AN4891
Application note

STM32H74x and STM32H75x system architecture and performance Expansion Package for STM32Cube

Introduction

The STM32H7 Series is the first series of STMicroelectronics microcontrollers in 40 nm-process technology. This technology allows STM32H7 devices to integrate large sizes of embedded Flash and SRAM memories that decrease the resource constraints typically complicating high-end embedded development. It also unleashes the performance of the core and enables ultra-fast data transfers through the system while, realizing major power savings.

The STM32H7 Series is the first series of Arm® Cortex®-M7-based 32-bit microcontrollers able to run up to 480 MHz[(a)], reaching new performance records of 1027 DMIPS and 2400 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 to industrial gateways, home automation, telecom equipment and smart consumer products, as well as to high-performance motor control, domestic appliances, and use in small devices with rich user interfaces such as smart watches..

This application note presents the global architecture of theSTM32H74x and STM32H75x 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 and STM32H75x device architecture performance in various memory partitioning configurations with different code and data locations.

This application note is provided with the X-CUBE-PERF-H7 embedded software package that 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.

a. Applies to revisions later than silicon rev Y. Up to and including rev Y revision, the device can not exceed 400 MHz.

Contents

List of tables

List of figures

1 General information

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

2 STM32H74x and STM32H75x system architecture overview

This section introduces the main architecture features of the STM32H74x and STM32H75x. *Table 1* defines the detailed product lines concerned.

Table 1. Applicable products.

Type	Products lines
Microcontroller	STM32H742, STM32H743/753 lines, STM32H750 value line

2.1 Cortex®-M7 core

The STM32H74x and STM32H75x devices are built on basis of a high-performance Arm® Cortex®-M7 32-bit RISC core operating up to 480 MHz frequency. The Cortex®-M7 core features a high-performance floating point unit (FPU). It features a double-precision floating point unit, which 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 STM32H74x and STM32H75x devices allow the definition of up to 16 MPU regions. A forward compatibility from the Cortex®-M4 to the Cortex®-M7 allows 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 allows the resolution of branches to anticipate the next branch and therefore decrease the number of cycles consumed by loop. The dual-instruction feature allows the core to execute two instructions simultaneously in order to increase instruction throughput.

2.2 Cortex®-M7 system caches

The devices embed the Cortex®-M7 with a level1 cache (L1-cache) which is split into two separated caches: the data cache (DCACHE) and the instruction cache (ICACHE) implementing a Harvard architecture bringing the best performance. These caches allow the

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

Cortex®-M7 to reach a performance of zero wait state even at high frequencies. The cache size for both instruction and data cache 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 are available in *Arm® Cortex®-M7 Processor - Technical Reference Manual*.

More details on L1-cache usage in STM32H74x and STM32H75x devices are available in the application note *Level 1 cache on STM32F7 and STM32H7 Series* (AN4839).

Table 2. STM32H74x and STM32H75x device cache sizes

Device	Instruction cache size	Data cache size
STM32H74x / STM32H75x	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, which is a 64-bit wide interface for more instruction fetch and data load bandwidth.

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

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

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

In order 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 STM32H74x and STM32H75x, 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 memory. 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 each: D0TCM and D1TCM.

The Cortex®-M7 CPU uses the 64-bit ITCM bus for fetching instructions from the ITCM and to have access to 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 STM32H74x and STM32H75x 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 STM32H74x and STM32H75x architecture and apart 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 2*.

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 STM32H74x and STM32H75x architecture, this interface connects the Cortex®-M7 core to AHB peripherals connected to a 32-bit AHB bus matrix which 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 1*. 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 STM32H74x and STM32H75x interconnect matrix

The STM32H74x and STM32H75x devices are the first STM32 microcontrollers that embed more than one bus matrix. 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 STM32H74x and STM32H75x feature three separate bus matrices. Each bus matrix is associated to a domain:

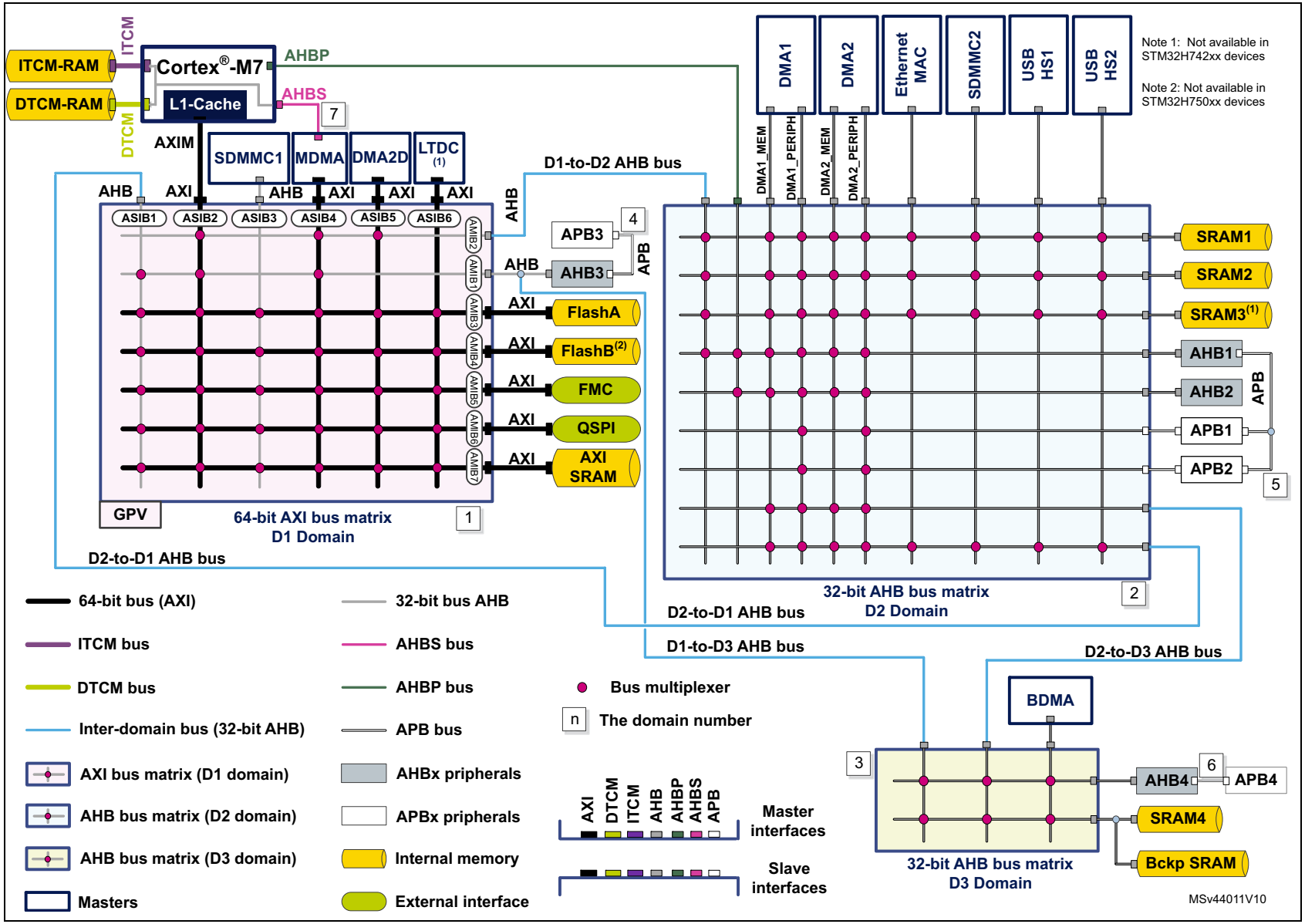
- 64-bit AXI bus matrix (in the D1 domain): It has a high-performance capability and is dedicated to operations requiring high speed. The high bandwidth peripherals are connected to the AXI bus matrix.
- 32-bit AHB bus matrix (in the D2 domain): communication peripherals and timers are connected to this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

All bus matrices can run up to 240 MHz. Only the Cortex®-M7, the ITCM-RAM and the DTCM-RAM can run up to 480 MHz.

All bus matrices are connected together by means of inter-domain buses to allow a master located in a given domain to have access to a slave located in another domain, except for BDMA master which access is limited to resources located in the D3 domain.

Figure 1 shows the overall system architecture of the STM32H74x and STM32H75x as well as the connections of the interconnect matrix.

Figure 1. 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 seven 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 up to 240 MHz. It is a 64-bit bus matrix that connects ASIBs and AMIBs and allows a number of parallel access paths between the core AXI bus, masters buses and the slaves buses enabling a concurrent access 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 such 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 STM32H74x and STM32H75x AXI bus matrix interconnects:

- Six bus masters:
  - The Cortex®-M7 AXI bus
  - The D2-to-D1 AHB inter-domain is 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 STM32H742xx devices)
  - The Chrom-Art Accelerator™ (DMA2D) 64-bit AXI bus
- Seven bus slaves:
  - The embedded Flash A memory on AXI bus
  - The embedded Flash B memory on AXI bus (not available in STM32H750xx devices)
  - The AXI SRAM memory up to 512 Kbytes accessed through an AXI bus
  - AHB3 peripherals including the AHB to APB bridge, APB3 peripherals and the D1-to-D3 AHB inter-domain
  - The FMC memory interface on AXI bus with 64-bit access
  - The Quad-SPI memory interface on AXI bus with 64-bit access
  - The D1-to-D2 inter-domain 32-bit AHB bus that connects the D1 domain to the D2 domain

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 inter-domain bus (refer to *Figure 1*). 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 located in domains D2 and D3 can run up to 240 MHz. They ensure and arbitrate concurrent accesses from multiple masters to multiple slaves. This allows 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:

- Ten bus masters:
  - The D1-to-D2 AHB inter-domain 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
- 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 (not available in STM32H742xx devices)
  - The AHB1 peripherals bus including the AHB to APB bridge that makes Cortex®-M7 to access APB1 and APB2 peripherals
  - The APB1 peripherals bus that allow DMA1 and DMA2 to access to APB1 peripherals
  - The APB2 peripherals bus that allow DMA1 and DMA2 to access to APB2 peripherals
  - The D2-to-D3 AHB inter-domain that connects the D2 domain to the D3 domain
  - The D2-to-D1 AHB inter-domain 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 inter-domain that connects the D1 domain to the D3 domain
  - The D2-to-D3 AHB inter-domain 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 1*) and APB4 peripherals
  - Internal SRAM4 of 64 Kbytes and the Backup SRAM of 4 Kbytes that shares the same AHB bus

2.4.3 Inter-domain buses

D1-to-D2 AHB inter-domain 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, 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 inter-domain bus

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

As a result, all the masters, DMA1, DMA2, Ethernet, SDMMC2, and the two USB HS located in the D2 domain can have access to the internal memories in the D1 domain (except TCM-RAMs) and to external memories as well as to the AHB3 and APB3 peripherals located in the D1 domain.

D1-to-D3 AHB inter-domain bus

This 32-bit AHB bus connects the D1 domain to the D3 domain AHB bus matrix. It allows 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 which are SRAM4, Backup SRAM, AHB4 and APB4 peripherals. SDMMC1, LTDC and DMA2D have no access to 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.

The D1-to-D3 AHB inter-domain 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 D2 which are Ethernet, SDMMC2 and the two USB HS do not have direct access to the resources located in D3 through the D2-to-D3 AHB inter-domain bus. Their access is done first through the D2-to-D1 AHB and then through the D1-to-D3 AHB inter-domain buses (refer to *Figure 2*: USBHS2 access to SRAM4: path in yellow).

D2-to-D3 AHB inter-domain bus

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

All masters in the D2 domain have the access to the resources located in the D3 domain. Only DMA1 and DMA2 have direct access to these resources through the D2-to-D3 AHB inter-domain bus. The access of the other masters is done first through the D2-to-D1 AHB and then through the D1-to-D3 AHB inter-domain buses.

Note that the basic-DMA (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 different masters to different slaves located in different domains.The numbering from 1 to 7 in *Table 4* refers to the numbers indicated in *Figure 1*. Each number refers to a domain.

Table 4. STM32H74x and STM32H75x bus-master-to-bus-slave possible interconnections

	Bus master / type[1]																			
	Cortex®-M7 - AXIM	Cortex®-M7 - AHBP	**Cortex®-M7 - ITCM**	**Cortex®-M7 -DTCM**	SDMMC1	**MDMA - AXI**	MDMA - AHBS	**DMA2D**	**LTDC[4]**	DMA1 - MEM	DMA1 - PERIPH	DMA2 - MEM	DMA2 - PERIPH	Eth. MAC - AHB	SDMMC2 - AHB	USBHS1 - AHB	USBHS2 - AHB	BDMA - AHB
Bus slave / type[1]	**Interconnect path and type[2]**																			
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	-
FlashA	**1**	-	-	-	*1*	**1**	-	**1**	**1**	*21*	*21*	-	*21*	*21*	-	*21*	*21*	*21*	*21*	-
FlashB[3]	**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	**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[4]	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	-	213	213	213	213	3
APB4 peripherals	136	-	-	-	-	136	-	-	-	236	236	-	236	236	-	213	213	213	213	36
SRAM4	13	-	-	-	-	13	-	-	-	23	23	-	23	23	-	213	213	213	213	3
Backup RAM	13	-	-	-	-	13	-	-	-	23	23	-	23	23	-	213	213	213	213	3

1. **Bold** font type denotes 64-bit bus, plain type denotes 32-bit bus.
2. 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,
   Multi-digit 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
3. Not available in STM32H750xx devices
4. Not available in STM32H742xx devices

Figure 2 shows some paths (10 examples of masters access paths) used by some masters located in the D1 and D2 domains to have access to resources located in the D1, D2, and D3 domains.

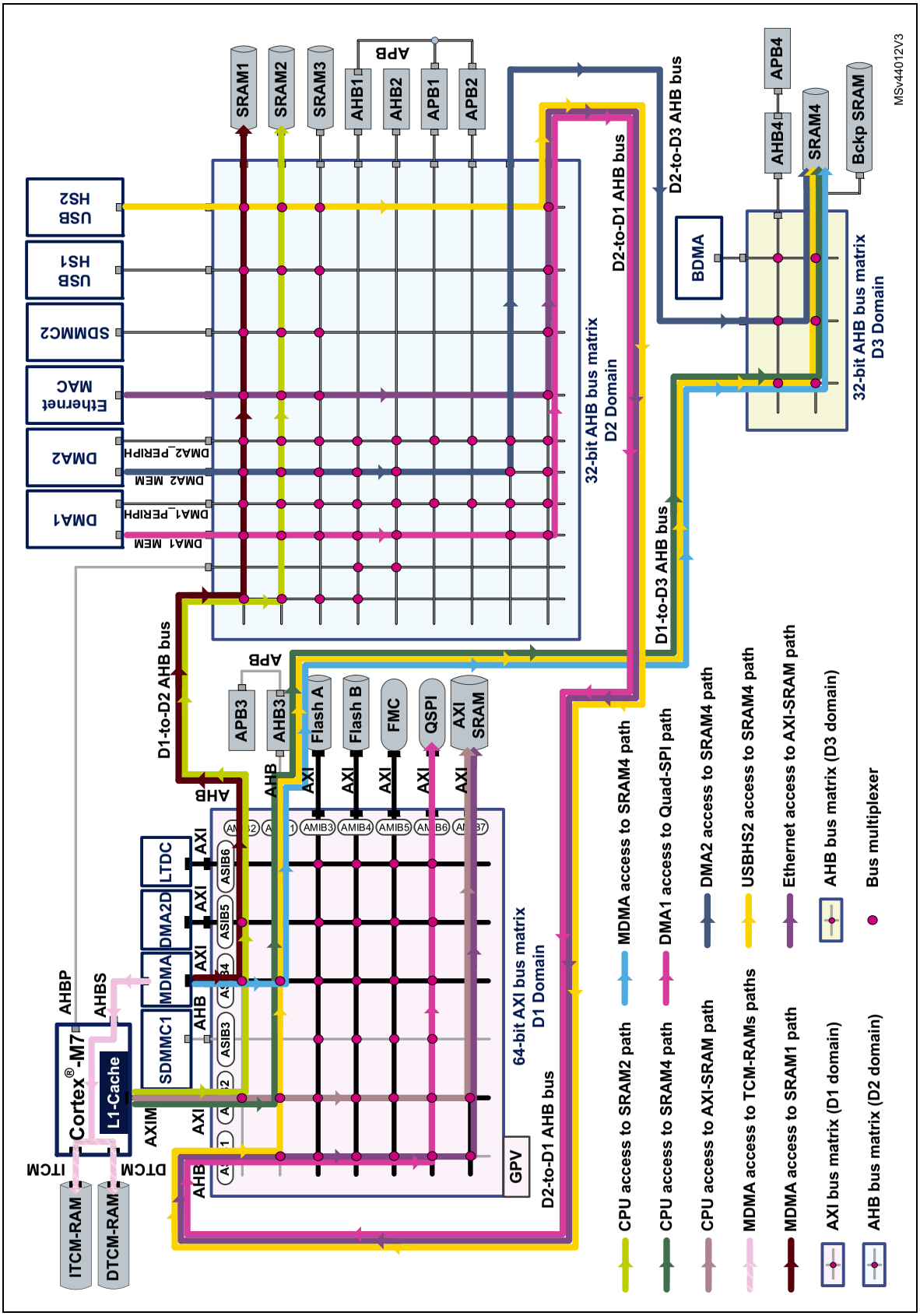


Figure 2. Examples of D1/D2 master accesses to memories in the D1, D2, and D3 domains

2.5 STM32H74x and STM32H75x memories

The STM32H74x and STM32H75x devices embed two independent Flash memory banks of up to 1 Mbyte each, an embedded SRAM scattered with different sizes and external memory interfaces such as the FMC and the Quad-SPI interfaces. The scattered architecture configuration of the STM32H74x and STM32H75x devices gives the flexibility to the user to partition his application memory resources following his needs and to get the right performance trade-off versus the application code size, data size, and power saving.

2.5.1 Embedded Flash memory

The STM32H74x and STM32H75x devices embed two independent Flash memory banks (Flash A and Flash B) of up to 1 Mbyte each, except STM32H750 devices that embed only one bank (Flash A).

The Flash memory is accessible for read or/and write accesses through the AXI bus. Each bank has its own AMIB connection to the AXI bus matrix which allows simultaneous operations. Two read/program/erase operations can be executed in parallel on the two banks which also avoids contention when two masters have access to the Flash at the same time (two masters access different banks).

The two banks are contiguous in term of address location for devices with a 2 Mbytes Flash. The start address of the Flash B memory is just after the end address of the Flash A memory. This allows the Cortex®-M7 to use all the Flash memory as a same block of 2 Mbytes of Flash as code location. The devices with 1 Mbyte of Flash do not allow this configuration thus, the two banks are not contiguous.

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).

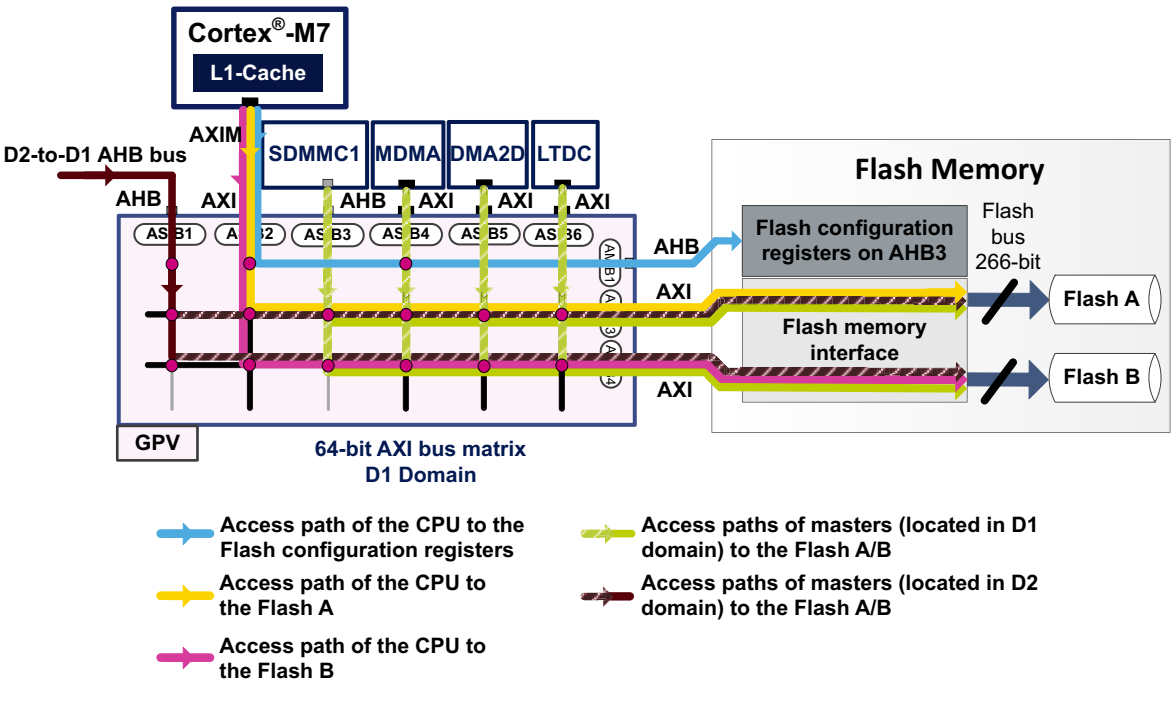
For control, configuration and status register accesses, the Flash memory interface is accessible through the AXIM/AHB3 path, which is a 32-bit AHB bus (refer to the blue path in *Figure 3*).

The STM32H7 is the first STM32 microcontroller that embeds a Flash that can run up to 70 MHz. Consequently, the number of wait states is decreased with respect to the previous STM32 devices for a same working frequency. This enhancement allows to decrease the latencies and, as a result, increases the performance of the system. For example, if the AXI bus runs at 200 MHz, the number of wait states is equal to two with 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 V_{CORE} range configured to VOS0 level (refer to the RM0433 reference manual for more details on V_{CORE} ranges).

The Flash A memory (the first bank) is accessed starting from address 0x0800 0000 while the Flash B memory (the second bank) is accessed starting from address 0x0810 0000. The instruction or/and data caches must be enabled to allow 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 3. STM32H74x and STM32H75x Flash memory accesses



2.5.2 Embedded RAM

The STM32H74x and STM32H75x features a large internal RAM of up to 1060 Kbytes (including 4 Kbytes of backup SRAM) that is divided into up to seven blocks and scattered in three domains:

RAM memories located in the D1 domain:

- The instruction RAM (ITCM-RAM) of 64 Kbytes is mapped at address 0x0000 0000. It is only accessible by the Cortex®-M7 and the MDMA (even in Sleep mode). It is accessible by the CPU through the ITCM bus (the bus colored in purple in *Figure 1*) and by the MDMA through the specific AHBS bus of the core (light pink path in *Figure 2*). 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 (480 MHz) without latency and with zero wait states.
- The DTCM-RAM of 128 Kbytes is located in the D1 domain and it is split into two DTCM-RAMs with 32-bit access each. Both 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 the address 0x2000 0000 and it is accessible only by the CPU and the MDMA. It's accessible by the Cortex®-M7 through the DTCM bus (light green bus in *Figure 1*) and by the MDMA through the specific AHBS bus of the Cortex®-M7 (light pink path in *Figure 2*). 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 (480 MHz) without latency. The concurrency access to the DTCM-RAM by the Cortex®-M7 and the

MDMA and 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*.

- The AXI SRAM of up to 512 Kbytes is mapped at the address 0x2400 0000 and it is accessible by all masters located only in D1 domain and D2 domain through D2-to-D1 AHB inter-domain bus. The BDMA located in 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 2* for some possible AXI SRAM accesses. The AXI SRAM can be used for read/write data storage as well as for code execution. The AXI SRAM is accessed at the same frequency as the AXI bus matrix (240 MHz maximum).

RAM memories located in the D2 domain:

The AHB SRAM1, SRAM2, and SRAM3 are accessible by all masters located only in the D1 domain and the D2 domain through the D1-to-D2 AHB inter-domain bus. They are accessible by bytes, half-words (16 bits) or words (32 bits). Refer to *Figure 2* for an illustration of some possible SRAM1 and SRAM2 accesses. They can be used for read/write data storage as well as for code execution. Those memories are accessed at the AHB bus matrix frequency (240 MHz max). Each memory has its own AHB bus that connects it to the AHB bus matrix. This allows the removal of bus contention and to keep high system performance when it is concurrently accessed by more than one master.

- 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 STM32H742xx devices.

These 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.

RAM memories located in the D3 domain:

- The AHB SRAM4 is accessible by:
  - The Cortex®-M7 and MDMA located in the D1 domain through the D1-to-D3 AHB inter-domain bus.
  - Any master located in the D2 domain through the D2-to-D3 AHB inter-domain bus or through AHB inter-domain 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 is accessible by bytes, half-words (16 bits) or words (32 bits). Refer to *Figure 2* 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 (240 MHz max).
- Backup SRAM of 4 Kbytes is accessible by most of the system masters at address 0x3880 0000 (all masters except SDMMC1, DMA2D and LTDC). It is accessible 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 the RM0433 reference manual for the way to use this memory).

All internal RAMs feature error correction code (ECC).

Table 5 summarizes the internal memory mapping and the memory sizes of the STM32H74x and STM32H75x devices:

Table 5. Internal memory summary of the STM32H74x and STM32H75x

Memory type	Memory region	Address start	Size	Access interfaces	Domain	Maximum frequency
FLASH	FLASH-A	0x0800 0000	1 Mbyte(1)	AXI (64 bits)	D1	240 MHz
	FLASH-B(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)	AHB (32 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 Mbytes depending on the device
2. Not available in STM32H750xx devices
3. Up to 512 Kbytes depending on the device
4. Up to 128 Kbytes depending on the device
5. Not available in STM32H742xx 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 STM32H74x and STM32H75x memories with the flexible memory controller (FMC) and the Quad-SPI controller.

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

Four external memory banks are dedicated to the FMC (plus one bank for SDRAM remap) and one dedicated to the Quad-SPI:

- Bank 1 is used for NOR/PSRAM memories.
- Bank 2 is used for SDRAM bank remap.
- 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 allows three configurable options:

- Default mapping.
- Remap of SDRAM bank2, thus allowing to access the SDRAM banks at two different address mappings.
- Swap 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 adding 256 Mbytes. It can be used to store data such as images in graphical applications or to contain code to execute the user application.

Figure 4 summarizes the memory mapping of the external memories, their respective address ranges after reset and their different allowed remap.

Figure 4. External memory mapping

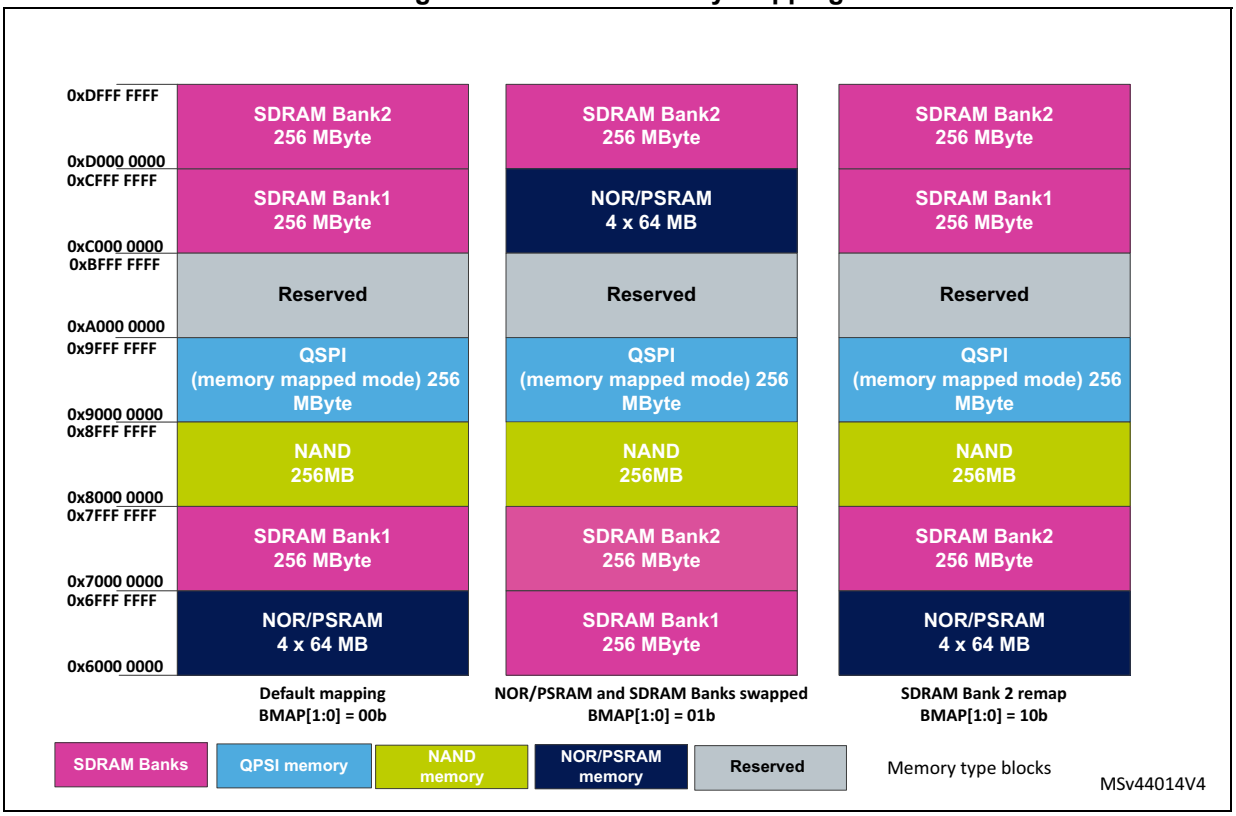


Figure 5 shows the possible paths that interconnect the Cortex®-M7 and the different DMAs with these external memories via the AXI and the AHB buses. As shown in the same figure, the external memories can benefit of the Cortex®-M7 cache, therefore, they can get the maximum of the performance whether they are used for data storage or for code execution. This allows to combine high performance and large memory size.

The path in pink, in *Figure 5*, represents the connection between the external memories, by means of the FMC, and the Cortex®-M7.

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

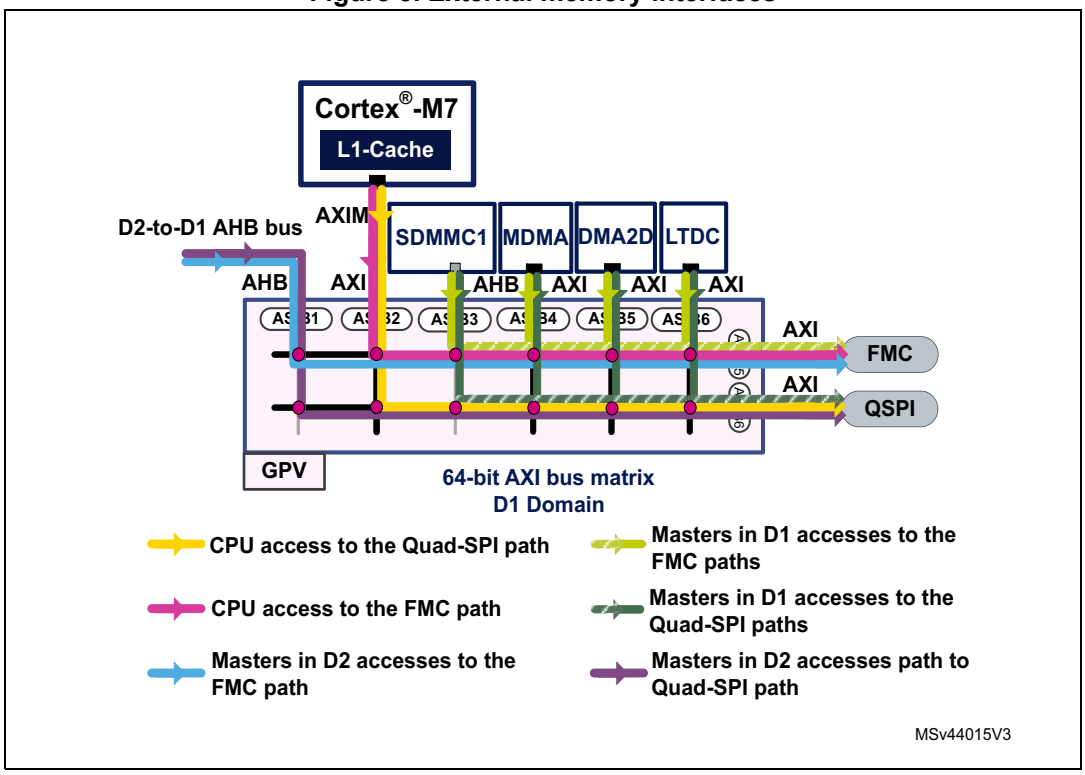
The path in light green represents the access paths of some masters in the D1 domain to external memories connected by means of the FMC controller.

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

The paths in purple and in blue represent respectively the accesses of masters located in 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 with the exception of the BDMA.

Figure 5. External memory interfaces



The flexible memory controller (FMC) interface

The STM32H74x and STM32H75x FMC controller interfaces 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) or for R/W data storage.

The STM32H74x and STM32H75x FMC features:

- 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 maximum frequency of 110 MHz (kernel clock divided by 2).

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. This is done by setting the field BMAP[1:0] bits in the FMC_BCR1 register. *Figure 4* 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 5*), allowing larger data or code sizes with maximum performance.

Quad-SPI interface

The STM32H74x and STM32H75x devices embed a Quad-SPI memory interface, 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 allows a multiplication by a factor of two the data/instruction through-put and therefore to increase the access efficiency to an external Quad-SPI Flash memory.

Works in one of the three following modes:

- Direct mode: all operations are performed through the Quad-SPI 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 and STM32H75x Quad-SPI interface supports the SDR mode which is the default mode and the DDR mode that allows double data throughput. The DDR mode boosts data load and instruction fetch performances. It is also useful when the Quad-SPI clock must be slow, for example in case of a PCB limitation that does not allow the Quad-SPI to operate at its maximum speed.

The Quad-SPI interface supports single-Flash and dual-Flash memory modes. The latter allows 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 Quad-SPI interface 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 remap is not needed (remap at the address 0x0000 0000).

Compared to the FMC, the Quad-SPI allows to connect an external Flash memory with a reduced cost with small packages (reducing PCB area) and a reduced GPIOs usage. Six GPIOs are used in single-QUADSPI mode (4 bits) for any Flash memory size or ten GPIOs in Dual-quad mode (8 bits).

As shown in *Figure 1*, the Quad-SPI is mapped on a dedicated layer on the 64-bit AXI bus matrix and can benefit from the L1-cache. This allows to execute the code and load the data from the Quad-SPI with higher performance (yellow path in *Figure 5*). Note that Quad-SPI registers are mapped on the AHB bus.

The Quad-SPI is also accessible by all the masters available on the system (dark green, yellow, and purple paths in *Figure 5*) except the BDMA. It is accessible by the Chrom-ART accelerator™ and the LCD-TFT, enabling an efficient data transfer, particularly images, for graphical applications where a high frame rate of display is needed.

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

2.6 Main architecture differences between STM32F7 Series and, STM32H74x and STM32H75x devices

Table 6 summarizes the differences between the STM32F7 Series devices and, STM32H74x and STM32H75x devices linked to the architecture and performances (peripheral differences not included).

Table 6. Architecture differences between the STM32F7 Series and, STM32H74x and STM32H75xdevices

-	STM32F7 Series	STM32H74x and STM32H75x devices
Cortex®-M7 revision	r0P1/r1P0 (1)	r1P1
Max CPU frequency	216 MHz	480 MHz
Instruction / data cache sizes	4 Kbytes to 16 Kbytes(1)	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 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 line width	128/256 bit(3)	256 bit for each bank for read and write
Flash wait states@200 MHz	6(5)	2
Dual Flash bank support	No/Yes (1)	Yes(6)
DTCM-RAM size	64 Kbytes or 128 Kbytes (1)	128 Kbytes
ITCM-RAM size	16 Kbytes	64 Kbytes
ITCM-RAM accessibility	CPU only	CPU + MDMA
SRAM size	Up to 384 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 (7) 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. Depends 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 STM32H750xx devices.
7. 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 and STM32H75x. 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 his application.

3.1 FFT demonstration

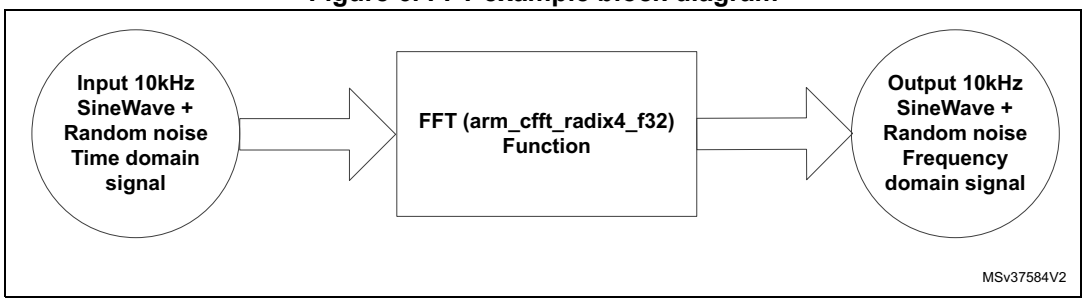
The FFT example is used as it benefits from the floating point unit of the Cortex®-M7, contains several loops and data load/store operations, and can be accessed in different paths/memories. The code can be executed from internal or external memories.

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 6* 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.

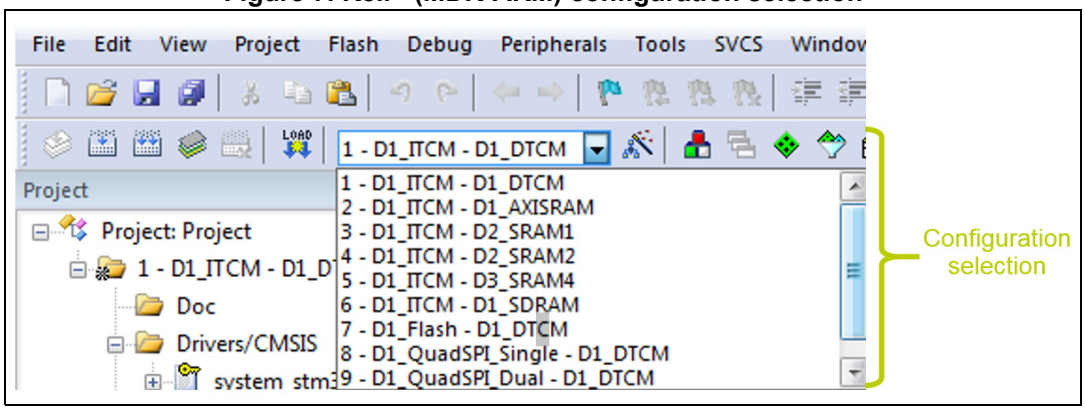
Figure 6. FFT example block diagram



3.2 Project configuration of the demonstration

The CPU memory access demonstration is provided with the Keil® (MDK-ARM), IAR™ (EWARM) and System Workbench for STM32 (SW4STM32) tool chains. The project is presented in ten sub-project workspaces that represent, each one, a configuration. Each configuration allows the selection of the data and code locations. *Figure 7* shows a screen shot of different configurations of MDK-ARM toolchain.

Figure 7. 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 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, the read-only data are located, in this case, in the internal Flash memory. This is the reason why the D-cache is also enabled.

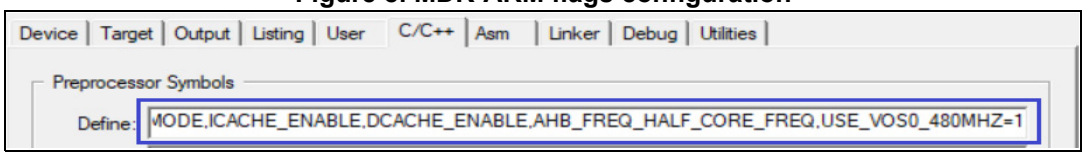
8 - D1_QuadSPI_Single - D1_DTCM: the program is executed from the Quad-SPI Flash memory with the I-cache and the D-cache enabled (since the constants are located in Quad-SPI Flash). The Quad-SPI Flash memory is configured in Single mode and runs with the DDR mode enabled at 60MHz if the system clock 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 60MHz 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 8* shows where these flags are defined for the MDK-ARM tool chain.

Figure 8. MDK-ARM flags configuration



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

Project flags description:

- **USE_VOS0_480MHZ** : defines the system clock frequency.
  - if USE_VOS0_480MHZ=1, the system clock frequency is set to 480MHz.
  - if USE_VOS0_480MHZ=0, the system clock frequency is set to 400MHz.
- **AHB_FREQ_*X*_CORE_FREQ**:
  Core and bus matrices operating frequencies. Where *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 frequency of the core.
  - **AHB_FREQ_EQU_CORE_FREQ**: The AXI bus matrix and the two AHB bus matrices run at the same frequency of 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**: Configures the number of wait states of the internal Flash:
  FLASH_WS = FLASH_LATENCY_X, where X ranges from 0 (0 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 wait state can be equal to 2.
If the flag USE_VOS0_480MHZ=1 and the user tries to set the Flash wait state to a value less than 4, a directive #error raises up during the compile time telling the user to increase the Flash 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**: 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 at 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 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**: Defines the Quad-SPI clock prescaler and the configuration is as follows:

  QSPI_CLK_PRESCALER=X, where X = 0 to 255.
- **QSPI_DDRMODE:** If defined in the configuration project, the Quad-SPI is configured in DDR mode.
- **QSPI_INSRUCTION_1_LINE, QSPI_INSRUCTION_4_LINES**: The first flag is to configure the Quad-SPI Flash instruction to operate in one line mode, while the second configures the instruction to operate in four lines 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 Quad-SPI 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.

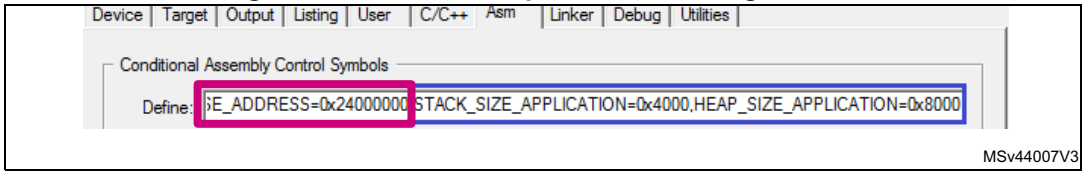
The user can create new configurations of code execution/data storage location based on these templates: by merging the adequate settings, by modifying the linker files and by setting the adequate Flash loader.

Note: The MDK-ARM tool chain and in order to modify the RAM regions in the scatter files, that is, the stack and heap regions, the user must modify accordingly the stack and heap sizes in the ASM menu. 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 in order to force their values in line with the heap/stack size regions configured in the scatter file.

Figure 9 shows where to modify these flags (framed in blue). 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 modified by Stack_Size_Init variable in the startup file. The initial stack pointer base address is configurable in the ASM menu as shown in *Figure 9* framed in red.

Figure 9. 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.

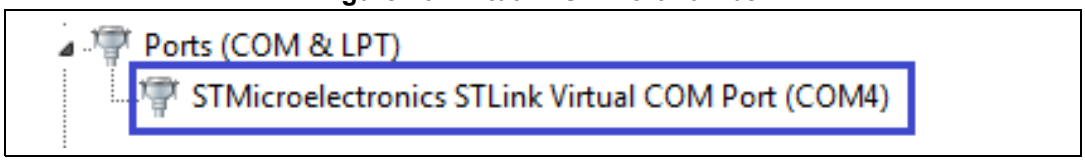
For the IAR™ (EWARM) toolchain, the linkers are located under EWARM\icf_files. For the System Workbench toolchain, the linkers are located under SW4STM32\<project folder configuration>.

The results are displayed on the Hyperterminal through the UART using the Virtual COM port with the following configuration:

- Baudrate: 115200
- Data bits: 7 bits
- Stop bits: 1 bit
- Parity: Odd
- HW flow control: none

To know which COM number the board is using, the user must connect the ST-Link of his board to his PC through a USB cable and go to "Control Panel" -> "System" -> "Device Manager" -> "Ports (COM & LPT)". Example: in *Figure 10*, the UART COM number is COM4.

Figure 10. Virtual COM Port number



4 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) tool chain, the STM32H7xx pack version 2.2.0 and the Cube firmware version 1.4.0.

The MDK-ARM code optimization configuration is 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 Results

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

- 480 MHz and 240 MHz (VOS0: only for silicon revision > rev Y, Flash wait state=4)
- 240 MHz and 240 MHz (VOS0: only for silicon revision > 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 STM32H742xx, STM32H743xx, STM32H750xx and STM32H753xx devices.

4.1.1 Data and instructions locations effects on performance

Table 7 and *Table 8* 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 > rev. Y.

Table 9 and *Table 10* 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 11 and *Table 12* 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 13 and *Table 14* 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 7*, *Table 9*, *Table 11* and *Table 13*, 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 8*, *Table 10*, *Table 12* and *Table 14*, the R/W data location is fixed in DTCM-RAM and the code and the RO data location is varied for each configuration.

Table 7. MDK-ARM results of data storage in different memory locations (execution location fixed in ITCM-RAM)CPU @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 8. MDK-ARM results of execution in different memory locations (data location fixed in DTCM-RAM) CPU @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 9. MDK-ARM results of data storage in different memory locations (execution location fixed in ITCM-RAM) CPU @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 10. MDK-ARM results of execution in different memory locations (data location fixed in DTCM-RAM) CPU @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 11. MDK-ARM results of data storage in different memory locations (execution location fixed in ITCM-RAM) CPU @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 12. MDK-ARM results of data storage in different memory locations (execution location fixed in ITCM-RAM) CPU @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 13. MDK-ARM results of data storage in different memory locations (execution location fixed in ITCM-RAM) CPU @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 14. MDK-ARM results of execution in different memory locations (data location fixed in DTCM-RAM) CPU @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

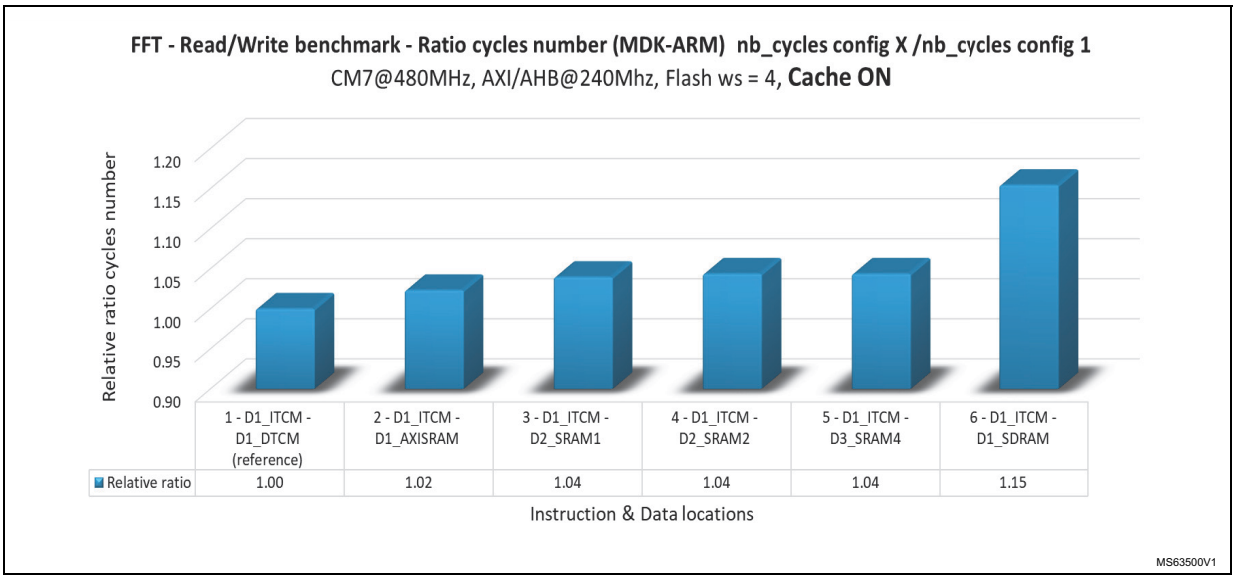
1. The execution time values may vary from one tool-chain version to another.

Relative ratio = execution_time_config_X / execution_time_config_1

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

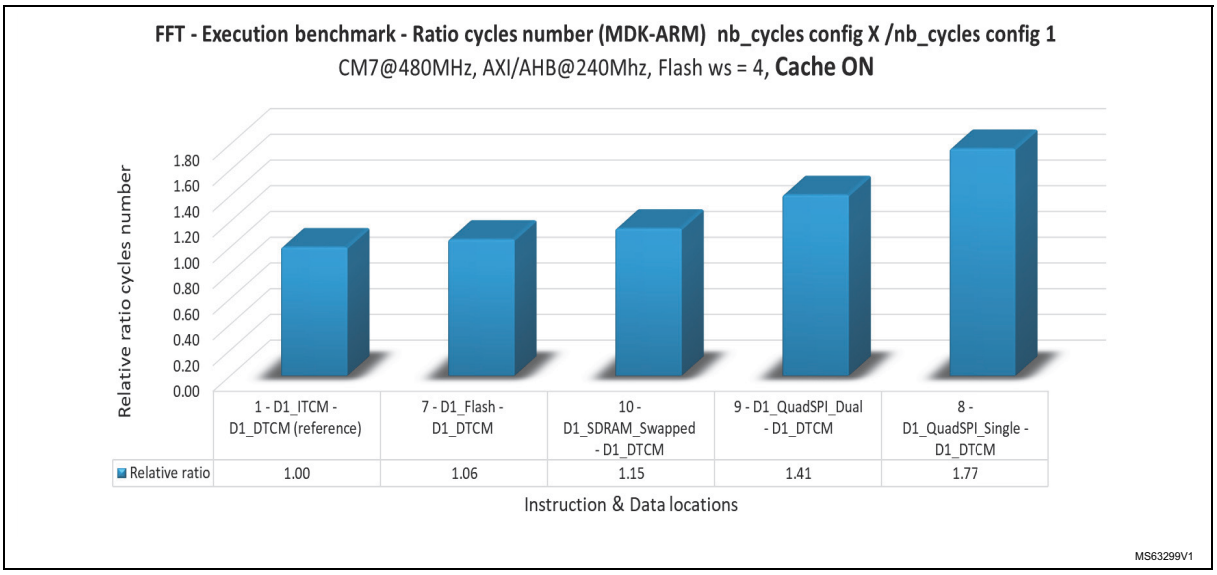
The chart in *Figure 11* shows the relative ratio of each configuration versus configuration 1 that represents the reference of this benchmark. This chart shows the FFT benchmark of data storage in different memory locations while the code location is fixed in ITCM-RAM with CPU running at 480 MHz.

Figure 11. STM32H74x and STM32H75x FFT benchmark: data storage in different memory locations (code in ITCM-RAM) at 480 MHz with MDK-ARM toolchain



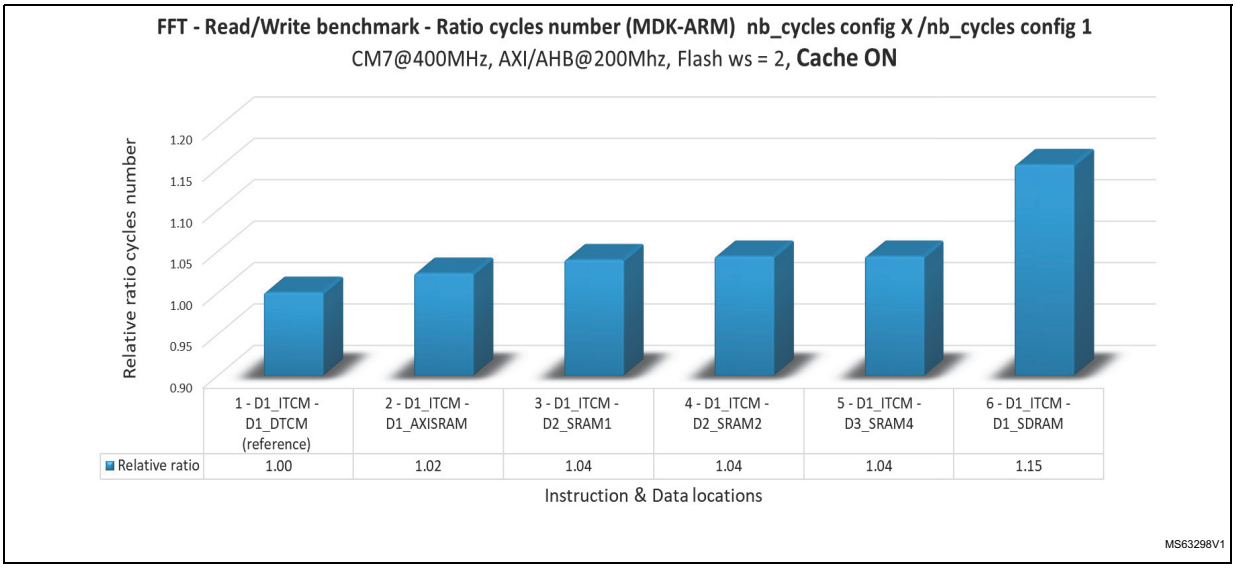
The chart in *Figure 12* shows the relative ratio of each configuration versus configuration 1 (D1_ITCM - D1_DTCM). It represents the FFT benchmark of the code execution in different memory locations while the data storage location is fixed in the DTCM-RAM with the CPU running at 480 MHz.

Figure 12. STM32H74x and STM32H75x FFT benchmark: code execution in different memory locations (R/W data in DTCM-RAM) at 480 MHz with MDK-ARM tool-chain



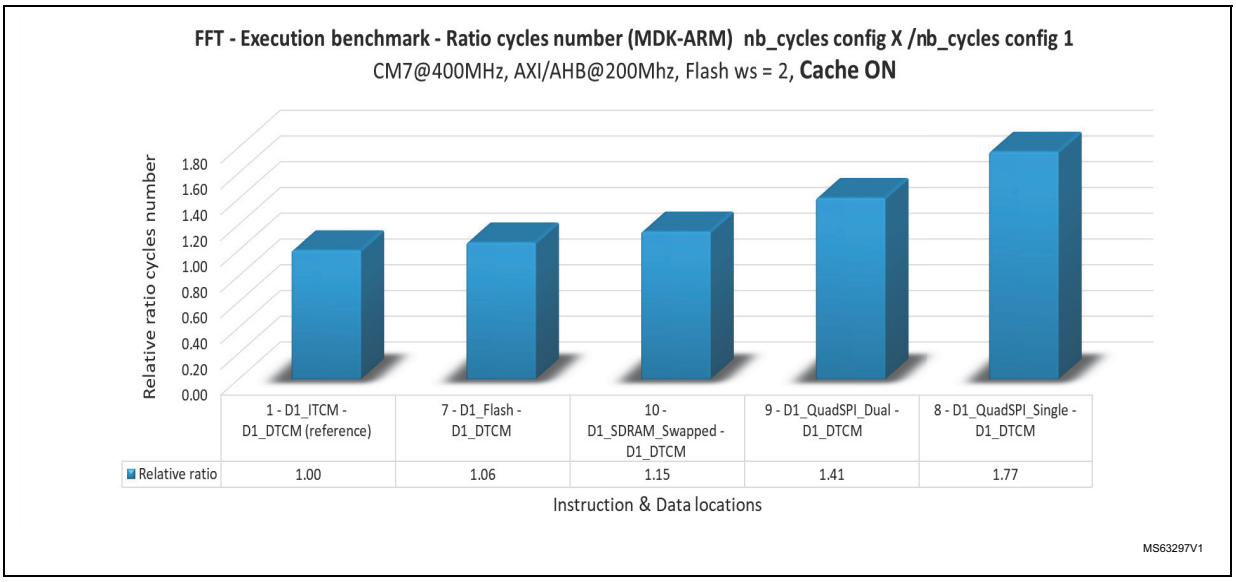
The chart in *Figure 13* shows the relative ratio of each configuration versus configuration 1 that represents the reference of this benchmark. This chart shows the FFT benchmark of data storage in different memory locations while the code location is fixed in ITCM-RAM with CPU running at 400 MHz.

Figure 13. STM32H74x and STM32H75x FFT benchmark: data storage in different memory locations (code in ITCM-RAM) at 400 MHz with MDK-ARM tool-chain



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 in different memory locations while the data storage location is fixed in the DTCM-RAM with the CPU running at 400 MHz.

Figure 14. STM32H74x and STM32H75x FFT benchmark: code execution in different memory locations (R/W data in DTCM-RAM) at 400 MHz with MDK-ARM tool-chain



4.1.2 Some basic parameters that can impact the performance

Flash wait state impacts

This section provides 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 by setting FLASH_WS parameter in the IDE Preprocessor.
Example: FLASH_WS=FLASH_LATENCY_4 to set the Flash wait states number to 4.

Table 15 summarizes the results of the wait state impacts on the performance with CPU and AXI running at 480 MHz and 240 MHz respectively.

Table 16 summarizes the results of the wait state impacts on the performance with CPU and AXI running at 400 MHz and 200 MHz respectively.

Table 15. Number of Flash wait states on the performance (MDK-ARM) /CPU @480MHz/ AXI @240MHz (VOS0)

7 - D1_Flash - D1_DTCM			
CPU @480MHz/AXI @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 16. Number of Flash wait states on the performance (MDK-ARM) /CPU @400MHz/ AXI @200MHz (VOS1)

7 - D1_Flash - D1_DTCM			
CPU @400MHz/AXI @200MHz (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 15* and *Table 16*) show a decrease of performance of about 0,54 % when the number of Flash wait states is incremented by 1.

SDRAM parameter and configuration impact

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 non-swapped banks). By default SDRAM mapping or non-swapped banks configuration (please refer to *Figure 4*), 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 17 provides the results obtained based on the 6 - D1_ITCM - D1_SDRAM configuration by adjusting the SDRAM Bus width and its clock frequency.

Table 17. SDRAM data read/write access performance versus its bus width and its clock frequency based on configuration 6 - D1_ITCM - D1_SDRAM

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 18*.

Table 18. Execution performance from SDRAM versus its bus width and its clock frequency based on configuration 10 - D1_SDRAM_Swapped - D1_DTCM

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 19 provides the performance results of SDRAM data read/write access in SDRAM swapped and non-swapped configurations based on the 6 - D1_ITCM - D1_SDRAM configuration.

Table 19. SDRAM data read/write access performance in swapped and non-swapped bank configurations based on configuration 6 - D1_ITCM - D1_SDRAM

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 20 provides the performance results of the code execution from SDRAM in SDRAM swapped and non-swapped configurations based on the 10 - D1_SDRAM_Swapped - D1_DTCM configuration.

Table 20. Execution performance from SDRAM in swapped and non-swapped bank configurations based on configuration 10 - D1_SDRAM_Swapped - D1_DTCM

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 Analysis

In the case of the data storage benchmark in different memory locations (*Figure 11* and *Figure 13*), 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 allows to benchmark 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 12* and *Figure 14*), the data location is fixed in the DTCM-RAM. This has the advantage that access to data is without latencies. In this way, any effect of the memory latency of the R/W data is excluded. Such a configuration allows to benchmark only instruction fetches in several memory locations without data access interference.

The relative ratio is calculated also for Cortex®-M7 running at 240 MHz (same frequency as bus matrices as shown in *Table 13* and *Table 14*) for different configurations to illustrate the real performances of the different memory interfaces versus Cortex®-M7 accesses.

Cache usage is recommended to get 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 allows to have almost the same level of performance for data storage in the DTCM-RAM and for code execution in the ITCM-RAM.

For highest data storage performance accesses by the Cortex®-M7, the best location of R/W data is DTCM-RAM, since it is accessed without latencies 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 13*).

The best location for code execution, in terms of performance, is ITCM-RAM, which is accessed without latencies, in a deterministic way, and runs at the same frequency as the Cortex®-M7. A similar level of performance is obtained when execution is performed in the internal Flash memory, even at 480 MHz. This is thanks to the cache size implemented in STM32H74x and STM32H75x and to the low number of wait states of the internal Flash memory (refer to *Figure 14*).

It is advised to use the Quad-SPI Flash in Dual-Flash memory mode to enhance performances. In the case of the FFT algorithm, 20 % performance increase is obtained using the Dual-Flash mode versus the Single-Flash mode.

For SDRAM, similar performance is obtained for code execution compared to execution in the internal Flash with cache enabled.

5 Software memory partitioning and tips

This section provides some tips about code and data partitioning in STM32H74x and STM32H75x 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.

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 480 MHz.

In real time applications that use RTOS, heap is generally massively used. If the DTCM-RAM size (128 Kbytes) is large enough for the application, a reasonable DTCM-RAM scattering is as follows: 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 his application needs but he must give higher priority to place the stack in DTCM-RAM with respect to heap and global variables. In a bare-metal applications model, 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 in the D1 domain, can be reserved for a graphic frame buffer in graphic applications using QVGA TFTs in 16-bit mode that need a large amount of graphic data and high display performance, achieved by the LCD-TFT and the DMA2D DMAs. 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.

The 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 I^2S 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 D1-to-D2 inter-domain bus with the usage of the data cache.

The SRAM4 located in 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 to lose 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 reference manual STM32H742/743/753 and STM32H750 *advanced Arm®-based 32-bit MCUs, Section 6: Low-power D3 domain* (RM0433), 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 or 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 graphic applications or non-critical data. At the same time, more priority is given for very critical data to be placed in the DTCM-RAM.

The Quad-SPI Flash memory could 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.

The Quad-SPI Flash memory can also be used to contain the application in the memory mapped mode up to 256 Mbytes and at the same time to save several GPIOs in smaller STM32H74x and STM32H75x 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 size and more execution performance the user can load his application in the Quad-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 performances to be reached in case of any memory access except for TCM-RAMs memories 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 480 MHz).

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 for code location of an application, care should 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 hard fault 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 the default executable regions of the Cortex®-M7.

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 hard fault exception, they must be enabled by programming the RCC_AHB2ENR register before the CPU can have access to them.

6 Conclusion

STM32H74x and STM32H75x devices are extending the range of ST high-performance 32-bit MCUs as a continuity of the STM32F7 Series.

The STM32H74x and STM32H75x, versus their predecessors, bring higher performance with an optimized power consumption thanks to their improved architecture, to their embedded L1-cache (instruction and data caches), to the Cortex®-M7 that can run up to 480 MHz, and to their 40 nm manufacturing technology.

The number of wait states of the embedded Flash is decreased with respect to the STM32F7 Series at the same working 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, to eliminate traditional constraints on applications development linked to resources, and to accelerate time to market for new products.

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

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

7 Revision history

Table 21. 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 21. 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: Project configuration of the demonstration* Added: – *Section 4.1.1: Data and instructions locations effects on performance* – *Figure 6* through *Figure 10* in *Section 3: Typical application*. – *Figure 11* through *Figure 14* in *Section 4: Results and analysis* – *Section 4.1.2: Some basic parameters that can impact the performance*