STM32Cube Expansion Package for STM32H7 Series MDMA

Introduction

This application note describes the use of the MDMA (master direct memory access) controller available in STM32H7 Series microcontrollers. The features of the MDMA controller, the STM32H7 system architecture, and the associated memory system contribute to the freeing up of CPU resources.

The MDMA optimizes the data transfer bandwidth and off-loads some basic data management operations from the CPU.

As a system controller, the MDMA is mainly used to manage direct data transfers between RAM data buffers without CPU intervention. It can also be used in a hierarchical structure that uses STM32 standard DMAs (DMA1 and DMA2) as first level data buffer interfaces for AHB peripherals, while the MDMA acts as a second level DMA with more advanced features, such as non-contiguous data increment, data packing, and data formatting.

This document focuses on all the features that are not available in other DMAs and provides the user with a good understanding of use cases where using the MDMA is advantageous.

This application note is provided together with the X-CUBE-MDMA Expansion Package. It details specific aspects of the MDMA by means of a use case walkthrough in order to allow developers to take full advantage of MDMA benefits with respect to a DMA-based solution. It presents a suitable implementation of different peripherals and subsystems.

Reference documents and firmware

The following documents and packages are available on *www.st.com*:

- *STM32H745/755 and STM32H747/757 advanced Arm®-based 32-bit MCUs* (RM0399)
- *STM32H742, STM32H743/753 and STM32H750 Value line advanced Arm®-based 32-bit MCUs* (RM0433)
- *STM32H7A3/7B3 and STM32H7B0 Value line advanced Arm®-based 32-bit MCUs* (RM0455)
- *STM32H723/733, STM32H725/735 and STM32H730 Value line advanced Arm®-based 32-bit MCUs* (RM0468)
- *Using the STM32F2, STM32F4 and STM32F7 Series DMA controller* (AN4031)
- *STM32Cube MCU Package for STM32H7 Series with HAL and dedicated middleware* (STM32CubeH7)
- *Data transfer efficiency using MDMA software expansion for STM32Cube* (X-CUBE-MDMA)

1 MDMA features

This chapter introduces the position of the MDMA in the global STM32H7 bus architecture and presents the MDMA specific functions that differentiate the MDMA controller from the DMA1 and DMA2 controllers.

This chapter is not intended to describe all registers. Refer to reference manuals *STM32H745/755 and STM32H747/757 advanced Arm®-based 32-bit MCUs* (RM0399), *STM32H742, STM32H743/753 and STM32H750 Value line advanced Arm®-based 32-bit MCUs* (RM0433), *STM32H7A3/7B3 and STM32H7B0 Value line advanced Arm®-based 32-bit MCUs* (RM0455) and *STM32H723/733, STM32H725/735 and STM32H730 Value line advanced Arm®-based 32-bit MCUs* (RM0468) for details.

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

1.1 The MDMA can interact with other masters without CPU intervention

The MDMA controller performs direct memory transfer. Like any other AXI/AHB master, it can take control of the AXI/AHB bus matrix to initiate AXI/AHB transactions. Figure 1 shows all the masters connected to the AXI/AHB matrix for the STM32H74xxx and STM32H75xxx microcontrollers.

Figure 1. Master connections of the AHB/AXI matrix for STM32H74xxx and STM32H75xxx

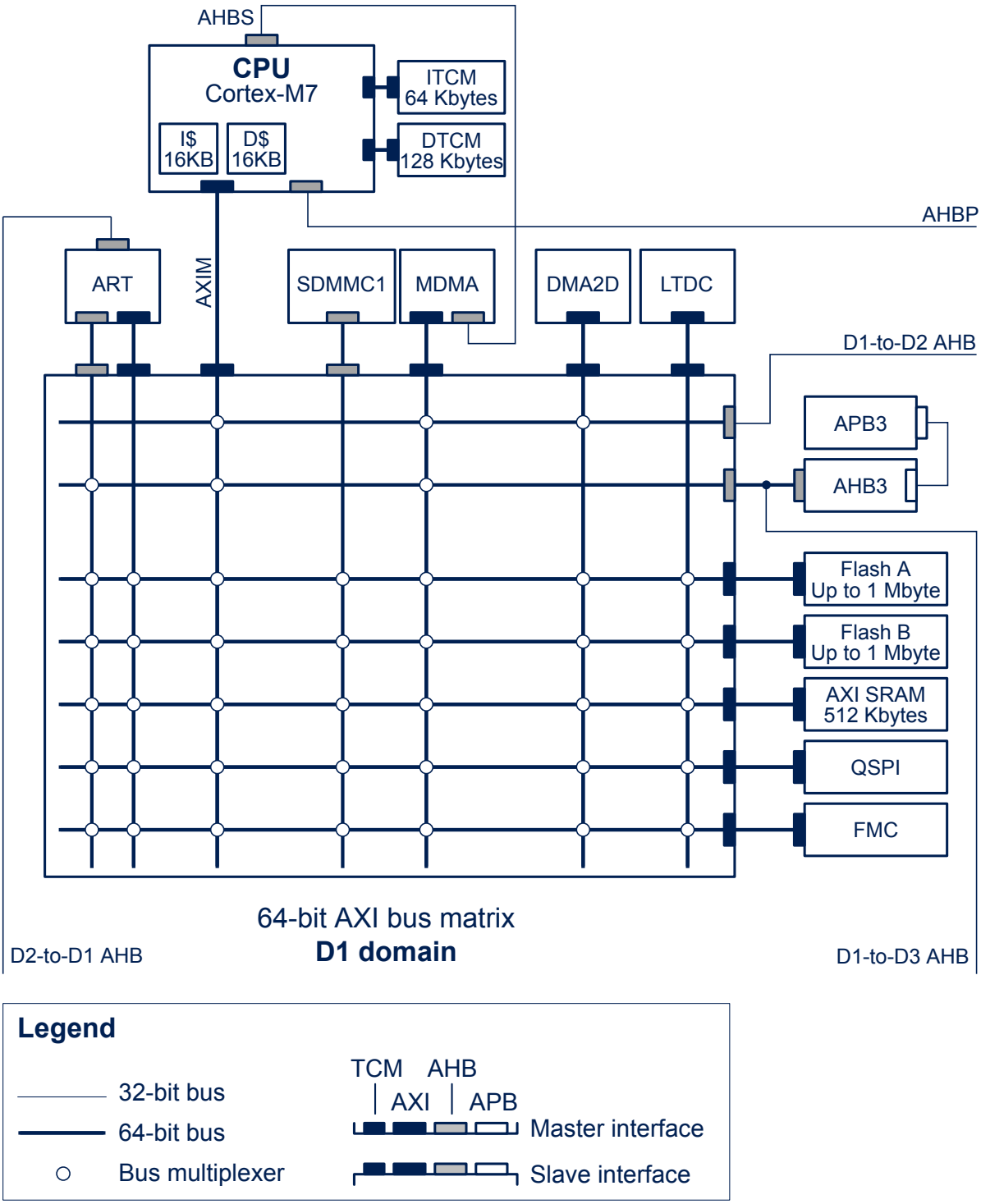
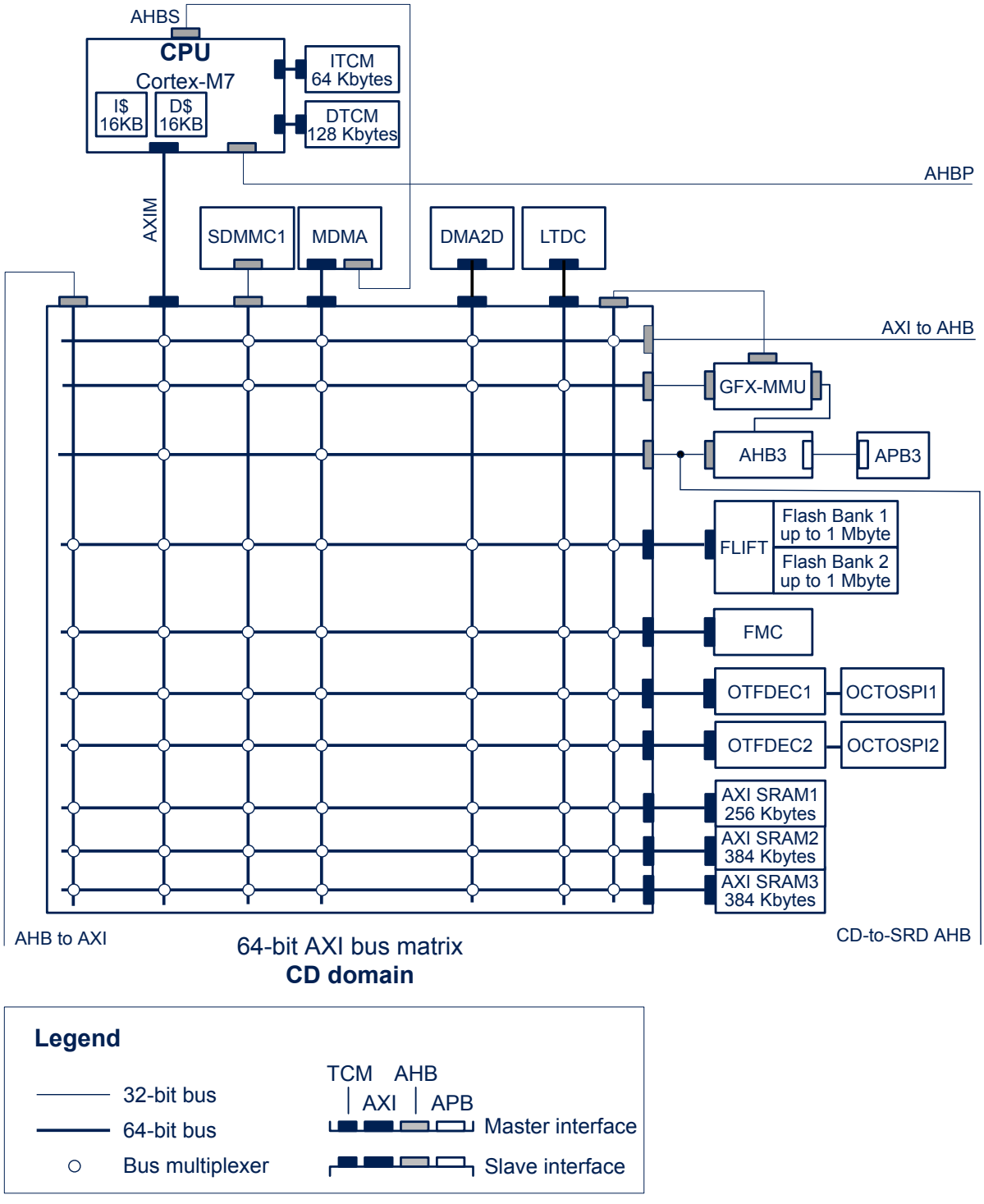


Figure 2 shows all the masters connected to the AXI/AHB matrix for the STM32H7Axxx and STM32H7Bxxx microcontrollers.

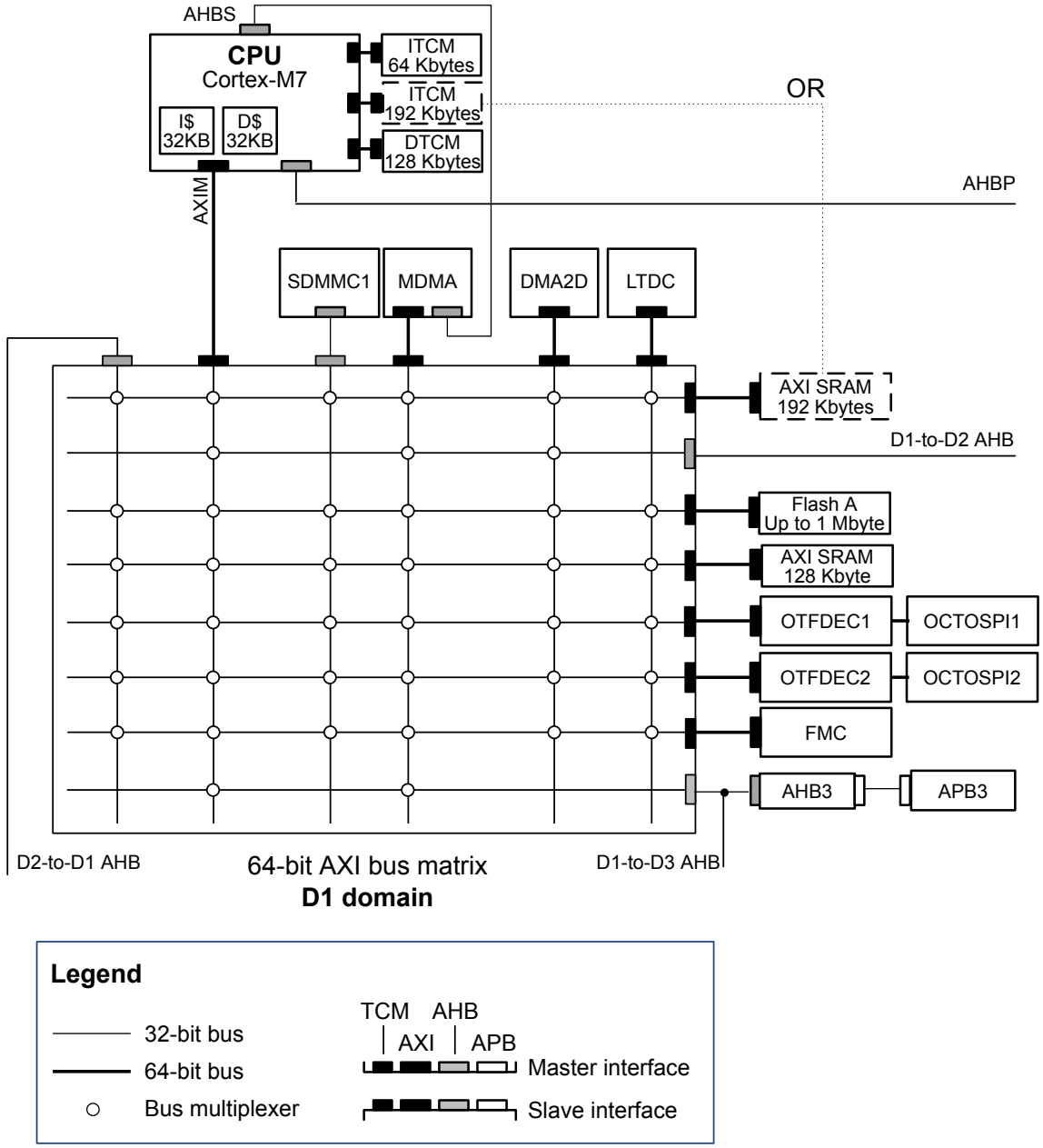
Figure 2. Master connections of the AHB/AXI matrix for STM32H7Axxx and STM32H7Bxxx



Note: All the masters shown in Figure 1 and Figure 2 can activate the MDMA.

The figure below shows all the masters connected to the AXI/AHB matrix for the STM32H72xxx and STM32H73xxx microcontrollers.

Figure 3. Master connections of the AHB/AXI matrix for STM32H72xxx and STM32H73xxx



1.2 The MDMA can be triggered by peripherals such as DMA1 and DMA2

In the D1 domain (STM32H72/73/74/75xxx) or CD domain (STM32H7Axxx and STM32H7Bxxx), the MDMA allows the transfer of memory data. It can be triggered by software or hardware. Furthermore it has direct connections with DMA1 and DMA2. This enables autonomous communication and synchronization between peripherals, thus saving CPU resources and power consumption. Each transmit complete flag (tcf) is linked to an MDMA stream (str). Table 1 presents the mapping of MDMA steams on DMA transmit flags for the D1 or CD domain.

Table 1. MDMA interconnection table with DMA1 and DMA2

Domain source	Bus	Peripheral	DMA transmit signal	MDMA stream signal	TSEL value
D2(1) or CD(2)	AHB1	DMA1	dma1_tcf0	mdma_str0	0x00
			dma1_tcf1	mdma_str1	0x01
			dma1_tcf2	mdma_str2	0x02
			dma1_tcf3	mdma_str3	0x03
			dma1_tcf4	mdma_str4	0x04
			dma1_tcf5	mdma_str5	0x05
			dma1_tcf6	mdma_str6	0x06
			dma1_tcf7	mdma_str7	0x07
		DMA2	dma2_tcf0	mdma_str8	0x08
			dma2_tcf1	mdma_str9	0x09
			dma2_tcf2	mdma_str10	0x0A
			dma2_tcf3	mdma_str11	0x0B
			dma2_tcf4	mdma_str12	0x0C
			dma2_tcf5	mdma_str13	0x0D
			dma2_tcf6	mdma_str14	0x0E
			dma2_tcf7	mdma_str15	0x0F

1. *D2 for STM32H72/73/74/75xxx MCUs.*
2. *CD for STM32H7Axxx and STM32H7Bxxx MCUs.*

DMA1 and DMA 2 work like other DMAs in similar products. A detailed description is available in application note *Using the STM32F2, STM32F4 and STM32F7 Series DMA controller* (AN4031).

Bit field TSEL[5:0] (Trigger SELection) in the trigger and bus selection register is used to program the hardware trigger (one stream for each MDMA channel). Bit field selects the hardware trigger (RQ) input for channel x. The acknowledge is sent on the ACK output having the same index value. The bits in this bit field are write-protected and can be written only when bit EN = 0 in the MDMA_CxCR register. The transfer is triggered by software writing 1 to the SWRQ (software request). When the SWRM bit is set (software request mode), this bit field is ignored.

Note: If multiple channels are triggered by the same event (meaning that they have the same TSEL value), all of them are triggered in parallel. However, only the channel with the lowest index acknowledges the request.

A write of the MDR (mask data register) value is also done at the address programmed in the MAR (mask address register). This allows the clean out of the RQ signal generated by the DMA by writing to its Interrupt Clear register.

A sequence with a hardware trigger from the DMA can be described as a sequence composed of the following steps:

1. Configure DMA for a peripheral (such as USART1) to memory transfer
2. Configure MDMA to be hardware triggered by configuring TSEL
3. Configure MAR and MDR
   a. In MAR, put the address of the DMA flag clear register
   b. In the MDR, put a value that corresponds to the flag to clear
4. Start the data transfer to the memory accessible by DMA1 (AHB SRAM)
5. Make the peripheral receive a data
6. The MDMA is triggered at the end of the transfer
7. The DMA transmit complete flag is automatically cleared by using MAR and MDR
8. Data transfer from memory (AHB SRAM) to memory only accessible by MDMA such as DTCM-RAM

1.3 Dynamic configuration

With the MDMA, both the source and destination transfers can address peripherals and memories in the entire 4-GByte area, at addresses comprised between 0x0000 0000 and 0xFFFF FFFF.

The MDMA can have access to the whole memory map. This allows the dynamic change of the configuration of peripherals such as I^2C or USART. DMA1 and DMA2 can be configured as well with a memory-to-memory transfer. All needed configurations can be placed in the RAM and then transferred without CPU intervention to the peripherals. It can occur after a trigger for example as presented in Section 1.2 The MDMA can be triggered by peripherals such as DMA1 and DMA2.

1.4 Block transfer and block repeat features

When the MDMA is triggered by hardware or software, it can trigger different kinds of transfer, which are called buffer transfer, block transfer, repeat block transfer, or whole data transfer.

This feature is set with the trigger mode (TRGM[1:0]) bit field located in the MDMA_CxTCR register (Transfer Configuration register of MDMA channel x).

Up to 64 Kbytes can be transferred with a single block. The size of this block is set in the block number of data bytes to transfer (BNDT[16:0]) located in the MDMA Channel x block number of data register (MDMA_CxBNDTR).

The number of blocks to be transferred is loaded in the block repeat count (BRC[11:0]) bit field located in the MDMA Channel x block number of data register (MDMA_CxBNDTR). Up to 4096 blocks can be sent. This field is decremented after each complete block transfer.

Also the source address register (MDMA_CxSAR) and destination address register (MDMA_CxDAR) are updated according to the block repeat source/destination mode (BRSUM/BRDUM) by adding or substracting the MDMA_CxSAR and MDMA_CxDAR respectively with the source address update value (SUV[15:0]) and the destination address value (DUV[15:0]), both located in the MDMA channel x Block Repeat address Update register (MDMA_CxBRUR).

When the block repeat count reaches 0, it means that the last block (or single block in case of non-repeat block transfer) is transferred.

1.5 Linked-list mode

A linked list is a linear collection of data elements, called nodes, each pointing to the next node by means of a pointer. The address of this pointer is stored in the CxLAR (Channel x Link Address Register). The structure pointed by this address must contain the ten MDMA configuration register values (CxTCR, CxBNDTR, CxSAR, CxDAR, CxBRUR, CxLAR, CxTBR, CxMAR and CxMDR) to be reloaded.

Note: CxLAR is reloaded as well to link to another node if its value is not 0. Otherwise, no register update is taking place. This mechanism occurs at the end of a (repeated) block transfer.

The linked-list mode allows the upload of a new MDMA configuration from the address given in the CxLAR register. This address must refer to a memory mapped on the AXI system bus.

Following this operation, the channel is ready to accept new requests, as defined in the block/repeated-block modes previously described, or continue the transfer if TRGM = 11.

Caution: The TRGM and SWRM values must not be changed when TRGM = 11.

A single request initiates the data array (collection of nodes) to be transferred until the linked-list pointer for the channel is null. The channel transfer complete of the last node is the end of transfer, unless both nodes are linked to each other; in such a case, the linked-list loops on to create a circular MDMA transfer.

The block size value is the length of the data block, which is described in a block structure of the MDMA linked list. It corresponds to one entry in the linked list.

Each channel can perform a linked-list transfer. When the transfer of the current data block (or last block in a repeat mode) is completed, a new block control structure is loaded from memory and a new block transfer is started.

It is a single block or the last block in a repeated block transfer: the next block information is loaded from the memory (using the linked-list address information, from the MDMA_CxLAR).

By setting the CTCIFx bit in the status register, and when the MDMA_CxBNDTR counter reaches zero, the block repeat counter is 0 and the linked-list pointer address is 0, and nn end of transfer is generated.

1.6 Up to 256 Mbytes per DMA request

The minimum amount of data to be transferred for each request (buffer size up to 128 bytes) is programmable. The total amount of data in a block is programmable up to 64 Kbytes and 4096 blocks (12-bit field):

$4096 \times 64\,Kbytes = 256\,Mbytes$

The use of the block repeat features allows a 256-Mbyte transfer.

For larger transfer sizes, the linked-list mechanism must be used as described in Section 1.5 Linked-list mode.

1.7 Non-contiguous data increment

When the step configured (half-word, word or double-word) is bigger than the data size transfer (byte, half-word or word), a non-contiguous data increment or decrement is obtained.

If the increment or decrement mode is enabled, the address of the next data transfer is the address of the previous one incremented or decremented by 1, 2, 4, or 8 depending on the increment size.

The increment or decrement step must at least be equal to the size of the data, which can be byte (8 bits), half-word (16 bits), word (32 bits), or double-word (64 bits) long.

Caution: If the increment mode is enabled and if the increment size is strictly inferior to the data size, the result is unpredictable. This applies to both the source and destination.

The source increment mode SINC[1:0], the destination increment mode DINC[1:0], the source increment size SINCOS[1:0], the destination increment size DINCOS[1:0], the source data size SSIZE[1:0] and the destination data size DSIZE[1:0] are all set by means of the the MDMA channel x Transfer Configuration register (MDMA_CxTCR).

A noncontiguous data increment occurs if the increment mode is enabled and if the increment size is strictly superior to the data size. For instance, if the size is programmed to be a byte and if the increment is programmed to be a word, then the bytes after the first one are transferred with a step of 4.

Note: Based on this separation, some more advanced packing and unpacking operations are available at software level. For instance, 2 × 16-bit data blocks may be interleaved together using two MDMA channels, in the destination memory, by simply programming the 2 channels with an increment step of 4 bytes and a data size of 16 bits together with a start address shifted by 2 between the two channels.

1.8 Data packing

Section 1.7 Non-contiguous data increment details the configuration of non-contiguous data steps. Furthermore, when source and destination data widths differ, the MDMA can pack and unpack the necessary data to optimize the bandwidth.

When the packing / unpacking feature is enabled with PKE (PacK Enable) in the MDMA_CxTCR (MDMA channel x Transfer Configuration Register), the source data is packed / unpacked into the destination data size. All data are right aligned, in little endian mode.

Data packing / unpacking is always done according to the little endian convention: the lower address in a data entity (double-word, word or half-word) always contains the lowest significant byte. This is independent of the address increment / decrement mode of both source and destination.

When the packing / unpacking feature is disabled and when the source size is the same as the destination size, the source data is written to the destination as is. If the sizes are not the same, two cases can occur:

- The source data size is smaller than the destination data size: source data are padded with zeros on the right or on the left according to the PAM (Padding/Alignement Mode) value with the sign extended or not.
- The source data size is larger than the destination data size: source data are truncated. The alignment is done according to the PAM[1] value. If right aligned, only the LSBs part of the source is written to the destination address. Otherwise, if left aligned, only the MSBs part of the source is written to the destination address. In both cases, the remainder part is discarded.

1.9 Little-endian and big-endian data format are supported

When an MDMA transfer occurs, it is possible to exchange the endianness of the data for double-word, word, or half-word data size. By default little endianness is preserved.

The Word Endianness eXchange (WEX) bit, the Half-word Endianness eXchange (HEX) bit, and the Byte Endianness eXchange (BEX) bit are located in the MDMA channel x control register (MDMA_CxCR).

When a data is exchanged, the higher address of the destination contains the data read from the lower address of the source.

The WEX is used to exchange words and is applicable to a destination with a double-word data size.

The HEX is used to exchange half word in each words and is applicable to word or double word.

The BEX is used to exchange byte in each half words and is applicable to half word, word or double word.

To obtain the big endianness, WEX, HEX, and BEX must be used simultaneously to completely reverse the byte order.

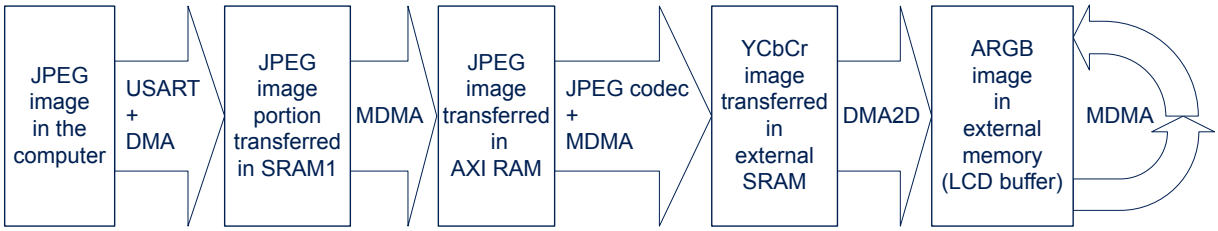
2 MDMA with USART, JPEG, and external RAM use cases

The demonstration examples provided in the X-CUBE-MDMA Expansion Package are developed for the STM32H743XI microcontroller on the STM32H743I-EVAL Evaluation board. They can easily be tailored for any other microcontroller in the STM32H7 Series.

In this chapter, the MDMA is used in conjunction with other masters and peripherals such as DMAs and JPEG to demonstrate how efficiently it can free CPU resources.

As an example, a JPEG image is sent with a USART and displayed on the screen of the STM32H743I-EVAL Evaluation board. Then, the joystick is used to move the image being displayed. The block diagram in Figure 4 provides an overview of the use case.

Figure 4. JPEG image use case block diagram



This use case is coded using the MCU Package for STM32H7 Series (STM32CubeH7) that can be dowloaded from *www.st.com*. It contains other useful MDMA examples.

2.1 MDMA with USART

With a hardware trigger from the DMA1 stream, the data go from USART1 to a buffer in AHB SRAM1 (D2 domain) and then to the AXI SRAM by means of the MDMA to be decoded with the JPEG codec. The buffer in SRAM1 is one-byte sized, and the USART is in circular mode, so the MDMA buffer transmission is used.

When a block is received (65536-byte size maximum), the external memory is no more filled. This is a non-repeat block transmission.

2.2 MDMA with JPEG

Once the AXI SRAM is filled, the JPEG codec is enabled. Then, the JPEG input buffer triggers MDMA transmission to send the JPEG file to the JPEG input buffer by means of the JPEG input FIFO not-full trigger. The JPEG output buffer triggers MDMA transmission as well to send the decoded data in YCbCr format to the external memory on the STM32H743I-EVAL board. Both JPEG input buffer and JPEG output buffer are 8-byte long.

YCbCr data are processed to ARGB by DMA2D (external memory to external memory) in order to be displayed on the screen.

2.3 MDMA with external SDRAM

In order to move the image on the LCD, it is displayed and managed by MDMA as a regular memory-to-memory word copy. The start address of the SDRAM is configured in the LCD-TFT display controller and hardware decoded with the flexible memory controller (FMC).

Each line is composed of pixels. Each pixel is 4 bytes in this case. Lines are not contiguous in the memory area. For this reason, a line is a block and there is an offset to go to the next line. A 320 × 240 regular image represents 240 blocks, each block being composed of 320 words. The first pixel of the screen (top-left corner) corresponds to the start address (the lowest address value) of the external memory. The screen has a resolution of 640 × 480 allowing the image to be moved inside the entire screen.

Upward move

To move the image upwards vertically, the process starts from the beginning of the image (its first pixel in the top-left image corner) and then increments the MDMA counter. The destination address is chosen slightly lower than the start address. The offset is a multiple of 640 in order to be vertically aligned. Therefore, the first block which corresponds to the first line is printed a few pixels higher. Then the block counter is incremented and the process is repeated line by line for the next lines in the entire image. The result obtained is that the image is slightly shifted upwards. In addition, some margin is taken to erase the previous image.

Downward move

To move the image downwards vertically, the process starts from the end of the image (its last pixel in the bottom-right image corner) and then decrements the MDMA counter. The destination address is chosen slightly higher than the start address. The offset is a multiple of 640 in order to be vertically aligned. Therefore, the first block which corresponds to the last line is printed a few pixels lower. Then the block counter is decremented and the process is repeated line by line for the previous lines in the entire image. The result obtained is that the image is slightly shifted downwards. Similarly, some margin is also taken to erase the previous image.

Left or right move

To move the image left or right horizontally is more complex. 240 blocks or slightly more are handled by the MDMA with a four-word size instead of 320. The destination address is chosen on the same line as the source to be horizontally aligned. 60 more runs complete the image.

Revision history

Table 2. **Document revision history**

Date	Version	Changes
21-Jun-2017	1	Initial release.
29-Jan-2020	2	Document scope extended to the STM32H7Axxx and STM32H7Bxxx microcontrollers: • Title updated • Updated Introduction, MDMA features, and The MDMA can be triggered by peripherals such as DMA1 and DMA2 • Added Figure 2. Master connections of the AHB/AXI matrix for STM32H7Axxx and STM32H7Bxxx
17-Jul-2020	3	Document scope extended to the STM32H72xxx and STM32H73xxx microcontrollers. Updated: • Introduction • New Figure 3. Master connections of the AHB/AXI matrix for STM32H72xxx and STM32H73xxx • Section 1.2 The MDMA can be triggered by peripherals such as DMA1 and DMA2

Contents

List of tables

List of figures