life.augmented

AN4891
Application note

STM32H72x, STM32H73x, and single-core STM32H74x/75x system architecture and performance

Introduction

The STM32H7 series is the first series of STMicroelectronics microcontrollers in 40 nm-process technology. This technology enables STM32H7 devices to integrate high-density embedded flash memory and SRAM that decrease the resource constraints typically complicating high-end embedded development. It also unleashes the performance of the core and enables ultrafast data transfers through the system while realizing major power savings.

In addition, the STM32H7 series is the first series of Arm® Cortex®-M7-based 32-bit microcontrollers able to run at up to 550 MHz, reaching new performance records of 1177 DMIPS and 2777 CoreMark®.

The STM32H7 series is the continuity of the STM32F7 series in terms of high-performance products with significant architecture improvement allowing a performance boost versus STM32F7 series devices.

The architecture and performance of STM32H7 series devices make them ideally suited for industrial gateways, home automation, telecom equipment and smart consumer products, high-performance motor control and domestic appliances, and use in small devices with rich user interfaces such as smart watches.

This application note focuses on STM32H72x, STM32H73x, STM32H742x, STM32H743/753x, and STM32H750x single-core microcontrollers, referred to herein as STM32H72x/73x/74x/75x (see *Table 1*). Dual-core devices are not covered by this document. 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 STM32H74x/75x Arm® Cortex®-M7 single-core architecture performance in various memory partitioning configurations with different code and data locations.

This application note is delivered with the X-CUBE-PERF-H7 Expansion Package dedicated to STM32H742x, STM32H743/753x and STM32H750x microcontrollers. This Expansion Package includes the *H7_single_cpu_perf* project aimed at demonstrating the performance of CPU memory accesses in different configurations with code execution and data storage in different memory locations using L1 cache. The project runs on the STM32H743I-EVAL board.

Reference documents

- Reference manual *STM32H723/733, STM32H725/735, and STM32H730 Value line advanced Arm®-based 32-bit MCUs* (RM0468)
- Reference manual *STM32H742, STM32H743/753, and STM32H750 Value line advanced Arm®-based 32-bit MCUs* (RM0433)

All documents are available from the STMicroelectronics website: *www.st.com*.

Contents

List of tables

List of figures

1 General information

This document applies to STM32 Arm®-based[a] microcontrollers.

arm

2 STM32H72x/73x/74x/75x system architecture overview

This section introduces the main architecture features of the STM32H72x/73x/74x/75x. *Table 1* defines the product lines concerned.

Table 1. STM32H7 lines targeted by this application note

Generic part numbers	Products lines
STM32H72x, STM32H73x	STM32H723/733, STM32H725/735, STM32H730 Value line
STM32H74x, STM32H75x	STM32H742, STM32H743/753 lines, STM32H750 Value line

2.1 Cortex®-M7 core

The STM32H72x/73x/74x/75x devices are built on a high-performance Arm® Cortex®-M7 32-bit RISC core operating at up to 480 MHz frequency (STM32H74x/75x) and 550 MHz (STM32H72x/73x). The Cortex®-M7 core features a high-performance floating-point unit (FPU) as well as a double-precision floating-point unit that supports 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 STM32H72x/73x/74x/75x devices enables defining up to 16 MPU regions. A forward compatibility from the Cortex®-M4 to the Cortex®-M7 enables the binaries, compiled for the Cortex®-M4, to run directly on the Cortex®-M7.

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 in order to increase instruction throughput.

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

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 given in *Table 2*.

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 is given in the *Arm® Cortex®-M7 Processor - Technical Reference Manual* available from the www.arm.com website.

More details on L1-cache usage in STM32H72x/73x/74x/75x devices are available in the application note *Level 1 cache on STM32F7 and STM32H7 series* (AN4839).

Table 2. STM32H72x/73x/74x/75x device cache sizes

Devices	Instruction cache size	Data cache size
STM32H72x/73x	32 Kbytes	32 Kbytes
STM32H74x/75x	16 Kbytes	16 Kbytes

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 consider 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 memory type is cacheable.

Additional information and general rules about memory attributes and behaviors are available in the *STM32F7 and STM32H7 series Cortex®-M7 processor programming manual* (PM0253).

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 3 summarizes the memory region attributes after Cortex®-M7 reset.

Table 3. Cortex®-M7 default memory attributes after reset

Address range	Region name	Type	Attributes	Execute never?
0x0000 0000-0x1FFF FFFF	Code	Normal	Cacheable, Write-Through, Allocate on read miss	No
0x2000 0000-0x3FFF FFFF	SRAM	Normal	Cacheable, Write-Back, Allocate on read and write miss	No
0x4000 0000-0x5FFF FFFF	Peripheral	Device	Non-shareable	Yes
0x6000 0000-0x7FFF FFFF	RAM	Normal	Cacheable, Write-Back, Allocate on read and write miss	No
0x8000 0000-0x9FFF FFFF	RAM	Normal	Cacheable, Write-Through, Allocate on read miss	No
0xA000 0000-0xBFFF FFFF	External device	Device	Shareable	Yes
0xC000 0000-0xDFFF FFFF	External device	Device	Non-shareable	Yes
0xE000 0000-0xE000 FFFF	Private peripheral bus	Strongly ordered	-	Yes
0xE001 0000-0xFFFF FFFF	Vendor system	Device	Non-shareable	Yes

In STM32H72x/73x/74x/75x, the 64-bit AXI master bus connects the core to the 64-bit AXI bus matrix (D1 domain).

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 (literal pool) 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 STM32H72x/73x/74x/75x architecture, only the CPU and the MDMA 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 STM32H72x/73x/74x/75x architecture and apart from Cortex®-M7, AHBS allows data transfer from/to DTCM-RAM and ITCM-RAM using only MDMA. The AHBS interface can be used when the core is in Sleep state, therefore, MDMA transfers from/to TCM-RAMs can be performed in low-power modes. This connection is represented by the paths colored in light pink in *Figure 3*.

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 STM32H72x/73x/74x/75x architecture, this interface connects the Cortex®-M7 core to the AHB peripherals connected to a 32-bit AHB bus matrix that is located in the D2 domain (see *Section 2.4*). The targets of this bus are the AHB1, AHB2, APB1, and APB2 peripherals located in the D2 domain. This connection is represented by the bus colored in dark green in *Figure 2*. The peripherals located in the D1 and D3 domains (see *Section 2.4*) are seen by the Cortex®-M7 through the AXI bus while the peripherals located in the D2 domain are seen by the Cortex®-M7 through the AHBP bus.

2.4 STM32H72x/73x/74x/75x interconnect matrix

The STM32H72x/73x/74x/75x devices are the first STM32 microcontrollers that embed more than one bus matrix, thus giving the best compromise between performance and power consumption. It also 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).

The STM32H72x/73x/74x/75x feature three separate bus matrices. Each bus matrix is associated with a domain:

- 64-bit AXI bus matrix (in the D1 domain): It has a high-performance capability and is dedicated to operations requiring high speed. The high-bandwidth peripherals are connected to the AXI bus matrix.
- 32-bit AHB bus matrix (in the D2 domain): communication peripherals and timers are connected to this bus matrix.
- 32-bit AHB bus matrix (in the D3 domain): reset, clock control, power management, and GPIOs are located in this domain.

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 by means of interdomain buses to enable a master located in a given domain to access to a slave located in another domain, except for BDMA master whose access is limited to resources located in the D3 domain.

Figure 1 and *Figure 2* show the overall system architecture of the STM32H72x/73x/74x/75x as well as the connections of the interconnect matrix.

Figure 1. STM32H72x and STM32H73x system architecture

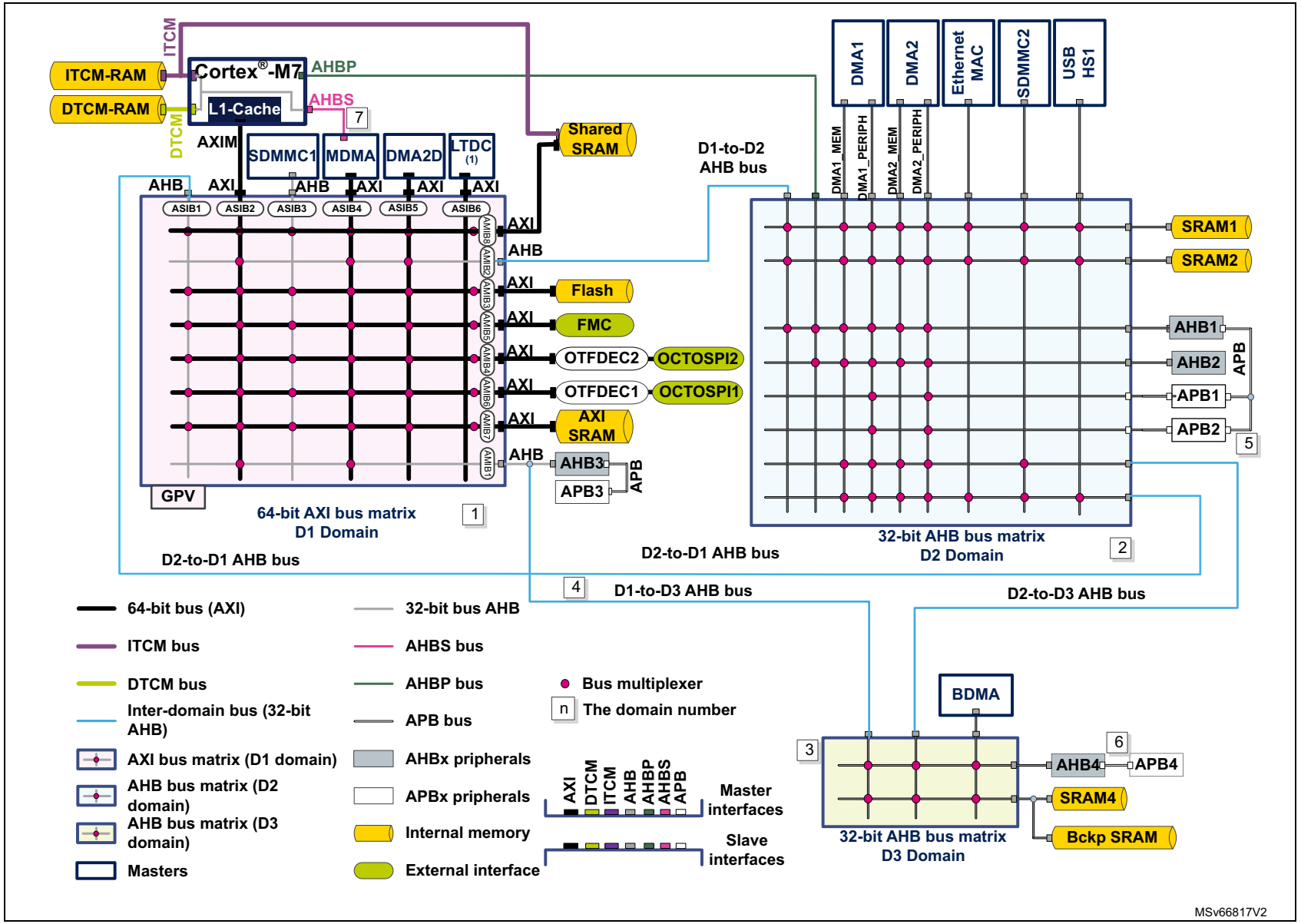
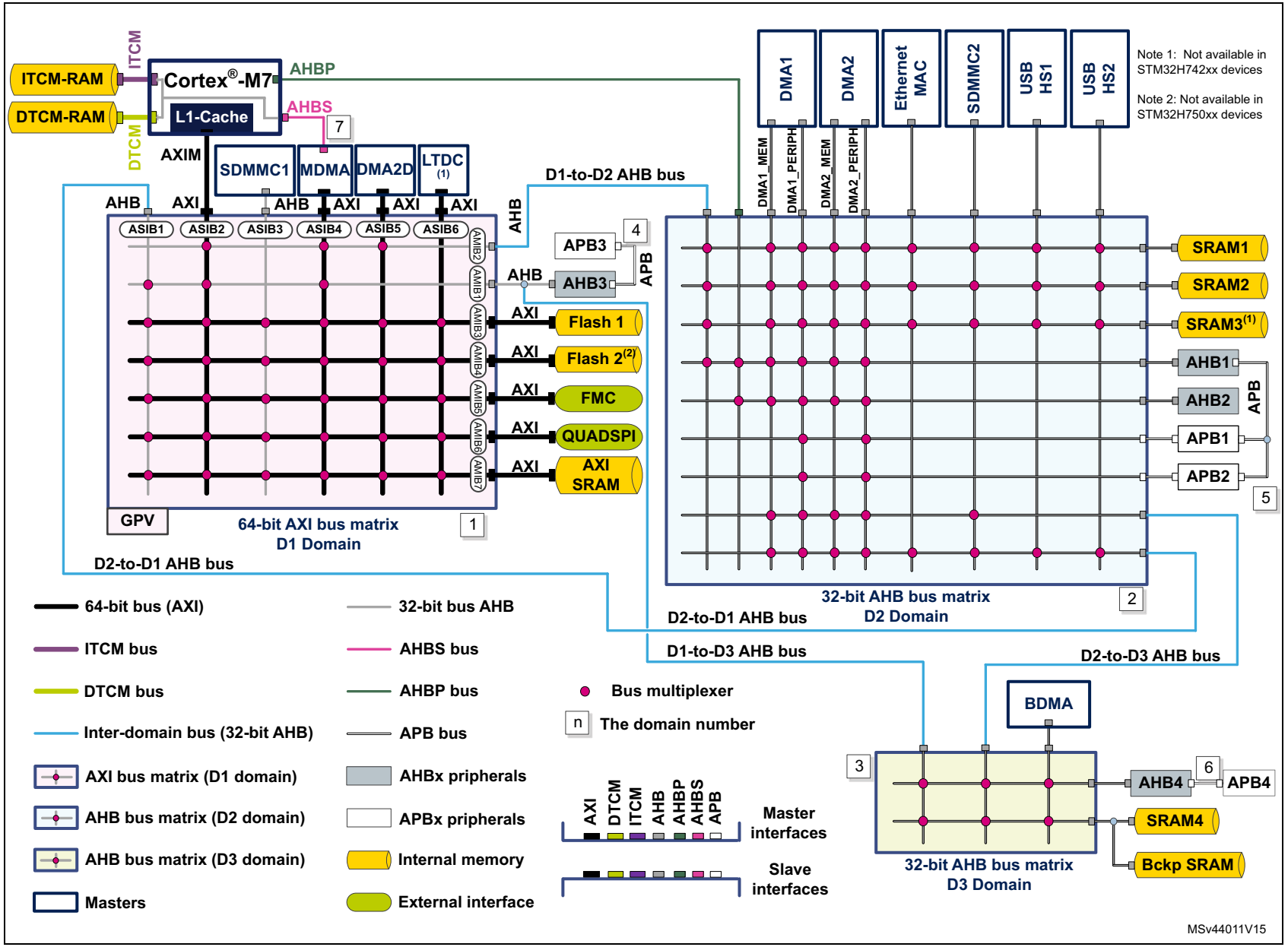


Figure 2. STM32H74x and STM32H75x system architecture



2.4.1 AXI bus matrix in the D1 domain

The AXI interconnect is based on the Arm® CoreLink™ NIC-400 Network Interconnect. It has six initiator ports called ASIBs (AMBA® slave interface blocks) where masters are connected, and up to eight target ports called AMIBs (AMBA® master interface blocks) where slaves are connected.

The AXI bus matrix is located in the D1 domain. It 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 simultaneously. 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 LTDC and of the DMA2D versus the Cortex®-M7 CPU.

The AXI bus matrix interconnects:

- Six bus masters:
  - The Cortex®-M7 AXI bus
  - The D2-to-D1 AHB interdomain, a 32-bit AHB bus that connects the D2 domain to the D1 domain
  - The SDMMC1 32-bit AHB bus
  - The MDMA 64-bit AXI bus
  - The LCD-TFT controller 64-bit AXI bus (not available in STM32H742x devices)
  - The Chrom-Art Accelerator (DMA2D) 64-bit AXI bus
- Up to eight bus slaves:
  - The embedded flash memory bank 1 on AXI bus
  - Up to 512 Kbytes of AXI SRAM memory accessed through the AXI bus
  - AHB3 peripherals including the AHB-to-APB bridge, APB3 peripherals and the D1-to-D3 AHB interdomain
  - The FMC memory interface on the AXI bus accessed by 64 bits
  - The D1-to-D2 interdomain 32-bit AHB bus that connects the D1 domain to the D2 domain
  - The quad-SPI memory interface on AXI bus with 64-bit access (available only in STM32H74x/75x)
  - The embedded flash memory bank 2 on AXI bus (available only in STM32H74x/75x except for STM32H750x)
  - SRAM shared between ITCM and AXI (available only in STM32H72x/73x)
  - OCTOSPI1 memory interface on AXI bus accessed by 64 bits (available only in STM32H72x)

  - OCTOSPI2 memory interface on AXI bus accessed by 64 bits (available only in STM32H72x)
  - OTFDEC1/OCTOSPI1 memory interface on AXI bus with 64-bit access (available only in STM32H73x)
  - OTFDEC1/OCTOSPI2 memory interface on AXI bus with 64-bit access (available only in STM32H73x)

The Cortex®-M7 has access to every resource available in the system. The AHB1 peripherals are accessed by the CPU through the AHBP bus and not through the AXI bus and D1-to-D2 AHB interdomain bus (refer to *Figure 2*). The MDMA has access to all resources available in the system except to the AHB2 resources located in the D2 domain.

2.4.2 AHB bus matrices in the D2 and D3 domains

The AHB bus matrices are located in the D2 and D3 domains. Their maximum frequency is half the maximum CPU frequency. They ensure and arbitrate concurrent accesses 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 in the D2 domain is dedicated to communication peripherals and timers. It interconnects the following buses:

- Up to 10 bus masters:
  - The D1-to-D2 AHB interdomain that connects the D1 domain to the D2 domain
  - The Cortex®-M7 AHB peripherals bus that makes the CPU to access AHB1 and AHB2 peripherals on D2 domain
  - The DMA1 memory AHB bus
  - The DMA1 peripheral AHB bus
  - The DMA2 memory AHB bus
  - The DMA2 peripheral AHB bus
  - The Ethernet DMA AHB bus
  - The SDMMC2 DMA AHB bus
  - The USB OTG high-speed 1 DMA AHB bus
  - The USB OTG high-speed 2 DMA AHB bus (available only in STM32H74x/75x)
- Up to nine bus slaves:
  - Internal SRAM1 up to 128 Kbytes with 32-bit AHB access
  - Internal SRAM2 up to 128 Kbytes with 32-bit AHB access
  - Internal SRAM3 of 32 Kbytes with 32-bit AHB access (available in STM32H74x/75x except STM32H742x)
  - The AHB1 peripherals bus including the AHB to APB bridge that makes Cortex®-M7 to access APB1 and APB2 peripherals
  - The AHB2 peripheral bus that connects speed peripherals
  - The APB1 peripheral bus that enables DMA1 and DMA2 to access to APB1 peripherals
  - The APB2 peripheral bus that enables DMA1 and DMA2 to access to APB2 peripherals

  - The D2-to-D3 AHB interdomain that connects the D2 domain to the D3 domain
  - The D2-to-D1 AHB interdomain that connects the D2 domain to the D1 domain

The AHB bus matrix in the D3 domain is dedicated to reset, clock control, power management and GPIOs. It interconnects:

- Three initiators:
  - The D1-to-D3 AHB interdomain that connects the D1 domain to the D3 domain
  - The D2-to-D3 AHB interdomain that connects the D2 domain to the D3 domain
  - The BDMA memory AHB bus
- Two bus slaves:
  - The AHB4 peripherals including the AHB to APB bridge (connection 6 in *Figure 2*) and APB4 peripherals
  - Internal SRAM4 up to 64 Kbytes and the backup SRAM of 4 Kbytes that shares the same AHB bus

2.4.3 Interdomain buses

D1-to-D2 AHB interdomain bus

This 32-bit AHB bus connects the AXI bus matrix located in the D1 domain to the AHB bus matrix located in the D2 domain. It allows some masters in the D1 domain to access resources (memories and peripherals) in the D2 domain.

Only the masters, Cortex®-M7, MDMA, and DMA2D located in the D1 domain can have access to the following resources located in the D2 domain: SRAM1, SRAM2, SRAM3 (for STM32H74x/75x), AHB1, APB1, and APB2 peripherals.

The SDMMC1 and LTDC have no access to the resources located in the D2 domain.

So if, for example, SDMMC1 or LTDC needs some data from SRAM1, the user can use MDMA or DMA2D to copy data from SRAM1 to AXI SRAM which is accessible by SDMMC1 and LTDC.

D2-to-D1 AHB interdomain bus

This 32-bit AHB bus connects the D2 domain to the AXI bus matrix located in the D1 domain. It enables bus masters in the D2 domain to access resources in the D1 domain and indirectly, via the D1-to-D3 AHB interdomain bus, the resources located in the D3 domain.

As a result, all the masters, DMA1, DMA2, SDMMC2, Ethernet, and a USB OTG_HS controller located in the D2 domain can have access to the internal memories in the D1 domain (except TCM-RAMs), to external memories, and for STM32H74x/75x to the AHB3 and APB3 peripherals located in the D1 domain.

D1-to-D3 AHB interdomain bus

This 32-bit AHB bus connects the D1 domain to the D3 domain AHB bus matrix. It enables masters in the D1 domain and indirectly some masters in the D2 domain to access resources located in the D3 domain.

Only the two masters, Cortex®-M7 and MDMA in the D1 domain, have access to the resources located in the D3 domain, that is SRAM4, backup SRAM, AHB4 and APB4 peripherals. SDMMC1, LTDC, and DMA2D cannot access these resources.

If SDMMC1, LTDC and DMA2D need some data, for example, from SRAM4, MDMA can be used to transfer them from SRAM4 to AXI SRAM.

For STM32H74x/75x, the D1-to-D3 AHB interdomain bus also makes possible for some masters located in the D2 domain to have access to the resources located in the D3 domain. Note that some masters in the D2 domain, such as USB OTG_HS, do not have direct access to the resources located in D3 through the D2-to-D3 AHB interdomain bus. The access is performed first through the D2-to-D1 AHB and then through the D1-to-D3 AHB inter-domain buses (refer to the USBHS2 access to SRAM4 path highlighted in yellow in *Figure 3*).

D2-to-D3 AHB interdomain bus

This 32-bit AHB bus connects the D2 domain to the D3 domain AHB bus matrix. It enables masters located in the D2 domain to access resources in the D3 domain.

DMA1, DMA2, and SDMMC2 located in the D2 domain have direct access to the resources located in the D3 domain through the D2-to-D3 AHB inter-domain bus. For STM32H74x/75x, the access of the other masters is performed first through the D2-to-D1 AHB and then through the D1-to-D3 AHB interdomain buses. The Ethernet peripheral does not have access to D3 resources.

Note that the basic-DMA controller (BDMA) located in the D3 domain has only access to the resources located in that domain including backup SRAM.

Table 4 summarizes the different possible interconnections and the different access path combinations of the different masters to the different slaves located in different domains. The numbering from 1 to 7 in *Table 4* refers to the numbers indicated in *Figure 2*. Each number refers to a given domain.

Table 4. Bus-master-to-bus-slave possible interconnections in STM32H72x/73x/74x/75x

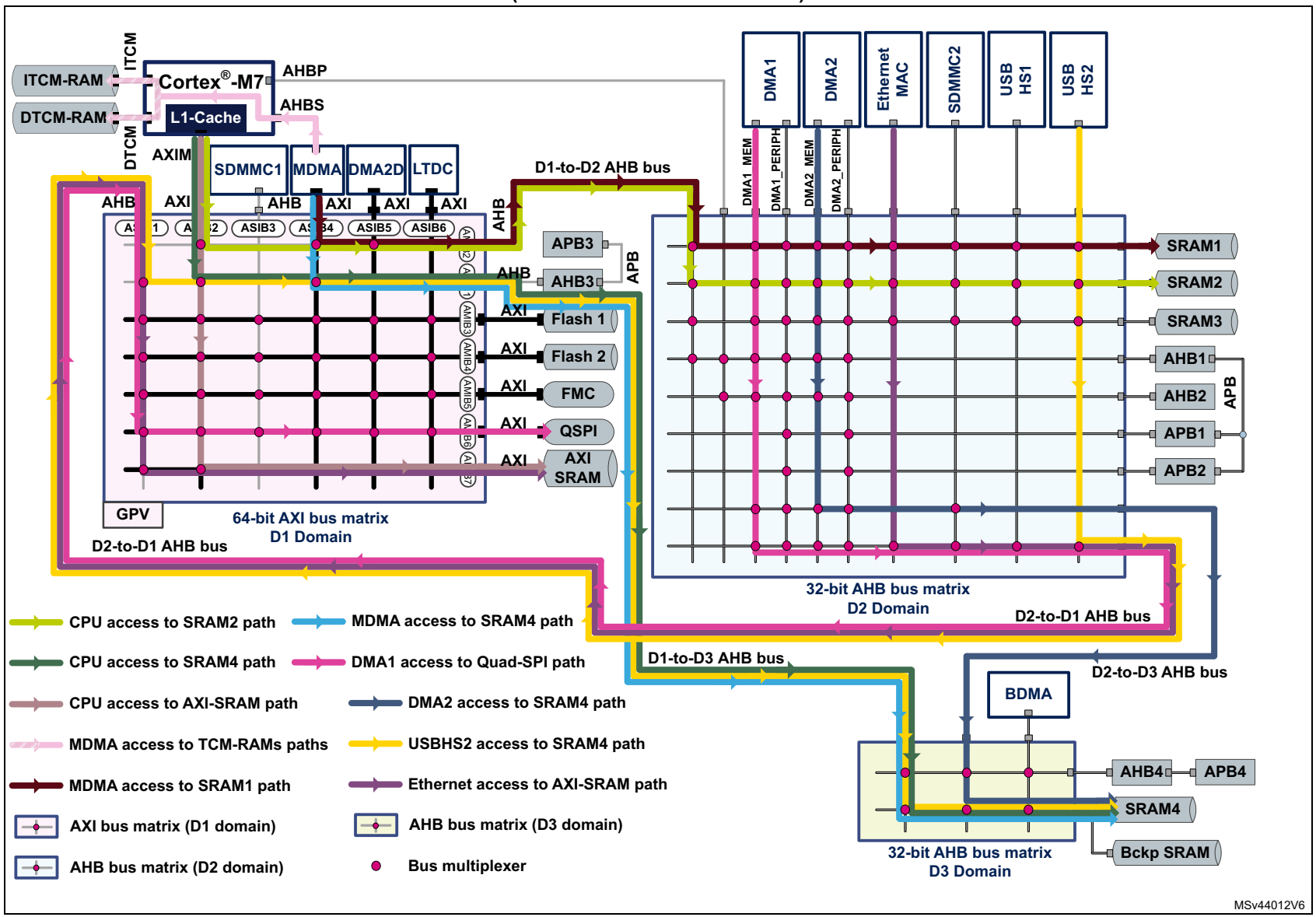
	Bus master / type[1]																			
	Cortex®-M7 - AXIM	Cortex®-M7 - AHBP	**Cortex®-M7 - ITCM**	**Cortex®-M7 -DTCM**	SDMMC1	**MDMA - AXI**	MDMA - AHBS	**DMA2D**	**LTDC[2]**	DMA1 - MEM	DMA1 - PERIPH		DMA2 - MEM	DMA2 - PERIPH		Ethernet MAC - AHB	SDMMC2 - AHB	USBHS1 - AHB	USBHS2 - AHB[5]	BDMA - AHB
Bus slave / type[1]	**Interconnect path and type[3]**																			
ITCM	-	-	**D**	-	-	-	7	-	-	-	-	-	-	-	-	-	-	-	-	-
DTCM	-	-	-	**D**	-	-	7	-	-	-	-	-	-	-	-	-	-	-	-	-
AHB3 peripherals	1	-	-	-	-	1	-	-	-	21	21	-	21	21	-	21	21	21	21	-
APB3 peripherals	14	-	-	-	-	14	-	-	-	214	214	-	214	214	-	214	214	214	214	-
Flash bank 1	**1**	-	-	-	*1*	**1**	-	**1**	**1**	*21*	*21*	-	*21*	*21*	-	*21*	*21*	*21*	*21*	-
Flash bank 2[4]	**1**	-	-	-	*1*	**1**	-	**1**	**1**	*21*	*21*	-	*21*	*21*	-	*21*	*21*	*21*	*21*	-
AXI SRAM	**1**	-	-	-	*1*	**1**	-	**1**	**1**	*21*	*21*	-	*21*	*21*	-	*21*	*21*	*21*	*21*	-
QUADSPI[5]	**1**	-	-	-	*1*	**1**	-	**1**	**1**	*21*	*21*	-	*21*	*21*	-	*21*	*21*	*21*	*21*	-
RAM shared between ITCM and AXI[6]	**1**	-	-	-	*1*	**1**	-	**1**	**1**	*21*	*21*	-	*21*	*21*	-	*21*	*21*	*21*	*21*	-
OCTOSPI[6]	**1**	-	-	-	*1*	**1**	-	**1**	**1**	*21*	*21*	-	*21*	*21*	-	*21*	*21*	*21*	*21*	-
FMC	**1**	-	-	-	*1*	**1**	-	**1**	**1**	*21*	*21*	-	*21*	*21*	-	*21*	*21*	*21*	*21*	-
SRAM 1	12	-	-	-	-	12	-	12	-	2	2	-	2	2	-	2	2	2	2	-
SRAM 2	12	-	-	-	-	12	-	12	-	2	2	-	2	2	-	2	2	2	2	-
SRAM 3[7]	12	-	-	-	-	12	-	12	-	2	2	-	2	2	-	2	2	2	2	-
AHB1 peripherals	12	2	-	-	-	12	-	12	-	2	2	-	2	2	-	-	-	-	-	-
APB1 peripherals	125	25	-	-	-	125	-	125	-	25	25	2	25	25	2	-	-	-	-	-
AHB2 peripherals	-	2	-	-	-	-	-	-	-	2	2	-	2	2	-	-	-	-	-	-
APB2 peripherals	125	25	-	-	-	125	-	125	-	25	25	2	25	25	2	-	-	-	-	-
AHB4 peripherals	13	-	-	-	-	13	-	-	-	23	23	-	23	23	-	-	23 (8)	213 (8)	213	3
APB4 peripherals	136	-	-	-	-	136	-	-	-	236	236	-	236	236	-	-	23 (8)	213 (8)	213	36
SRAM4	13	-	-	-	-	13	-	-	-	23	23	-	23	23	-	-	23 (8)	213 (8)	213	3
Backup RAM	13	-	-	-	-	13	-	-	-	23	23	-	23	23	-	-	23 (8)	213 (8)	213	3

1. **Bold** font type denotes 64-bit bus. Plain type denotes 32-bit bus.
2. LTDC is not available on STM32H742x devices.

3. Cells in the table body indicate access possibility, utility, path, and type:
   Access possibility and utility:
   Any figure = access possible, "-" = access not possible, gray shading = access useful/usable
   Access path:
   D = direct, 1 = via AXI bus matrix, 2 = via AHB bus matrix in D2, 3 = via AHB bus matrix in D3, 4 = via AHB/APB bridge in D1, 5 = via AHB/APB bridge in D2, 6 = via AHB/APB bridge in D3, 7 = via AHBS bus of Cortex®-M7,
   Multidigit numbers = interconnect path goes through more than one matrix or/and bridge, in the order of the digits.
   Access type:
   Plain = 32-bit, *Italic*=32-bit on bus master end / 64-bit on bus slave end, **Bold**=64-bit
4. Flash bank 2 is available only in STM32H74x/75x except for STM32H750x devices.
5. QUADSPI and USBHS2 are available only in STM32H74x/75x devices.
6. Available only in STM32H72x/73x.
7. SRAM3 is available only in STM32H74x/75x except STM32H742x.
8. Connection available only in STM32H74x/75x.

Figure 3 shows some paths (10 examples of master access paths) used by some masters located in the D1 and D2 domains to access to resources located in the D1, D2, and D3 domains. This example is based on STM32H74x/75x. However, the paths are the same for STM32H72x/73x except for the USBHS2 access to SRAM4 path.

Figure 3. Examples of D1/D2 master accesses to memories in D1, D2, and D3 domains (STM32H74x and STM32H75x)



2.5 STM32H72x/73x/74x/75x memories

The STM32H74x/75x devices (except for STM32H750x) embed two independent flash memory banks of up to 1 Mbyte each. The STM32H750x feature one single flash memory bank of 128 Kbytes while the STM32H72x/73x embed one bank of up to 1 Mbyte.

All devices feature an embedded scattered SRAM of different sizes, and external memory interfaces such as FMC, quad-SPI (STM32H74x/75x) or octo-SPI (STM32H72x/73x). The scattered architecture configuration gives the flexibility to partition the memory resources following the application requirements in order to obtain the right performance trade-off versus application code size, data size, and power saving.

2.5.1 Embedded flash memory

The flash memory is accessible for read or/and write accesses through the AXI bus.

On dual-bank devices, the banks have their own AMIB connection to the AXI bus matrix, which enables simultaneous operations: two read/program/erase operations can be executed in parallel on the two banks. This also avoids contention when two masters have access to the flash memory at the same time since the two masters can access different banks. The two banks are contiguous in term of address location for devices with a 2-Mbyte flash memory. The start address of bank 2 is just after the end address of bank 1. This enables the Cortex®-M7 core to use the whole flash memory as a single block of 2 Mbytes to store code. The STM32H74x/75x devices, which feature 1 Mbyte of flash memory organized in two banks, do not support this configuration. Therefore, the two banks are not contiguous. The STM32H72x/73x devices feature one single bank.

STM32H72x/73x devices feature one single bank.

The flash memory is organized as 256-bit. It is accessible in read and write with 256-bit wide access for instructions and data, and 10 bits for error code correction (ECC).

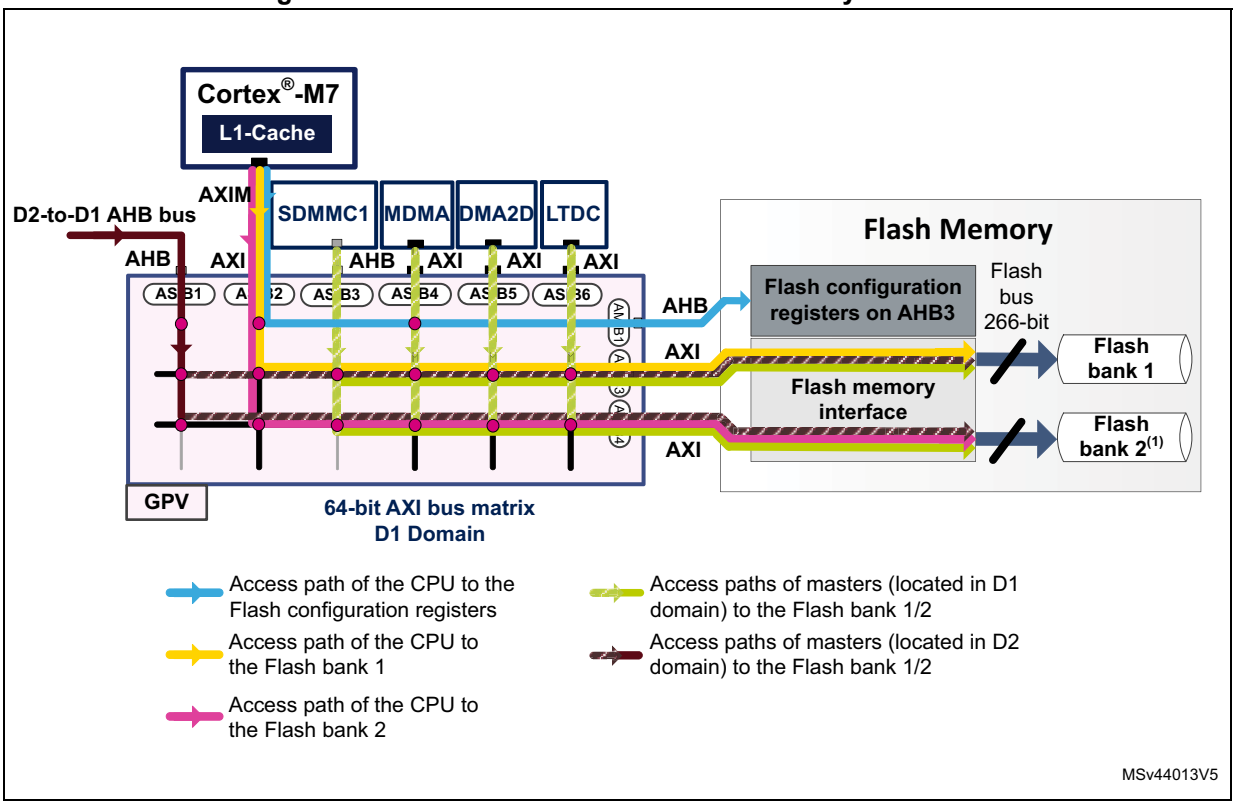
For control, configuration and status register accesses, the flash memory interface is accessible through the AXIM/AHB3 path, which is a 32-bit AHB bus (refer to the blue path in *Figure 4*).

The STM32H7 devices are the first STM32 microcontrollers that embed a flash memory able to run at up to 70 MHz. The number of wait states is consequently decreased with respect to the previous STM32 devices working at the same frequency. This enhancement enables decreasing latency and, as a result, increasing system performance. For example, if the AXI bus runs at 200 MHz, the number of wait states is equal to two with the V_{CORE} range configured to VOS1 level. If the AXI bus runs at 240 MHz, the number of wait states is equal to four with the V_{CORE} range configured to VOS0 level (refer to RM0433 and RM0468 reference manuals for more details on V_{CORE} ranges).

Flash bank 1 (the first bank) is accessed starting from address 0x0800 0000 while flash bank 2 (the second bank), when available, is accessed starting from address 0x0810 0000. The instruction or/and data caches must be enabled to achieve a zero-wait-state-like access of the CPU to the flash memory.

Figure 3 shows the different access paths of different masters to the flash memory. The MDMA can also have access to the flash configuration registers.

Figure 4. STM32H72x/73x/74x/75x flash memory accesses



1. Bank 2 is available only in STM32H74x/75x (except for STM32H750x).

2.5.2 Embedded RAM

The STM32H74x/75x devices feature a large internal RAM of up to 1060 Kbytes (including 4 Kbytes of backup SRAM) divided into up to seven blocks and scattered in three domains, while the STM32H72x/73x feature an internal RAM of up to 564 Kbytes (including 4 Kbytes of backup SRAM) divided into up to seven blocks and scattered in three domains.

RAMs located in the D1 domain

The RAMs located in the D1 domain are the following:

- 64 Kbytes of instruction RAM (ITCM-RAM)

  The ITCM-RAM is mapped at address 0x0000 0000.

  It can only be accessed by the Cortex®-M7 core and the MDMA (even in Sleep mode). It is accessible by the CPU through the ITCM bus (the bus colored in purple in *Figure 2*) and by the MDMA through the specific AHBS bus of the core (light pink path in *Figure 3*). It is accessible by the Cortex®-M7 by bytes, half-words (16 bits), words (32 bits) or double words (64 bits). The ITCM-RAM can be accessed at the maximum CPU clock speed without latency and with zero wait states. To perform accesses at a

CPU speed higher than 520 MHz, the STM32H72x/73x must disable the ECC on this RAM through the CPUFREQ_BOOST option byte.

For STM32H72x/73x the ITCM-RAM size can be increased up to 192 Kbytes with a 64 Kbyte granularity at the expense of AXI-SRAM size.

- 128 Kbytes of DTCM-RAM

  The DTCM-RAM is located in the D1 domain.

  It is split into two DTCM-RAMs with 32-bit access each. Both memories are connected respectively to the D0TCM and D1TCM ports of the Cortex®-M7 (not represented in the figures) and can be used in parallel (for load/store operations) thanks to the Cortex®-M7 dual issue capability. The DTCM-RAM is mapped on the DTCM interface at address 0x2000 0000. It is accessible only by the CPU and the MDMA and can be accessed by the Cortex®-M7 through the DTCM bus (light green bus in *Figure 2*) and by the MDMA through the specific AHBS bus of the Cortex®-M7 (light pink path in *Figure 3*). It is accessible by the Cortex®-M7 by bytes, half-words (16 bits), words (32 bits) or double words (64 bits).

  The DTCM-RAM is accessible at the maximum Cortex®-M7 clock speed without any latency. To perform accesses at a CPU speed higher than 520 MHz, the STM32H72x/73x must disable the ECC on this RAM through the CPUFREQ_BOOST option byte.

  Concurrent accesses to the DTCM-RAM by the Cortex®-M7 and the MDMA as well as their priorities can be handled by the slave control register of the Cortex®-M7 itself (CM7_AHBSCR register). A higher priority can be given to the Cortex®-M7 to access the DTCM-RAM versus the MDMA. For more details of this register, refer to *Arm® Cortex®-M7 processor - technical reference manual*.

- Up to 512 Kbyte of AXI SRAM

  The AXI SRAM is mapped at address 0x2400 0000.

  It can be accessed by all masters located only in D1 domain and D2 domain through the D2-to-D1 AHB interdomain bus. The BDMA located in the D3 domain cannot access this memory. The AXI SRAM is connected to the AXI bus matrix through a 64-bit wide AXI bus and can be accessed as bytes (8 bits), half-words (16 bits), full-words (32 bits) or double-words (64 bits). Refer to *Figure 3* for some possible AXI SRAM accesses. The AXI SRAM can be used for read/write data storage as well as for code execution. It is accessed at the same frequency as the AXI bus matrix (half the maximum CPU frequency).

- 192 Kbytes of RAM shared between ITCM and AXI RAM (available only in STM32H72x/73x)

  It is located in the D1 domain and can be shared between instruction TCM or AXI SRAM with a 64-Kbyte granularity.

  When used as ITCM-RAM, it is accessed as the 64-Kbyte ITCM-RAM, except that it is mapped at address 0x0001 0000, contiguous to the fixed ITCM-RAM.

  When used as AXI SRAM, it is accessed as the 128-Kbyte AXI SRAM, except that it is mapped at address 0x2402 0000 contiguous to the fixed AXI SRAM.

  This feature can be configured through the TCM_AXI_SHARED[1:0] option byte.

RAMs located in the D2 domain

The AHB SRAM1, SRAM2, and SRAM3 can be accessed by all masters located only in the D1 domain and the D2 domain through the D1-to-D2 AHB interdomain bus. They are accessible by bytes, half-words (16 bits) or words (32 bits). Refer to *Figure 3* for an illustration of some possible SRAM1 and SRAM2 accesses. They can be used for read/write data storage as well as code execution. These memories are accessed at the AHB bus matrix frequency (half the maximum CPU frequency). Each memory has its own AHB bus that connects it to the AHB bus matrix. This enables removing bus contention and keeping high system performance when it is concurrently accessed by more than one master.

Below the D2 SRAM base addresses:

- STM32H74x/75x
  - The AHB SRAM1 of up to 128 Kbytes is mapped at address 0x3000 0000.
  - The AHB SRAM2 of up to 128 Kbytes is mapped at address 0x3002 0000.
  - The AHB SRAM3 of 32 Kbytes is mapped at address 0x3004 0000. This memory is not available in STM32H742x devices.
- For STM32H72x/73x:
  - The AHB SRAM1 of up to 16 Kbytes is mapped at address 0x3000 0000.
  - The AHB SRAM2 of up to 16 Kbytes is mapped at address 0x3000 4000.

The above SRAMs can be used by local DMAs (located in the D2 domain) as buffers for data from/to peripherals in that domain. Data can be transferred to the D1 domain at the end of a transfer using MDMA for Cortex®-M7 high-speed processing.

Note: After reset, SRAM1, SRAM2, and SRAM3 clocks are disabled thus they are not accessible. To enable their clocks, the bits SRAM1EN, SRAM2EN, and SRAM3EN must be set in the register RCC_AHB2ENR respectively for SRAM1, SRAM2 and SRAM3.

RAMs located in the D3 domain

The RAMs located in the D3 domain are the following:

- The AHB SRAM4 is accessible by:
  - The Cortex®-M7 and MDMA located in the D1 domain through the D1-to-D3 AHB interdomain bus.
  - Any master located in the D2 domain through the D2-to-D3 AHB interdomain bus or through AHB interdomain buses D2-to-D1 and D1-to-D2. The combination depends on the master accessing SRAM4.
  - BDMA through the AHB bus matrix located in the D3 domain.

  SRAM4 is mapped at address 0x3800 0000. It can be accessed by bytes, half-words (16 bits) or words (32 bits). Refer to *Figure 3* for an illustration of some possible SRAM4 accesses. SRAM4 can be used for read/write data storage as well as for code execution or for retaining data while the D1 and D2 domains enter DStandby mode. SRAM4 is accessed at the AHB bus matrix frequency (half the maximum CPU frequency).
- The backup SRAM of 4 Kbytes accessible by most of the system masters at address 0x3880 0000 (all masters except SDMMC1, DMA2D, LTDC, Ethernet, and USB OTG_HS for STM32H72x/73x). It can be accessed by bytes, half-words (16 bits) or words (32 bits). It can be considered as an internal EEPROM when VBAT is always present (refer to RM0433 and RM0468 reference manuals for the way to use this memory).

All internal RAMs feature error correction code (ECC). The ECC must be disabled on TCM-RAMs, through CPUFREQ_BOOST option byte, when STM32H73x/74x devices are running above 520 MHz.

Table 5 and *Table 6* summarize the internal memory mapping and the memory sizes of the STM32H72x/73x/74x/75x devices:

Table 5. Internal memory summary of the STM32H72x and STM32H73x

Memory type	Memory region	Address start	Size	Access interfaces	Domain	Maximum frequency
Flash memory	FLASH-1	0x0800 0000	1 Mbyte(1)	AXI (64 bits)	D1	275 MHz
RAM	DTCM-RAM	0x2000 0000	128 Kbytes	DTCM (64 bits)	D1	550 MHz
	ITCM-RAM	0x0000 0000	(2)	ITCM (64 bits)	D1	550 MHz
	AXI SRAM	0x2400 0000	(2)	AXI (64 bits)	D1	275 MHz
	SRAM1	0x3000 0000	16 Kbytes	AHB (32 bits)	D2	275 MHz
	SRAM2	0x3000 4000	16 Kbytes	AHB (32 bits)	D2	275 MHz
	SRAM4	0x3800 0000	64 Kbytes	AHB (32 bits)	D3	275 MHz
	Backup SRAM	0x3880 0000	4 Kbytes	AHB (32 bits)	D3	275 MHz

1. Up to 1 Mbytes depending on the device
2. Refer to table *ITCM/DTCM/AXI configuration* from RM0468 reference manuals.

Table 6. Internal memory summary of the STM32H74x and STM32H75x

Memory type	Memory region	Address start	Size	Access interfaces	Domain	Maximum frequency
Flash memory	FLASH-1	0x0800 0000	1 Mbyte(1)	AXI (64 bits)	D1	240 MHz
	FLASH-2(2)	0x0810 0000	1 Mbyte(1)	AXI (64 bits)	D1	240 MHz
RAM	DTCM-RAM	0x2000 0000	128 Kbytes	DTCM (64 bits)	D1	480 MHz
	ITCM-RAM	0x0000 0000	64 Kbytes	ITCM (64 bits)	D1	480 MHz
	AXI SRAM	0x2400 0000	512 Kbytes(3)	AXI (64 bits)	D1	240 MHz
	SRAM1	0x3000 0000	128 Kbytes(4)	AHB (32 bits)	D2	240 MHz
	SRAM2	0x3002 0000	128 Kbytes(4)	AHB (32 bits)	D2	240 MHz
	SRAM3(5)	0x3004 0000	32 Kbytes	AHB (32 bits)	D2	240 MHz
	SRAM4	0x3800 0000	64 Kbytes	AHB (32 bits)	D3	240 MHz
	Backup SRAM	0x3880 0000	4 Kbytes	AHB (32 bits)	D3	240 MHz

1. Up to 1 Mbyte depending on the device
2. Not available in STM32H750x devices
3. Up to 512 Kbytes depending on the device
4. Up to 128 Kbytes depending on the device
5. Not available in STM32H742x devices

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 STM32H72x/73x/74x/75x memories with the flexible memory controller (FMC) and the quad-SPI controller (STM32H74x/75x) or octo-SPI interface (STM32H72x/73x).

The external memory space is divided into fixed-size banks of 256 Mbytes each.

STM32H72x and STM32H73x external memories

Four external memory banks are dedicated to the FMC (plus one bank for SDRAM remapping) and two to the octo-SPI controller:

- Bank 1 is used for NOR/PSRAM memories.
- Bank 2 is used for OCTOSPI2 memory.
- Bank 3 is used for NAND memory.
- Bank 4 is used for OCTOSPI1 memory.
- Banks 5 and 6 are used for the two SDRAM banks.

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

The octo-SPI interface is a specialized communication interface targeting single, dual, quad, or octal SPI flash memories. It can extend the internal flash memory by 256 Mbytes for each octo-SPI interface, and can be used to store data such as images in graphical applications or to store the user application. On STM32H73x, the data stored can be encrypted before being written. In that case the OTFDEC performs an on-the-fly decryption while reading data (refer to the application note *How to use OTFDEC for encryption/decryption in trusted environment on STM32 MCUs (AN5281)*).

Figure 5 summarizes the memory mapping of the external memories, their respective address ranges after reset and the possible remapping.

Figure 5. STM32H72x and STM32H73x external memory mapping

STM32H74x and STM32H75x external memories

Four external memory banks are dedicated to the FMC (plus one bank for SDRAM remapping) and one dedicated to the QUADSPI:

- Bank 1 is used for NOR/PSRAM memories.
- Bank 2 is used for SDRAM bank remapping.
- Bank 3 is used for NAND memory.
- Bank 4 is used for quad-SPI memory.
- Banks 5 and 6 are used for the two SDRAM banks.

The FMC bank mapping can be modified through the BMAP[1:0] bits in the FMC_BCR1 register. The BMAP bank memory-mapping bits allow three configurable options:

- Default mapping
- Remapping of SDRAM bank2, thus allowing to access the SDRAM banks at two different address mappings
- Swapping of the NOR/PSRAM bank with SDRAM banks

The quad-SPI interface is a specialized communication interface targeting single, dual or quad SPI flash memories. It can extend the internal flash memory by 256 Mbytes and can be used to store data such as images in graphical applications or to contain code to execute the user application.

Figure 6 summarizes the memory mapping of the external memories, their respective address ranges after reset and their possible remapping.

Figure 6. STM32H74x and STM32H75x external memory mapping

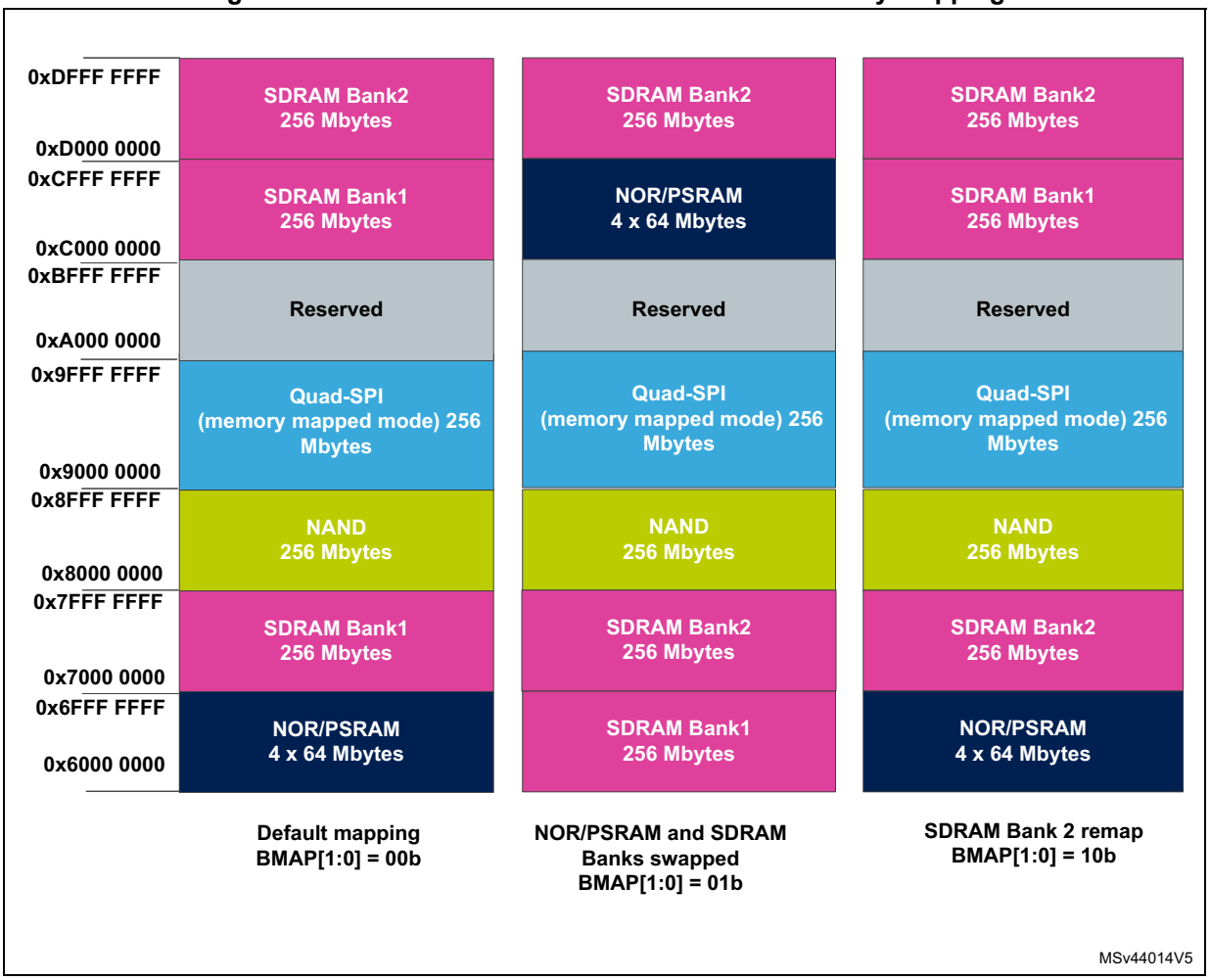


Figure 7 shows the possible paths that interconnect the Cortex®-M7 and the different DMAs with these external memories via the AXI and the AHB buses. The example covers STM32H74x/75x. However, it also applies to STM32H72x/73x by replacing the quad-SPI interface by octo-SPI. As shown in *Figure 7*, the external memories can benefit from the Cortex®-M7 cache and therefore maximize the performance whatever their usage (data storage or code execution). This enables combining high performance and large memory size.

The path highlighted in pink in *Figure 7* represents the connection between the external memories and the Cortex®-M7, by means of the FMC.

The path highlighted in yellow represents the connection between the external memories and the Cortex®-M7, by means of the quad-SPI controller.

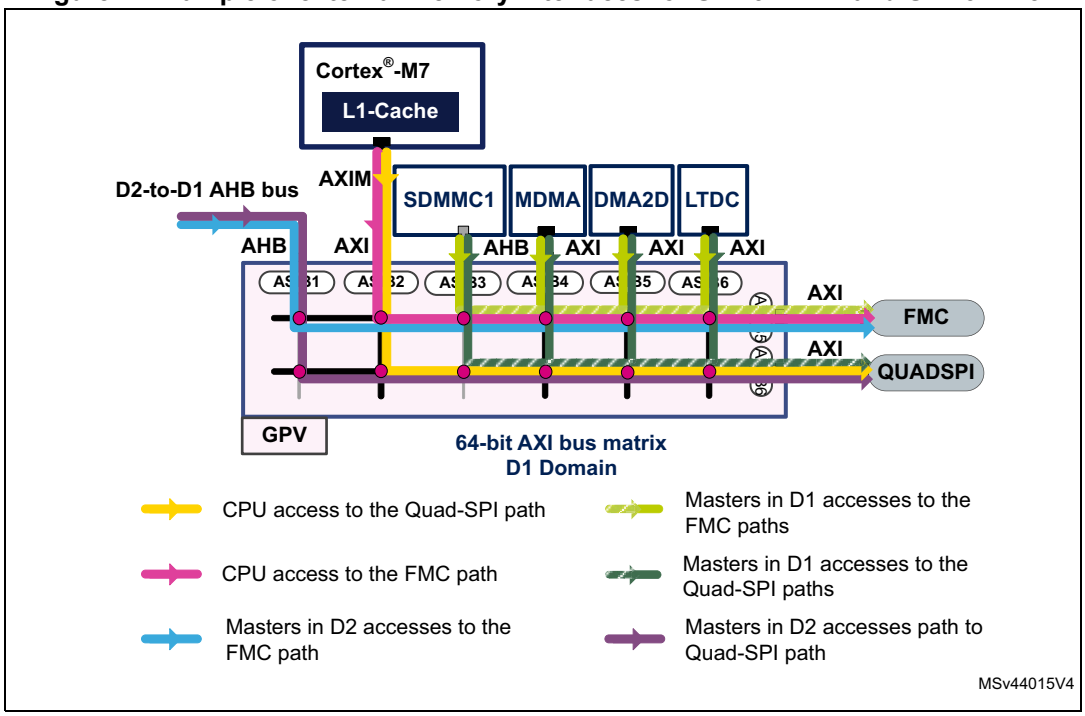
The path highlighted in light green represents the access paths of some masters located in the D1 domain to external memories connected by means of the FMC controller.

The path highlighted in dark green represents the access path of some masters located in the D1 domain to external memories connected by means of the quad-SPI controller.

The paths highlighted in purple and in blue represent respectively the accesses of masters located in the D2 domain to external memories connected by means of the quad-SPI controller and the FMC controller that are located in D1 domain.

All external memories are accessible by any master available in the system except for the BDMA.

Figure 7. Example of external memory interfaces for STM32H74x and STM32H75x



Flexible memory controller interface (FMC)

The STM32H72x/73x/74x/75x FMC controller makes the interface with the memory-mapped devices including SRAMs, ROMs, NOR/NAND flash memories and SDRAM devices. It is used either for program execution (except for NAND flash memory) or for R/W data storage.

The FMC main features are the following:

- 4 banks to support different memories at the same time
- Independent chip select control for each memory bank
- Independent configuration for each memory bank
- Programmable timings to support a wide range of devices
- 8/16/32-bit data bus
- External asynchronous wait control
- Interfaces with synchronous DRAM (SDRAM) that provides two SDRAM banks.
- Possible independent clock source (independent PLL) for more clock flexibility.
- Synchronous memories can be accessed at a maximum frequency of the kernel clock divided by 2 or 3.

All the FMC external memories share the same addresses, data, and control signals.

Each external device is accessed with a unique chip select. The FMC performs only one access at a time to an external device.

The two default regions of SDRAM banks are not cacheable. Even if the cache is enabled, the data or instructions do not go through the cache. To benefit from cache acceleration, the SDRAM banks can be remapped from 0xC000 0000 and 0xD000 0000 to 0x6000 0000 and 0x7000 0000, respectively, which are, by default, cacheable regions (for STM32H72x/73x, only SDRAM bank1 can be remapped from 0xC000 0000 to 0x6000 0000). This is done by setting the field BMAP[1:0] bits in the FMC_BCR1 register. *Figure 6* shows the different external memory mappings and their corresponding BMAP[1:0] values.

If the remapping is not suitable for the application, the Cortex®-M7 MPU can be used to modify the propriety of the default SDRAM memory region to be cacheable.

All the external memories that are connected to the FMC benefit from the L1-cache for data and instructions (pink and yellow paths in *Figure 7*), allowing larger data, or code sizes with maximum performance.

Quad-SPI memory interface (QUADSPI)

The STM32H74x/75x devices embed a quad-SPI memory interface (QUADSPI), which is a specialized communication interface targeting single, dual, or quad-SPI flash memories. This multiple-width interface supports a traditional SPI single bit serial input and output as well as two-bit and four-bit serial commands. In addition, the interface supports double data rate (DDR) read commands, meaning that the address transfer and data read are performed on both edge of the communication clock. It enables multiplying by a factor of two the data/instruction throughput and therefore increasing the access efficiency to external quad-SPI flash memories.

The QUADSPI works in one of the following three modes:

- Indirect mode: all operations are performed through the QUADSPI registers.
- Status polling mode: the external flash memory status register is periodically read and an interrupt is generated in case of flag setting.
- Memory mapped mode: the external flash is memory mapped. It is then seen by the system as an internal memory.

The STM32H74x/75x QUADSPI supports the SDR mode, which is the default mode, and the DDR mode that enables double data throughput. The DDR mode boosts data load and instruction fetch performances. It is also useful when the QUADSPI clock must be slow, for example when the QUADSPI cannot operate at its maximum speed because of a PCB limitation.

The QUADSPI supports single-flash and dual-flash memory modes. The latter enables the microcontroller to communicate with two external memory devices at the same time. This mode is useful to double throughput and to double flash memory size. The throughput of two bytes per cycle with dual-flash memory in DDR quad-SPI mode is reachable, which allows a high performance access to these external Flash memories either for data storage or for code execution.

The QUADSPI is able to manage up to 256-Mbytes flash memory starting from 0x9000 0000 to 0x9FFF FFFF in the memory mapped mode. It is mapped on an executable area, so the remapping is not needed (remapping at the address 0x0000 0000).

Compared to the FMC, the QUADSPI enables connecting an external flash memory at a reduced cost with small packages (reducing PCB area) and a reduced GPIO usage. Six GPIOs are used in single quad-SPI mode (4 bits) for any flash memory size or 10 GPIOs in dual quad-SPI mode (8 bits).

As shown in *Figure 2*, the QUADSPI is mapped on a dedicated layer on the 64-bit AXI bus matrix and can benefit from the L1-cache. This enables executing the code and loading the data from the QUADSPI with higher performance (see yellow path in *Figure 7*). Note that QUADSPI registers are mapped on the AHB bus.

The QUADSPI can also be accessed by all the masters available in the system (see dark green, yellow, and purple paths in *Figure 7*) except for the BDMA. It is accessible by the Chrom-ART accelerator and the LCD-TFT controller, enabling an efficient data transfer, particularly images, for graphical applications where a high frame display rate is needed.

For more details on QUADSPI usage, refer to the application note *quad-SPI (QUADSPI) interface on STM32 microcontrollers* (AN4760).

Octo-SPI memory interface (OCTOSPI)

The STM32H72x/73x devices embed two octo-SPI memory interfaces (OCTOSPI), which are specialized communication interfaces targeting single, dual, quad or octo-SPI flash memories. These multiple-width interfaces support a traditional SPI single-bit serial input and output as well as two-bit, four-bit, and eight-bit serial commands. In addition, OCTOSPIs support double data rate (DDR) read commands, meaning that the address transfer and data read operations are performed on both edges of the communication clock. They enable multiplying by a factor of two the data/instruction throughput and therefore increasing the access efficiency to external octo-SPI flash memories.

OCTOSPIs work in one of the following three modes:

- Indirect mode: all operations are performed through the OCTOSPI registers.
- Status polling mode: the external flash memory status register is periodically read and an interrupt is generated in case of flag setting.
- Memory mapped mode: the external flash memory is memory mapped. It is then seen by the system as an internal memory.

The OCTOSPIs support the following protocols, and their associated frame formats:

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

The STM32H72x/73x OCTOSPIs support the SDR mode, which is the default mode, and the DDR mode that enables double data throughput. The DDR mode boosts data load and instruction fetch performances. It is also useful when the OCTOSPI clock must be slow, for example when the OCTOSPI cannot operate at its maximum speed because of a PCB limitation.

Each OCTOSPIs can manage up to 256 Mbytes of flash memory starting from 0x9000 0000 to 0x9FFF FFFF for OCTOSPI1 and 0x7000 0000 to 0x7FFF FFFF for OCTOSPI2 in memory mapped mode. They are mapped on an executable area, so the remapping is not needed (remapping at the address 0x0000 0000).

Compared to the FMC, the OCTOSPIs enable connecting an external flash memory with a reduced cost with small packages (reducing PCB area) and a reduced GPIO usage. Ten GPIOs are used in octal mode whatever the flash memory size.

Each OCTOSPI is mapped on a dedicated layer on the 64-bit AXI bus matrix and can benefit from the L1-cache. This enables executing code and loading data from the OCTOSPI with higher performance. Note that OCTOSPI registers are mapped on the AHB bus.

The OCTOSPIs can also be accessed by all the masters available in the system except for the BDMA. They are accessible by the Chrom-ART accelerator and the LCD-TFT, enabling an efficient data transfer, particularly images, for graphical applications where a high frame display rate is needed.

For STM32H73x, all the data read from the OCTOSPIs can be decrypted on-the-fly before being sent to the AXI (refer to the application note *How to use OTFDEC for encryption/decryption in trusted environment on STM32 MCUs (AN5281)*).

2.6 Main architecture differences between STM32F7 series and, STM32H72x, STM32H73x, STM32H74x, and STM32H75x devices

Table 7 summarizes the differences between the STM32F7 series devices and, STM32H72x/73x/74x/75x devices in terms of architecture and performances (peripheral differences not included).

Table 7. Architecture differences between the STM32F7 Series and STM32H72x, STM32H73x, STM32H74x and STM32H75x devices

-	STM32F7 Series	STM32H72x and STM32H73x devices	STM32H74x and STM32H75x devices
Cortex®-M7 revision	r0P1/r1P0 (1)	r1P2	r1P1
Max CPU frequency	216 MHz	550 MHz	480 MHz
Instruction / data cache sizes	4 Kbytes to 16 Kbytes(1)	32 Kbytes	16 Kbytes
FPU	Single/double precision floating point (1)	Single and double precision floating point	
Bus matrix	AXI to AHB bridge + 1 x AHB bus matrix	1 x AXI bus matrix + 2 x AHB bus matrices + inter-domains buses	
Flash memory access	Through 32-bit or 64-bit AHB bus (2) shared with two banks (3) or through 64-bit ITCM bus	Through dedicated AXI bus (64 bit) for each bank (4) / No ITCM access	
Flash memory line width	128/256 bit(3)	256 bit for each bank for read and write	
Flash memory wait states@200 MHz	6(5)	2	
Flash memory dual-bank support	No/Yes (1)	No	Yes(6)
DTCM-RAM size	64 Kbytes or 128 Kbytes (1)	128 Kbytes	
ITCM-RAM size	16 Kbytes	64 to 256 Kbytes(7)	64 Kbytes
ITCM-RAM accessibility	CPU only	CPU + MDMA	
SRAM size	Up to 384 Kbytes	Up to 564 Kbytes	Up to 864 Kbytes
SRAM access bus	32-bit AHB	64-bit AXI (AXI SRAM) 32-bit AHB (SRAMs in D2 and D3)	
DMAs	2x General-purpose DMAs	4x General-purpose DMAs (8) that can be used with DMAMux for more flexibility	
DMA1 transfers	Only peripheral to memory/Memory to peripheral transfers are permitted	All transfers are permitted	
MPU protected area number	8	16	

1. Depending on the STM32F7 device.
2. The CPU access is 64-bit and the DMA access is 32-bit.
3. Some devices have only 128-bit access with single bank, some others have only 256-bit access with single bank, and others have 128/256-bit access following the bank mode used (single and dual bank).
4. Two read/program/erase operations can be executed in parallel on the two banks at the same time.
5. Frequency at voltage range 2.7 V - 3.6 V.
6. This feature is not available in STM32H750x devices.
7. The ITCM-RAM size can be configured through an option byte.
8. 1x high-speed general-purpose master direct memory access controller (MDMA), 2x dual-port DMAs with FIFO and 1x basic DMA.

3 Typical application

This application note provides a software example that demonstrates the performance of the STM32H74x/75x devices. The selected example is based on the FFT example provided in the CMSIS library. The *H7_single_cpu_perf* project can be used as a skeleton where the user can integrate their application.

3.1 FFT demonstration

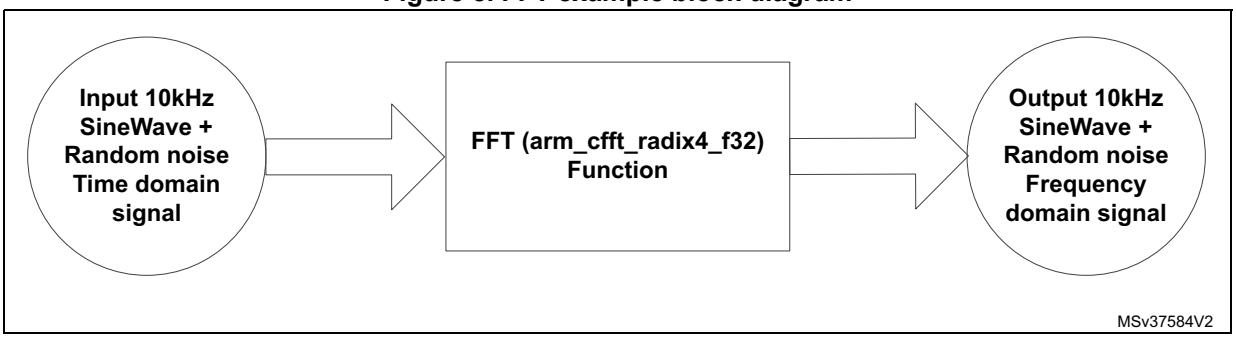
The FFT example is used as it benefits from the floating-point unit of the Cortex®-M7, contains several loops and data load/store operations, and can be accessed in different paths/memories. The code can be executed from internal or external memories.

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 8* 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 STM32H743I-EVAL board and the results (provided in nanosecond) are shown on the HyperTerminal through the UART using the virtual COM port.

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 SDRAM or quad-SPI.

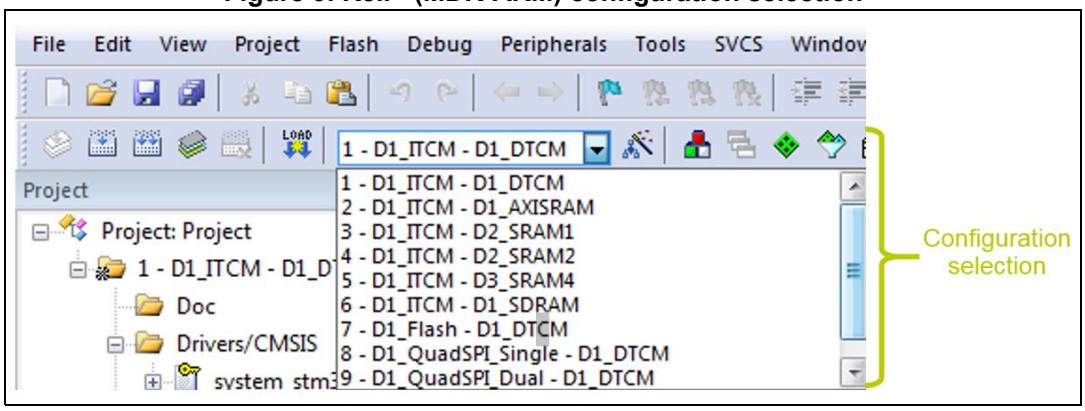
Figure 8. FFT example block diagram



3.2 Configuring demonstration projects

The CPU memory access demonstration is provided with the Keil® (MDK-ARM), IAR™ (EWARM) and system workbench for STM32 (SW4STM32) toolchains. The project is presented in 10 subproject workspaces, each one representing a configuration. Each configuration enables selecting the data and code locations. *Figure 9* shows a screenshot of different configurations of MDK-ARM toolchain.

Figure 9. Keil® (MDK-ARM) configuration selection



The configurations are named using the following rules:

N - InstructionLocation - DataLocation

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 loader and copied afterward (at the scatter 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.

The following configurations are proposed:

1 - D1_ITCM - D1_DTCM: the program is executed from the ITCM-RAM and the data storage is done in the DTCM-RAM located in the D1 domain.

2 - D1_ITCM - D1_AXISRAM: the program is executed from the ITCM-RAM and the data storage is done in the AXI SRAM located in the D1 domain with the D-cache enabled.

3 - D1_ITCM - D2_SRAM1: the program is executed from the ITCM-RAM and the data storage is done in the SRAM1 located in the D2 domain with the D-cache enabled.

4 - D1_ITCM - D2_SRAM2: the program is executed from the ITCM-RAM and the data storage is done in the SRAM2 located in the D2 domain with the D-cache enabled.

5 - D1_ITCM - D3_SRAM4: the program is executed from the ITCM-RAM and the data storage is done in the SRAM4 located in the D3 domain with the D-cache enabled.

6 - D1_ITCM - D1_SDRAM: the program is executed from the ITCM-RAM and the data storage is done in the SDRAM located in the D1 domain with the D-cache enabled.

7 - D1_Flash - D1_DTCM: the program is executed from the internal flash memory and the data storage is done in the DTCM-RAM with the I-cache and the D-cache enabled. The FFT algorithm uses huge constants. In this case, the read-only data are located in the internal Flash memory. This is the reason why the D-cache is also enabled.

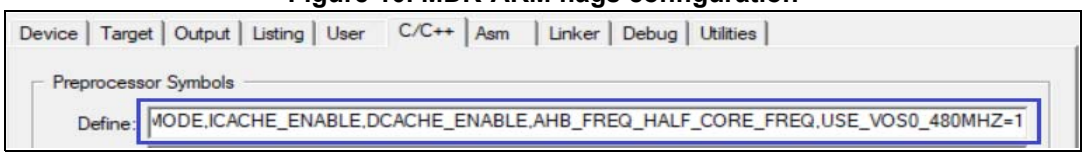
8 - D1_QuadSPI_Single - D1_DTCM: the program is executed from the quad-SPI flash memory with the I-cache and the D-cache enabled (since the constants are located in quad-SPI flash). The quad-SPI flash memory is configured in single mode and runs with the DDR mode enabled at 60 MHz if the system clock frequency is set at 480 MHz and 50 MHz if the system clock is set to 400 MHz.

9 - D1_QuadSPI_Dual - D1_DTCM: the program is executed from the quad-SPI flash memory with the I-cache and the D-cache enabled (since the constants are located in quad-SPI flash). The quad-SPI flash memory is configured in dual mode and runs with DDR mode enabled at 60 MHz if the system clock frequency is set at 480 MHz, and 50 MHz if the system clock frequency is set at 400 MHz.

10 - D1_SDRAM_Swapped - D1_DTCM: the program is executed from the FMC-SDRAM with bank 2 swapped address (0xD000 0000 -> 0x7000 0000) and the data storage is done in the DTCM-RAM. The I-cache and D-cache are enabled (constants are located in the SDRAM) and the SDRAM runs at 100 MHz.

Each configuration has its own flag set. These flags are settable on the configuration project. *Figure 10* shows where these flags are defined for the MDK-ARM toolchain.

Figure 10. MDK-ARM flags configuration



The code is optimized for time level 3 for all the configurations.

Project flags are the following:

- **USE_VOS0_480MHZ:** defines the system clock frequency.
  - If USE_VOS0_480MHZ = 1, the system clock frequency is set to 480 MHz.
  - If USE_VOS0_480MHZ = 0, the system clock frequency is set to 400 MHz.
- **AHB_FREQ_*X*_CORE_FREQ**: defines the core and bus matrices operating frequencies.

  X is equal to HALF or EQU. There are two configurations:
  - **AHB_FREQ_HALF_CORE_FREQ**: the AXI bus matrix and the two AHB bus matrices run at the half core frequency.
  - **AHB_FREQ_EQU_CORE_FREQ**: the AXI bus matrix and the two AHB bus matrices run at the same frequency as the core.

The default value is AHB_FREQ_HALF_CORE_FREQ. The core runs at twice the frequency of AXI and AHB bus matrices.

- **DCACHE_ENABLE**: if defined in the configuration project, the data cache is enabled.
- **ICACHE_ENABLE**: if defined in the configuration project, the instruction cache is enabled.
- **FLASH_WS**: this flag configures the number of wait states of the internal flash:

  FLASH_WS = FLASH_LATENCY_X, where X ranges from 0 (zero wait state) to 15 (15 wait states). The default number of wait states in the project configuration is two (X = 4).

Note: If the flag USE_VOS0_480MHZ = 0, the flash memory wait state can be equal to 2.

If the flag USE_VOS0_480MHZ = 1 and the user tries to set the flash memory wait state to a value less than four, a directive #error is raised during the compile time telling the user to increase the flash memory wait state value.

- **DATA_IN_ExtSDRAM**: if defined in the configuration project, the SDRAM is configured and ready to be used either for data storage or for code execution.
- **SDRAM_MEM_BUS_WIDTH**: this flag configures the width of the external SDRAM bus as follows:

  SDRAM_MEM_BUS_WIDTH = FMC_SDRAM_MEM_BUS_WIDTH_X with X being 8, 16 or 32.
- **SDRAM_ADDRESS_SWAPPED**: if defined in the configuration project, the address of SDRAM bank 2 is remapped from 0xD000 0000 to 0x7000 0000. This remapping relocates the SDRAM in cacheable and executable memory.

Note: If this flag is removed, the SDRAM bank 2 address is set to its default address 0xD000 0000. The MPU is used in system_stm32h7xx.c file to configure that region as cacheable and executable. In such a case, the SDRAM address must be adapted accordingly in the corresponding linker file to avoid a hard fault exception, that is, replace 0x70000000 by 0xD0000000.

- **DATA_IN_QSPI**: if defined in the configuration project, the quad-SPI flash memory is configured and ready to be used for code/data location.
- **QSPI_DUAL_FLASH**: if defined in the configuration project, the quad-SPI flash memory is configured in dual mode. In this mode, the data bus is 8-bit wide.
- **QSPI_CLK_PRESCALER**: this flag defines the QUADSPI clock prescaler. It is configured as follows:

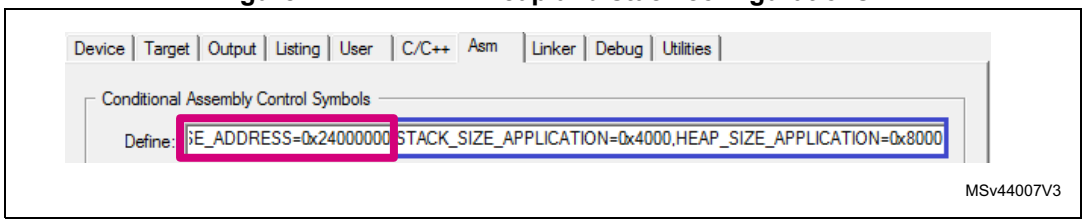
  QSPI_CLK_PRESCALER = X, where X = 0 to 255.
- **QSPI_DDRMODE:** if defined in the configuration project, the QUADSPI is configured in DDR mode.
- **QSPI_INSRUCTION_1_LINE, QSPI_INSRUCTION_4_LINES**: the first flag is used to configure the quad-SPI flash instruction to operate in one-line mode, while the second configures the instruction to operate in four-line mode. If no flag is present, the quad-SPI instruction is configured in one-line mode.
- **QSPI_XIP_MODE**: if defined in the configuration project, the QUADSPI is configured in XIP mode: only the instruction is sent first. Note that XIP mode has almost no effect on the performance when the 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 scatter files (stack and heap regions), the user must modify accordingly the stack and heap sizes in the **ASM** menu of the MDK-ARM toolchain. The size of the region in the scatter file is not considered as the real stack size of the main application. The user must modify the STACK_SIZE_APPLICATION and HEAP_SIZE_APPLICATION flag values to force their values in line with the heap/stack size regions configured in the scatter file.*

Figure 11 shows where to modify these flags (blue framed). There is also an initial stack pointer that is used when external memories are used for data storage. Its size is 1 Kbyte and can be changed by modifying the *Stack_Size_Init* variable in the startup file. The initial stack pointer base address can be configured in the **ASM** (assembly control symbols) menu as shown in *Figure 11* (see red frame).

Figure 11. MDK-ARM heap and stack configurations



The scatter files of different configurations are located under *MDK-ARM\scatter_files* path in the project of the demonstration.

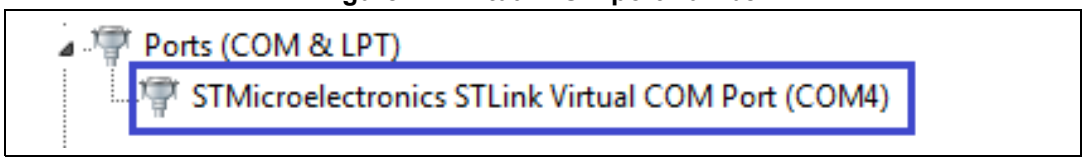
For the IAR™ (EWARM) toolchain, the linkers are located under *EWARM\icf_files*. For the system workbench toolchain, the linkers are located under *SW4STM32\<project folder configuration>*.

The results can be displayed on a HyperTerminal PC application through the UART using the virtual COM port with the following configuration:

- Baud rate: 115200
- Data bits: 7 bits
- Stop bits: 1 bit
- Parity: Odd
- Hardware flow control: none

To know which COM number the board is using, the user must connect the board ST-LINK to the PC through a USB cable and go to **Control Panel** > **System** > **Device Manager** > **Ports (COM & LPT)**. *Figure 12* shows an example where the UART COM number is COM4.

Figure 12. Virtual COM port number



4 Benchmark results and analysis

This section explains each feature activation (I-cache and D-cache) following the configuration used and presents the corresponding results obtained (time spent by the FFT algorithm in nanosecond).

The results are obtained with the Keil® MDK-ARM (v5.27.1) toolchain, the STM32H7xx Pack version 2.2.0 and the STM32CubeH7 MCU Package version 1.4.0.

The MDK-ARM code optimization configuration is set to level 3 (optimized for time).

If the instruction fetch is done through the AXIM bus of the CPU, the I-cache must be enabled to increase the performance of the code execution.

If the data is fetched through the AXIM bus of the CPU, the D-cache must be enabled to increase the performance of the data access to memories and remove their latencies.

If the code is not located in the ITCM-RAM, the I-cache must be enabled as for the following configurations:

- 7 - D1_Flash - D1_DTCM
- 8 - D1_QuadSPI_Single - D1_DTCM
- 9 - D1_QuadSPI_Dual - D1_DTCM
- 10 - D1_SDRAM_Swapped - D1_DTCM

In these configurations, the instructions are fetched from the AXIM bus of the CPU.

The D-cache must be enabled for all configurations that do not have the read-write data located in the DTCM-RAM as for the following configurations:

- 2 - D1_ITCM - D1_AXISRAM
- 3 - D1_ITCM - D2_SRAM1
- 4 - D1_ITCM - D2_SRAM2
- 5 - D1_ITCM - D3_SRAM4
- 6 - D1_ITCM - D1_SDRAM

The D-cache must also be enabled when the read-only data are not located in the ITCM-RAM as for the following configurations:

- 7 - D1_Flash - D1_DTCM
- 8 - D1_QuadSPI_Single - D1_DTCM
- 9 - D1_QuadSPI_Dual - D1_DTCM
- 10 - D1_SDRAM_Swapped - D1_DTCM

In the case of the FFT algorithm used in the demonstration, a very large amount of read-only data is used. Disabling the data cache for configurations 7, 8, 9, and 10 drastically decreases performances.

For configurations 6 and 10, the SDRAM is swapped (remapped from 0xD000 0000 to 0x7000 0000 in order to allow cache usage and that memory region to be executable) since the default MPU attribute region starting from 0xA000 0000 to 0xDFFF FFFF is neither cacheable nor executable as it is a Device memory type region.

4.1 Benchmark results

The results are obtained with the STM32H743I-EVAL board. In the provided projects, the Arm® Cortex®-M7 and the bus matrices can run respectively at:

- 480 MHz and 240 MHz (VOS0: only for silicon revision higher than rev Y, flash wait state=4)
- 240 MHz and 240 MHz (VOS0: only for silicon revision higher than rev Y, flash wait state=4)
- 400 MHz and 200 MHz (VOS1: all silicon revisions, flash wait state=2)
- 200 MHz and 200 MHz (VOS1: all silicon revisions, flash wait state=2).

Note: The performance results provided in this document apply to all STM32H742x, STM32H743x, STM32H750x and STM32H753x devices.

4.1.1 Effects of data and instructions locations on performance

Table 8 and *Table 9* show the FFT demonstration performance results obtained with the MDK-ARM in each configuration and the Cortex®-M7 running at 480 MHz. These tables are available only for silicon revisions higher than rev. Y.

Table 10 and *Table 11* show the same type of results with the CPU running at 240 MHz (CPU and bus matrices running at the same frequency). These tables are available only for silicon revisions > rev. Y.

Table 12 and *Table 13* show the FFT demonstration performance results obtained with the MDK-ARM in each configuration with the Cortex®-M7 running at 400 MHz. These tables are available for all silicon revisions.

Table 14 and *Table 15* show the same type of results with the CPU running at 200 MHz (CPU and bus matrices running at the same frequency). These tables are available for all silicon revisions.

In *Table 8*, *Table 10*, *Table 12* and *Table 14*, the code and the read-only (RO) data location are fixed in ITCM-RAM while the read/write (R/W) data location is varied for each configuration. Memories (internal/external) used for R/W data are located in different domains.

In *Table 9*, *Table 11*, *Table 13* and *Table 15*, the R/W data location is fixed in DTCM-RAM and the code and the RO data location is varied for each configuration.

Table 8. MDK-ARM results of data storage for different memory locations (execution location fixed in ITCM-RAM), CPU running at 480 MHz (AHB_FREQ_HALF_CORE_FREQ, USE_VOS0_480MHZ = 1, flash ws = 4)

Cache configuration	Configuration	Execution time (ns)(1)	Relative ratio
-	1 - D1_ITCM - D1_DTCM (reference)	260327	1.00
D-cache ON	2 - D1_ITCM - D1_AXISRAM	266345	1.02
D-cache ON	3 - D1_ITCM - D2_SRAM1	270612	1.04
D-cache ON	4 - D1_ITCM - D2_SRAM2	271689	1.04
D-cache ON	5 - D1_ITCM - D3_SRAM4	271693	1.04
D-cache ON	6 - D1_ITCM - D1_SDRAM	300495	1.15

1. The execution-time values may vary from one tool-chain version to another.

Table 9. MDK-ARM results of execution in different memory locations (data location fixed in DTCM-RAM), CPU running at 480 MHz (AHB_FREQ_HALF_CORE_FREQ, USE_VOS0_480MHZ = 1, flash ws = 4)

Cache configuration	Configuration	Execution time in ns(1)	Relative ratio
-	1 - D1_ITCM - D1_DTCM (reference)	260327	1.00
I-cache + D-cache ON	7 - D1_Flash - D1_DTCM	276135	1.06
I-cache + D-cache ON	8 - D1_QuadSPI_Single - D1_DTCM	459691	1.77
I-cache + D-cache ON	9 - D1_QuadSPI_Dual - D1_DTCM	366166	1.41
I-cache + D-cache ON	10 - D1_SDRAM_Swapped - D1_DTCM	298218	1.15

1. The execution-time values may vary from one tool-chain version to another.

Table 10. MDK-ARM results of data storage in different memory locations (execution location fixed in ITCM-RAM) CPU running at 240 MHz (AHB_FREQ_EQU_CORE_FREQ, USE_VOS0_480MHZ = 1, flash ws = 4)

Cache configuration	Configuration	Execution time in ns(1)	Relative ratio
-	1 - D1_ITCM - D1_DTCM (reference)	520650	1.00
D-cache ON	2 - D1_ITCM - D1_AXISRAM	528150	1.01
D-cache ON	3 - D1_ITCM - D2_SRAM1	531033	1.02
D-cache ON	4 - D1_ITCM - D2_SRAM2	532091	1.02
D-cache ON	5 - D1_ITCM - D3_SRAM4	532100	1.02
D-cache ON	6 - D1_ITCM - D1_SDRAM	560283	1.08

1. The execution time values may vary from one tool-chain version to another.

Table 11. MDK-ARM results of execution in different memory locations (data location fixed in DTCM-RAM) CPU running at 240 MHz (AHB_FREQ_EQU_CORE_FREQ, USE_VOS0_480MHZ = 1, flash ws = 4)

Cache configuration	Configuration	Execution time in ns(1)	Relative ratio
-	1 - D1_ITCM - D1_DTCM (reference)	520650	1.00
I-cache + D-cache ON	7 - D1_Flash - D1_DTCM	542854	1.04
I-cache + D-cache ON	8 - D1_QuadSPI_Single - D1_DTCM	670491	1.29
I-cache + D-cache ON	9 - D1_QuadSPI_Dual - D1_DTCM	611062	1.17
I-cache + D-cache ON	10 - D1_SDRAM_Swapped - D1_DTCM	550070	1.06

1. The execution-time values may vary from one tool-chain version to another.

Table 12. MDK-ARM results of data storage in different memory locations (execution location fixed in ITCM-RAM) CPU running at 400 MHz (AHB_FREQ_HALF_CORE_FREQ, USE_VOS0_480MHZ = 0, flash ws = 2)

Cache configuration	Configuration	Execution time in ns (1)	Relative ratio
-	1 - D1_ITCM - D1_DTCM (reference)	312382	1.00
D-cache ON	2 - D1_ITCM - D1_AXISRAM	319605	1.02
D-cache ON	3 - D1_ITCM - D2_SRAM1	324735	1.04
D-cache ON	4 - D1_ITCM - D2_SRAM2	326027	1.04
D-cache ON	5 - D1_ITCM - D3_SRAM4	326032	1.04
D-cache ON	6 - D1_ITCM - D1_SDRAM	352642	1.13

1. The execution-time values may vary from one tool-chain version to another.

Table 13. MDK-ARM results of data storage in different memory locations (execution location fixed in ITCM-RAM) CPU running at 400 MHz (AHB_FREQ_HALF_CORE_FREQ, USE_VOS0_480MHZ = 0, flash ws = 2)

Cache configuration	Configuration	Execution time in ns (1)	Relative ratio
-	1 - D1_ITCM - D1_DTCM (reference)	312382	1.00
I-cache + D-cache ON	7 - D1_Flash - D1_DTCM	327817	1.05
I-cache + D-cache ON	8 - D1_QuadSPI_Single - D1_DTCM	592282	1.90
I-cache + D-cache ON	9 - D1_QuadSPI_Dual - D1_DTCM	473580	1.52
I-cache + D-cache ON	10 - D1_SDRAM_Swapped - D1_DTCM	345840	1.11

1. The execution-time values may vary from one tool-chain version to another.

Table 14. MDK-ARM results of data storage in different memory locations (execution location fixed in ITCM-RAM) CPU running at 200 MHz (AHB_FREQ_EQU_CORE_FREQ, USE_VOS0_480MHZ = 0, flash ws = 2)

Cache configuration	Configuration	Execution time in ns (1)	Relative ratio
-	1 - D1_ITCM - D1_DTCM (reference)	624765	1.00
D-cache ON	2 - D1_ITCM - D1_AXISRAM	633775	1,01
D-cache ON	3 - D1_ITCM - D2_SRAM1	637235	1,02
D-cache ON	4 - D1_ITCM - D2_SRAM2	638505	1,02
D-cache ON	5 - D1_ITCM - D3_SRAM4	638515	1,02
D-cache ON	6 - D1_ITCM - D1_SDRAM	664720	1,06

1. The execution-time values may vary from one tool-chain version to another.

Table 15. MDK-ARM results of execution in different memory locations (data location fixed in DTCM-RAM) CPU running at 200 MHz ((AHB_FREQ_EQU_CORE_FREQ, USE_VOS0_480MHZ = 0, flash ws = 2)

Cache configuration	Configuration	Execution time in ns (1)	Relative ratio
-	1 - D1_ITCM - D1_DTCM (reference)	624765	1.00
I-cache + D-cache ON	7 - D1_Flash - D1_DTCM	647915	1.04
I-cache + D-cache ON	8 - D1_QuadSPI_Single - D1_DTCM	821015	1.31
I-cache + D-cache ON	9 - D1_QuadSPI_Dual - D1_DTCM	748810	1.20
I-cache + D-cache ON	10 - D1_SDRAM_Swapped - D1_DTCM	654390	1.05

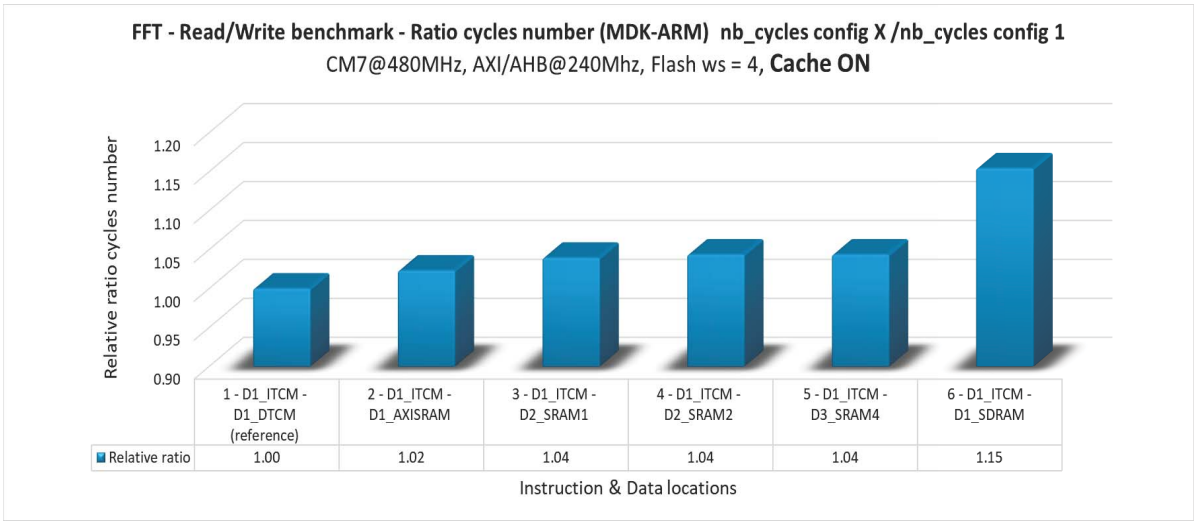
1. The execution time values may vary from one tool-chain version to another.

$$\text{Relative ratio} = \text{execution\_time\_config\_X} / \text{execution\_time\_config\_1}$$

The relative ratio calculation enables comparing the performance of a given configuration versus the configuration having the best performance (1 - D1_ITCM - D1_DTCM) as well as comparing the performance of a given configuration with another one.

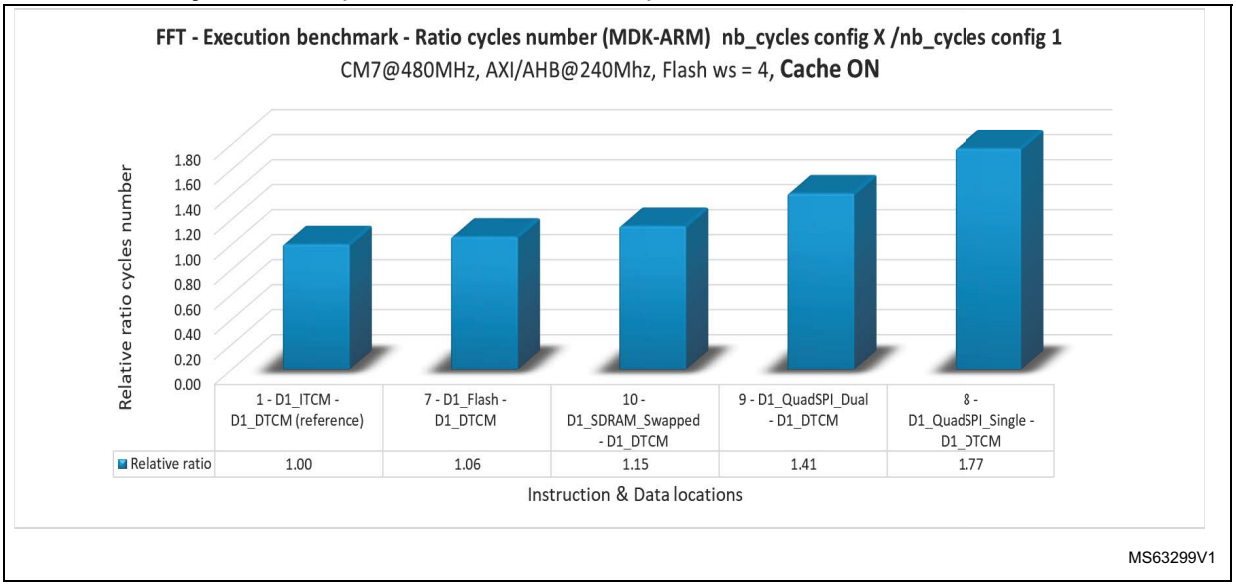
The chart in *Figure 13* shows the relative ratio of each configuration versus configuration 1 that represents the reference of this benchmark. This chart shows the FFT benchmark of data storage in different memory locations while the code location is fixed in ITCM-RAM with a CPU running at 480 MHz.

Figure 13. STM32H74x and STM32H75x FFT benchmark: data storage in different memory locations (code in ITCM-RAM) at 480 MHz, with MDK-ARM toolchain



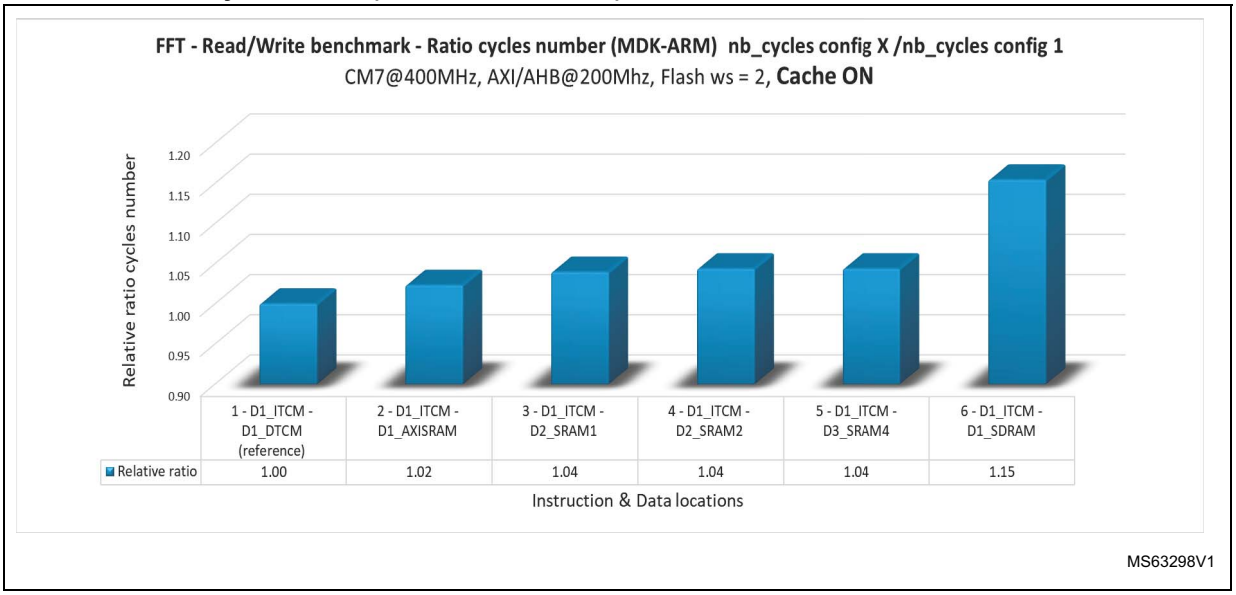
The chart in *Figure 14* shows the relative ratio of each configuration versus configuration 1 (D1_ITCM - D1_DTCM). It represents the FFT benchmark of the code execution from different memory locations while the data storage location is fixed in the DTCM-RAM with the CPU running at 480 MHz.

Figure 14. STM32H74x and STM32H75x FFT benchmark: code execution from different memory locations (R/W data in DTCM-RAM) at 480 MHz, with MDK-ARM toolchain



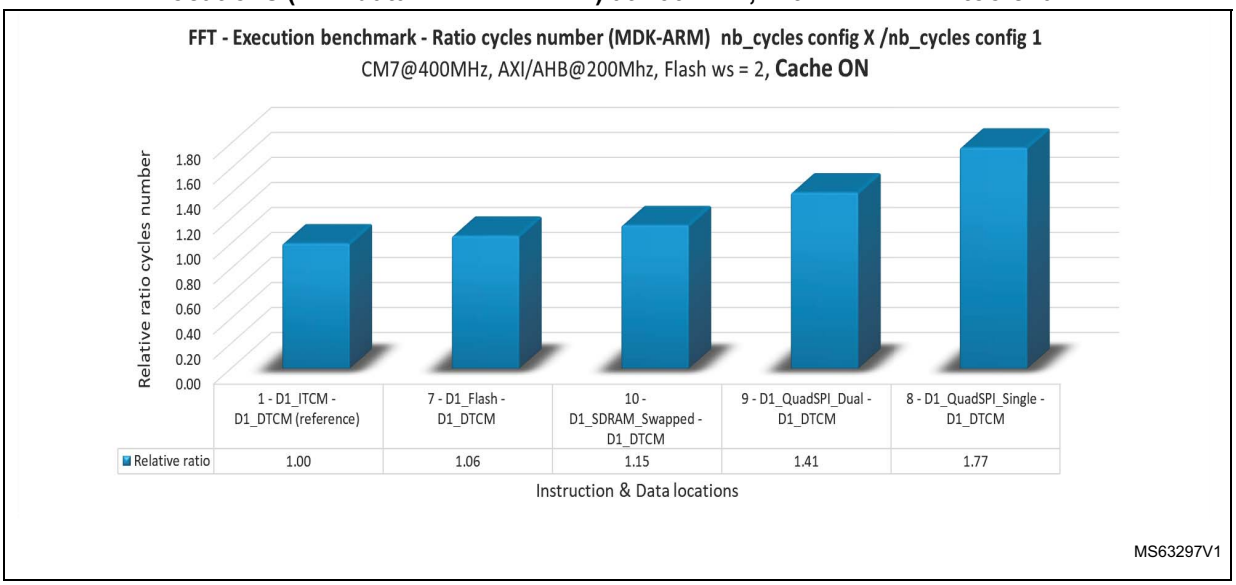
The chart in *Figure 15* shows the relative ratio of each configuration versus configuration 1 that represents the reference of this benchmark. This chart shows the FFT benchmark of data storage in different memory locations while the code location is fixed in ITCM-RAM with CPU running at 400 MHz.

Figure 15. STM32H74x and STM32H75x FFT benchmark: data storage in different memory locations (code in ITCM-RAM) at 400 MHz, with MDK-ARM toolchain



The chart in *Figure 16* shows the relative ratio of each configuration versus configuration 1 (D1_ITCM - D1_DTCM). It represents the FFT benchmark of the code execution from different memory locations while the data storage location is fixed in the DTCM-RAM with the CPU running at 400 MHz.

Figure 16. STM32H74x and STM32H75x FFT benchmark: code execution from different memory locations (R/W data in DTCM-RAM) at 400 MHz, with MDK-ARM toolchain



4.1.2 Impact of basic parameters on performance

Some basic parameters can impact the performance:

- Number of flash memory wait states
- SDRAM parameters such as bus width, clock frequency or remapping

Number of flash memory wait states

This section describes the performance impact resulting from the number of flash memory wait states.

The results below are obtained with the configuration 7 - D1_Flash - D1_DTCM with FLASH_WS parameter set in the IDE preprocessor.

Example: FLASH_WS=FLASH_LATENCY_4 sets the number of flash wait states to 4.

Table 16 summarizes the results of the wait state impacts on the performance with CPU and AXI running at 480 MHz and 240 MHz respectively.

Table 17 summarizes the results of the wait state impacts on the performance with CPU and AXI running at 400 MHz and 200 MHz respectively.

Table 16. Number of flash wait states vs performance (MDK-ARM)/CPU running at 480 MHz/AXI running at 240 MHz (VOS0)

7 - D1_Flash - D1_DTCM			
CPU frequency = 480 MHz, AXI frequency = 240MHz (VOS0)	**Execution time (ns)**	**Relative ratio**	**Decrease (%)**
Flash ws = 4 (reference)	276135	1,00	-
Flash ws = 5	277627	1,01	0,54
Flash ws = 6	279177	1,01	1,10
Flash ws = 7	280685	1,02	1,65
Flash ws = 8	282247	1,02	2,21
Flash ws = 9	283737	1,03	2,75
Flash ws = 10	285195	1,03	3,28

Table 17. Number of flash wait states vs performance (MDK-ARM) /CPU running at 400 MHz/AXI running at 200 MHz (VOS1)

7 - D1_Flash - D1_DTCM			
CPU frequency = 400 MHz, AXI frequency = 200 MHz (VOS1)	**Execution time (ns)**	**Relative ratio**	**Decrease (%)**
Flash ws = 2 (reference)	327817	1,00	-
Flash ws = 3	329597	1,01	0,54
Flash ws = 4	331397	1,01	1,09
Flash ws = 5	333187	1,02	1,64
Flash ws = 6	335107	1,02	2,22
Flash ws = 7	336857	1,03	2,76
Flash ws = 8	338622	1,03	3,30

The above results (*Table 16* and *Table 17*) show a decrease of performance of about 0.54 % when the number of flash wait states is incremented by 1.

SDRAM parameters and configuration

This section focuses on the performance impact of some of the SDRAM parameters such as the SDRAM bus width, its clock frequency, as well as the remapping of the SDRAM (swapped and nonswapped banks). By default SDRAM mapping is set to nonswapped banks configuration (refer to *Figure 6*), the used SDRAM is mapped at address 0xD000 0000 which is in neither a cacheable nor an executable region. To use this configuration, remove SDRAM_ADDRESS_SWAPPED definition from the IDE preprocessor. For more details on SDRAM_ADDRESS_SWAPPED flag, refer the description in the *Section 3.2*.

Table 18 provides the results obtained with a 6 - D1_ITCM - D1_SDRAM configuration by adjusting the SDRAM bus width and its clock frequency.

Table 18. SDRAM data read/write access performance vs bus width and clock frequency based on 6 - D1_ITCM - D1_SDRAM configuration

6 - D1_ITCM - D1_SDRAM	Execution time in ns	Decrease (%)
Bus width 32 bit / SDRAM clock = 100MHz	300495	-
Bus width 32 bit / SDRAM clock = 66.7MHz	322895	7,45
Bus width 16 bit / SDRAM clock = 100MHz	321995	7,15
Bus width 16 bit / SDRAM clock = 66.7MHz	352914	17,44
Bus width 8 bit / SDRAM clock = 100MHz	364193	21,20
Bus width 8 bit / SDRAM clock = 66.7MHz	419377	39,56

The same benchmark has been done for 10 - D1_SDRAM_Swapped - D1_DTCM configuration. The results are provided in *Table 19*.

Table 19. Execution performance from SDRAM versus bus width and clock frequency based on 10 - D1_SDRAM_Swapped - D1_DTCM configuration

10 - D1_SDRAM_Swapped - D1_DTCM	Execution time in ns	Decrease (%)
Bus width 32 bit / SDRAM clock = 100MHz	298218	-
Bus width 32 bit / SDRAM clock = 66.7MHz	315235	5,71
Bus width 16 bit / SDRAM clock = 100MHz	320314	7,41
Bus width 16 bit / SDRAM clock = 66.7MHz	352239	18,11
Bus width 8 bit / SDRAM clock = 100MHz	377681	26,65
Bus width 8 bit / SDRAM clock = 66.7MHz	447493	50,06

Table 20 provides the SDRAM data read/write access results for SDRAM swapped and nonswapped configurations based on the 6 - D1_ITCM - D1_SDRAM configuration.

Table 20. SDRAM data read/write access performance in swapped and non-swapped bank configurations based on 6 - D1_ITCM - D1_SDRAM configuration

6 - D1_ITCM - D1_SDRAM	Execution time in ns	Decrease (%)
1 - D1_ITCM - D1_DTCM (reference)	260327	-
SDRAM swapped 32 bit / SDRAM clock = 100MHz	300495	15,43
SDRAM Non-swapped 32 bit / SDRAM clock = 100MHz	300520	15,44

Table 21 provides the results of the code execution from SDRAM in SDRAM swapped and nonswapped configurations using the 10 - D1_SDRAM_Swapped - D1_DTCM configuration.

Table 21. Execution performance from SDRAM in swapped and non-swapped bank configurations based on 10 - D1_SDRAM_Swapped - D1_DTCM configuration

10 - D1_SDRAM_Swapped - D1_DTCM	Execution time in ns	Decrease (%)
1 - D1_ITCM - D1_DTCM (reference)	260327	-
SDRAM swapped 32 bit / SDRAM clock = 100MHz	298218	14,56
SDRAM Non-swapped 32 bit / SDRAM clock = 100MHz	298116	14,52

4.2 Result analysis

In the case of the data storage benchmark in different memory locations (*Figure 13* and *Figure 15*), the code location is fixed in the ITCM-RAM. This has the advantage that the code is executed with real zero-wait-state access. In this way, any memory latency effect of the code execution is excluded. This enables benchmarking only R/W data accesses in several memory locations excluding interferences of instruction fetches.

In the case of the code execution benchmark in different memory locations (*Figure 14* and *Figure 16*), the data location is fixed in the DTCM-RAM. This has the advantage that access to data is performed without any latency. In this way, any effect of the memory latency on the R/W data is excluded. Such a configuration enables benchmarking only instruction fetches in several memory locations without data access interference.

To illustrate the real performance of the different memory interfaces versus Cortex®-M7 accesses, the relative ratio is also calculated for the Cortex®-M7 core running at 240 MHz (same frequency as bus matrices as shown in *Table 14* and *Table 15*) for different configurations.

Cache usage is recommended to achieve the highest performance of the product. It can be used for all memory accesses, either internal or external memories. Only TCM-RAMs do not require cache usage. The cache hides the latency introduced by the bridges and the interconnections. This enables having almost the same level of performance for data storage in the DTCM-RAM and code execution in the ITCM-RAM.

For highest data storage access performance by the Cortex®-M7, the best location of R/W data is DTCM-RAM, since it is accessed without latency with deterministic accesses (no cache maintenance is performed). DTCM-RAM runs at the same frequency as the Cortex®-M7 (up to 480 MHz). The other RAM locations, either internal or external memories and located in different domains, do not provide deterministic access when cache is enabled while still showing similar performance access as DTCM-RAM (refer to *Figure 15*).

The best location for code execution, in terms of performance, is ITCM-RAM, which is accessed without any latency in a deterministic way, and runs at the same frequency as the Cortex®-M7 core. A similar level of performance is obtained when code execution is performed from the internal flash memory, even at 480 MHz. This is due to the cache size implemented in STM32H74x/75x and to the low number of wait states of the internal flash memory (refer to *Figure 16*).

It is advised to use the quad-SPI flash in dual-flash memory mode to enhance performance. If the FFT algorithm is used, a 20% performance increase is obtained by using the dual-flash mode versus the single-flash mode.

For SDRAM, a similar performance is obtained for code execution compared to execution from the internal flash memory with cache enabled.

5 Software memory partitioning and tips

This section provides some tips about code and data partitioning in STM32H72x/73x/74x/75x memory in order to get the best trade-off between performance and code/data sizes. Recommendations about product optimal configuration and issue avoidance are also provided.

5.1 Software memory partitioning

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

Therefore, the ITCM-RAM 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. STM32H72x/73x ITCM-RAM size is configurable and can thus be adapted to the application requirements (see *Section 2.6*).

The DTCM-RAM 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 are located, for example, in AXI SRAM in the D1 domain, or SRAM1 in the D2 domain, or SRAM4 in the D3 domain, MDMA can be used to move the data from these memories to the DTCM-RAM in order to be processed at the CPU clock speed.

In real-time applications that use RTOS, the heap is generally massively used. If the DTCM-RAM size (128 Kbytes) is large enough for the application, the reasonable DTCM-RAM scattering is 16 Kbytes for the stack, 92 Kbytes for the heap, and 20 Kbytes for the global variables. The user can scatter the DTCM-RAM to fit her/his application needs but she/he must give higher priority to place the stack in DTCM-RAM with respect to heap and global variables. In a bare-metal application model, the heap is less used and the DTCM-RAM is scattered between stack and global variables only.

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-RAM.

The AXI SRAM located in the D1 domain can be reserved for a graphic frame buffer in the graphical applications using QVGA TFTs in 16-bit mode that need a large amount of graphic data and high display performance, ensured by the LCD-TFT and the DMA2D DMAs. To achieve this, STM32H72x/73x shared SRAM must be configured as an AXI memory. This memory can also be used for data storage when no available space is left in DTCM-RAM. 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.

SRAM1, SRAM2, and SRAM3, located in the D2 domain, can be used as buffers to store in/out data of peripherals located in the D2 domain such as Ethernet and USB. These data can be buffers and descriptors or I2S audio frames or others.

Audio data frames can be processed in DTCM-RAM by copying them from SRAM1, SRAM2, or SRAM3 to DTCM-RAM using MDMA. These memories can also be used for global variables since the CPU can access them via the D1-to-D2 interdomain bus with the usage of the data cache.

The SRAM4 located in the D3 domain is generally used to store data for low-power part of the user application. It can be used to retain some application data when D1 and D2 enter DStandby mode. The low-power application data can be R/W data for the CPU or buffers to be transferred by peripherals (located in the D3 domain) such as LPUART1, I2C4, and others.

Note: The data cache needs to be cleared before switching domain D1 to standby. Clearing the data cache avoids losing data transferred to SRAM4 for retention.

SRAM4 remains available as long as all the system is not in Standby mode. Otherwise, it is still possible to use the backup SRAM to retain some data with a battery connected to VBAT. However, the backup SRAM size is optimized to 4 Kbytes in order to reduce leakage (refer to RM0433 and RM0468 reference manuals), that provides a low-power application example and describes the usage of the D3 domain in low-power mode applications.

SRAM4 can also be used for CPU regular data storage as an internal memory extension for non-low-power applications.

When the application needs more memory and when its code, data, or both, do not fit in the internal memories, the external memories can be used to expand the memory size without a loss of performance.

For example, an external NOR flash memory up to 64 Mbytes connected through the FMC can contain the application instructions with the cache enabled.

In case of a lack of internal RAMs, the data storage can be achieved in an external SRAM or in an SDRAM through the FMC interface while enabling the data cache. These memories can contain either frame buffers for graphical applications or noncritical data. At the same time, more priority is given for very critical data to be placed in the DTCM-RAM.

Quad-SPI and octo-SPI flash memories can be used to store read-only data (relatively huge image or audio files) with keeping almost the same level of performance as an internal flash memory access by enabling the data cache.

Quad-SPI and octo-SPI flash memories can also be used to store the application code in the memory-mapped mode up to 256 Mbytes and at the same time to save several GPIOs in smaller STM32H72x/73x/74x/75x device packages, compared to parallel flash memories that have to be connected to the FMC interface. In that case, when the CPU accesses regularly to read-only data, the latter can be mapped in the internal flash memory. If the application needs more memory space and more execution performance, the user can load her/his application in the quad-SPI/octo-SPI (load region) and use an external SDRAM, where the application is copied (at the scatter load phase) and executed (execution region).

5.2 Recommendations and tips

Enabling the cache allows the best performance to be reached for all memory accesses except for TCM-RAMs for which there is no effect on Cortex®-M7 access performance.

Whenever possible, if the data is located in SRAM1, SRAM2 or SRAM3, the user can copy it into the DTCM-RAM using the MDMA so that it is processed afterward by the CPU with more determinism and better performance (computation at the CPU clock speed).

When an external memory is used for the data storage of the application and it is not mapped in a cacheable region, simply use the memory remapping when the relocation to a cacheable region is possible, which is the case for SDRAM banks (setting BMAP[1:0] field in the FMC_BCR1 register) or use the MPU to configure the memory MPU attributes as cacheable region.

When an external memory is used to store application code, care must be taken if this memory is mapped in a region having the default attribute, Execute-Never (XN). In that case the user has to modify the memory into an executable region using the MPU, otherwise a HardFault exception occurs. This is the case of the SDRAM bank regions after reset (configurations 6 and 10 described in *Section 3.2*). Refer to *Table 3* for Cortex®-M7 default executable regions.

In the case of concurrent accesses of MDMA and Cortex®-M7 to the DTCM-RAM, the application speed can decrease if the Cortex®-M7 does not have the highest priority access. This priority can be managed by software using the CM7_AHBSCR register in the critical code section that loads/stores data by the Cortex®-M7 in the DTCM-RAM.

Using the QUADSPI in dual-flash memory mode enhances the performance versus the single-flash mode configuration. It is advised to choose this mode whenever possible.

After reset, the SRAMs located in the D2 domain are disabled by default. To avoid HardFault exception, they must be enabled by programming the RCC_AHB2ENR register before the CPU can have access to them.

6 Conclusion

STM32H72x/73x/74x/75x devices are extending the range of ST high-performance 32-bit microcontrollers as a continuity of the STM32F7 series.

The STM32H72x/73x/74x/75x, versus their predecessors, bring higher performance with an optimized power consumption, thanks to their improved architecture, their embedded L1-cache (instruction and data caches), the Cortex®-M7 core that can run at up to 550 MHz, and their 40 nm manufacturing technology.

The number of embedded flash memory wait states is decreased with respect to the STM32F7 series working at the same frequency. This considerably increases performance and reduces latency.

The internal memory sizes are also increased to better serve ever more memory consuming applications and, as a result, eliminate traditional constraints related to resources on application development, and accelerate time-to-market for new products.

In addition, the internal memories have a more scattered architecture than in STM32F7 series devices, thus offering more flexibility to the user to place the code and data with the lowest CPU access latencies.

The benchmarking and the results provided in this application note show that, whatever the memory location of code and data, the performance remains similar either with internal or external memories.

7 Revision history

Table 22. Document revision history

Date	Revision	Changes
15-Jun-2017	1	Initial release.
24-Apr-2019	2	Replaced: – All the STM32H7x3 reference with STM32H74x and STM32H75x where needed. – Clock frequencies from 200 to 240 MHz and 400 to 480MHz where applicable. – The CPU acronym with Cortex®-M7 throughout the document. Updated: – SRAM bus frequency 200MHz to 240MHz throughout the document – *: Introduction* with the current performance figures. – *Section 2.2: Cortex®-M7 system caches*; with the supported series. – *Section 2.3.1: AXI bus interface*; removed the “None share” term. – *Figure 2.4.1: AXI bus matrix in the D1 domain*: The AXI memory support description changes to “up to 512 Kbytes” – *Section 2.4.3: Inter-domain buses*: Changed the Basic DMA definition. – *Section 2.5.1: Embedded Flash memory*: added the term “up to of 1 Mbyte” and added the exception for the STM32H750, changed the Flash addressing process, AXI wait state definition has been enhanced. – *Section 2.5.2: Embedded RAM*: term “up to” added in front 1060Kbytes to the internal RAM definition, CPU clock speed changed to 480MHz, AXI SRAM capacity definition updated with the term “up to”, AXI bus matrix frequency changed to 240MHz, AHB SRAM 1 & 2 capacity definition changed by adding the term “up to”, for the AHB SRAM3 added the exception to STM32H742xx devices, – *Section 2.5: STM32H74x and STM32H75x memories*: added the term “up to of 1 Mbyte” – *Section 2.5.3: External memories*: Synchronous memory access frequency definition changed to 110Mhz. – *Table 6: Architecture differences between the STM32F7 Series and, STM32H74x and STM32H75xdevices*: updated table with foot note and SRAM information. – *Section 5.1: Software memory partitioning*: updated text with “bare-metal applications model”, changed the wording on the sentence “These memory modules can also be used for global variables since the CPU can access them via D1-to-D2 inter-domain bus with the usage of the data cache.”, note changed to support the following devices STM32H742/743/753 and STM32H750 together with the Reference Manual details – *Section 4.1: Results*: added the supported devices. Added: – *Figure 1: STM32H74x and STM32H75x system architecture*: notes 1 & 2. – *Table 4: STM32H74x and STM32H75x bus-master-to-bus-slave possible interconnections*: foot note 3 & 4. – *Table 5: Internal memory summary of the STM32H74x and STM32H75x devices*; Added Foot note (1), (2), (3) & (4), updated supported frequencies 240 & 480MHz.

Table 22. Document revision history (continued)

Date	Revision	Changes
16-Jul-2019	3	Updated: – *Introduction*: stm32h7x3_cpu_perf replaced with H7_single_cpu_perf. – *Section 3: Typical application*: stm32h7x3_cpu_perf replaced with H7_single_cpu_perf. – *Section 3.2: Configuring demonstration projects* Added: – *Section 4.1.1: Effects of data and instructions locations on performance* – *Figure 8* through *Figure 12* in *Section 3: Typical application*. – *Figure 13* through *Figure 15* in *Section 4: Benchmark results and analysis* – *Section 4.1.2: Impact of basic parameters on performance*
16-Sep-2020	4	Updated document to cover STM32H72x and STM32H73x microcontrollers. Added reference to RM0468 as well as *Section : Reference documents*. Removed Chrom-Art Accelerator trademark. Changed “X-CUBE-PERF-H7 embedded software package” into “X-CUBE-PERF-H7 Expansion Package”. Replaced Flash A and B by Flash bank 1 and bank 2, respectively. Specified that only STM32H74x and SMT32H75x (except for STM32H750x) feature two banks. Updated *Section 2.5.1: Embedded Flash memory*. Replaced QSPI interface (QSPI) by Quad-SPI interface (QUADSPI). *Section : Flexible memory controller interface (FMC)*: changed update frequency for synchronous memories to kernel clock divided by 2 or 3. In *Section 3.2: Configuring demonstration projects*, updated note related to the modification of RAM regions in scattered files. Renamed *Section 4: Benchmark results and analysis* and *Section 4.1.2: Impact of basic parameters on performance*.
9-May-2022	5	In *Section : Introduction*, specified that X-CUBE-PERF-H7 targets only STM32H742x, STM32H743/753x and STM32H750x microcontrollers. Removed connection between Ethernet peripheral and D2-to-D1 AHB inter-domain bus in *Figure 2: STM32H74x and STM32H75x system architecture* and *Section : D2-to-D1 AHB inter-domain bus*. Removed Ethernet from *Section : D1-to-D3 AHB inter-domain bus*. In *Section : D2-to-D3 AHB inter-domain bus*, removed USB OTG_HS from the list of peripherals that do not have access to D3 resources. In *Table 4: Bus-master-to-bus-slave possible interconnections in STM32H72x/73x/74x/75x*, removed Ethernet peripheral connection to AHB4, APB4, SRAM4 and Backup SRAM. In *Figure 3: Examples of D1/D2 master accesses to memories in D1, D2, and D3 domains (STM32H74x and STM32H75x)*, removed connection between Ethernet peripheral and D2-to-D1 AHB inter-domain bus as well as connection with AXI SRAM. Removed Backup SRAM access from Ethernet in *Section : RAMs located in the D3 domain*. Updated access interface for AXI SRAM in *Table 5: Internal memory summary of the STM32H72x and STM32H73x* and *Table 6: Internal memory summary of the STM32H74x and STM32H75x*.

Table 22. Document revision history (continued)

Date	Revision	Changes
29-Aug-2022	6	Restored connection between Ethernet peripheral and D2-to-D1 AHB bus in *Figure 2: STM32H74x and STM32H75x system architecture* and *Section : D2-to-D1 AHB interdomain bus*. Added connection between SDMMC2 peripheral and D2-to-D3 AHB bus in *Figure 3: Examples of D1/D2 master accesses to memories in D1, D2, and D3 domains (STM32H74x and STM32H75x)*.
07-Jul-2025	7	Updated SDRAM bank1 remap addresses in *Section : Flexible memory controller interface (FMC)*. Minor text improvements.