. The user is expected to enter the correct PCB connections into the STM32CubeMX DDR tool.

Table 13. Swizzle parameters for the DDR3 validation board PCB

Swizzle: UIS_			
UIS_SWIZZLE_0	0x00000018	UIS_SWIZZLE_24	0x00000007
UIS_SWIZZLE_1	0x0000000E	UIS_SWIZZLE_25	0x0000000D
UIS_SWIZZLE_2	0x00000000	UIS_SWIZZLE_26	0x08000E18
UIS_SWIZZLE_3	0x00000008	UIS_SWIZZLE_27	0x19041A01
UIS_SWIZZLE_4	0x00000001	UIS_SWIZZLE_28	0x10140213
UIS_SWIZZLE_5	0x0000001A	UIS_SWIZZLE_29	0x0A00090B
UIS_SWIZZLE_6	0x00000004	UIS_SWIZZLE_30	0x00060C00
UIS_SWIZZLE_7	0x00000019	UIS_SWIZZLE_31	0x07050000
UIS_SWIZZLE_8	0x00000013	UIS_SWIZZLE_32	0x0000000D
UIS_SWIZZLE_9	0x00000002	-	-
UIS_SWIZZLE_10	0x00000014	-	-
UIS_SWIZZLE_11	0x00000010	-	-
UIS_SWIZZLE_12	0x0000000B	-	-
UIS_SWIZZLE_13	0x00000009	-	-
UIS_SWIZZLE_14	0x00000000	-	-
UIS_SWIZZLE_15	0x0000000A	-	-
UIS_SWIZZLE_16	0x00000000	-	-
UIS_SWIZZLE_17	0x00000000	-	-
UIS_SWIZZLE_18	0x0000000C	-	-
UIS_SWIZZLE_19	0x00000006	-	-
UIS_SWIZZLE_20	0x00000000	-	-
UIS_SWIZZLE_21	0x00000000	-	-
UIS_SWIZZLE_22	0x00000000	-	-
UIS_SWIZZLE_23	0x00000005	-	-

7 DDR4 configuration

7.1 General considerations about DDR4

Below is general information about the DDR mode support: DDR4

For more information, refer to JEDEC JESD79-4D [5.], [2.], and [7.].

Supplies and references

- The power supplies are the following:
  V_{DDQ} = 1.20 V
  V_{REF} = 0.6 V
  An additional V_{PP} = 2.5 V is required.
- The voltage references are:
  External V_{REF} and VTT on PCB for CA bus
  Internal V_{REF} and ODT at SDRAM for DQ bus
  On the device, V_{REF} is generated internally according to a fixed PhyV_{REF} = 0x56 so V_{REF} = 67.2% V_{DDQ}.

Density

DDR4 is commonly available in BGA with 16-bit interface and densities of 2, 4, 8, or 16 Gbit/s.

Note: DDR4 is also available in BGA with twin-die in 8 and 16-Gbit density per die. This configuration is supporting a total density of 32 Gbit with a 16-bit memory interface. This specific mode is set by setting the parameter 'device_width' to 8 instead of 16.

DDR4 supported packages are:

- 16 bits with single BGA in p2p
- 32 bits with two BGA in fly-by topology

Note: Single rank only is required to support up to 4-Gbyte total density.

Frequency

DDR4 frequency can be set with the following constraints:

- A frequency between 625 MHz and 1200 MHz, knowing that 625 MHz is the DDR4 lower limit with DLL ON.
- DDR4 with DLL off when the frequency is equal to 125 MHz or less.

Impedance

R_{ON} is set to 40 Ω on the device according to the address Tx impedance control register (ATxImpedance) that is equal to 0x28 and Tx impedance control register (TxImpedance) TxImpedance that is equal to 0x28.

R_{ON} is set to 48 Ω at DDR4 by MR2.

Terminations

ODT is set to 53 Ω on the device according to on-die termination impedance (ODTImpedance) that is equal to 0x3C.

ODT is set to 60 Ω at DDR4 by MR1 (RTT_NOM) and MR5 (RTT_PARK).

RTT_wr is off and a constant 60 Ω termination is present a DDR4, except during the Read mode. This is insensitive to ODT signal position.

Preamble and Postamble

One clock Read Preamble is fixed at all frequencies < 1200 MHz (two clocks are never needed).

One clock Write Preamble is fixed at all frequencies < 1200 MHz.

0.5 clock Read Postamble is fixed at all frequencies < 1200 MHz.

0.5 clock Write Postamble is fixed at all frequencies < 1200 MHz.

Address mapping

The proposed address mapping is fixed to row > BA1 / BA0 / BG0 > columns and DDRCTRL ADDRMAP registers determined accordingly. BG1 is only applicable to devices in 8 bits. It should be mapped between BA0 and BG0.

There are 10 column bits (2-Kbyte page for a 16-bit device).

Other parameters

LPASR (low-power ASW according to T) is set to normal (MR2).

DBI read are not enabled by default. DBI write is not supported (MR5).

FGRM (Fine grained refresh mode) is set to fixed 1X (MR3).

7.2 DDR4 configuration example

Below is an example of DDR4 configuration using the following parameters:

- Frequency = 1200 MHz
- DDR4-2400R 16-16-16 configured with ‘worse’ settings
- Full populated
- 8 Gbit/channel (2 Gbytes in total)
  The total density is reported by the DDR tool as 16 Gbit.
- RBC addressing
- Modified QoS
- Command/address according to PCB connections for A/C group in Table 17

Note: To simplify the speed grade and bin selection, it is possible to apply the ‘worse’ setting in the STM32CubeMX DDR tool. The objective is to use the most conservative timings with little impact to performances.

Note: The following table is provided as a typical configuration example. Refer to actual values provided by the STM32CubeMX DDR tool.

Table 14. DDRCTRL parameters

Registers	Values	Registers	Values	Registers	Values	Registers	Values
MSTR	0x01040010	DIMMCTL	0x00000000	DFITMG0	0x038F8209	SCHED	0x80001B00
MRCTRL0	0x00000030	RANKCTL	0x0000066F	DFITMG1	0x00080303	SCHED1	0x00000000
MRCTRL1	0x00000000	DRAMTMG0	0x11152815	DFILPCFG0	0x07004111	PERFHPR1	0x04000200
MRCTRL2	0x00000000	DRAMTMG1	0x0004051E	DFILPCFG1	0x000000F0	PERFLPR1	0x08000080
DERATEEN	0x00000000	DRAMTMG2	0x0609060D	DFIUPD0	0xC0300018	PERFWR1	0x08000400
DERATEINT	0x0124F800	DRAMTMG3	0x0050400C	DFIUPD1	0x005700B4	DBG0	0x00000000
DERATECTL	0x00000000	DRAMTMG4	0x0904050A	DFIUPD2	0x80000000	DBG1	0x00000000
PWRCTL	0x00000008	DRAMTMG5	0x06060403	DFIMISC	0x00000041	DBGCMD	0x00000000
PWRTMG	0x00130001	DRAMTMG6	0x02020005	DFITMG2	0x00000F09	SWCTL	0x00000000
HWLPCTL	0x00000002	DRAMTMG7	0x00000606	DFITMG3	0x00000000	POISONCFG	0x00000000
RFSHCTL0	0x00210010	DRAMTMG8	0x04041007	DBICTL	0x00000001	PCCFG	0x00000000
RFSHCTL1	0x00000000	DRAMTMG9	0x0002040A	DFIPHYMSTR	0x80000000	PCFGR_0	0x00704100
RFSHCTL3	0x00000000	DRAMTMG10	0x001C180A	ADDRMAP0	0x0000001F	PCFGW_0	0x00004100
RFSHTMG	0x009200D2	DRAMTMG11	0x4408021C	ADDRMAP1	0x003F0909	PCTRL_0	0x00000000
RFSHTMG1	0x008C0000	DRAMTMG12	0x0C020010	ADDRMAP2	0x00000000	PCFGQOS0_0	0x0021000C
CRCPARCTL0	0x00000000	DRAMTMG13	0x1C200004	ADDRMAP3	0x00000000	PCFGQOS1_0	0x01000080
CRCPARCTL1	0x00001000	DRAMTMG14	0x000000A0	ADDRMAP4	0x00001F1F	PCFGWQOS0_0	0x00000C07
INIT0	0xC0020002	DRAMTMG15	0x00000000	ADDRMAP5	0x070F0707	PCFGWQOS1_0	0x04000200
INIT1	0x00010002	ZQCTL0	0x01000040	ADDRMAP6	0x07070707	PCFGR_1	0x00704100
INIT2	0x00000D00	ZQCTL1	0x2000493E	ADDRMAP7	0x00000F0F	PCFGW_1	0x00004100
INIT3	0x09400103	ZQCTL2	0x00000000	ADDRMAP8	0x00003F08	PCTRL_1	0x00000000
INIT4	0x00180000	ODTCFG	0x06000618	ADDRMAP9	0x07070707	PCFGQOS0_1	0x00100007
INIT5	0x00100004	ODTMAP	0x00000001	ADDRMAP10	0x07070707	PCFGQOS1_1	0x04000200
INIT6	0x00080440	-	-	ADDRMAP11	0x00000007	PCFGWQOS0_1	0x01100C07
INIT7	0x00000C0F	-	-	-	-	PCFGWQOS1_1	0x04000200

Table 15. PHY basic and advanced parameters

Basic parameters: UIB_		Advanced parameters: UIA_			
UIB_DRAMTYPE	0x00000001	UIA_LP4RXPREAMBLEMODE_0	0x00000000	UIA_LP4RL_0	0x00000000
UIB_DIMMTYPE	0x00000004	UIA_LP4POSTAMBLEEXT_0	0x00000000	UIA_LP4WL_0	0x00000000
UIB_LP4XMODE	0x00000000	UIA_D4RXPREAMBLELENGTH_0	0x00000001	UIA_LP4WLS_0	0x00000000
UIB_NUMDBYTE	0x00000004	UIA_D4TXPREAMBLELENGTH_0	0x00000000	UIA_LP4DBIRD_0	0x00000000
UIB_NUMACTIVEDBYTEDFI0	0x00000004	UIA_EXTCALRESVAL	0x00000000	UIA_LP4DBIWR_0	0x00000000
UIB_NUMACTIVEDBYTEDFI1	0x00000000	UIA_IS2TTIMING_0	0x00000000	UIA_LP4NWR_0	0x00000000
UIB_NUMANIB	0x00000008	UIA_ODTIMPEDANCE_0	0x0000003C	UIA_LP4LOWPOWERDRV	0x00000000
UIB_NUMRANK_DFI0	0x00000001	UIA_TXIMPEDANCE_0	0x00000028	UIA_DRAMBYTESWAP	0x00000000
UIB_NUMRANK_DFI1	0x00000001	UIA_ATXIMPEDANCE	0x00000028	UIA_RXENBACKOFF	0x00000000
UIB_DRAMDATAWIDTH	0x00000010	UIA_MEMALERTEN	0x00000000	UIA_TRAINSEQUENCECTRL	0x00000000
UIB_NUMPSTATES	0x00000001	UIA_MEMALERTPUIMP	0x00000000	UIA_SNPSUMCTLOPT	0x00000000
UIB_FREQUENCY_0	0x000003A5	UIA_MEMALERTVREFLEVEL	0x00000000	UIA_SNPSUMCTLF0RC5X_0	0x00000000
UIB_PLLBYPASS_0	0x00000000	UIA_MEMALERTSYNCBYPASS	0x00000000	UIA_TXSLEWRISEDQ_0	0x0000000F
UIB_DFIFREQRATIO_0	0x00000001	UIA_DISDYNADRTRI_0	0x00000001	UIA_TXSLEWFALLDQ_0	0x0000000F
UIB_DFI1EXISTS	0x00000001	UIA_PHYMSTRTRAININTERVAL_0	0x00000000	UIA_TXSLEWRISEAC	0x0000000F
UIB_TRAIN2D	0x00000000	UIA_PHYMSTRMAXREQTOACK_0	0x00000000	UIA_TXSLEWFALLAC	0x0000000F
UIB_HARDMACROVER	0x00000003	UIA_WDQSEXT	0x00000000	UIA_DISABLERETRAINING	0x00000001
UIB_READDBIENABLE_0	0x00000000	UIA_CALINTERVAL	0x00000009	UIA_DISABLEPHYUPDATE	0x00000000
UIB_DFIMODE	0x00000000	UIA_CALONCE	0x00000000	UIA_ENABLEHIGHCLKSKEWFIX	0x00000000
-	-	UIA_PHYVREF	0x00000040	UIA_DISABLEUNUSEDADDRLNS	0x00000001
-	-	UIA_SEQUENCECTRL_0	0x0000031F	UIA_PHYINITSEQUENCENUM	0x00000000
-	-	-	-	UIA_ENABLEDFICSPOLARITYFIX	0x00000000

Table 16. Mode registers

MRs	
MR0_0	0x00000940
MR1_0	0x00000103
MR2_0	0x00000018
MR3_0	0x00000000
MR4_0	0x00000008
MR5_0	0x00000460
MR6_0	0x00000C0F

Mode registers [16] are configured according to the following parameters:

- CL = 16, CWL = 12, WR/RTP = 9
- RPRE = WPRE = 1CK
- ODI = 34 Ω; RTT_NOM = RTT_PARK = 60 Ω
- TCCDL = 6
- V_{REF} = 69.75% (code 0xF)

Table 17. DDR4 validation board PCB connections

Device DDR ball	SDRAM
DDR_A8	A3
DDR_A9	BA1
DDR_A10	A12
DDR_A11	RASN
DDR_A12	A6
DDR_A13	A0
DDR_A14	A2
DDR_A15	A8
DDR_A17	ACTN
DDR_A18	BG0
DDR_A20	WEN
DDR_A21	BA0
DDR_A22	A4
DDR_A23	A10
DDR_A25	A13
DDR_A26	A5
DDR_A27	A9
DDR_A28	A11
DDR_A29	A7
DDR_A30	CASN
DDR_A31	A1

Note: DDR_A0 to DDR_A7 are primary signals, which cannot be swapped. The PCB connections are used as an example to illustrate the swizzle configuration. The user is expected to enter the correct PCB connections into the STM32CubeMX DDR tool.

Table 18. Swizzle parameters for DDR4 validation board

Swizzle: UIS_			
UIS_SWIZZLE_0	0x0000000C	UIS_SWIZZLE_24	0x00000002
UIS_SWIZZLE_1	0x00000005	UIS_SWIZZLE_25	0x00000018
UIS_SWIZZLE_2	0x00000013	UIS_SWIZZLE_26	0x1A13050C
UIS_SWIZZLE_3	0x0000001A	UIS_SWIZZLE_27	0x19010309
UIS_SWIZZLE_4	0x00000009	UIS_SWIZZLE_28	0x0D0A0407
UIS_SWIZZLE_5	0x00000003	UIS_SWIZZLE_29	0x00000014
UIS_SWIZZLE_6	0x00000001	UIS_SWIZZLE_30	0x000B0600
UIS_SWIZZLE_7	0x00000019	UIS_SWIZZLE_31	0x02080000
UIS_SWIZZLE_8	0x00000007	UIS_SWIZZLE_32	0x00000018
UIS_SWIZZLE_9	0x00000004	-	-
UIS_SWIZZLE_10	0x0000000A	-	-
UIS_SWIZZLE_11	0x0000000D	-	-
UIS_SWIZZLE_12	0x00000014	-	-
UIS_SWIZZLE_13	0x00000000	-	-
UIS_SWIZZLE_14	0x00000000	-	-
UIS_SWIZZLE_15	0x00000000	-	-
UIS_SWIZZLE_16	0x00000000	-	-
UIS_SWIZZLE_17	0x00000000	-	-
UIS_SWIZZLE_18	0x00000006	-	-
UIS_SWIZZLE_19	0x0000000B	-	-
UIS_SWIZZLE_20	0x00000000	--	-
UIS_SWIZZLE_21	0x00000000	-	-
UIS_SWIZZLE_22	0x00000000	-	-
UIS_SWIZZLE_23	0x00000008	-	-

8 LPDDR4 configuration

8.1 General considerations about LPDDR4

Below is general information about the DDR mode support: LPDDR4

For more information, refer to JEDEC JESD209-4D in [6.], [2.], and [7.].

Supplies and references

V_{DDQ} = 1.1 V

V_{REF} for AC and DQ is generated internally. SoC side V_{REF} is generated internally. LPDDR4 is commonly available in BGA200 internally and is configured with either one or two 16-bit channels. The two channels (CAA and CAB) are used in parallel.

Density

Single-channel devices have density from 1 to 16 Gbit and 8 Gbit density is the most common. Dual-channel devices have density from 2 to 32 Gbit.

LPDDR4 supported packages are:

- 16 bits: single channel and half-populated interface.
- 32 bits: two channels and a full-populated interface.

Note:

- *Single rank only is required, with 8 Gbit per channel (x16) to support a capacity of 2 Gbytes.*
- *Single rank only is required, with 16 Gbit per channel (x16) to support a capacity of 4 Gbytes.*

Frequency

LPDDR4 frequency can be set with the following constraints:

- The frequency is between 10 MHz and 1200 MHz.
- PLL is bypassed when the frequency is equal to 333 MHz or less.

There is no speed grade/speed bin for LPDDR4 below 3200 Mbps.

Note: All timing parameters, including RL, WL, N_{WR}, and N_{RT}, are calculated from frequency (MEMCLK in MHz) using ≤ formula.

For example: 799 MHz and 800 MHz are in the same range. 801 MHz is in the next range.

Impedances

- The device R_{ON} is set to 40 Ω based on the values in the address Tx impedance control register (ATxImpedance) and Tx impedance control register (TxImpedance).
- DRAM: PDDS is set to 40 Ω.
- PU_CAL is set to 1/3.

Terminations

- Device: ODT is set to 53 Ω according to the on-die termination impedance parameter.
- SDRAM: AC and DQ ODT set to 40 Ω according to MR11.

Preamble and postamble

- Read: preamble static, postamble 1.5 clock
- Write: preamble 2 CK, postamble 0.5 clock

Other parameters

- Device side V_{REF} is set according to PU_CAL for the PHYVREF parameter.
- DRAM CA and DQ V_{REF} are determined from device impedances and DRAM ODTs and programmed via MR12/MR14.
- MR4 polling is activated by default on the DDRCTRL configuration using 20 ms poll period for refresh and timing derating according to temperature.
- Per bank refresh is proposed by default.
- DBI read and write are not enabled by default.
- DDC (DRAM drift compensation) requires periodic PHY retraining. This feature is enabled by default according to UIA_PHYMSTRTRAININTERVAL_0 using the DFI PHYMSTR. It may impact system latency by 2uS. DDC period is set to maximum value (~224 ms).

8.2 LPDDR4 configuration example

Below is an example of the LPDDR4 configuration using the following parameters:

- The frequency equals 1200 MHz.
- It is fully populated.
- It has 16 Gbit per channel (4 Gbytes in total).
  The total density is reported by the DDR tool as 32 Gbit.
- RBC addressing is included.
- It has default QoS.
- One-to-one PCB connections for CAA, CAB, and DQ groups are set.

Note: The following table is provided as a typical configuration example. Refer to actual values provided by the STM32CubeMX DDR tool.

Table 19. DDRCTRL registers

Registers	Values	Registers	Values	Registers	Values	Registers	Values
MSTR	0x01080020	DIMMCTL	0x00000000	DFITMG0	0x0395820A	SCHED	0x80001B00
MRCTRL0	0x00000030	RANKCTL	0x0000066F	DFITMG1	0x000A0303	SCHED1	0x00000000
MRCTRL1	0x00000000	DRAMTMG0	0x1718141A	DFILPCFG0	0x07F04111	PERFHPR1	0x04000200
MRCTRL2	0x00000000	DRAMTMG1	0x00050524	DFILPCFG1	0x000000F0	PERFLPR1	0x08000080
DERATEEN	0x00000203	DRAMTMG2	0x060C0E11	DFIUPD0	0x4040000C	PERFWR1	0x08000400
DERATEINT	0x0124F800	DRAMTMG3	0x0090900C	DFIUPD1	0x0040007F	DBG0	0x00000000
DERATECTL	0x00000000	DRAMTMG4	0x0B04060B	DFIUPD2	0x00000000	DBG1	0x00000000
PWRCTL	0x00000108	DRAMTMG5	0x02030909	DFIMISC	0x00000041	DBGCMD	0x00000000
PWRTMG	0x00130001	DRAMTMG6	0x02020007	DFITMG2	0x0000150A	SWCTL	0x00000000
HWLPCTL	0x00000002	DRAMTMG7	0x00000302	DFITMG3	0x00000000	POISONCFG	0x00000000
RFSHCTL0	0x00210074	DRAMTMG8	0x03034405	DBICTL	0x00000001	PCCFG	0x00000000
RFSHCTL1	0x00000000	DRAMTMG9	0x0004040D	DFIPHYMSTR	0x80000001	PCFGR_0	0x00704100
RFSHCTL3	0x00000000	DRAMTMG10	0x001C180A	ADDRMAP0	0x0000001F	PCFGW_0	0x00004080
RFSHTMG	0x81240054	DRAMTMG11	0x440C021C	ADDRMAP1	0x00080808	PCTRL_0	0x00000000
RFSHTMG1	0x00360000	DRAMTMG12	0x1A020010	ADDRMAP2	0x00000000	PCFGQOS0_0	0x0021000C
CRCPARCTL0	0x00000000	DRAMTMG13	0x0B100002	ADDRMAP3	0x00000000	PCFGQOS1_0	0x01000080
CRCPARCTL1	0x00001000	DRAMTMG14	0x000000E9	ADDRMAP4	0x00001F1F	PCFGWQOS0_0	0x01100C07
INIT0	0xC0020002	DRAMTMG15	0x00000000	ADDRMAP5	0x070F0707	PCFGWQOS1_0	0x04000200
INIT1	0x00010002	ZQCTL0	0x02580012	ADDRMAP6	0x07070707	PCFGR_1	0x00704100
INIT2	0x00000D00	ZQCTL1	0x01E0493E	ADDRMAP7	0x00000F07	PCFGW_1	0x00004100
INIT3	0x00C40024	ZQCTL2	0x00000000	ADDRMAP8	0x00003F3F	PCTRL_1	0x00000000
INIT4	0x00310008	ODTCFG	0x04000400	ADDRMAP9	0x07070707	PCFGQOS0_1	0x00100007
INIT5	0x00100004	ODTMAP	0x00000000	ADDRMAP10	0x07070707	PCFGQOS1_1	0x01000080
INIT6	0x00660047	-	-	ADDRMAP11	0x00000007	PCFGWQOS0_1	0x01100C07
INIT7	0x00040047	-	-	-	-	PCFGWQOS1_1	0x04000200

Table 20. PHY basic and advanced parameters

Basic parameters: UIB_		Advanced parameters: UIA_			
UIB_DRAMTYPE	0x00000002	UIA_LP4RXPREAMBLEMODE_0	0x00000000	UIA_LP4RL_0	0x00000004
UIB_DIMMTYPE	0x00000004	UIA_LP4POSTAMBLEEXT_0	0x00000001	UIA_LP4WL_0	0x00000004
UIB_LP4XMODE	0x00000000	UIA_D4RXPREAMBLELENGTH_0	0x00000001	UIA_LP4WLS_0	0x00000000
UIB_NUMDBYTE	0x00000004	UIA_D4TXPREAMBLELENGTH_0	0x00000000	UIA_LP4DBIRD_0	0x00000000
UIB_NUMACTIVEDBYTEDFI0	0x00000002	UIA_EXTCALRESVAL	0x00000000	UIA_LP4DBIWR_0	0x00000000
UIB_NUMACTIVEDBYTEDFI1	0x00000002	UIA_IS2TTIMING_0	0x00000000	UIA_LP4NWR_0	0x00000004
UIB_NUMANIB	0x00000008	UIA_ODTIMPEDANCE_0	0x00000035	UIA_LP4LOWPOWERDRV	0x00000000
UIB_NUMRANK_DFI0	0x00000001	UIA_TXIMPEDANCE_0	0x00000028	UIA_DRAMBYTESWAP	0x00000000
UIB_NUMRANK_DFI1	0x00000001	UIA_ATXIMPEDANCE	0x00000028	UIA_RXENBACKOFF	0x00000000
UIB_DRAMDATAWIDTH	0x00000010	UIA_MEMALERTEN	0x00000000	UIA_TRAINSEQUENCECTRL	0x00000000
UIB_NUMPSTATES	0x00000001	UIA_MEMALERTPUIMP	0x00000000	UIA_SNPSUMCTLOPT	0x00000000
UIB_FREQUENCY_0	0x000004B0	UIA_MEMALERTVREFLEVEL	0x00000000	UIA_SNPSUMCTLF0RC5X_0	0x00000000
UIB_PLLBYPASS_0	0x00000000	UIA_MEMALERTSYNCBYPASS	0x00000000	UIA_TXSLEWRISEDQ_0	0x0000000F
UIB_DFIFREQRATIO_0	0x00000001	UIA_DISDYNADRTRI_0	0x00000001	UIA_TXSLEWFALLDQ_0	0x0000000F
UIB_DFI1EXISTS	0x00000001	UIA_PHYMSTRTRAININTERVAL_0	0x0000000A	UIA_TXSLEWRISEAC	0x0000000F
UIB_TRAIN2D	0x00000000	UIA_PHYMSTRMAXREQTOACK_0	0x00000005	UIA_TXSLEWFALLAC	0x0000000F
UIB_HARDMACROVER	0x00000003	UIA_WDQSEXT	0x00000000	UIA_DISABLERETRAINING	0x00000000
UIB_READDBIENABLE_0	0x00000000	UIA_CALINTERVAL	0x00000009	UIA_DISABLEPHYUPDATE	0x00000001
UIB_DFIMODE	0x00000000	UIA_CALONCE	0x00000000	UIA_ENABLEHIGHCLKSKEWFIX	0x00000000
-	-	UIA_PHYVREF	0x00000014	UIA_DISABLEUNUSEDADDRLNS	0x00000001
-	-	UIA_SEQUENCECTRL_0	0x0000131F	UIA_PHYINITSEQUENCENUM	0x00000000
-	-	-	-	UIA_ENABLEDFICSPOLARITYFIX	0x00000000

Table 21. Mode registers

MRs	
MR0_0	0x00000000
MR1_0	0x000000C4
MR2_0	0x00000024
MR3_0	0x00000031
MR4_0	0x00000000
MR5_0	0x00000000
MR6_0	0x00000000
MR11_0	0x00000066
MR12_0	0x00000047
MR13_0	0x00000008
MR14_0	0x00000047
MR22_0	0x00000005

The LPDDR4 PCB connections are as follows:

- DQ0 to DQ31 are connected one to one from SoC to SDRAM
- CAA0 to CAA6 and CAB0 to CAB6 are connected from SoC to SDRAM as shown in Table 7.

Table 22. Swizzle parameters for LPDDR4 validation board

Swizzle: UIS_			
UIS_SWIZZLE_0	0x00000003	UIS_SWIZZLE_24	0x00000007
UIS_SWIZZLE_1	0x00000002	UIS_SWIZZLE_25	0x00000004
UIS_SWIZZLE_2	0x00000000	UIS_SWIZZLE_26	0x00000005
UIS_SWIZZLE_3	0x00000001	UIS_SWIZZLE_27	0x00000006
UIS_SWIZZLE_4	0x00000006	UIS_SWIZZLE_28	0x00000002
UIS_SWIZZLE_5	0x00000007	UIS_SWIZZLE_29	0x00000003
UIS_SWIZZLE_6	0x00000005	UIS_SWIZZLE_30	0x00000001
UIS_SWIZZLE_7	0x00000004	UIS_SWIZZLE_31	0x00000000
UIS_SWIZZLE_8	0x00000005	UIS_SWIZZLE_32	0x00000000
UIS_SWIZZLE_9	0x00000004	UIS_SWIZZLE_33	0x00000001
UIS_SWIZZLE_10	0x00000007	UIS_SWIZZLE_34	0x00000002
UIS_SWIZZLE_11	0x00000006	UIS_SWIZZLE_35	0x00000003
UIS_SWIZZLE_12	0x00000000	UIS_SWIZZLE_36	0x00000004
UIS_SWIZZLE_13	0x00000003	UIS_SWIZZLE_37	0x00000005
UIS_SWIZZLE_14	0x00000002	UIS_SWIZZLE_38	0x00000000
UIS_SWIZZLE_15	0x00000001	UIS_SWIZZLE_39	0x00000001
UIS_SWIZZLE_16	0x00000005	UIS_SWIZZLE_40	0x00000002
UIS_SWIZZLE_17	0x00000007	UIS_SWIZZLE_41	0x00000003
UIS_SWIZZLE_18	0x00000006	UIS_SWIZZLE_42	0x00000004
UIS_SWIZZLE_19	0x00000004	UIS_SWIZZLE_43	0x00000005
UIS_SWIZZLE_20	0x00000000	-	-
UIS_SWIZZLE_21	0x00000001	-	-
UIS_SWIZZLE_22	0x00000003	-	-
UIS_SWIZZLE_23	0x00000002	-	-

Note: The PCB connections are used as an example to illustrate the swizzle configuration. The user is expected to enter the correct PCB connections into the STM32CubeMX DDR tool.

9 Testing DDR with STM32CubeMX tool suite

STM32CubeMX DDRFW-UTIL supports an extensive DDR test suite. Selective tests can be launched to verify DDR configuration robustness or to catch potential elusive errors. Tests are described in Section 9.1.

Test fails can have different causes and possible corrective actions are suggested in Section 9.2. Tests are downloaded to SRAM. Test download and execution are controlled from the STM32CubeMX. Extensive on line information on test purpose, parameters, eventual restrictions, and possible root cause of failure are available in the STM32CubeMX tool.

9.1 Tests description

Basic tests

Basic tests are simple and run fast. They are intended to capture major configuration or hardware issues showing off immediately.

Intensive tests

Intensive tests use extensive coverage of data and address patterns for noise and SSO characteristics, high throughput traffic, interleaved Read and Write. Depending on parameters, the runtime test can be long. Intensive test can be deployed progressively with test trail before launching long and exhaustive test sequences.

Stress tests

Stress tests are intensive tests executed with stretched conditions. They are done either with a small frequency increase (10-20 MHz), or with a skew of parameters (for example a fine step delay increase). It can be also with a specific frequency selective patterns. They are intended to catch low margin issues of a configuration that might cause elusive errors and eventual crashes later during runtime. A stress test campaign should always be done during system bring-up. Stress tests may also be run in case of suspicious failure. The test and its skewed parameters should be directed to pinpoint the observed failure (for example when errors are related to specific bit or byte). A summary of the tests is shown in Table 23.

Table 23. **DDR tests list**

Number	Test name	Arguments	Basic	Intensive	Stress	Description
0	All tests	-	y	-	-	Execute all DDR tests
1	Databus simple	[addr]	y	-	-	Verifies each data line
2	Databus walking 0	[loop] [addr]	y	-	-	Walking zeroes
3	Databus walking 1	[loop] [addr]	y	-	-	Walking ones
4	Address bus	[size] [addr]	y	-	-	Check addressing
5	Mem device	[size] [addr]	-	y	-	Physical memory integrity test
6	Simultaneous switching output	[size] [addr]	-	y	-	Address bus stressing
7	Noise	[pattern] [addr]	-	y	-	Read/write test with SSO
8	Noise burst	[size] [pattern] [addr]	-	y	-	Burst transfer & SSO
9	Random	[size] [loop] [addr]	-	y	-	Read/write test using pseudo rand
10	Frequency selective	[size] [loop] [addr]	-	y	y	Read/write using frequency selecting submultiple of CLK frequency
11	Block sequential variant	[size] [loop] [addr]	-	y	-	Memtester adapted with incremental
12	Checkerboard variant	[size] [loop] [addr]	-	y	-	Memtester adapted with checkerboard
13	Bit spread variant	[size] [loop] [addr]	-	y	-	Memtester adapted with bit pread
14	Bit split variant	[size] [loop] [addr]	-	y	-	Memtester adapted with bit flip pattern
15	Walking zeroes variant	[size] [loop] [addr]	-	y	-	Memtester adapted with walking
16	Walking ones variant	[size] [loop] [addr]	-	y	-	Memtester adapted with walking

9.2 Failure classification

Catastrophic failures

The failures that happen with a basic test generally result from a major configuration or hardware issue. The cause can be more or less obvious from the test report.

Sporadic failures

Basic tests do not detect these failures and are rather infrequent. It is essential to identify failure commonalities and patterns to pinpoint their root cause. Inspecting the test report enables checking which bit or byte lane that they affect, or if they concern Read or Write.

Revision history

Table 24. **Document revision history**

Date	Version	Changes
21-Mar-2024	1	Initial release.
28-Jun-2024	2	Updated: • Section 3.1: System parameters • Section 6.1: General considerations about DDR3L • Section 6.2: DDR3L configuration example • Section 7.1: General considerations about DDR4 • Section 7.2: DDR4 configuration example • Section 8.1: General considerations about LPDDR4 • Section 8.2: LPDDR4 configuration example
21-Aug-2024	3	Updated: • Section 7.1: General considerations about DDR4 • Section 8.1: General considerations about LPDDR4 • Section 5.4: Swizzle control parameters • Section 8.2: LPDDR4 configuration example

Contents

1 General information 2
2 Overview 3
3 Description of configuration parameters 8
3.1 System parameters 8
3.2 Mission mode parameters 8
3.2.1 DDR refresh controller 10
3.2.2 PHY training 11
4 Low-power features 13
5 Configuration parameters 14
5.1 DDRCTRL parameters 14
5.2 PHY parameters 15
5.3 SDRAM mode parameters 17
5.4 Swizzle control parameters 18
6 DDR3L configuration 22
6.1 General considerations about DDR3L 22
6.2 DDR3L configuration example 23
7 DDR4 configuration 27
7.1 General considerations about DDR4 27
7.2 DDR4 configuration example 29
8 LPDDR4 configuration 33
8.1 General considerations about LPDDR4 33
8.2 LPDDR4 configuration example 35
9 Testing DDR with STM32CubeMX tool suite 39
9.1 Tests description 39
9.2 Failure classification 40
Revision history 41
List of tables 43
List of figures 44

List of tables

Table 1. Acronyms . . . 2
Table 2. Bus master port and QoS settings table . . . 9
Table 3. DDRCTRL parameters . . . 14
Table 4. Basic PHY parameters . . . 15
Table 5. Advanced PHY parameters . . . 16
Table 6. Swizzle parameters . . . 18
Table 7. AC pin mapping by device versus protocol (internal signal swap) . . . 20
Table 8. DQ pin mapping by device (internal signal swap) . . . 21
Table 9. DDRCTRL parameters . . . 23
Table 10. PHY basic and advanced parameters . . . 24
Table 11. Mode registers . . . 25
Table 12. DDR3 validation board PCB connections . . . 25
Table 13. Swizzle parameters for the DDR3 validation board PCB . . . 26
Table 14. DDRCTRL parameters . . . 29
Table 15. PHY basic and advanced parameters . . . 30
Table 16. Mode registers . . . 31
Table 17. DDR4 validation board PCB connections . . . 31
Table 18. Swizzle parameters for DDR4 validation board . . . 32
Table 19. DDRCTRL registers . . . 35
Table 20. PHY basic and advanced parameters . . . 36
Table 21. Mode registers . . . 37
Table 22. Swizzle parameters for LPDDR4 validation board . . . 38
Table 23. DDR tests list . . . 40
Table 24. Document revision history . . . 41

List of figures

Figure 1. DDR subsystem . . . 3
Figure 2. DDRPHYC overview . . . 4
Figure 3. PHY initialization and training sequence . . . 5
Figure 4. Configuration and initialization flow diagram . . . 6
Figure 5. DDR configuration software architecture . . . 6
Figure 6. Simplified diagram of the training hardware . . . 7