Introduction to STM32H7Rx/7Sx system architecture and performance

Introduction

The STM32H7Rx/7Sx devices are based on the high-performance Arm® Cortex®-M7 32-bit RISC core operating at up to 600 MHz, reaching new performance records of 1284 DMIPS and 3183 CoreMark®.

STM32H7Rx/7Sx is part of STM32H7 series, which is the first series of STMicroelectronics microcontrollers in 40 nm-process technology. These devices incorporate high-speed embedded memories, including 64 Kbytes of user flash memory and 128 Kbytes of system flash memory, and up to 620 Kbytes of RAM. They also feature an extensive range of enhanced I/Os and peripherals connected to APB buses, AHB buses, a 32-bit multi-AHB bus matrix, and a multi-layer AXI interconnect supporting internal and external memory access. To improve application robustness, all memories feature error code correction (one error correction, two error detections).

The Cortex-M7 core features a floating-point unit (FPU) which supports Arm double-precision (IEEE 754 compliant), single-precision data-processing instructions, and data types. The Cortex -M7 core includes 32 Kbytes of instruction cache and 32 Kbytes of data cache. STM32H7Rx/7Sx devices support a full set of DSP instructions and a memory protection unit (MPU) to enhance application security.

This application note focuses on STM32H7R3, STM32H7R7, STM32H7S3, and STM32H7S7 microcontrollers. Its objective is to present the global architecture of the devices as well as their memory interfaces and features, which provide a high degree of flexibility to achieve the best performance and additional code and data size trade-off.

The application note also provides the results of a software demonstration of the STM32H7S7 Arm Cortex-M7 architecture performance in various memory partitioning configurations with different code and data locations.

This document is delivered with the X-CUBE-PERF-H7RS Expansion Package dedicated to STM32H7Sx microcontrollers running on STM32H7S78-DK board. Some configurations that do not use ciphering could be supported by STM32H7Rx. This Expansion Package includes two projects:

- The *bandwidth_firmware* project, which measures the bandwidth of data transfer between internal RAMs or between internal and external memories with or without activating the on-the-fly encryption and decryption.
- The *FFT_firmware* project, which demonstrates the performance of CPU memory accesses in different configurations with code execution and data storage in different memory locations using L1 cache with or without activating the on-the-fly encryption and decryption.

Both projects are targeting the STM32H7S78-DK board.

Table 1. Applicable products

Type	Microcontrollers
Microcontrollers	• STM32H7R3/7S3 line • STM32H7R7/7S7 line

1 General information

This application note applies to STM32H7Rx/7Sx series microcontrollers that are Arm® Cortex®-M7 core-based devices.

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

Reference documents

[1] Reference manual *STM32H7Rx/7Sx Arm®-based 32-bit MCUs* (RM0477)

[2] Application note *Level 1 cache on STM32F7 Series and STM32H7 Series* (AN4839)

[3] Programming manual *STM32F7 and STM32H7 Series Cortex®-M7 processor* (PM0253)

External links

[4] *Arm® Cortex®-M7 Processor Technical Reference Manual* at www.arm.com

2 STM32H7Rx/7Sx system architecture overview

2.1 Cortex-M7 core

The STM32H7Rx/7Sx devices are built on a high-performance Arm Cortex-M7 32-bit RISC core operating at up to 600 MHz frequency. The Cortex-M7 core features a high-performance floating-point unit (FPU) supporting all Arm single-precision and double-precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU) that enhances the application security. The MPU embedded in STM32H7Rx/7Sx devices enables defining up to 16 MPU regions.

The Cortex-M7 features a 6/7-stage superscalar pipeline with a branch prediction and dual-issue instructions. The branch prediction feature enables the resolution of branches to anticipate the next branch and therefore decrease the number of cycles consumed by loop. The dual-instruction feature enables the core to execute two instructions simultaneously to increase instruction throughput.

2.2 Cortex-M7 system caches

The devices embed the Cortex-M7 with a level1 cache (L1-cache), which is split into two separate caches: the data cache (DCACHE) and the instruction cache (ICACHE) implementing a Harvard architecture bringing the best performance. These caches enable the Cortex-M7 to reach a performance of zero wait state even at high frequencies. The cache size for both instruction and data caches are equal to 32 Kbytes.

By default, the instruction cache and the data cache are both disabled.

The Arm CMSIS library provides two functions that enable data and instruction caches:

- SCB_EnableICache() to enable the instruction cache.
- SCB_EnableDCache() to enable and invalidate the data cache.

Additional information about enabling and invalidating the cache are given in document [4].

More details on L1-cache usage in STM32H7Rx/7Sx devices are available in document [2].

2.3 Cortex-M7 memory interfaces

The Cortex-M7 has five interfaces: AXIM, ITCM, DTCM, AHBS, and AHBP. This section describes each of them.

2.3.1 AXI bus interface

AXI stands for advanced extensible interface. The Cortex-M7 implements the AXIM AMBA® 4, a 64-bit wide interface for more instruction fetch and data load bandwidth.

Any access that is not for the TCM or the AHBP interface, is handled by the appropriate cache controller if the cache is enabled. The user must take into account that the memory regions are not all cacheable. Cacheability depends on memory type:

- Shared memory, device, or strongly-ordered memory regions are not cacheable.
- Only normal nonshared memory type is cacheable.

Additional information and general rules about memory attributes and behaviors are available in document [3].

To modify the type and the attribute of a memory region, the MPU can be used to configure it to be a cacheable region. This is done by configuring the TEX field and S, C, and B bits in the MPU_RASR register.

Table 2 summarizes the memory region attributes after Cortex-M7 reset.

Table 2. Cortex-M7 default memory attributes after reset

Address range	Region name	Type	Attributes	Execute never?
0x00000000-0x1FFF FFFF	Code	Normal	Write-through cache attribute	No
0x20000000-0x3FFF FFFF	SRAM	Normal	Write-back, Write-allocate cache attribute	No
0x40000000-0x5FFF FFFF	Peripheral	Device	Nonshareable	Yes
0x60000000-0x7FFF FFFF	RAM	Normal	Write-back, Write-allocate cache attribute	No

Address range	Region name	Type	Attributes	Execute never?
0x80000000-0x9FFF FFFF	RAM	Normal	Write-through cache attribute	No
0xA0000000-0xBFFF FFFF	External device	Device	Shareable	Yes
0xC0000000-0xDFFF FFFF	External device	Device	Nonshareable	Yes
0xE0000000-0xE000 FFFF	Private peripheral bus	Strongly ordered	-	Yes
0xE0010000-0xFFFF FFFF	Vendor system	Device	Nonshareable	Yes

In STM32H7Rx/7Sx, the 64-bit AXI master bus connects the core to the 64-bit AXI bus matrix.

2.3.2 TCM bus interface

The TCM (tightly-coupled memory) is provided to connect the Cortex-M7 to an internal RAM. The TCM interface has a Harvard architecture with ITCM (instruction TCM) and DTCM (data TCM) interfaces. The ITCM has one 64-bit memory interface while the DTCM is split into two 32-bit wide ports, D0TCM and D1TCM.

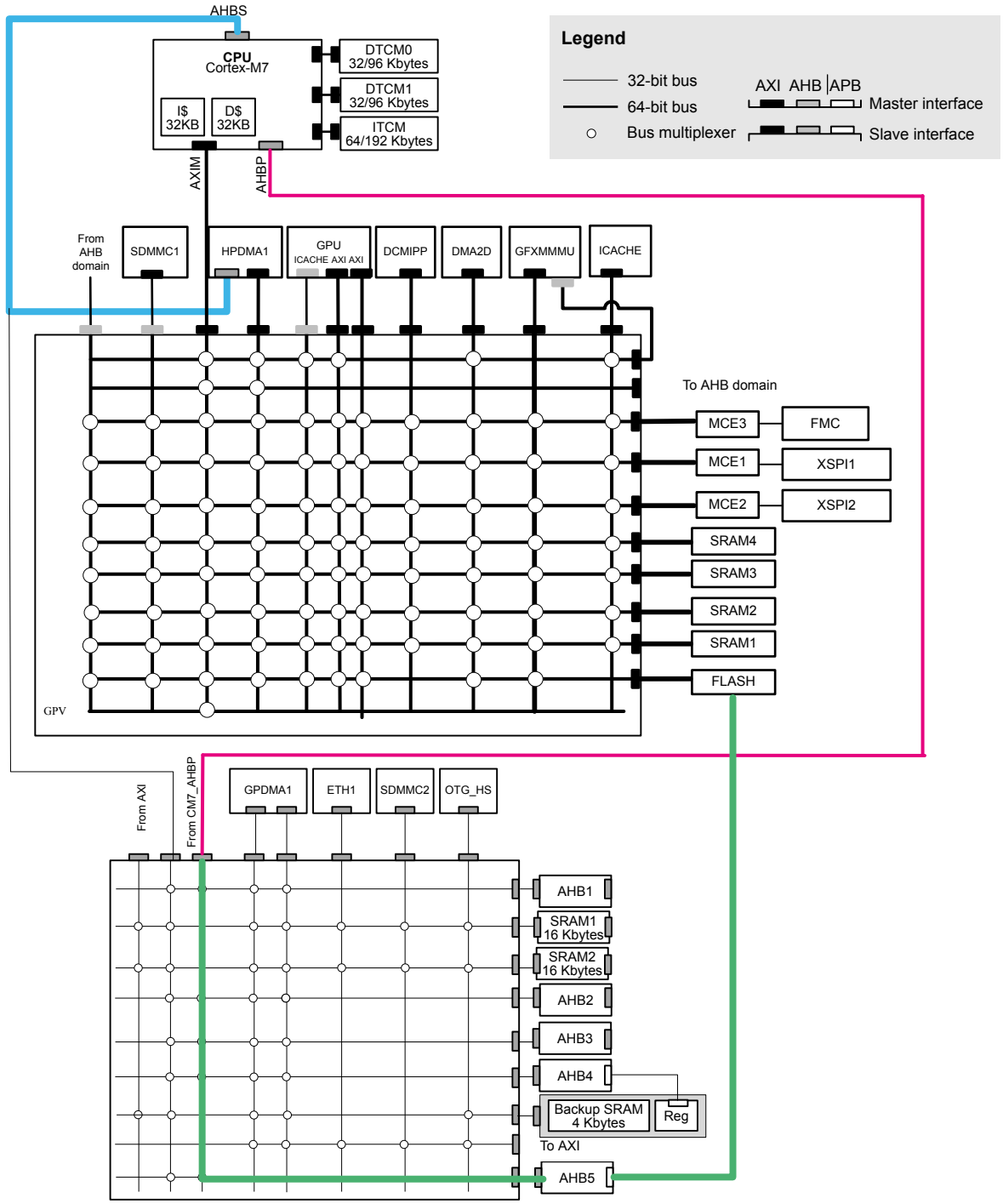
The Cortex-M7 CPU uses the 64-bit ITCM bus for fetching instructions from the ITCM and to access the data located in the ITCM-RAM. The ITCM is accessed by the Cortex-M7 at CPU clock speed with zero wait state. The DTCM interface can also fetch instructions.

In the STM32H7Rx/7Sx architecture, only the CPU and the HPDMA1 can have access to memories connected to the ITCM and DTCM interfaces.

2.3.3 AHBS bus interface

The Cortex-M7 AHBS (AHB slave) is a 32-bit wide interface that provides system access to the ITCM, D1TCM, and D0TCM. However, in the STM32H7Rx/7Sx architecture and apart Cortex-M7, AHBS allows data transfer from/to DTCM-RAM and ITCM-RAM using only HPDMA1. The AHBS interface can be used when the core is in sleep state, therefore, HPDMA1 transfers from/to TCM-RAMs can be performed in low-power modes. This connection is represented by the path colored in blue in Figure 1.

Figure 1. **STM32H7Rx/7Sx system architecture**



2.3.4 AHBP bus interface

The AHBP interface (AHB peripheral) is a single 32-bit wide interface that is dedicated to the connection of the Cortex-M7 to the peripherals. It is only used for data access. Instruction fetches are never performed on this interface.

In the STM32H7Rx/7Sx architecture, this interface connects the Cortex-M7 core to the AHB1, AHB2, AHB3, AHB4, AHB5, APB1, APB2, APB4, and APB5 peripherals via the AHB bus matrix. This connection is represented by the bus colored in pink in Figure 1.

2.4 STM32H7Rx/7Sx interconnect matrix

The STM32H7Rx/7Sx devices embed two bus matrices: one 64-bit AXI matrix and one 32-bit AHB matrix. This configuration allows efficient simultaneous operation of high-speed peripherals and removes bus congestion when several masters are simultaneously active (different masters located in separated bus matrices).

Each bus matrix ensures and arbitrates concurrent accesses from multiple masters to multiple slaves. This allows efficient simultaneous operation of high-speed peripherals. The arbitration uses a round-robin algorithm.

The maximum bus matrix frequency is half the maximum CPU frequency. Only the Cortex- M7, the ITCM-RAM, and the DTCM-RAM can run at the CPU frequency.

All bus matrices are connected together by means of inter-domain buses to enable a master located in a given matrix to access a slave located in another matrix.

Figure 1 shows the overall system architecture of the STM32H7Rx/7Sx with the connections of the interconnect matrix and Table 3 the bus-master-to-bus-slave interconnect.

Table 3. Bus-master-to-bus-slave possible interconnections in STM32H7Rx/7Sx

Bus slave/ type	Bus master/type																	
	Cortex-M7 - AXIM	Cortex-M7 - AHBP	Cortex-M7 - ITCM	Cortex-M7 - DTCM	SDMMC1	HPDMA1 - AXI	HPDMA1 - AHB	GPDMA - AHB	ICACHE_AHB	GPU2D_AXI1	GPU2D_AXI2	DCMIPP	GFXMMU	DMA2D	LTDC	ETH1 - AHB	SDMMC2 - AHB	OTG_HS - AHB
	Interconnect path and type(1)																	
ITCM	-	-	X	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-
DTCM	-	-	-	X	-	-	X	-	-	-	-	-	-	-	-	-	-	-
FLASH	X	-	-	-	X	X(2)	-	X	-	-	-	-	-	X	-	X	-	X
AXI SRAM1	X	-	-	-	X	X(2)	-	X	X	X	X	X	X	X	X	X	X	X
AXI SRAM2	X	-	-	-	X	X(2)	-	X	X	X	X	X	X	X	X	X	X	X
AXI SRAM3	X	-	-	-	X	X(2)	-	X	X	X	X	X	X	X	X	X	X	X
AXI SRAM4	X	-	-	-	X	X(2)	-	X	X	X	X	X	X	X	X	X	X	X
AHB SRAM1/2	X	X	-	-	-	X(2)	-	X	-	-	-	-	-	-	-	X	X	X
XSPI1	X	-	-	-	X	X(2)	-	X	X	X	X	X	X	X	X	X	X	X
XSPI2	X	-	-	-	X	X(2)	-	X	X	X	X	X	X	X	X	X	X	X
FMC	X	-	-	-	X	X(2)	-	X	X	X	X	X	X	X	X	X	X	X
GFXMMU	X	-	-	-	-	-	-	-	X	-	-	-	-	X	X	X	X	X
AHB1 peripherals	-	X	-	-	-	-	X	X	-	-	-	-	-	-	-	-	-	-
APB1 peripherals	-	X	-	-	-	-	X	X	-	-	-	-	-	-	-	-	-	-
AHB2 peripherals	-	X	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-
APB2 peripherals	-	X	-	-	-	-	X	X	-	-	-	-	-	X	-	-	-	-
AHB3 peripherals	-	X	-	-	-	-	-	X	-	-	-	-	-	X	-	-	-	-
AHB4 peripherals	-	X	-	-	-	-	-	X	-	-	-	-	-	X	-	-	-	-

Bus slave/ type	Bus master/type																	
	Cortex-M7 - AXIM	Cortex-M7 - AHBP	Cortex-M7 - ITCM	Cortex-M7 - DTCM	SDMMC1	HPDMA1 - AXI	HPDMA1 - AHB	GPDMA - AHB	ICACHE_AHB	GPU2D_AXI1	GPU2D_AXI2	DCMIPP	GFXMMU	DMA2D	LTDC	ETH1 - AHB	SDMMC2 - AHB	OTG_HS - AHB
	Interconnect path and type(1)																	
APB4 peripherals	-	X	-	-	-	-	X	X	-	-	-	-	-	X	-	-	-	-
AHB5 peripherals	-	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
APB5 peripherals	-	X	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-
Backup RAM	X	X	-	-	-	X(2)	-	X	-	-	-	-	-	X	-	-	X	-

1. *"X" = access possible, "-" = access not possible.*
2. *Every transfer can be done in 64 bits.*

2.4.1 AXI bus matrix

The AXI interconnect is based on the Arm CoreLink™ NIC-400 Network Interconnect. It has eleven initiator ports called ASIBs (AMBA slave interface blocks) where masters are connected, and ten target ports called AMIBs (AMBA master interface blocks) where slaves are connected. Where an ASIB or AMIB is connected to an AHB, it converts between the AHB and the AXI protocol.

The AXI bus matrix can run at up to half the maximum CPU frequency. It is a 64-bit bus matrix that connects ASIBs and AMIBs and enables a number of parallel access paths between the core AXI bus, masters buses and the slaves buses thus making possible concurrent accesses and efficient operation even when several high-speed peripherals are running. An internal arbiter resolves the conflicts and the bus concurrency of masters on the bus matrix using a round-robin algorithm with QoS capability (quality of service).

Each master has programmable read channel and write channel priorities from 0 to 15, configured respectively in AXI_INIx_READ_QOS and AXI_INIx_WRITE_QOS registers so that the higher the value, the higher the priority. If two masters attempt to access the same slave at the same time, the one having the higher priority transaction accesses to the given slave before the other master. If the two transactions have the same QoS value, then a least-recently-used (LRU) priority scheme is adopted. The QoS is configurable in the Global Programmer View (GPV) that contains registers for configuring some parameters, such as the QoS level at each ASIB.

The QoS is useful for tasks such as graphics processing to boost the priority of the display controller (LTDC) compared to the Cortex-M7 CPU. Table 4 and Table 5.present the masters and slaves connected to the AXI bus matrix respectively.

Table 4. ASIB configuration

ASIB	Connected master	Protocol	Bus width	R/W issuing
INI 1	AHB from AHB peripherals	AHB-lite	32	1/1
INI 2	SDMMC1	AHB-lite	32	1/1
INI 3	HPDMA1	AXI4	64	8/8
INI 4	Cortex-M7	AXI4	64	7/32
INI 5	GPU	AHB	64	1/1
INI 6		AXI4	64	1/2
INI 7		AXI4	64	1/0

ASIB	Connected master	Protocol	Bus width	R/W issuing
INI 8	DCMIPP	AXI4	64	0/1
INI 9	GFXMMU	AXI4	64	1/2
INI 10	LTDC	AXI4	64	2/0
INI 11	DMA2D	AXI4	64	2/1

Table 5. AMIB configuration

AMIB	Connected slave	Protocol	Bus width	R/W/total acceptance
TARG1	GFXMMU	AXI4	64	2/2/4
TARG2	AHB SRAM	AXI4(1)	32	1/1/2
TARG3	FMC	AXI4	64	1/1/2
TARG4	XSPI1	AXI4	64	1/1/2
TARG5	XSPI2	AXI4	64	1/1/2
TARG6	AXI SRAM(2)	AXI4	64	2/2/4
TARG7	AXI SRAM(3)	AXI4	64	2/2/4
TARG8	AXI SRAM	AXI4	64	2/2/4
TARG9	AXI SRAM(4)	AXI4	64	2/2/4
TARG10	FLASH	AXI4	64	2/2/4

1. *Conversion to AHB protocol is done via an AXI/AHB bridge sitting between AXI interconnect and the connected slave.*
2. *SRAM shared with the ECC (Section 2.5.2: Embedded RAM).*
3. *SRAM shared with the DTCM (Section 2.5.2: Embedded RAM).*
4. *SRAM shared with the ITCM (Section 2.5.2: Embedded RAM).*

2.4.2 AHB bus matrix

The AHB bus matrix operates at half the maximum CPU frequency. It ensures and arbitrates concurrent access from multiple masters to multiple slaves. This enables efficient simultaneous operation of high-speed peripherals and memories.

An internal arbiter resolves the conflicts and the bus concurrency of masters on the bus matrix using a round-robin algorithm.

The AHB bus matrix is dedicated to communication peripherals and timers. It interconnects the following buses:

- Eight bus masters:
  - The AXI-to-AHB inter-domain that connects the AXI domain to the AHB domain
  - The Cortex-M7AHB peripherals bus that makes the CPU to access AHB domain
  - The HPDMA1 AHB bus
  - The GPDMA memory AHB bus
  - The GPDMA peripheral AHB bus
  - The Ethernet AHB bus
  - The SDMMC2 DMA AHB bus
  - The USB OTG high-speed 1 DMA AHB bus

- Nine bus slaves:
  - The AHB1 peripherals bus including the AHB to APB bridge that makes Cortex-M7 access APB1 peripherals
  - Internal SRAM1 up to 16 Kbytes with 32-bit AHB access
  - Internal SRAM2 up to 16 Kbytes with 32-bit AHB access
  - The AHB2 peripherals bus that connects to APB2 peripherals
  - The AHB3 peripherals
  - The AHB4 peripherals bus that connects to APB4 peripherals
  - The AHB5 peripherals bus that controls the AXI masters and the external memories controllers
  - The AHB-to-AXI inter-domain that connects the AHB domain to the AXI domain

2.5 STM32H7Rx/7Sx memories

STM32H7Rx/7Sx devices incorporate:

- 64 Kbytes of user flash memory
- 128 Kbytes of system flash memory
- High-speed embedded memories
- 64 Kbytes exclusively ITCM
- 64 Kbytes exclusively DTCM
- Up to 492 Kbytes of RAM including:
  - 128 Kbytes that can be shared between ITCM and AXI-SRAM1
  - 128 Kbyte DTCM that can be shared between DTCM and AXI-SRAM3
  - 128 Kbyte AXI-SRAM2 non shareable
  - 32 Kbytes AHB
  - 4 Kbytes of backup RAM
  - Up to 72 Kbytes in AXI-SRAM4 when ECC is disabled
- An extensive range of enhanced I/Os and peripherals connected to APB buses, AHB buses
- A 32-bit multi-AHB bus matrix
- Multi-layer AXI interconnect supporting internal and external memory access

To improve application robustness, all memories feature error code correction (one error correction, two error detections).

2.5.1 Embedded flash memory

The STM32H7Rx/7Sx is considered as the first device bootflash. Its flash memory size is limited to 64 Kbytes of user flash memory and 128 Kbytes of system flash memory. Large application can be implemented in external memories.

The embedded flash memory is accessible for read or/and write accesses through the AXI bus. The read operations support multiple lengths (64 bits, 32 bits, 16 bits, or one byte). The flash memory programming by 128 bits (user area) and 16 bits (OTP area). The Error Code Correction (ECC) is one error detection/correction or two error detections per 128 bit flash word using 9 ECC bits.

For control, configuration and status register accesses, the flash memory interface is accessible through the 32-bit AHB5 bus. Refer to the green path in Figure 1.

To correctly read data from the flash memory, the number of wait states (LATENCY) must be correctly programmed in the flash memory access control register (FLASH_ACR) according to the embedded flash memory AXI interface clock frequency (sys_ck), and the internal voltage range of the device (V_{core}).

Table 6 shows the correspondence between the number of wait states (LATENCY), the programming delay parameter (WRHIGHFREQ), the embedded flash memory clock frequency and its supply voltage ranges.

Table 6. FLASH recommended read wait states and programming delays

Wait states per flash word read (LATENCY)	Read latency (cycles per flash word)	Programming delay (WRHIGH FREQ)	AXI interface clock frequency vs V_{CORE} range	
			VOS low range 1.15 V - 1.26 V	VOS high range 1.20 V - 1.40 V
0x0	1	00	[0 MHz ; 36 MHz]	[0 MHz ; 40 MHz]
0x1	2		[36 MHz ; 72 MHz]	[40 MHz ; 80 MHz]
0x2	3	01	[72 MHz ; 108 MHz]	[80 MHz ; 120 MHz]
0x3	4	01	[108 MHz ; 144 MHz]	[120 MHz ; 160 MHz]
0x4	5	10	[144 MHz ; 180 MHz]	[160 MHz ; 200 MHz]
0x5	6	10	[180 MHz ; 216 MHz]	[220 MHz ; 240 MHz]
0x6	7	11	N/A	[240 MHz ; 280 MHz]
0x7	8	11	N/A	[280 MHz ; 320 MHz]

2.5.2 Embedded RAM

The STM32H7Rx/7Sx devices include up to 456 Kbytes of AXI-SRAM mapped onto the AXI bus divided and shared as follow.

- Up to 192 Kbytes of instruction TCM RAM as described in Table 7. This feature can be configured through the ITCM_AXI_SHARE[1:0] option byte in the FLASH_OBW2SRP register as described in the FLASH option byte word 2 status register (FLASH_OBW2SR).
- Up to 192 Kbytes of data TCM RAM as described in Table 8. This feature can be configured through the DTCM_AXI_SHARE[1:0] option byte in described in the FLASH option byte word 2 status register (FLASH_OBW2SR).
- Up to 72 Kbytes when ECC is disabled as described in Table 9. This feature can be configured through the ECC_ON_SRAM option byte in the FLASH_OBW2SRP register as described in the FLASH option byte word 2 status register (FLASH_OBW2SR). 128 Kbytes not shared and without ECC.
- 16 Kbytes AHB_SRAM1.
- 16 Kbytes AHB_SRAM2.

The system AHB SRAM can be accessed as bytes, half-words (16-bit units) or words (32-bit units), while the system AXI SRAM can be accessed as bytes, half-words, words, or double-words (64-bit units). These memories can be addressed at maximum system clock frequency without wait state.

Table 7. SRAM1/ITCM configurations

Allocation	SRAM1/ITCM split (Kbytes)		
SRAM1 allocation	0	64	128
ITCM allocation	192	128	64

Table 8. SRAM3/DTCM configurations

Allocation	SRAM3/DTCM split (Kbytes)		
SRAM3 allocation	0	64	128
DTCM0 allocation	96	64	32
DTCM1 allocation	96	64	32

Table 9. SRAM4/ECC configuration

ECC	SRAM4 allocation (Kbytes)
ON	0
OFF	72

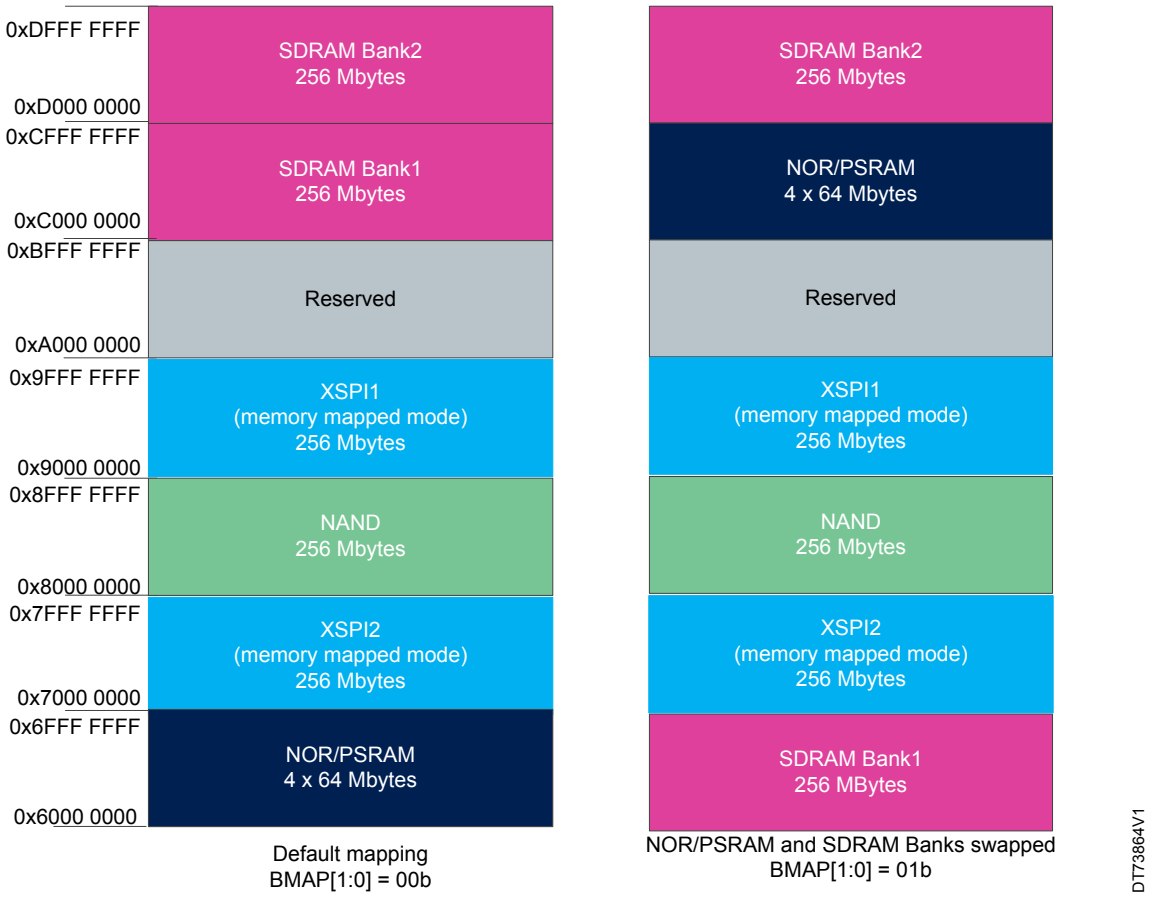
The backup RAM is mapped at the address 0x3880 0000 and is accessible by most of the system masters. With a battery connected to the VBAT pin, the backup SRAM can be used to retain data during low-power mode (Standby and VBAT mode).

2.5.3 External memories

In addition to the internal memories and storage controllers such as the USB and the SDMMC, the user can extend the STM32H7Rx/7Sx memories with the flexible memory controller (FMC) and the two XSPI memory interfaces.

The external memory space is divided into fixed-size banks of 256 Mbytes each as shown in Figure 2.

Figure 2. STM32H7Rx/7Sx external memory mapping



2.5.3.1 *Flexible memory controller*

Four external memory banks are dedicated to the FMC and two to the XSPI controller:

- Bank 1 is used to address up to four NOR flash memory or PSRAM devices. This bank is split into four NOR/PSRAM subbanks with four dedicated chip selects.
- Bank 3 is used to address NAND flash memory devices. The MPU memory attribute for this space must be reconfigured by software to device.
- Bank 5 and 6 are used to address SDRAM devices (one device per bank).

For each bank the type of memory to be used can be configured by the user application through the configuration register.

The FMC bank mapping can be modified through the BMAP[1:0] bits of the FMC_BCR1 register. The BMAP bank memory mapping bits support two configurable options:

- Default mapping
- Swapping of the NOR/PSRAM bank with SDRAM banks

Figure 2 shows the two configurations of bank memory mapping.

2.5.3.2 *Extended-SPI interface (XSPI)*

The XSPI is a specialized communication interface targeting single, dual, quad, octal, or 16-bit SPI memories. The STM32H7Rx/7Sx devices embed up to two separate XSPI interfaces. Each XSPI interface supports single/dual/quad/octal and 16-bit SPI formats. The XSPI memory interfaces supports

- One Quad-SPI memory in STM32H7R/S3R8 devices.
- One Octo-SPI memory in STM32H7R/S3I8, STM32H7R/S3Z8, STM32H7R/S3A8, and STM32H7R/S3V8 devices.
- Two Octo-SPI memories in STM32H7R/S3L8, STM32H7R/S7Z8, STM32H7R/S7A8, STM32H7R/S7I8, and STM32H7R/S7L8 devices.
- One Octo-SPI and one Hexa-SPI in STM32H7R/S3L8H6H and STM32H7R/S7L8H6H devices.

The XSPI supports most external serial memories such as serial PSRAMs, serial NAND and serial NOR flash memories, HyperRAM™ and HyperFlash™ memories, with the following functional modes:

- Indirect mode: all the operations are performed using the XSPI registers to preset commands, addresses, data, and transfer parameters.
- Automatic status-polling mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting. This feature is only available in regular-command protocol.
- Memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting both read and write operations.

The XSPI supports the following protocols with associated frame formats:

- The regular-command frame format with the command, address, alternate byte, dummy cycles, and data phase.
- The HyperBus™ frame format.

The XSPI can extend the internal flash memory by 256 Mbytes for each XSPI interface, and can be used to store data such as images in graphical applications or to store the user application.

2.5.3.3 *Memory cipher engine*

On STM32H7Sx, the data stored in nonvolatile or volatile external memories can be encrypted on-the-fly while writing and decrypted on-the-fly while reading by the memory cipher engine (MCE).

The STM32H7Sx devices include three MCE instances: MCE1 that uses the AES encryption and decryption algorithm, and MCE2 and MCE3 that use the Noekeon algorithm. All the instances have two cipher engines. Table 10 summarizes the MCE features.

The possible configurations of MCE provided in STM32H7Sx can be summarized as follow:

Table 10. STM32H7Rx/7Sx MCE features

-	MCE1	MCE2	MCE3
External memory interface	XSPI1/2	XSPI1/2	FMC
Cipher engines	AES x2	12 rounds Noekeon x2	
Encryption modes	Block, fast block, stream	Block, fast block	
Number of regions	4 with 4-Kbyte granularity		
Cipher context(s)	2	0	
Derive key function	Normal, fast		
Master key	2		
Features	• Automatic key-erase in case of tamper. • AHB configuration port, privileged aware. • AXI system bus master/slave interfaces (64-bit). • One set of write-only and lockable master key registers per block cipher (normal, fast). • For MCE1, two sets of lockable cipher contexts (128-bit key, IV), usable for stream and block ciphers.		

According to the XSPI I/O manager (XSPIM) configuration which is a low-level interface that enables an efficient XSPI pin assignment and multiplex of single/dual/quad/octal/16-bit SPI interfaces over the same bus, MCE1 can be applied to the XSPIM Port 1 or XSPIM Port 2 as well as for MCE2.

Each MCE instance defines four regions of encryption and two contexts for MCE1. When a read or write occurs from or in an encrypted region, extra cycles are added to the transaction according to the encryption/decryption algorithm and mode. Table 11 presents the MCE latencies needed.

AES 128-bit block

AES 128-bit block cipher is compatible with the ECB mode specified in the NIST FIPS publication 197 advanced encryption standard (AES) with a key derivation function based on the Keccak-f400 algorithm. AES 128-bit block is the most secure ciphering proposed solution since it has full side channel protection with no bit-flip attacks. However, it is the most time-consuming solution since it needs 36 more cycles for reading 256 bits and 46 more cycles to write 256 bits into external memory. The key derivation needs 14 cycles on its own.

AES 128-bit fast block

AES 128-bit fast block cipher has only one difference compared to AES 128-bit normal block regarding the key derivation: it is faster since it only needs four cycles, but is more sensitive to side-channel attacks. It reduces the MCE latency by 10 cycles per 256 bits read or written sequentially.

Noekeon 128-bit block

Noekeon 128-bit cipher is another iterated block cyphering solution based on 128-bit key length with key derivation function based on Keccak-f400 algorithm ensuring the side channel protection. Noekeon enhancement is the use of simple binary operations as a substitute to AES algorithm which can reduce the generated delay (7 cycles Vs 11 cycles for AES for 128-bit encryption).

Noekeon 128-bit fast block

Noekeon 128-bit fast block uses the same key derivation solution as AES 128-bit fast block. It is faster (4 cycles in fast mode Vs 14 cycles in normal mode), but is sensitive to side channel attacks.

AES 128-bit stream

AES 128-bit cipher in stream mode is compatible with CTR. It consists of XOR-ing plaintext with a sequence of pseudorandom variables. AES 128-bit stream has the lowest latency, but with limitations in terms of security. It has no side channel protection, and it is subject to bit-flip attacks.

Table 11. MCE cipher latencies

Cipher type	Enciphering mode	ENC bit in MCE_REGCRx	Latency for 16 bytes data (in AXI cycles)	Latency for 32 bytes data (in AXI cycles)	Optimization for sequential accesses
AES	Block	10	14 + 11 = 25	14 + 11 × 2 = 36	No
	Fast block	11	4 + 11 = 15	4 + 11 × 2 = 26	
	Stream	01	11	11 × 2 = 22	Yes
Noekoen	Block	10	14 + 7 = 21	14 + 7 × 2 = 28	No
	Fast block	11	4 + 7 = 11	4 + 7 × 2 = 18	

MCE includes a special AXI-64 read-write arbitration scheme, designed to speed-up multiple reads. More specifically, when no write is ongoing in MCE, the two cipher cores can be allocated to decrypt two reads in parallel. In this case an incoming write request is stalled until the cipher core allocated to writes is free. Best performances are obtained when transactions are aligned on two 64-bit words for regions where the block cipher is selected.

However, when MCE is used in conjunction with XSPI that supports an external flash memory, it is required to use the DMA to perform writes to the flash memory using 16-bytes bursts.

2.6 Main architecture differences between STM32H7Rx/7Sx, STM32H730, and STM32H750 devices

The STM32H7Rx/7Sx architecture features a 64-bit AXI and 32-bit multilayer AHB bus matrix, and bus bridges that allow interconnecting bus masters with bus slaves, as illustrated in Figure 1. In STM32H730 and STM32H750 devices, there are three domains: an AXI bus matrix in D1 domain, and two AHB bus matrices in D2 and D3 domains.

Table 12 summarizes the differences between the STM32H7Rx/7Sx devices and, STM32H730 and STM32H750 devices in terms of architecture and performances (peripheral differences not included).

Table 12. Peripheral summary for STM32H7Rx/7Sx, STM32H730, and STM32H750 devices

Peripherals		STM32H750	STM32H730	STM32H7Sxx	STM32H7Rxx
Maximum CPU frequency		480 MHz	550 MHz	600 MHz	
MPU region number		16	16		
Data cache (Kbytes)		16	32	32	
Instruction cache (Kbytes)		16	32	32	
User flash memory (Kbytes)		128 Kbytes	128 Kbytes	64 Kbytes	
SRAMs (Kbytes)		864	Up to 368(1)	Up to 488(2)	
TCM RAM (Kbytes)	ITCM	64	Up to 256(1)	Up to 192(2)	
	DTCM	128	128	Up to 192(2)	
Backup SRAM (Kbytes)		4	4	4	
External memories	FMC	1	1		
	SDMMC	2	2		
	XSPI	One QUADSPI interface	Two OCTOSPI interfaces	Two XSPI interfaces	

1. *Includes 192 Kbytes shared between ITCM and AXI.*
2. *Includes 128 Kbytes shared between ITCM and AXISRAM1 and 128 Kbytes shared between DTCM and AXISRAM3 and 72 Kbytes shared between ECC and AXISRAM4.*

3 Typical application

This application note provides two software examples that demonstrate the performance of the STM32H7Rx/7Sx devices. The first example is based on the FFT example provided in the CMSIS library. The second example, the bandwidth_firmware is proposed to measure the bandwidth of data transfer between internal RAMs or between internal and external memories with or without activating the on-the-fly encryption and decryption. Both projects run on the STM32H7S78-DK board and can be used as a skeleton where the user can integrate their application.

3.1 FFT demonstration

The FFT example is used as it benefits from the floating point unit of the Cortex-M7, containing several loops and data load/store operations, and can be accessed in different paths/memories. The code can be executed from internal or external memories.

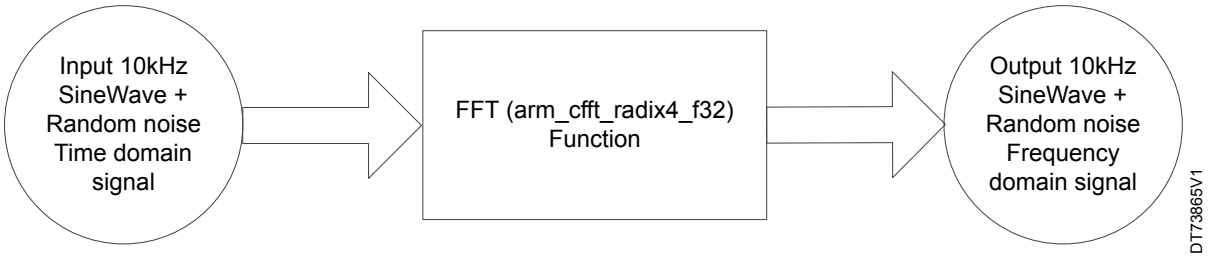
3.1.1 FFT presentation

The example consists in the calculation of the maximum energy in the frequency domain of an input signal with the use of complex FFT, complex magnitude, and maximum functions. It uses the FFT 1024 points and the calculation is based on a single precision floating point. The input signal is a 10 kHz sine wave merged with white noise. Figure 3 shows the block diagram of the transformation.

The number of cycles consumed by the FFT process is also calculated based on the system-tick timer. The example is run on STM32H7S78-DK and the results (provided in number of cycles) are shown on the I/O terminal of IAR IDE.

The FFT demonstration displays the current project configuration such as the system frequency, the configuration of caches (ON/OFF) and the external memory configuration of PSRAM or OCTO flash memory NOR.

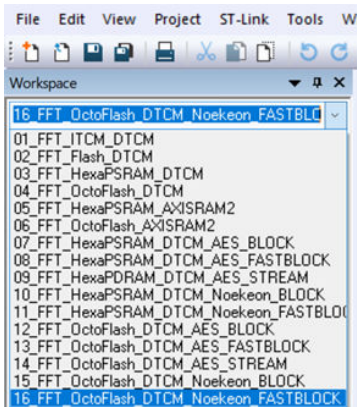
Figure 3. FFT example block diagram



3.1.2 Configuring demonstration projects

The CPU memory access demonstration is provided with the IAR™ (EWARM) toolchains. The project is presented in sixteen subprojects workspaces, each one representing a configuration. Each configuration enables selecting the data and code locations. Figure 4 shows a screenshot of different configurations of EWARM toolchains.

Figure 4. IAR™ (EWARM) configuration selection



DT73866V1

The configurations are named using the following rules:

```
N _FFT_ InstructionLocation _ DataLocation_ EncryptionAlgorithm
```

Where:

- N: The configuration number.
- InstructionLocation: The memory location of the user code with its respective domain location. The user must differentiate between the execution region and the load region. The execution region is the memory location where the application is executed. The load region is the memory location where the application is initially loaded by the flash memory loader and copied afterward (at the linker load phase), in the execution region if the address locations of execution region and load region are different.
- DataLocation: The memory location of RW/Zero initialized data, stack and heap and the domain location of the memory.
- EncryptionAlgorithm: Involves the algorithm used for encryption and decryption, if enabled, as well as the MCE mode.

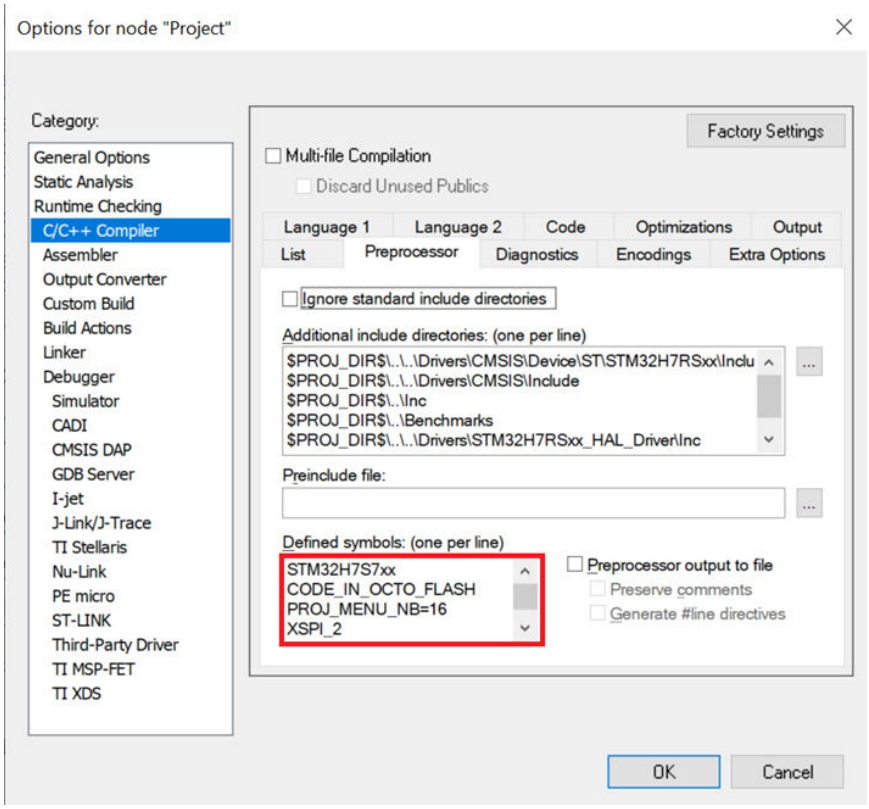
Since FFT coefficients are declared as constant and are stored in the internal flash memory, the data cache is enabled in all the proposed configurations enumerated below:

1. FFT_ITCM_DTCM: The program is executed from the ITCM. The instruction cache is disabled and the data is loaded/stored in the DTCM.
2. FFT_Flash_DTCM: The program is executed from the flash memory at seven wait states with the instruction cache enabled and the data is loaded/stored in the DTCM.
3. FFT_HexaPSRAM_DTCM: The program is executed from the external PSRAM memory with the instruction cache enabled and the data is loaded/stored in the DTCM.
4. FFT_OctoFlash_DTCM: The program is executed from the external flash memory with the instruction cache enabled and the data is loaded/stored in the DTCM.
5. FFT_HexaPSRAM_AXISRAM2: The program is executed from the external PSRAM memory with the instruction cache enabled and the data is loaded/stored in the SRAM2 with the DCACHE enabled.
6. FFT_OctoFlash_AXISRAM2: The program is executed from the external flash memory with the instruction cache enabled and the data is loaded/stored in the SRAM2 with the data cache enabled.
7. FFT_HexaPSRAM_DTCM_AES_BLOCK: The program is executed from the external PSRAM memory with the instruction cache enabled and the data is loaded/stored in the DTCM. MCE AES clock is enabled.
8. FFT_HexaPSRAM_DTCM_AES_FASTBLOCK: The program is executed from the external PSRAM memory with the instruction cache enabled and the data is loaded/stored in the DTCM. MCE AES fast block is enabled.
9. FFT_HexaPSRAM_DTCM_AES_STREAM: The program is executed from the external PSRAM memory with the instruction cache enabled and the data is loaded/stored in the DTCM. MCE AES stream is enabled.
10. FFT_HexaPSRAM_DTCM_Noekeon_BLOCK: The program is executed from the external PSRAM memory with the instruction cache enabled and the data is loaded/stored in the DTCM. MCE Noekeon block is enabled.

11. FFT_HexaPSRAM_DTCM_Noekeon_FASTBLOCK: The program is executed from the external PSRAM memory with the instruction cache enabled and the data is loaded/stored in the DTCM. MCE Noekeon fast block is enabled.
12. FFT_OctoFlash_DTCM_AES_BLOCK: The program is executed from the external flash memory with the instruction cache enabled and the data is loaded/stored in the DTCM. MCE AES block is enabled.
13. FFT_OctoFlash_DTCM_AES_FASTBLOCK: The program is executed from the external flash memory with the instruction cache enabled and the data is loaded/stored in the DTCM. MCE AES fast block is enabled.
14. FFT_OctoFlash_DTCM_AES_STREAM: The program is executed from the external flash memory with the instruction cache enabled and the data is loaded/stored in the DTCM. MCE AES stream is enabled.
15. FFT_OctoFlash_DTCM_Noekeon_BLOCK: The program is executed from the external flash memory with the instruction cache enabled and the data is loaded/stored in the DTCM. MCE Noekeon block is enabled.
16. FFT_OctoFlash_DTCM_Noekeon_FASTBLOCK: The program is executed from the external flash memory with the instruction cache enabled and the data is loaded/stored in the DTCM. MCE Noekeon fast block is enabled.

Each configuration has its own flag set. These flags are settable on the options of the project. Figure 5 shows where these flags are defined for the EWARM toolchains.

Figure 5. EWARM flags configuration



DT73867V1

Figure 5 shows an example of configuration where the code is executed from OCTO flash memory using XSPI2 instance and with MCE2 (Noekeon) enabled with fast block configuration. This corresponds to configuration 16 in X-CUBE-PERF-H7RS.

The code is highly optimized targeting the highest speed and with no size constraints for all the configurations. The CPU is operating at 600 MHz for all configurations and external memories are operating at 190 MHz. Project flags are the following:

- CODE_IN_OCTO_FLASH: Defines if the FFT code is in the external flash memory.
- CODE_IN_HEXA_PSRAM: Defines if the FFT code is in the external PSRAM memory.

Note: Those two flags are useful for defining which XSPIM port is used as well as for XSPI controller initialization.

- XSPI_1: Defines if the first instance of XSPI controller is used.

- XSPI_2: Defines if the second instance of XSPI controller is used.

Note: Those two flags are useful for defining which XSPI controller is used since the IO Manager is swapped in some configuration to enable MCE on every external memory.

- AES: Indicates that MCE1 is enabled.
- NOEKEON: Indicates that MCE2 is enabled.
- ENCRYPT: Indicates that encryption is enabled.
- BLOCK: Indicates that the enabled MCE mode is the block encryption.
- FAST: Indicates that the enabled MCE mode is the fast block encryption.
- STREAM: Indicates that the enabled MCE mode is the stream encryption.
- CACHE_I_ON: If defined in the configuration project, the instruction cache is enabled.
- CACHE_D_ON: If defined in the configuration project, the data cache is enabled.

The user can create new configurations of code execution/data storage location based on these templates. This is done by merging the adequate settings, by modifying the linker files, and by setting the adequate flash loader.

Note: To modify the RAM regions in the linker files (stack and heap regions), the user must modify accordingly the stack and heap sizes in option menu in Project tab of the EWARM toolchains. The size of the region in the scatter file is not considered as the real stack size of the main application.

The scatter files of different configurations are located under EWARM project directory. The results can be displayed on the IO terminal.

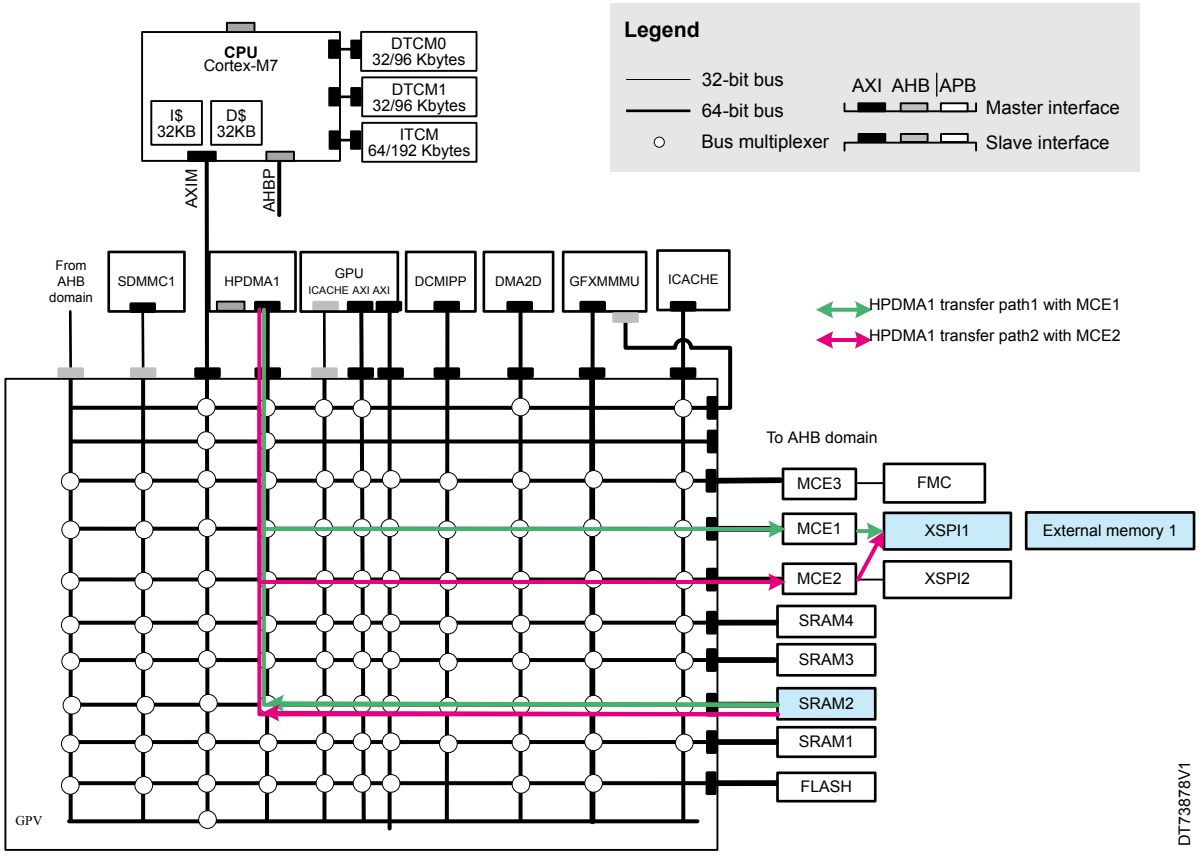
3.2 Bandwidth measurement demonstration

As it is the first device of STM32H7 family that is bootflash, it is very important to measure the bandwidth of data transfer from and to the external memories.

3.2.1 Bandwidth measurement presentation

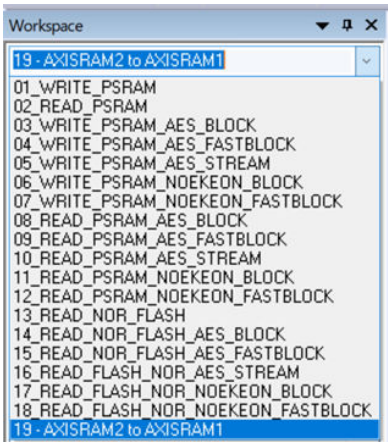
The example consists in the measurement of the bandwidth between a destination and a source where one of them is an external memory. This project consists of using the HPDMA1 for a transfer of 16 KBytes between different location in the device. The time needed for the transfer is measured leading to a given bandwidth expressed in MBytes/s. Figure 6 presents an example of two independant transfers from AXISRAM2 to the external PSRAM while enabling MCE1 or MCE2. Each independant transfer is drawn by a different color.

Figure 6. **Example of transfer paths between AXISRAM2 and external PSRAM using MCE1 or MCE2**



The bandwidth measurement demonstration is provided with the IAR® (EWARM) toolchains. The project is presented in nineteen subprojects workspaces, each one representing a configuration. Each configuration enables selecting the direction of the HPDMA1 transfer. Figure 7 shows a screenshot of different configurations of EWARM toolchains.

Figure 7. **IAR™ (EWARM) configuration selection for bandwidth measurement project**



The configurations are named using the targeted bandwidth measurement. All the configurations consider the external memories except for the last configuration which measure the bandwidth when a data transfer occurs between AXISRAMs.

The configurations are named using the following rules: AccessDirection_memory_EncryptionAlgorithm

This example provides 19 workspaces:

1. WRITE_PSRAM: Bandwidth measurement when 16 Kbytes is written to the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory.
2. READ_PSRAM: Bandwidth measurement when 16 Kbytes is read from the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory.
3. WRITE_PSRAM_AES_BLOCK: Bandwidth measurement when 16 Kbytes is written to the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory; MCE1 is enabled with block configuration.
4. WRITE_PSRAM_AES_FASTBLOCK: Bandwidth measurement when 16 Kbytes is written to the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory; MCE1 is enabled with fast block configuration.
5. WRITE_PSRAM_AES_STREAM: Bandwidth measurement when 16 Kbytes is written to the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory; MCE1 is enabled with stream configuration.
6. WRITE_PSRAM_NOEKEON_BLOCK: Bandwidth measurement when 16 Kbytes is written to the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory, MCE2 is enabled with block configuration.
7. WRITE_PSRAM_NOEKEON_FASTBLOCK: Bandwidth measurement when 16 Kbytes is written to the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory, MCE2 is enabled with fast block configuration.
8. READ_PSRAM_AES_BLOCK: Bandwidth measurement when 16 Kbytes is read from the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory, MCE1 is enabled with block configuration.
9. READ_PSRAM_AES_FASTBLOCK: Bandwidth measurement when 16 Kbytes is read from the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory, MCE1 is enabled with fast block configuration.
10. READ_PSRAM_AES_STREAM: Bandwidth measurement when 16 Kbytes is read from the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory, MCE1 is enabled with stream configuration.
11. READ_PSRAM_NOEKEON_BLOCK: Bandwidth measurement when 16 Kbytes is read from the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory, MCE2 is enabled with block configuration.
12. READ_PSRAM_NOEKEON_FASTBLOCK: Bandwidth measurement when 16 Kbytes is read from the external Hexa-PSRAM via HPDMA1, code is executed from the embedded flash memory, MCE2 is enabled with fast block configuration.
13. READ_NOR_FLASH: Bandwidth measurement when 16 Kbytes is read from the OctoFlash NOR via HPDMA1, code is executed from the embedded flash memory.
14. READ_NOR_FLASH_AES_BLOCK: Bandwidth measurement when 16 Kbytes is read from the OctoFlash NOR via HPDMA1, code is executed from the embedded flash memory, MCE1 is enabled with block configuration.
15. READ_NOR_FLASH_AES_FASTBLOCK: Bandwidth measurement when 16 Kbytes is read from the OctoFlash NOR via HPDMA1, code is executed from the embedded flash memory, MCE1 is enabled with fast block configuration.
16. READ_NOR_FLASH_AES_STREAM: Bandwidth measurement when 16 Kbytes is read from the OctoFlash NOR via HPDMA1, code is executed from the embedded flash memory, MCE1 is enabled with stream configuration.
17. READ_NOR_FLASH_NOEKEON_BLOCK: Bandwidth measurement when 16 Kbytes is read from the OctoFlash NOR via HPDMA1, code is executed from the embedded flash memory, MCE2 is enabled with block configuration.
18. READ_NOR_FLASH_NOEKEON_FASTBLOCK: Bandwidth measurement when 16 Kbytes is read from the OctoFlash NOR via HPDMA1, code is executed from the embedded flash memory, MCE2 is enabled with fast block configuration.
19. AXISRAM2 to AXISRAM1: Bandwidth measurement when 16 Kbytes is read from the AXISRAM1 and written to AXISRAM2 via HPDMA1, code is executed from the embedded flash memory.

The code is highly optimized targeting the highest speed and with no size constraints for all the configurations. The CPU is operating at 600 MHz for all configurations and external memories are operating at 190 MHz. Only the Instruction cache is enabled for all the configurations. Project flags are the following:

- CODE_IN_OCTO_FLASH: Defines whether if the transfer is from or to the external flash memory.
- CODE_IN_HEXA_PSRAM: Defines whether if the transfer is from or to the external PSRAM memory.

Note: Those two flags are useful for defining which XSPIM port is used as well as for XSPI controller initialization.

- READ: Indicates that a read of 16kB occurs from an external memory.
- IO_MNG: Indicates that the IO XSPI Manager is swapped.
- AES: Indicates that MCE1 is enabled
- NOEKEON: Indicates that MCE2 is enabled
- ENCRYPT: Indicates that encryption is enabled
- BLOCK: Indicates that the enabled MCE mode is the block encryption
- FAST: Indicates that the enabled MCE mode is the fast block encryption
- STREAM: Indicates that the enabled MCE mode is the stream encryption

Two conditions of HPDMA1 configuration are proposed in the project. The first one is enabled via the macro SEQUENTIAL_TRANSFER. It configures the HPDMA1 at its best performance allowing a sequential transfer from the destination to the source. The second configuration simulates a multiple master access to AXI bus. In this case, the HPDMA1 transfer is reduce to a burst of 8 × 32 bits or 4 × 64 bits. This configuration is ensured by the macro NONSEQUENTIAL_TRANSFER. In such configuration, the transfer is not sequential. An offset of 16 bits is added to the source or the destination according to a bandwidth measurement of a read or write.

4 Benchmark results and analysis

This section explains each feature activation in each project and the analysis of the results obtained. The FFT results are provided in the form of instruction count, while the measured bandwidth is expressed in Mbyte/s.

4.1 FFT benchmark results and analysis

As explained in Section 3, depending on the chosen configuration of the firmware, the fast Fourier transform algorithm is executed from ITCM, embedded flash memory and external memories to measure the impact of changing code location on the STM32H7Rx/7Sx performances. In all these configurations, instruction, and data cache are enabled due to the constant buffer's declaration. In our case, an 8-Kbyte buffer containing the FFT coefficient is declared as constant and stored in the internal flash memory. The FFT code size is about 4 Kbytes.

Table 13. FFT number of cycles, CPU @ 600MHz, external memories @190 MHz

ID	Configuration	Number of cycles	Relative ratio
1	ITCM_DTCM	123136	100%
2	Flash_DTCM	131946	107.2%
3	Hexa PSRAM_DTCM	134453	109.2%
4	Octo flash - DTCM	135406	110.0%
5	Hexa PSRAM_AXISRAM2	137213	111.4%
6	Octo flash_AXISRAM2	138283	112.3%

The relative ratio calculation allows the comparison of the performance of a given configuration against the configuration having the best performance (ITCM_DTCM). When data is stored in DTCM, the performance of STM32H7Rx/7Sx devices is reduced by 9.2% when the execution is from Hexa PSRAM, and by 10% when the code is executed from the external octo flash memory. However, more performance reduction is seen when data is stored in internal RAMs.

4.2 Bandwidth measurements and analysis

In this section, the bandwidth measurements are presented for the most common use case. First, bandwidth estimation is performed, and then followed by the presentation of actual measured bandwidth based on the external memory frequency.

4.2.1 Bandwidth use case

The purpose of the benchmark provided in X-CUBE-PERF-H7RS is to measure the bandwidth of realistic access to external memories. Several types of transactions can be done depending on which operation is done and by which master.

For any application, four types of scenarios with different transactions with external memories can be distinguished:

Execute in place

The execute in place feature executes the code directly from the external memory configured in memory-mapped mode. In such use case, the access by the CPU to the external memory to fetch instructions with the ICACHE enabled is granted by an AXI transfer with a fixed length of four double words or eight words on the AXI bus. This kind of access is usually nonsequential.

Code download/graphic assets

The load and run feature is utilized when the code is stored in external flash memory, but is executed from internal or external RAM. In this scenario, all the code is copied from external flash memory, which requires a long and sequential read access. This is also applicable to graphic applications where the frame buffer content is created by copying graphical primitives from external flash memory.

C variables initialization/Code upload/Graphic frame buffer initialization

The external PSRAM memory is accessed sequentially to write the variables at the initialization step of an application, to write a code copied from the nonvolatile external memory or to create the frame buffers for graphic applications. In all these cases, the write access to the external memory is sequential.

C variable accesses/graphic frame buffer

When an application is running, the external PSRAM is accessed to update the values of the stored variables. Such access is nonsequential and is often a read followed by a write. It is the same for the graphic frame buffer where the pixels are first read then modified then written back.

Benchmarking configuration

This benchmark consists of measuring the bandwidth for these four scenarios using the HPDMA1:

- Sequential read from the external flash memory
- Sequential write into the external PSRAM memory
- Nonsequential read from the external flash memory
- Nonsequential read modify write into the external PSRAM

To measure the bandwidth of the considered accesses, a buffer of 16 Kbytes of data is transferred between internal and external memories using the HPDMA1. In the firmware, the HPDMA1 is configured either to provide the maximum performance for a sequential transfer or to simulate a nonsequential transfer with a length of four double words or eight words. Both HPDMA1 configurations can be selected by commenting/ uncommenting sequential or nonsequential macro in the main core text.

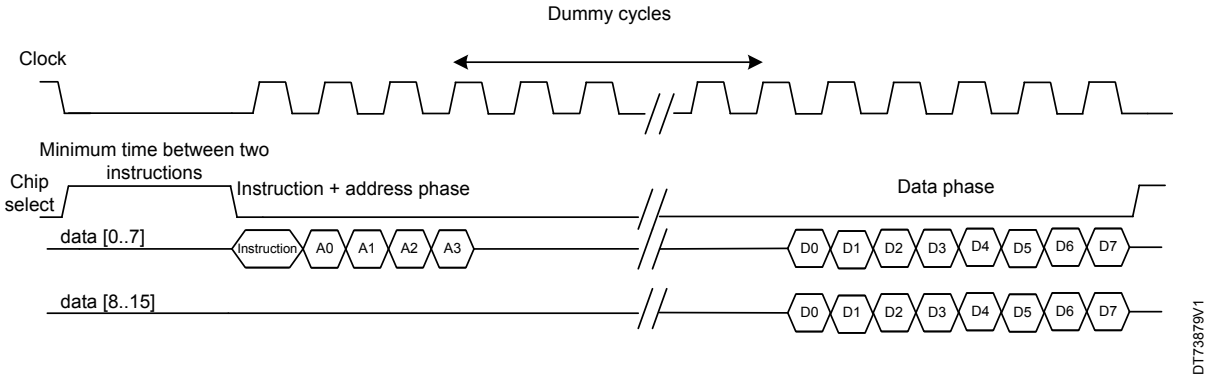
4.2.2 Bandwidth estimation

The XSPI protocol has the advantage of having a very low pin count. However, this implies having dozens of cycles of latency to send the instruction and the address to the memory prior to accessing any data.

External XSPI RAM

In average, an external XSPI RAM have around 25 cycles of setup (termination of the previous transaction + minimum time between two transactions + command phase + address phase + dummy cycles). After these cycles of setup, the phase of data transmission starts. Figure 8 shows an example of HexaSPI RAM data transmission.

Figure 8. Example of data transmission on HexaSPI RAM



For nonsequential access (8 × 32-bit read or write), the data phase duration of a HexaSPI RAM in DDR mode is eight cycles. For such access, the average efficiency of the XSPI is around 25% (8 cycles/(25+8) cycles).

If the memory is for example operating at 190MHz, the expected peak of bandwidth is equal to 190 × 4 × 0.25 = 190 MBps in DDR HexaSPI without considering all the refresh operations and potential collisions.

Doing sequential access increase efficiency as the transactions is automatically merged into a very long one. Usually, an efficiency of 80% could be reached as the latency becomes much smaller than the data phase.

If the memory is for example operating at 190MHz, the expected peak of bandwidth is equal to 190 × 4 × 0.80 = 608 MBps in DDR HexaSPI without considering all the refresh operations and potential collisions.

External XSPI flash memory

For an external flash memory, the number of dummy cycles is mostly higher than a RAM, and the width of the bus is usually 8-bit rather than 16-bit for the RAM. This results in nearly equal efficiency of around 25% for nonsequential transactions. If the memory is, for example, operating at 190MHz, a peak around 190 × 2 × 0.25 = 95 MBps in DDR Octo-SPI can be expected.

For sequential accesses, the efficiency is slightly higher as a flash memory can support much longer transactions than a RAM. 90% to 95% of efficiency is targeted. If the memory is for example at 190MHz, a peak above 190 × 2 × 0.95 = 361 MBps in DDR Octo-SPI can be expected.

4.2.3 Bandwidth measurements

When the CPU is at 600 MHz, and when the external memory is at 190 MHz, the measured bandwidth for the sequential read from Octo-SPI flash memory is equal to 364 MBps. The measured sequential write into the external HexaSPI PSRAM is equal to 578 MBps.Figure 9 and Figure 10 present the bandwidth of the sequential read and sequential write according to the frequency of the external memory.

Figure 9. Measured bandwidth for sequential read from flash NOR

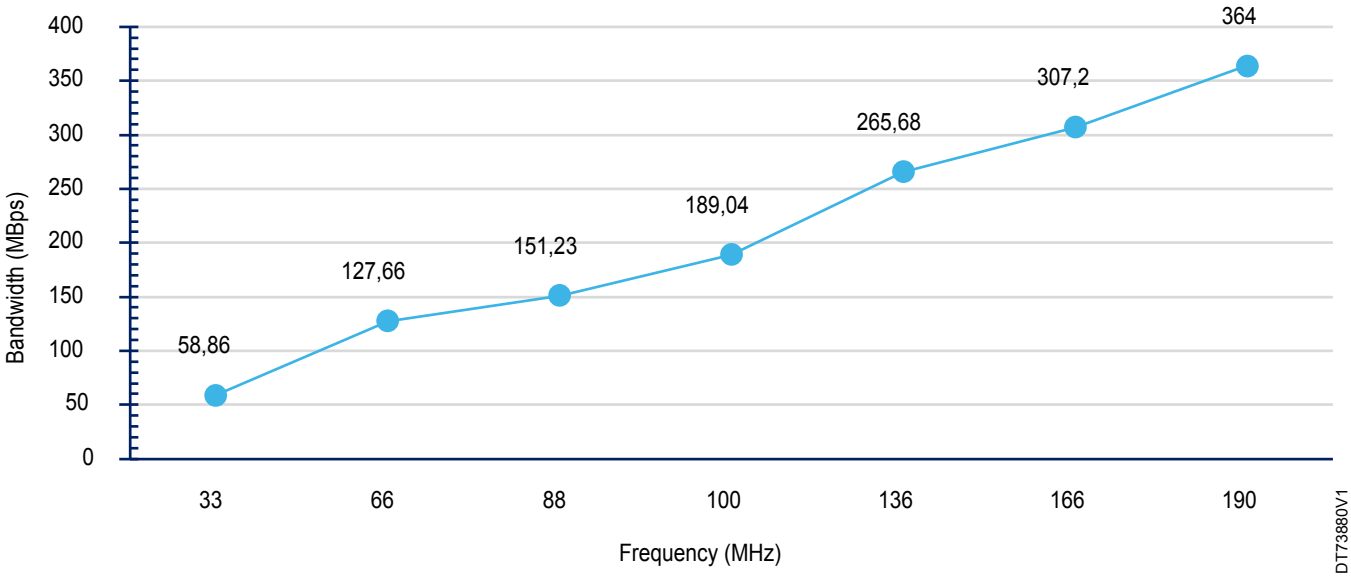
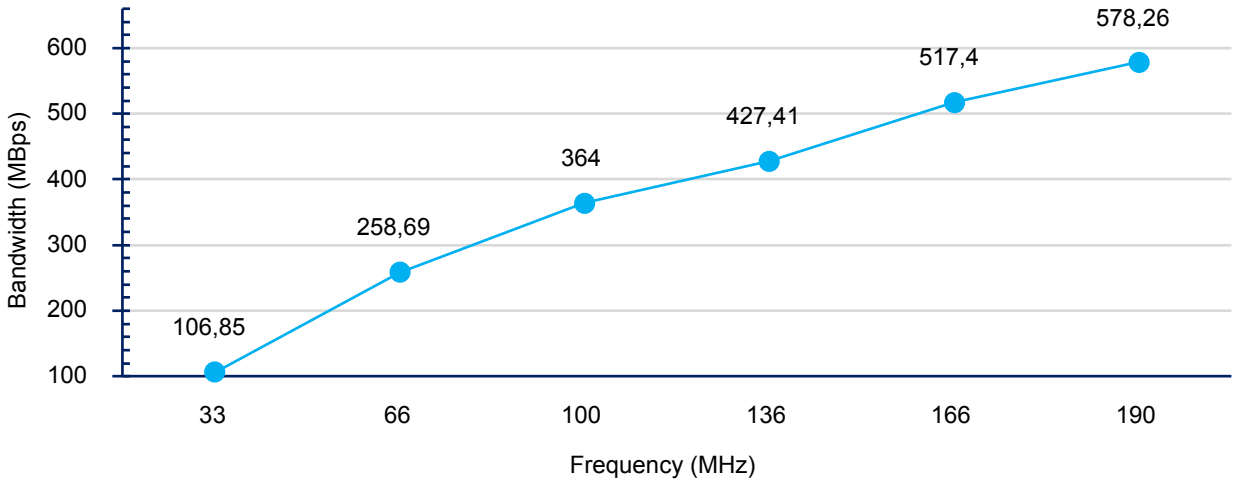


Figure 10. **Measured bandwidth for sequential write into PSRAM**



DT73881V1

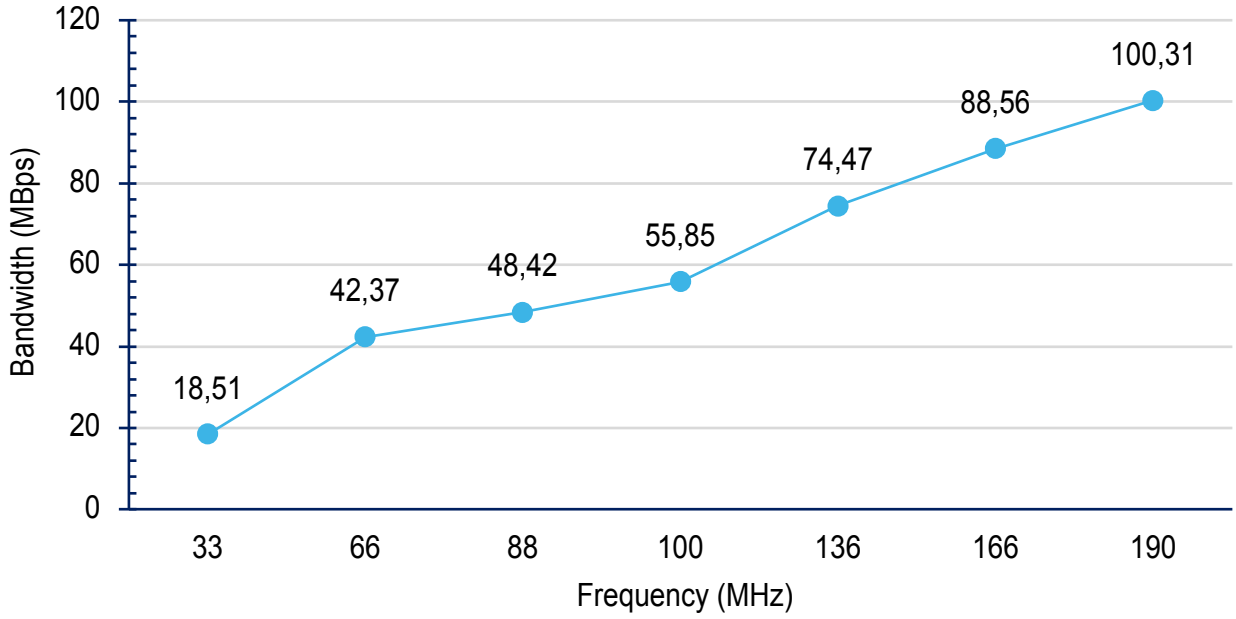
For the nonsequential accesses, the measured bandwidth for the read from flash memory is equal to 100 MBps, while the bandwidth for a read modify-write from PSRAM is equal to 162 MBps. Table 12 summarizes the measured bandwidth for the considered use cases. Figure 11 and Figure 12 illustrate the measured bandwidth regarding the memory frequency.

Table 14. **Measured bandwidth for external memories @190 MHz**

Execution step	XSPI configuration	Measured bandwidth	Bandwidth gain compared to XSPI configuration
Sequential read from flash memory	Octo	364 MBps	-
Sequential write in the PSRAM	Hexa	578 MBps	+65%
	Octo	351 MBps	
Nonsequential read from flash memory	Octo	100 MBps	-
Read-modify write in PSRAM	Hexa	162 MBps	+18%
	Octo	138 MBps	

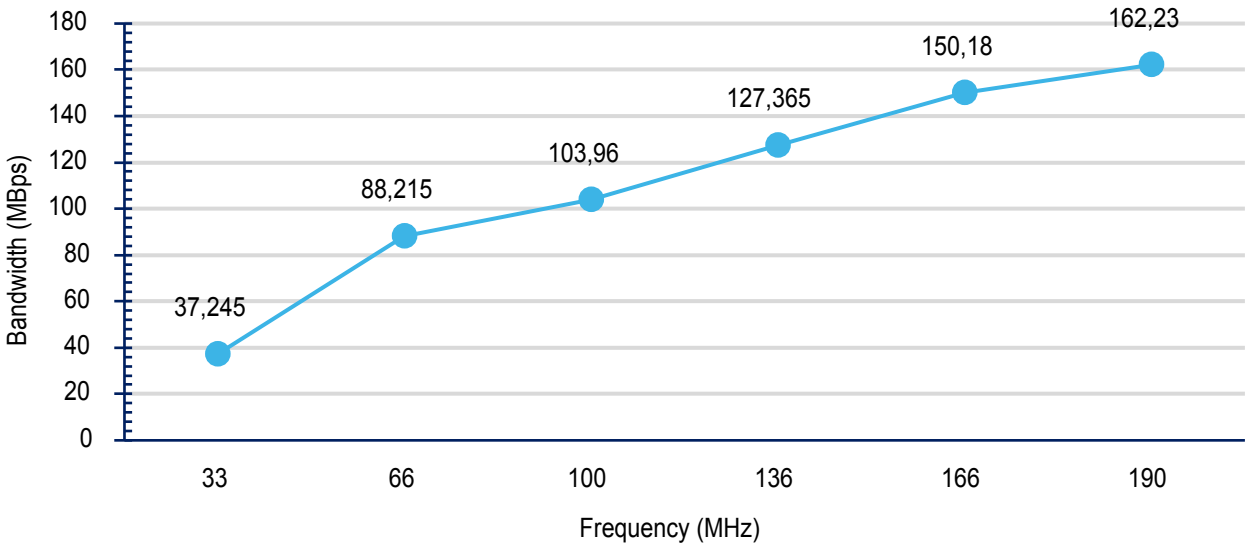
Table 14 shows that the bandwidth for HexaSPI RAM is increased by 65% when compared to OctoSPI RAM for sequential access. For the nonsequential ones, the increase of bandwidth is equal to 18%.

Figure 11. **Measured bandwidth for nonsequential read from flash NOR**



DT73882V1

Figure 12. **Measured bandwidth for read while write in PSRAM**



DT73883V1

5 Impact of MCE

In this section, the impact of enciphering and deciphering is measured according to the number of needed cycles or the measured bandwidth. For each external memory, all the MCE algorithms are tested with both proposed benchmarks. In some cases, the XSPI IO Manager is used in swapped mode to apply the MCE1 (AES) on the XSPI2 and MCE2 (Noekeon) on the XSPI1.

The MCE activation results in an additional delay as presented in Section 2.5.3.3. This delay differs from a read or write access.

5.1 MCE impact on the execution time

As an execution example, the FFT described in Section 3: Typical application is considered to measure the impact of MCE in case of execute in place. When MCE is enabled and configured, any access to the external memory is decrypted in case of read, or encrypted in case of write. When a code is executed, all the instructions are decrypted on-the-fly.

For the case of the FFT, the code size is about 4 Kbytes and can fit in the instruction cache. When the instruction cache is enabled, the CPU reads and decrypts all the code at once. To provide an idea about the impact of MCE in case no instruction cache is enabled, the needed cycles to execute the FFT with relative ratio compared to the case where no MCE is enabled (ID3 and ID4) are given in the Table 15.

Table 15 and Table 15 give an estimation of the variation of the number of cycles according to the encryption algorithm and its configuration.

Table 15. MCE impact on the execution from PSRAM@190 MHz, CPU @600 MHz

-		Number of cycles	Relative ratio	Number of cycles	Relative ratio
ID	Configuration	Instruction cache ON		Instruction cache OFF	
3	Hexa PSRAM _DTCM	134453	100%	929823	100%
7	Hexa PSRAM _DTCM_AES_Block	135807	101%	1180522	126.9%
8	Hexa PSRAM _DTCM_AES_FastBlock	135811	101%	1180286	126.9%
9	Hexa PSRAM _DTCM_AES_Stream	134861	100.3%	969580	104.2%
10	Hexa PSRAM _DTCM_Noekeon_Block	135205	100.5%	1037245	111.5%
11	Hexa PSRAM _DTCM_ Noekeon_FastBlock	135215	100.5%	1037951	111.6%

Table 16. MCE impact on the execution from PSRAM@190 MHz, CPU @600 MHz

-		Number of cycles	Relative ratio	Number of cycles	Relative ratio
ID	Configuration	Instruction cache ON		Instruction cache OFF	
4	Octo flash - DTCM	135406	100%	1000840	100%
12	Octo flash _DTCM_AES_Block	136796	101%	1271185	127%
13	Octo flash _DTCM_AES_FastBlock	136833	101%	1271355	127%
14	Octo flash _DTCM_AES_Stream	135465	100%	1010591	100.9%
15	Octo flash _DTCM_Noekeon_Block	136133	100.5%	1196825	119.5%
16	Octo flash _DTCM_ Noekeon_FastBlock	136155	100.5%	1196731	119.5%

When an execution occurs from an external memory with instruction cache enabled, multiple reads are addressed. MCE is designed to speed up the multiple reads. More specifically, when no write is on-going in MCE the two cipher cores can be allocated to decrypt two reads in parallel. Best performances are obtained when transactions are aligned on two 64-bit words for regions where the block cipher is selected. In such case, and when the instruction cache is enabled, the MCE impact is limited to 1% of more instructions for the AES block ciphering.

However, when the instruction cache is not enabled, MCE is requested at each instruction fetch, and the delay of decryption is added at each instruction execution. In such case, the impact of MCE is at its peak and is about 27% of the needed cycles.

Depending of the user application, the MCE impact can vary from 1% of additional cycles when the application can fit into the cache instruction, to 27% of additional cycles when no cache instruction is enabled. A rough estimate of MCE additional cycles could be done by comparing the user application size to the 32 Kbytes size of the instruction cache.

5.2 MCE impact on the bandwidth

The impact of ciphering and deciphering is measured for the considered uses cases described in Section 4:

- Sequential read from external Octo flash NOR,
- Sequential write into external Hexa PSRAM,
- Nonsequential read from external Octo flash NOR
- Nonsequential read-modify write into external Hexa PSRAM.

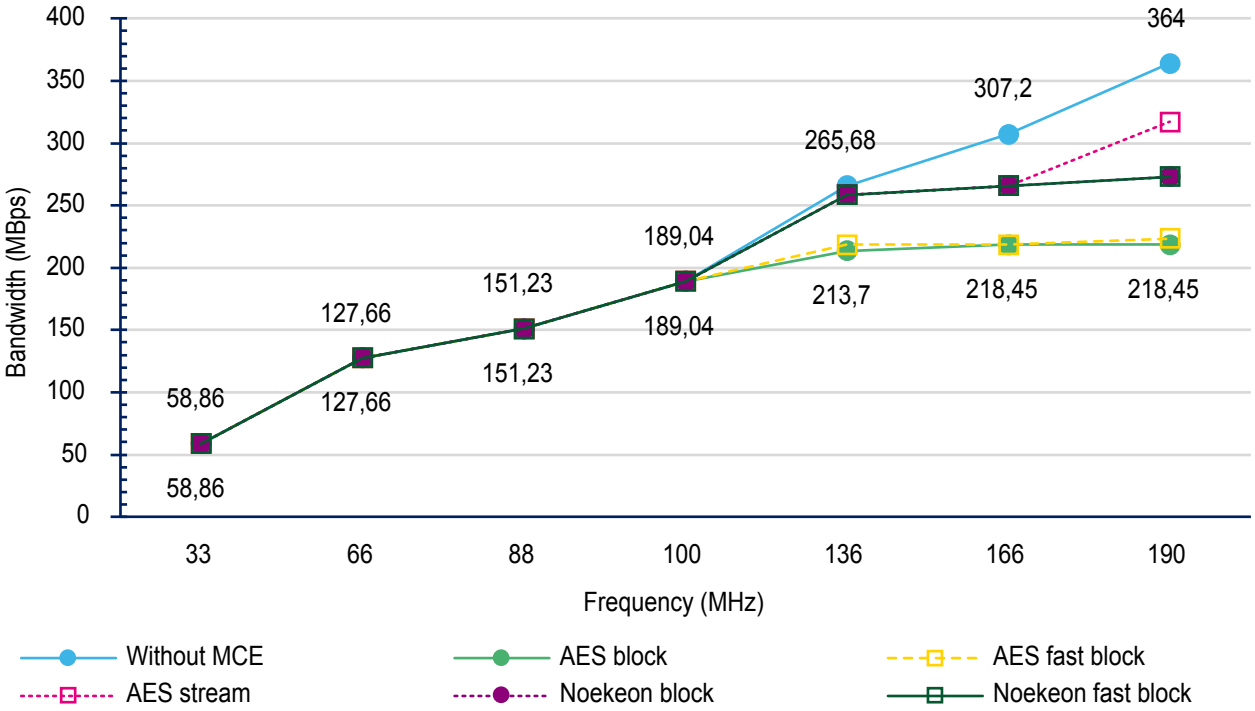
When CPU operates at 600 MHz and the external memories at 190 MHz, the measured bandwidths according to the use cases and to the MCE configuration possibilities are summarized in Table 17. Section 5.2 and Section 5.2 present the bandwidth variation according to the frequency of the external memory.

Table 17. MCE impact on the bandwidth, external memories @190 MHz

	Id	Configuration	Bandwidth (MBps)	Relative ratio
Sequential read from flash memory	13	Without MCE	364	
	14	AES block	218.45	-40%
	15	AES fast block	223.41	-39%
	16	AES stream	317.11	-13%
	17	Noekeon block	273	-25%
	18	Noekeon fast block	273	-25%
Sequential write into PSRAM	1	Without MCE	578.26	
	3	AES block	136.53	-76.4%
	4	AES fast block	158.55	-72.6%
	5	AES stream	351.09	-39.3%
	6	Noekeon block	223.41	-61.4%
	7	Noekeon fast block	289.13	-50.0%
Nonsequential read from flash memory	13	Without MCE	100.31	
	14	AES block	91	-9.3%
	15	AES fast block	91	-9.3%
	16	AES stream	100.31	0.0%
	17	Noekeon block	93.62	-6.7%
	18	Noekeon fast block	93.62	-6.7%
Read modify write into PSRAM	1&2	Without MCE	162.23	
	3&8	AES block	127.67	-21.3%
	4&9	AES fast block	133,215	-17.9%
	5&10	AES stream	162.23	0.0%

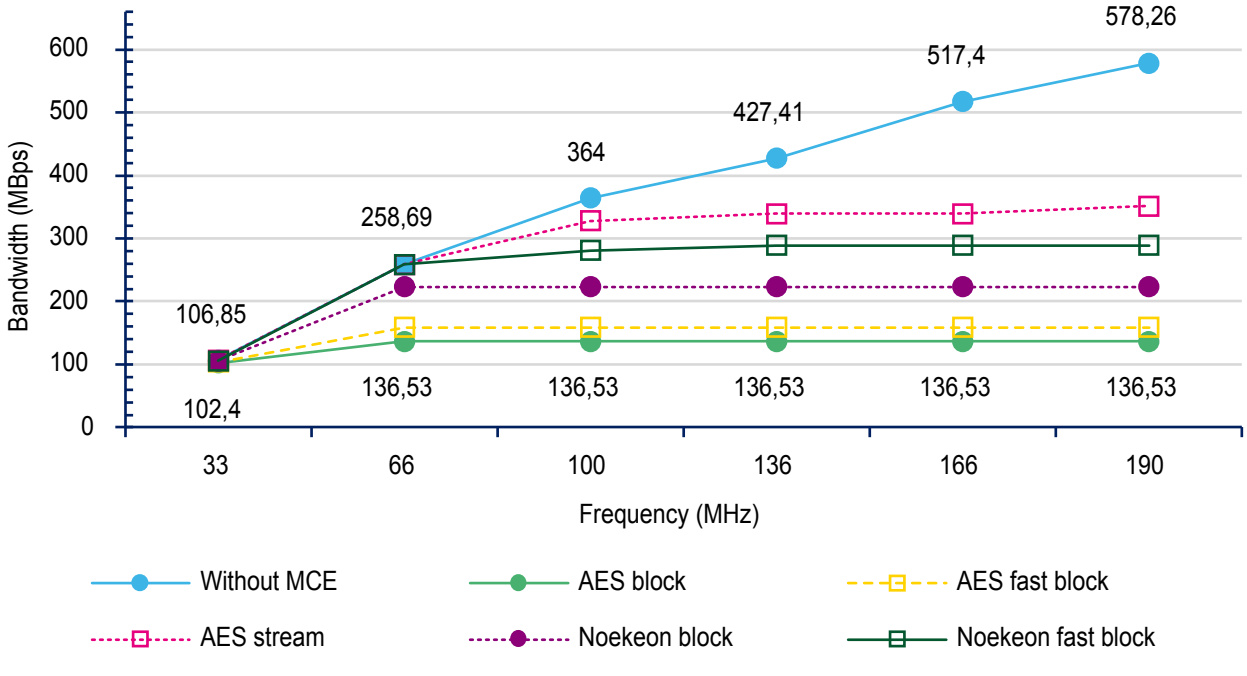
	Id	Configuration	Bandwidth (MBps)	Relative ratio
Read modify write into PSRAM	6&11	Noekeon block	150,915	-7.0%
	7&12	Noekeon fast block	155.67	-4.0%

Figure 13. Measured bandwidth for sequential read from flash NOR according to MCE configurations



DT73884V1

Figure 14. Measured bandwidth for sequential write into Hexa PSRAM according to MCE configurations



DT73885V1

All the optimizations of the MCE are focused on the read. Write access is halted when encryption is in progress until the read with decryption is completed. In addition, any encryption leads to 10 more cycles of latency compared to the decryption. These additional cycles are needed for the key expansion. All these considerations result in a further decrease in bandwidth for write access compared to read access. The impact of MCE is also more visible on sequential access.

The MCE locks the maximal value of the bandwidth to a given value depending on the latency. Since the MCE is driven by AXI clock, this maximal value does not depend on the frequency of the external memory. Table 18 presents the maximum theoretical value of the bandwidth if MCE is enabled.

Table 18. MCE maximum theoretical bandwidth for 32 bytes

MCE configuration		AES block	AES fast block	AES stream	Noekeon block	Noekeon fast block
Write	Delay	46	36	22	28	18
	Theoretical bandwidth	208	266	436	343	533
Read	Delay	36	26	22	28	18
	Theoretical bandwidth	266	369	436	343	533

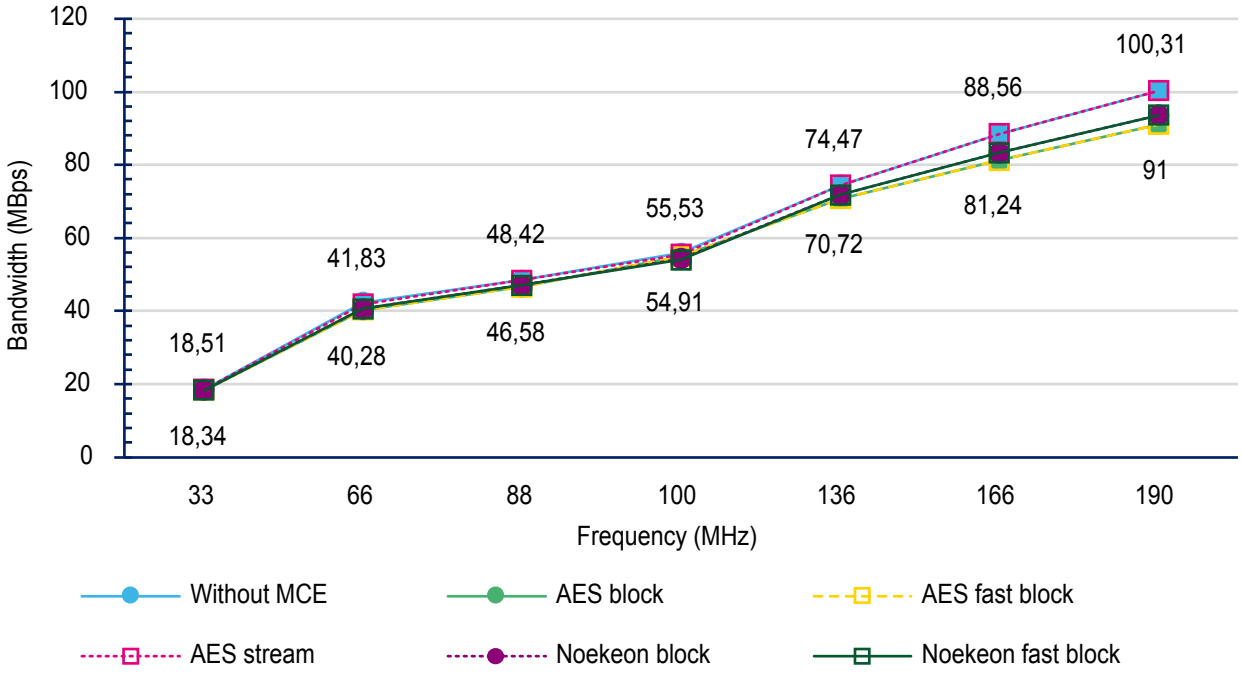
For the PSRAM memory, the X-CUBE-PERF-H7RS firmware provides the bandwidth measurement of separated write and read of 16 Kbytes. The value mentioned in Table 17 of read modify write in the PSRAM is equal to the mean of read and write bandwidths when the external memory attribute is configured as write-back. In this case, all access lengths are equal to 256 bits.

When an access to the PSRAM is not a multiple of 128 bits, and when block or fast block ciphering is configured, read modify write is automatically performed.

For instance, if a write access of four bytes occurs when the MCE block or fast block is activated, all the block of 128 bits containing the four bytes is read, decrypted, and modified with the four considered bytes. The whole block of 128 bits is then encrypted, and finally written to the external memory. To avoid such latency, the write-back attributes of external memory region is highly recommended.

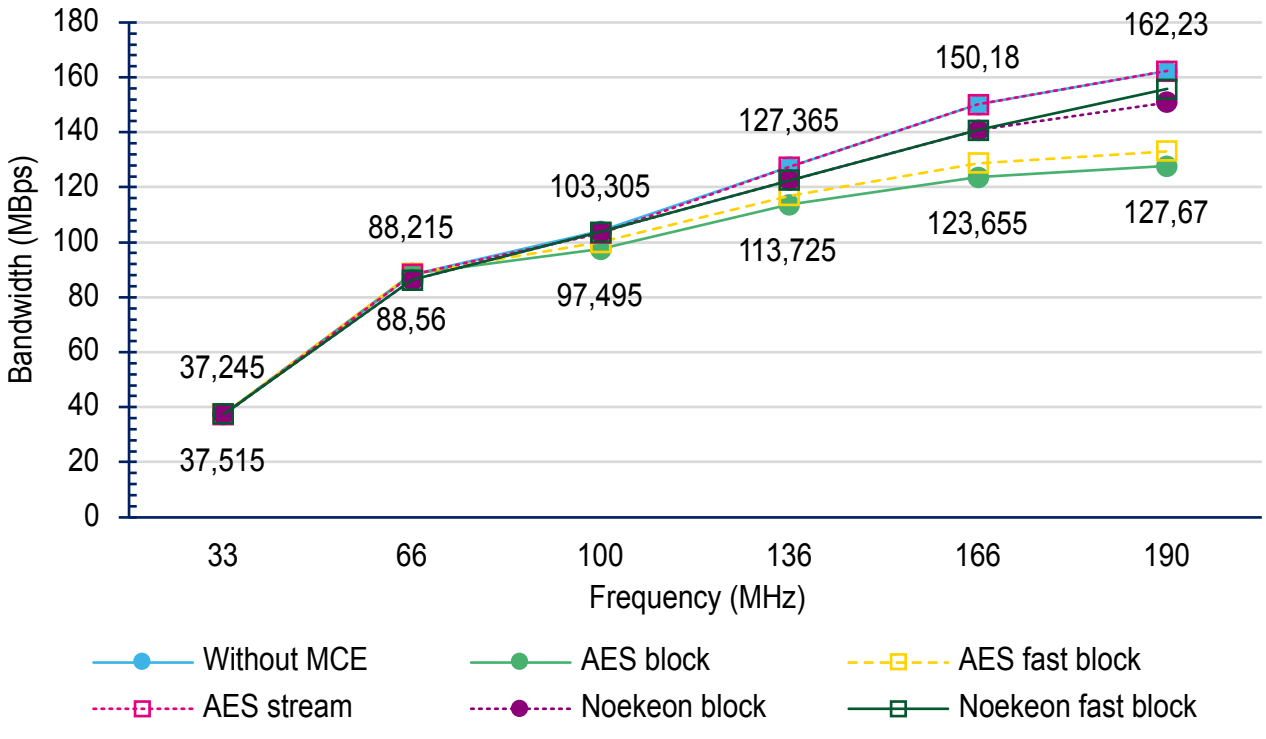
Section 5.2 and Section 5.2 present the bandwidth measured values according the external memory values for nonsequential read from flash NOR and nonsequential read-modify-write in PSRAM.

Figure 15. Measured bandwidth for nonsequential read from flash NOR according to MCE configurations



DT73886V1

Figure 16. Measured bandwidth for read modify write in hexa PSRAM according to MCE configurations



DT73887V1

6 Recommendations and tips

In this section are given some recommendations and conclusions about software memory partitioning, external memories frequency and performances, and MCE configurations.

6.1 Tips on data and code locations

ITCM and DTCM

Since the Cortex-M7 has a 64-bit wide direct access to TCM memories with zero wait state, the DTCM and the ITCM locations are the best locations for the read/write of data and instruction fetch respectively

Therefore, the ITCM is reserved for critical code with deterministic execution, such as interrupt handlers that cannot wait for cache misses, as well as for some critical control loops targeting, for example, motor control applications.

The DTCM is reserved for regular access to R/W data and for critical real time data, such as stack and heap that need determinism. For higher-speed computation and if the data is located for example in AXI SRAM, HPDMA1 can be used to move the data from these memories to the DTCM to be processed at the CPU clock speed.

In real-time applications that use RTOS, the heap is generally massively used. The user can scatter the DTCM to fit her/his application needs but she/he must give higher priority to place the stack in DTCM with respect to heap and global variables. In a bare-metal application model, the heap is less used and the DTCM is scattered between stack and global variables only.

Embedded RAMs

AXI SRAMs can be used for data storage when no available space is left in DTCM. In that case, a region from the AXI SRAM, with the data cache enabled, can be reserved for global variables to leave more space for critical data needing deterministic access.

AHB SRAMs can be used as buffers to store in/out data of peripherals located in the AHB domain such as Ethernet and USB. These data can be buffers and descriptors or I2S audio frames or others. Be careful to declare these shared buffers as “sharable” to prevent them from being cached or to manage the cache carefully to avoid data inconsistency.

Embedded flash memory

When the code size of the user application fits into the internal flash memory (with cache enabled), the latter is the reasonable location for code execution (in terms of performance) while the critical code needing determinism is placed in ITCM.

External memories

The external octo flash memory or the external hexa PSRAM are almost equivalent alternatives for code execution.

XSPI flash memory or PSRAM memories can be used to store the application code in the memory mapped mode up to 256 Mbytes with less needed pins compared to parallel flash memories that have to be connected to the FMC interface.

The most suitable scattering for large applications that need access to large R/W data is to load the user application in the XSPI flash memory and to use the external PSRAM for data storage with caches enabled. In fact, for graphical applications that need a large amount of graphic data, the external PSRAM in memory mapped configuration can be reserved for a graphic frame. High display performance is then ensured by the LTCD, the GFXMMU, GPU2D, and the DMA2D.

6.2 Bandwidth and memory configurations

The measured bandwidths of the most common accesses to the XSPI external memory shows a high data rate in sequential data transfer that exceeds the estimated bandwidth of the external flash memory and 95% of the estimated bandwidth of the external PSRAM memory.

A comparison between the octal and hexadecimal configuration of the external PSRAM memory in term of bandwidth shows an improvement of 65% of bandwidth for hexadecimal configuration for a sequential write. For a nonsequential read modify write, this improvement is equal to 18%.

6.3 Tips on MCE configuration

To ensure data confidentiality, code and data encryption and decryption on-the-fly is provided by the MCE. For STM32H7S7x, AES and Noekeon are provided as cyphering algorithms. Both support read and write in memory mapped mode.

AES 128-bit block

AES 128-bit block cipher is the most secured MCE provided solution compatible but with high generated delays. It is shown by the measurement of the number of cycles needed when code is executed from external memories, that the MCE enabling impact is limited to 1% more cycles when instructions cache is enabled and when the application fit into the STM32H7S7x caches. If the application is larger than the latter, the impact depends of its size and can at the worst case lead to 27% more cycles if instruction cache is off.

In terms of bandwidth, AES block leads to a maximum value of bandwidth of 136 MBps reached from the 66 MHz frequency for the write and 218 MBps reached from the 133 MHz for the read. The plateau curves demonstrate that for applications that need sequential access with high confidentiality on the R/W data, it is recommended to choose a PSRAM operating at 133 MHz.

AES 128-bit fast block

AES 128-bit fast block cipher is equivalent to AES 128-bit block but with more sensitivity to side channel attack at the key derivation step. The AES 128-bit fast block impact on bandwidth is up to 4% lower than that of AES 128-bit block for write access. 158,55 MBps are measured when writing in a PSRAM operating at frequency starting from 66 MHz. For the read access, the implemented FIFO reduces the impact of key generation and no difference is seen in terms of bandwidth between normal or fast block ciphering. 223 MBps are measured in this case starting from the frequency 133 MHz.

Noekeon 128-bit block

Noekeon 128-bit cipher considers the same key derivation with the same level of security compared to AES 128-bit which guarantees full side channel protection, and which takes 14 cycles as for AES 128-bit block. The Noekeon ciphering process is simpler compared to AES, but is more sensitive.

223 MBps of bandwidth are measured for a sequential write into PSRAM that operates at a frequency starting from 100 MHz. The maximum bandwidth in a sequential read is equal to 273 MBps when the flash NOR is operating at 190 MHz.

Noekeon 128-bit fast block

Noekeon 128-bit fast block uses the same key derivation solution as AES 128-bit fast block and conserves the same ciphering process as Noekeon 128-bit block. It is then faster than the normal block configuration but more sensitive to side channel attacks. In such configuration, the bandwidth is equal to 351 MBps for a sequential write in PSRAM at 190 MHz and to 273 MBps for sequential read from flash NOR at 190 MHz.

AES 128-bit stream

AES 128-bit stream has the lowest latency but with limitations in terms of security: no side channel protection with possibility of bit-flip attacks. It is highly recommended to use AES stream for read-only region to maintain a given level of security. It basically has no impact on the execution time and number of cycles. The AES stream impact can only be seen on the sequential access with a bandwidth reduction of 13% for the read from flash memory and 39% for the write into PSRAM.

Tips on MCE use

To ensure the best performance, a 128-bit multiple access to external memories is recommended for block and fast block ciphering. For flash NOR memories, DMA is also highly recommended to be used with granularity of 16 bytes.

As explained in Section 5: Impact of MCE, when MCE block configuration is selected and a write access of data size less than 128 bits occurs, a read of 128 bits with decryption, modification of the needed data and encryption of the whole 128-bit block are generated. These steps produce a huge delay and can highly reduce the bandwidth. Consequently, it is highly recommended to enable the instruction and data caches and to maintain the attribute of the external memory encrypted region as write-back.

As a conclusion, if high confidentiality is needed only on the external flash NOR, AES Block (MCE1) is advised.

However, if there is a need for higher bandwidth, AES stream is recommended for flash NOR for executing code for instance.

To protect the R/W data, MCE1 or MCE 2 can be used with data cache enabled.

Due to the small difference of bandwidth, the normal block configuration is preferred to the fast block configuration thanks to its side channel protection.

Revision history

Table 19. **Document revision history**

Date	Version	Changes
22-Mar-2024	1	Initial release.

Contents

List of tables

List of figures