How to use the GPDMA for STM32 MCUs

Introduction

This application note relates to the general purpose DMA (GPDMA). The GPDMA is a system peripheral, and is a dual-port master on the AHB bus. It is used to transfer data between peripherals and/or memories via linked lists. All the GPDMA programmable transfers provide higher performance at system level, and unload the CPU of performing these data transfer tasks.

The aim of this document is not to rewrite the GPDMA dedicated section in the reference manual, but to provide some performance-oriented programming guidelines to the system developer.

This document is based on the combined capabilities of the GPDMA, and of the peripheral assisted by the GPDMA. It focuses on the key points to consider in order to optimize system performance, and to match the application requirements.

This application note explains the rationale, and gives recommendations about:

- GPDMA channel allocation
- GPDMA ports allocation
  - for the transfer from the memory-mapped source location
  - for the transfer to the memory-mapped destination location
- GPDMA transfer priority assignment
- GPDMA source/destination burst programming, in terms of data width and burst length

Table 1. Applicable products

Type	Products
Microcontrollers	• STM32H5 series • STM32H7R3/7S3 line • STM32H7R7/7S7 line • STM32U5 series

1 General information

This application note applies to the STM32 series microcontrollers that are Arm® Cortex® core-based devices.

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

Reference documents

[1] Reference manual *STM32H523/33xx, STM32H562/63xx, and STM32H573xx Arm®-based 32-bit MCUs* (RM0481)

[2] Reference manual *STM32H503 line Arm®-based 32-bit MCUs* (RM0492)

[3] STM32H503xx datasheet (DS14053)

[4] STM32H503xx erratasheet (ES0561)

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

[6] STM32H7Sxx8 datasheet (DS14359)

[7] STM32H7Rxx8 datasheet (DS14360)

[8] Reference manual *STM32U5 Series Arm®-based 32-bit MCUs* (RM0456)

[9] STM32U585xx datasheet (DS13086)

[10] STM32U575xx datasheet (DS13737)

2 GPDMA general guidelines

2.1 GPDMA overview and instances

The GPDMA controller is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked lists, upon the control of an off-loaded CPU. The GPDMA is a dual AHB master and system peripheral. Most of peripherals and memories are connected to it, giving a lot of flexibility and increasing system performance when data transfers are required. A linked list is a programmed data structure in the memory to prepare each GPDMA channel for chaining and scheduling DMA data transfers.

A GPDMA instance features up to 16 channels.

The following table depicts the GPDMA instances for each STM32 series.

Table 2. GPDMA instances

Product	GPDMA instances
STM32H5	GPDMA1 8 channels + GPDMA2 8 channels
STM32H7Rx/7Sx	GPDMA1 16 channels
STM32U5	GPDMA1 16 channels

In STM32H5 series :

- At the hardware level, there are 2 instances of the GPDMA with 8 channels each, named as GPDMA1 and GPDMA2 in the STM32H5 Series microcontrollers. The two instances are fully identical and equally connected to the AHB bus matrix and to the peripherals and memories, and any addressed AHB/APB target. They are also equally connected in terms of input signals like DMA requests or DMA triggers, and in terms of output signals like DMA channels transfer complete signals.
- At the software level, the programmer's view is then simplified with a set of 16 GPDMA channels, where GPDMA1 channels are indexed from 0 to 7 and GPDMA2 channels are indexed from 16 to 31. Consequently, in the following sections, the reader must understand that when it is written about a GPDMA channel x with x = 0 to 7, it does apply both to the channel x of GPDMA1, and to the channel x + 8 of GPDMA2.
  For more details about GPDMA 1 and 2, refer to the STM32H5 reference manual.

2.2 GPDMA channel allocation

The user must allocate one channel for a GPDMA transfer. The GPDMA implements a dedicated FIFO to a given GPDMA channel, in order to be able to handle concurrently the GPDMA transfers from the source (read access) and the GPDMA transfers to the destination (write access).

The unit of a FIFO cell is one byte. The FIFO size dictates the maximum DMA burst size (burst length x data width) that the channel can effectively perform. For a given channel, the maximum DMA burst size is half of its FIFO size. Generally the larger the burst is, the better the overall system performances are: higher throughput/ bandwidth transfers, lower system bus occupancy.

Given that the system bus is 32-bit word wide, programming a DMA source/destination data width of 32 bits (S/DDW_LOG2[1:0] in GPDMA_CxTR1) is recommended for a minimized bus utilization.

Table 3. GPDMA channel number, FIFO size, and addressing mode

Product	Channel number	FIFO size	Addressing mode
STM32H5	0 to 3 and 8 to 11	8 bytes (2 words)	linear
	4, 5 and 12, 13	32 bytes (8 words)	linear
	6, 7 and 14, 15		2D
STM32H7Rx/7Sx STM32U5	0 to 11	8 bytes (2 words)	linear
	12 to 15	32 bytes (8 words)	2D

The following channel allocation is recommended:

- Channels with an 8-byte FIFO must be allocated for AHB/APB peripheral to SRAM transfers (in both ways) unless a 2D addressing is required for the memory, or unless the AHB peripheral supports a burst request Programming burst as a single word (half of the FIFO size) is then recommended, unless the application requires to handle an 8- or 16-bit data width.
- Channels with 32-byte FIFO must be allocated for memory-to-memory transfers
  Programming bursts of 4 words (half of the FIFO size) is then recommended by default, for performances. These channels are also preferred for:
  - transfers from AHB peripherals supporting a burst request such as OCTOSPI, HASH, and ADC1. Half transfers from/to the peripheral are typically programmed as a burst. The (half) transfer from/to memory is then recommended to be as a 4-word burst.
  - transfers from/to some of the AHB peripherals with higher-bandwidth requirements. The (half) transfer from/to memory is then also recommended to be programmed as a 4-word burst.

Refer to the corresponding product reference manual to know their channel types and the FIFO sizes.

2.3 GPDMA port selection

The user must assign one port for the transfer from the source (SAP in GPDMA_CxTR1), and one port for the transfer to the destination (DAP in GPDMA_CxTR1). This assignment is dynamically updated via the next linked-list item and data structure before the execution of the next data transfer.

The bus topology can be summarized as follows (see Section 4 for more details):

- The GPDMA port 0 is directly connected to APB1 and APB2 peripherals, without crossing to the AHB matrix.
- The default slave of the AHB matrix is:
  - AHB1 peripherals for the GPDMA port 0
  - SRAM1 for the GPDMA port 1

It is recommended to use the two GPDMA master ports as follows:

- Port 0 is allocated for the (half) transfers from/to a peripheral (whatever it is an AHB or APB peripheral). The other (half) transfer, respectively to/from memory, is allocated to the other port 1. For APB1 and APB2 peripherals, port 0 avoids crossing through the interconnect matrix, and reduces the overall latency on the corresponding channel. This also reduces the AHB bus activity including and behind the interconnect bus matrix.
- Port 1 is allocated for memory-to-memory transfers (especially if accessing SRAM1).

The advantages of using port 0 for peripherals and port 1 for any memory, are:

- bandwidth balance over the two ports during peripheral to/from memory transfers
- avoiding bursts to memory from impacting directly the latency of a peripheral access

This is a typical and recommended configuration for performance. The user is of course free to select any port which do have access to the source location, and any port to the destination location.

The GPDMA allocated link port for loading the next linked-list item is defined by the user (LAP in GPDMA_CxCR) at channel level when the channel is not active. A next linked-list item is prepared and resides in the memory. Port 1 is then recommended to be allocated for loading the next linked-list item.

2.4 GPDMA channel priority

One priority value is assigned to each (half) transfer for competing versus other concurrent ones. The GPDMA arbiter can then grant and schedule the (half) transfer on one master port. At user level, this is prepared at channel level via PRIO[1:0] in GPDMA_CxCR when the channel is not active.

The GPDMA has two master ports for parallelization of AHB transfers. Simultaneous transfers via the two ports is possible. A GPDMA arbitration occurs for each port as follows:

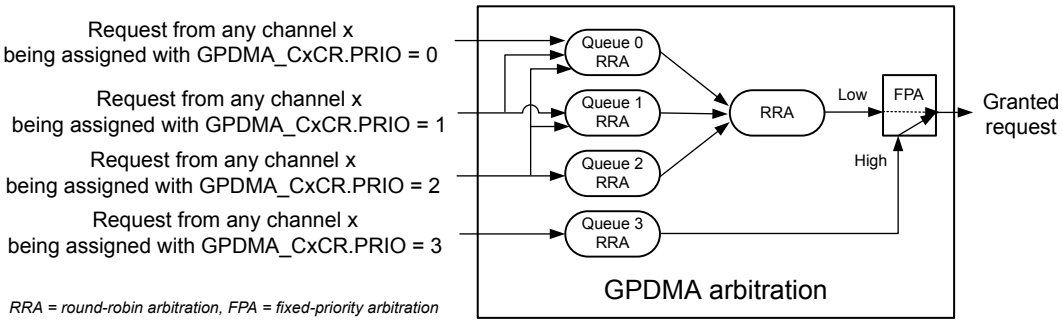
- a priority-based arbitration (see Figure 1) between the 8 possible requested FIFO-based read bursts
- a priority-based arbitration (see Figure 1) between the 8 possible requested FIFO-based write bursts
- a final round-robin arbitration stage between read and write

The GPDMA arbitration phase may introduce 1-cycle clock latency for the GPDMA to generate the AHB address of the granted burst on the allocated master port (see Section 2.5 for more information).

The GPDMA implements a programmable arbitration logic, that allows the user to adjust the channel bandwidth and latency, according to the following rules:

- The priority of the requested burst transfer is programmable from 0 to 3.
- Requests having the same priority are handled with a round-robin arbitration scheme.
- Priority 3 is recommended for time-sensitive requests because it is handled with a fixed higher priority scheme over priorities 0 to 2.
- The residual bandwidth is shared by requests of priority 0 to 2 by implementing a weighted round-robin allocation for these non time-sensitive channels.
- The different weights are monotonically resulting from the programmed channel priorities, queue 0 having the lowest weight.

Figure 1. GPDMA arbitration policy



The user needs to assign the correct level of priority to a GPDMA channel on which peripherals/memories are connected, in order to get an acceptable service quality for this channel:

- no timing error (no peripheral register underrun/overrun) on peripheral side
- acceptable latency between the data transfer request and the servicing of this request
- acceptable impact on service quality of the other channels

Channel priorities must be carefully assigned in order to fit the application timing requirements. A requested FIFO-based transfer of priority 0 to 2 that is ready to be scheduled again on a master port, may be delayed by the round-robin arbiter for the switch, and the execution of up to 7 concurrent (single/burst) transfers of priority 0 to 2. Such a request is also the first preempted by any time-sensitive requested transfers.

A GPDMA channel mapped with a request from a critical timer is typically allocated to the time-sensitive queue, so that the related TIMER registers can be updated with the lowest latency. GPDMA requests from other peripherals may be assigned a priority 2 or 1, and the memory-to-memory transfers may be finally assigned with the lowest priority 0 (best effort traffic). Peripherals mapped to a channel 12 to 15 and having a burst capability can be typically assigned to the intermediate priority 1.

This recommendation about channel priority may be tuned to fit specific application requirements (see the reference manual for more details about arbitration and priority).

A data transfer is performed at a linked-list level for a given channel. The user may have chained it with a next data transfer via the same channel, by having programmed a linked-list data structure in memory. Once the data transfer of a given linked-list item (LLI) is completed, the GPDMA reads/fetches automatically the next linked-list data structure, and updates internally its registers, before executing the next data transfer as described.
The link-transfer priority is given by the assigned priority to its channel, in the same way as the data transfer.
The link-transfer consists of a series of 32-bit single reads (up to six single reads for channel (0 to 5 and 8 to 13) for STM32H5 and 0 to 11 for others, and up to eight single reads for channel (6 to 7 and 14 to 15) for STM32H5 and 12 to 15 for others. Between every single 32-bit read for the LLI update, the GPDMA takes one extra clock cycle for the arbitration phase.

2.5 GPDMA burst

An elementary programmed data transfer is a GPDMA burst (burst of data read from the source, or burst of data written to the destination), and is defined by:

- a programmed data width: 8-, 16-, or 32-bit (via S/DDW_LOG2[1:0] in GPDMA_CxTR1)
- a programmed burst length: 1 to 64 (via S/DBL_1[5:0] in GPDMA_CxTR1)

The burst size can be separately programmed for the source and the destination.

A burst is a series of n = 1 to 64 beats. Each beat is one data transfer, and has the same data width. For example, a 4-word burst is a burst of four 32-bit words. A burst with a burst length of 1 is named single.

The GPDMA issues a beat with the data width such as programmed: the programmed data width is never changed by the GPDMA implementation.

The GPDMA does not always issue a burst with the burst length as programmed on the allocated AHB master port. A GPDMA burst is implemented in hardware in one of the following ways:

- with a same AHB transaction on the allocated master port
- by a series of bursts of lower length or/and singles, due to any of the following conditions:
  - Burst size > half of the FIFO size.
  - Block size (defined as the source block size) is not a multiple of the source data width.
  - AHB constraints:
    - on a 1-Kbyte address boundary crossing
    - burst that must be an incremental burst of 4, 8 or 16 beats

The GPDMA implementation guarantees the data integrity versus the programmed addressing, and maximizes the performance by implementing the largest allowed burst size compared to the programmed burst.

It is recommended to program a GPDMA burst of half of the FIFO size of the allocated GPDMA channel (typically a single word for channels (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others, and 4-word burst for channels (4 to 7) for STM32H5 and 12 to 15 for others.).

Table 4. Maximum achievable GPDMA burst length versus data width and channel number

Channel number	Data width access (bits)	Maximum burst length (beats)
STM32H5 0 to 3 or 8 to 11 (8-byte FIFO) STM32H7Rx/7Sx and STM32U5 0 to 11 (8-Byte FIFO)	8	Up to 4
	16	Up to 2
	32	1
STM32H5 4 to 7 or 12 to 15 (32-byte FIFO) STM32H7Rx/7Sx and STM32U5 12 to 15 (32-byte FIFO)	8	Up to 16
	16	Up to 8
	32	Up to 4

A GPDMA burst is an elementary chunk of data that is at fixed address or at a contiguously incremented address. For channels (6 to 7 and 14 to 15) for STM32H5 and 12 to 15 for others (2D addressing capable), a first address jump may be performed after a burst. For getting a 2D buffer in memory, the user may have to program a burst, which is different from the typical 4-word burst. The GPDMA implementation automatically optimizes the performances, and includes this programmed constraint.

One GPDMA arbitration clock cycle is added before each programmed burst, provided that this burst size is lower than half of the FIFO size. Each data transfer can be a single data transfer, with a data arbitration performed each time a data has to be sent on the AHB bus master, or grouped into a burst data transfer to send few data back-to-back, without any possible GPDMA preemption after a GPDMA single arbitration phase. This minimizes the data transfer time (overall latency), and maximizes the throughput and the bus efficiency.

The main advantages of a transfer considering burst with more than one beat/data are the following:

- The GPDMA arbitration is reduced to only one phase, which is applied for all the data defined by the burst size.
- The burst is not interruptible at the GPDMA (arbitration) level, provided that its size is lower than half of the FIFO size. The latency for a set of number of read/written data is reduced and earlier completed (except for the very first beat).
- If the targeted source or destination is an internal memory, the programmed addressed-incremented burst is implemented by incremented AHB bursts on the AHB bus. This consumes less bus cycles, and frees bus bandwidth for other possible concurrent masters, or enables the bus to be programmed at a lower bus frequency for a given traffic/bandwidth.
- If the targeted source or destination is a peripheral register at a fixed address supporting a burst (with a FIFO mechanism like ADC1, OCTOSPI, or HASH), program a GPDMA 4-word fixed burst (which is after GPDMA arbitration split to four single transfers for being AHB compliant) increases the back-to-back transfer throughput since the arbitration clock cycle penalty occurs once.

Having a GPDMA burst with an AHB burst transaction increases the latency to serve other concurrent GPDMA transfers over the same allocated AHB master port, meanwhile it is performed. All other GPDMA requested transfers on the same GPDMA port are pending during this time (as well as all other masters intending to read or write data from/to the same target peripheral/memory). It is recommended to manage the different application requirements by programming channel priorities, as highlighted in Section 2.4.

2.6 GDMA request

The GPDMA transfer can be initiated by setting a software request (via SWREQ in GPDMA_CxTR2), potentially associated to a hardware trigger defined by TRIGSEL[5:0] in the same register. This is the case for memory-to-memory transfers, which do not rely on any hardware request signal coming from the memory.

The hardware trigger may be driven by a peripheral (such as timers), by the GPDMA itself (channel transfer complete), or by other peripherals. Refer to the product reference manual for more details about the trigger functionalities, and more specifically in the GPDMA implementation section for the product-dependent list of triggers, which can be mapped to the GPDMA.

The memory-to-memory mode is also used for some timer configurations in DMA mode and for GPIOs configuration in DMA mode (see Section 3.4 and Section 3.6).

One GPDMA dead clock cycle is inserted between two back-to-back half data transfers (one dead cycle between two reads, one dead cycle between two writes) of a same channel in memory-to-memory mode.

As an alternative to a software request, a GPDMA transfer may be requested and initiated by an input hardware request coming from a peripheral, for peripheral-to-memory and memory-to-peripheral GPDMA transfers. The user must program SWREQ = 0 and REQSEL[6:0] in GPDMA_CxTR2 in order to select the proper hardware peripheral request among the product-dependent list of the mapped GPDMA request signals. Refer to the GPDMA implementation section of the product reference manual for this list and mapping.

Back-to-back data transfers to/from the peripheral are limited in performance by the fact that at least four AHB clock cycles must occur between two occurrences of a same peripheral request, due to the hardware protocol of the request/acknowledge between the peripheral and the GPDMA. This limitation does not happen for transfers to/from memory.

A hardware requested transfer can also be conditioned and associated to a hardware trigger, like for the memory-to-memory mode.

3 Peripherals, memories, and GPDMA configuration

Most of analog and digital peripherals in the STM32 devices with GPDMA have capabilities to handle data transfer without CPU intervention. They propose different possible configurations to fit with most of the application requirements.

The overall system performance results from the GPDMA configuration on each peripheral, depending on its location (bus type), its priority for the data transfer, and its capability to offer internal FIFOs for burst operation (the burst size is directly linked to the FIFO size of the peripheral, and the embedded FIFO connected to the associated GPDMA channel).

In the following sections, an exhaustive list of peripherals/memories embedded into the STM32 devices is presented to guide the developers for optimal GPDMA configuration, and to get the best benefit in term of performance.

For each peripheral, the following list of items is covered to help for peripheral and GPDMA configuration:

- Peripheral request signals to GPDMA
  - Most of peripherals requests are of single type or burst type.
  - On the contrary, the lptimer_ue request (lptimer update event) requires a block-type GPDMA specific programming. BREQ must be set in GPDMA_CxTR2. This BREQ field is present in any channel.
  - I3C receive data request to GPDMA (of single type) is specific: the number of data to be received during an I3C SDR private read message may be early completed by the external I3C device. In order to be supported by the GPDMA hardware, the field PFREQ must be set in DMA_CxTR2. It is to note that this field is present only on channels 0, 1, 8, and 9. This peripheral-flow control is different from any other peripheral DMA request (used in DMA-flow control): the programmed DMA block size (BNDT[15:0] of DMA_CxBR1) is used for the block-level transfer completion.
- Peripheral bus (which AHB/APB)
- Peripheral FIFO (unit cell, size)
- Peripheral register access
  - Most of the GPDMA requests are tightly coupled to one specific peripheral register.
  - On the contrary, timers offer the flexibility to address different peripheral registers to fit different application requirements.
- GPDMA read/write access
- GPDMA port preference if any
- GPDMA channel preference if any
- GPDMA data width: 8-, 16-, or 32-bit
- GPDMA burst length: 1 or 4

3.1 GPDMA configuration for internal memories

The STM32 devices embed a flash memory and multiple internal SRAMs. All these memories can be targeted as source or/and destination by the GPDMA to perform data transfer during respectively a requested memory-to-peripheral/peripheral-to-memory transfer.

A GPDMA programmed 32-bit data width is preferred to reduce the 32-bit AHB bus load on the allocated source/destination port to read from/write to the memory. This reduces the traffic to the memory by a ratio of four versus performing byte access, and by half versus handling half-words. This also reduces, in the same way by four or by half, the memory footprint, allowing a memory compacted with contiguous data. This configuration must be used as much as possible because the GPDMA, with its channel FIFO, can decorrelate the source data width versus the destination data width, and can perform FIFO queuing, data handling like packing and unpacking.

Using a different data width in source and destination provides constraints to the application:

- peripheral-to-memory mode: if 8-/16-bit reads are performed from the peripheral data register to the FIFO, meanwhile the FIFO performs byte-to-word packing and 32-bit writes are performed to the compacted memory, the application must guarantee that the read source block size is a multiple of 4 bytes (which may not be acceptable).
- memory-to-peripheral mode: if 32-bit reads are performed from the compacted memory to the FIFO, meanwhile the FIFO performs word-to-byte unpacking, and 8-bit writes are performed to the peripheral data register, the read source block size must be a multiple of 4 bytes.

If the application cannot guarantee that the buffer/block size is a multiple of 4 bytes or 2 half-words, then the user must program a lower data width than the 32-bit AHB bus, and the AHB bus load is not optimized. Another implementation alternative for the application is to split the block transfer into two LLI/block transfers, with one block multiple of 4 bytes for most of transfers, and another block with the residual byte(s).

For example, the I2C peripheral has a data register containing only 8-bit lsb data, and must be handled as a byte-stream on the peripheral side. The other transfer to/from the memory may take the benefit of the 32-bit AHB bus, performs 32-bit data width access, and a contiguously incremented memory address (provided that the allocated buffer/block size is a multiple of 4 bytes). This reduces by four the required bandwidth on the memory, and by four the memory footprint versus a byte-based addressing.

Another example is the ADC1 used in DMA mode with an 8-bit resolution mode:

- The ADC must be configured 8-bit reads to the fixed peripheral data register address (with a right alignment, must have a 32-bit aligned address). The benefit is to be able to push only significant bytes into the GPDMA channel FIFO. As a channel 0 to 11 has an 8-byte FIFO, the source burst can be programmed as a 4-byte burst (half size of the FIFO), which minimizes the GPDMA arbitration overhead.
- The memory can be accessed with 32-bit words, GPDMA packing, and GPDMA write bursts with a 32-bit data width, and a contiguously incremented memory address (provided that the allocated buffer/block size is a multiple of 4 bytes). This reduces by four the required bandwidth on the memory, and by four the memory footprint versus a byte-based addressing.

For peripheral-to-memory or memory-to-peripheral GPDMA transfers, the GPDMA transfer is initiated when the hardware request (selected by REQSEL[6:0] in GPDMA_CxTR2) is asserted from the peripheral, and after that the GPDMA arbiter acknowledges the peripheral request. Refer to the product reference manual for more details.

Back-to-back data transfers to/from the peripheral are limited in performance by the fact that at least four AHB clock cycles must occur between two occurrences of a same peripheral request, due to the hardware protocol of the request/acknowledge between the peripheral and the GPDMA. This limitation does not happen for transfers to/from memory.

All these memories can be used for memory-to-memory transfers, with a recommended 32-bit data width for source and destination. If a channel 12 to 15 is allocated, memory transfers can benefit from being programmed and implemented as 4-word bursts (half of the FIFO size), in order to get higher throughput and back-to-back data transfers.

For memory-to-memory transfers, channels 12 to 15 must be allocated if a 2D addressing mode is required.

The lowest priority (priority 0) is recommended for such a channel with memory-to-memory burst transfers in order to be used as best effort traffic, and preempted by the time-sensitive traffic (priority 3), but also other traffic to/from hardware peripherals (priority 1 and 2). See Section 2.2.

The memory-to-memory transfer does not rely on any hardware request directly from the memory. The GPDMA transfer is initiated by setting SWREQ in GPDMA_CxTR2, associated to potential triggers defined by TRIGSEL[5:0] in the same register. Refer to the product reference manual for more details.

The memory-to-memory mode can be triggered by a GPDMA trigger coming from a peripheral (such as timers), or thanks to some additional triggers sent by the GPDMA itself or coming from other peripherals. GPIOs are accessed in memory-to-memory mode.

3.2 GPDMA configuration for analog peripherals

3.2.1 Analog-to-digital converter ADC

The STM32H503 and STM32U535/U545/U575/U585 devices embed one analog-to-digital converter (ADC1). The STM32H573/563/562, STM32U59x/5Ax/5Fx/5Gx, and STM32H7Rx/7Sx devices embed two analog-to-digital converters (ADC1 and ADC2).

This analog-to-digital (A/D) converter implementation proposes a DMA mode. The GPDMA request signal is used by the hardware to request a new single converted sample to be read by the GPDMA whenever the peripheral needs it. The GPDMA reads the ADC_DR register. This ADC 16-bit or 32-bit data register must be accessed by a 32-bit word-aligned read access, with a programmed source data width being typically either 8-bit, 16-bit, or 32-bit and with a fixed addressing.

The table below summarizes the main peripheral characteristics: hardware request signal to the GPDMA, peripheral bus on which the ADC is connected, and peripheral register from/to which the GPDMA must be programmed to read or write.

Table 5. ADC peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
adcx_dma (x = 1, 2)	AHB2	None or 8-word FIFO depending on the STM32	Read ADC_DR	The ADC1/2 data register (ADC_DR) must be read with a 32-bit aligned address, whatever the data width.

The table below shows the supported and recommended GPDMA settings to be used for serving the ADC peripheral request: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length.

Table 6. GPDMA recommendation for ADC

GPDMA source port	GPDMA source burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	8-, 16-, or 32-bit	1 or 4	For STM32H5 (0 to 3 or 8 to 11) For others STM32 (0 to 11 in single data mode 12 to 15 in burst mode)	The default recommended setting balances bandwidth over each port (source port 0 to read from peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. For STM32H5, it is better to use channel 0 to 3 or 8 to 11 to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For others STM32 , For improved burst performances, the channel allocation can be one of channels 12 to 15 (with 8-word FIFO).

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory if no specific requirements.*

The ADC generates an analog watchdog trigger signal which can be used as trigger for any GPDMA transfer.

3.2.2 Analog-to-digital converter ADC4

This analog-to-digital (A/D) converter implementation only in STM32U5 proposes a DMA mode.

The GPDMA request signal is used by the hardware to request a new single converted sample to be read by the GPDMA whenever the peripheral needs it. The GPDMA reads the ADC_DR register.

This ADC 16-bit data register must be accessed by a 32-bit word-aligned read access, with a programmed source data width being typically either 8- or 16-bit, and with a fixed addressing.

The ADC4 implementation uses no FIFO for ADC_DR. No burst performance can be achieved for the read access to the ADC1. A channel 0 to 11 can be allocated.

The table below summarizes the main peripheral characteristics: hardware request signal to the GPDMA, peripheral bus on which the ADC4 is connected, and peripheral register from/to which the GPDMA must be programmed to read or write.

Table 7. ADC4 peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
adc4_dma	AHB3	None	Read ADC_DR	The ADC4 data register (ADC_DR) must be read with a 32-bit aligned address, whatever is the data width.

The table below shows the supported and recommended GPDMA settings to be used for serving the ADC1/ ADC12 peripheral request: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length.

Table 8. GPDMA recommendation for ADC4

GPDMA source port	GPDMA source burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	8-, 16-, or 32-bit	1	0 to 11 typically	The default recommended setting balances bandwidth over each port (source port 0 to read from peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel 0 to 11 to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations.

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory, if no specific requirements.*

The ADC4 generates an analog watchdog trigger signal (adc4_awd1) which can be used as trigger for any GPDMA transfer.

3.2.3 Digital-to-analog converter DAC

The STM32 devices have a digital-to-analog (D/A) converter DAC except for STM32H7Rx/7Sx.

The digital-to-analog converter DAC is DMA capable without embedded FIFO. Even if the DAC has a dual channel GPDMA capability (completely independent), it can also use a single GPDMA channel to transfer within a single GPDMA channel the data to the dual converter. The DAC proposes then a GPDMA optimization reducing the number of GPDMA channel to use from two to one. It also reduces the number of transactions on the AHB bus by packing the data of the two converters in a single data transfer (in dual mode).

Table 9. DAC peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
dac_ch1_dma dac_ch2_dma	AHB2	None	Write (any) DAC_DHRyRx with y = 12, 8 and x = 1, 2	The DAC data hold register must be written at the specific 32-bit aligned address, depending on the used DAC format (8- or 12-bit, left-aligned or right-aligned) and depending on the considered channel (1 or 2).

The table below shows the supported and recommended GPDMA settings to be used for serving the DAC peripheral request: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 10. GPDMA recommendation for DAC

GPDMA destination port	GPDMA destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	8-, 16-, or 32-bit	1	For STM32H5 (0 to 3 or 8 to 11) For others STM32 (0 to 11)	The default recommended setting balances bandwidth over each port (source port 0 to read from peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and (0 to 11) for other STM32 to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performances, the channel allocation can be one of channels 4 to 7 or 12 to 15 (with 8-word FIFO).

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory, if no specific requirements.*

3.3 GPDMA configuration for graphic peripherals

3.3.1 DCMI (digital camera interface)

Only STM32H573/563/562 and STM32U5 devices embed the DCMI.

The DCMI is not burst capable when doing GPDMA transfers, even if an embedded FIFO is present. All transfers need to be handled in single data mode by the GPDMA controller since the DCMI sends a DMA request each time 32-data are available.

The GPDMA reads the 32-bit DCMI_DR register. This 32-bit data register must be accessed with a programmed source data width being 32-bit and with a fixed addressing.

No burst performance can be achieved for the read access to the DCMI. A channel 0 to 11 for STM32U5 and 0 to 3 or 8 to 11 for the other STM32 can be allocated, unless a channel 0 to 11 for STM32U5 and 4 to 7 or 12 to 15 for other STM32 is free/unallocated and the required performances in terms of bandwidth are significant for this AHB peripheral. A channel 12 to 15 for STM32U5 and 4 to 7 or 12 to 15 for other STM32 may be also allocated if the accessed memory is external.

The table below summarizes the main peripheral characteristics: hardware request signal to the GPDMA, peripheral bus on which the peripheral is connected, and peripheral register from/to which the GPDMA must be programmed to read.

Table 11. DCMI peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
dcmi_dma	AHB2	8-word FIFO	Read 32-bit DCMI_DR	The DCMI can group two pixels within a single data transfer (data packing), depending on the image format sent by the camera sensor, for optimized utilization of the 32-bit AHB bus.

The table below shows the supported and recommended GPDMA settings to be used for serving the DCMI peripheral request: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 12. GPDMA recommendation for DCMI

GPDMA source port	GPDMA source burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	32-bit	1	0 to 3 or 8 to 11 and 0 to 11 for STM32U5 typically 4 to 7 or 12 to 15 especially if remains unallocated/free channels for such AHB peripherals which may require significant bandwidth	The default recommended setting balances bandwidth over each port (source port 0 to read from peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel 0 to 3 or 8 to 11 and 0 to 11 for STM32U5 to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performance, the channel allocation can be one of channels (with 8-word FIFO).

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory, if no specific requirements.*

3.3.2 JPEG codec

This section only applies to STM32U5Fx/5Gx devices.

The hardware 8-bit JPEG codec encodes uncompressed image data streams or decodes JPEG-compressed image data streams. This peripheral has a DMA mode for streaming the input/output buffer from/to memory. The JPEG is burst capable, and includes a separated 16-word FIFO for input and output.

When the JPEG input FIFO is (at least) half empty (8-word free space), a new peripheral request jpeg_rx_dma is asserted: the DMA can write eight times the JPEG_DIR input data register.

When the JPEG output FIFO is half full (it contains an 8-word data), a new peripheral request jpeg_tx_dma is asserted: the DMA can read eight times the JPEG_DOR output data register.

An 8-word fixed burst must be programmed on the DMA side to write/read the JPEG input/output data register, whatever the DMA channel allocation. GPDMA transfers are then implemented by a series of either single words or 2-word, 3-word, or 4-word DMA bursts. For improved burst performances (with lower GPDMA arbitration cycles), one GPDMA channel from 12 to 15 (8-word FIFO) must be allocated.

For JPEG encoding, the output JPEG-compressed stream must be managed with peripheral-flow control. The allocated DMA channel x must be programmed with DMA_CxTR2.PFREQ = 1.

The table below summarizes the main peripheral characteristics: hardware request signal to the GPDMA, peripheral bus on which the peripheral is connected, and peripheral register from/to which the GPDMA must be programmed to read/write.

Table 13. JPEG peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
jpeg_rx_dma	AHB1	16-word input FIFO	Write 32-bit JPEG_DIR	Request for 8 words as soon and as long as the input FIFO level is half empty (8-word space)
jpeg_tx_dma		16-word output FIFO	Read 32-bit JPEG_DOR	Request for 8 words as soon and as long as the input FIFO level is half full (8-word)

The table below shows the supported GPDMA settings to be used for serving the JPEG peripheral request in terms of data width and burst length.

Table 14. GPDMA recommendation for JPEG

GPDMA source/ destination port	GPDMA source/destination burst		Comments
	GPDMA data width	GPDMA burst length	
Port 0[1]	32-bit	8	The default recommended setting balances bandwidth over each port (port 0 for peripheral, port 1 for the memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. During JPEG encode, the allocated DMA channel x for the output compressed stream must be programmed with the peripheral flow control (DMA_CxTR2.PFREQ = 1). An 8-word burst must be DMA programmed for any channel. The DMA can issue a single word or a 2-, 3-, or 4-word burst (or an 8-word only for HPDMA channels 12 to 15). For improved burst performance (lower GPDMA arbitration), one of the GPDMA channels 12 to 15 can be allocated.

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory, if no specific requirements.*

3.4 GPDMA configuration for communication, audio and mathematical peripherals

3.4.1 Peripheral synchronous slave interface (PSSI)

Only STM32H573/563/562, STM32H7Rx/7Sx, and STM32U5 devices embed the PSSI.

The PSSI is not burst capable when doing GPDMA transfer, even if an embedded FIFO is present. All transfers must be handled in single data mode by the GPDMA controller, since the PSSI sends a new GPDMA request each time a data is needed to be read/written, depending on the PSSI receive/transmit mode. The GPDMA performs a 32-bit aligned access to read/write from/to the PSSI_DR register. This 32-bit data register can be accessed with a programmed source/destination data width being either 8-, 16-, or 32-bit, and with a fixed addressing. It is recommended to use a 32-bit data read/write to PSS_DR in order to optimize bus bandwidth and reduce the bus load.

The PSSI is exclusive with the DCMI since they share the same kernel except for STM32H7Rx/7Sx have only PSSI without DCMI.

No burst performance can be achieved for the read/write access to the PSSI. A channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others can be allocated, unless a channel 4 to 7 (only for STM32H5) or 12 to 15 is free/unallocated and the required performances in terms of bandwidth are significant for this AHB peripheral. A channel 4 to 7 (only for STM32H5) or 12 to 15 may be also allocated if the accessed memory is external.

The table below summarizes the main peripheral characteristics: hardware request signal to the GPDMA, peripheral bus on which the peripheral is connected, and peripheral register from/to which the GPDMA must be programmed to read/write.

Table 15. PSSI peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
pssi_dma	AHB2	8-word FIFO	Read/ write 32-bit PSSI_DR	The PSSI can transmit or receive parallel data. Must be read/ write with a 32-bit aligned address, whatever is the data width.

The table below shows the supported and recommended GPDMA settings to be used for serving the PSSI peripheral request: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 16. GPDMA recommendation for PSSI

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	32-bit (recommended)	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others 4 to 7 (only for STM32H5) or 12 to 15 especially if remains unallocated/free channels for such AHB peripherals which may require significant bandwidth	For optimized bus utilization, it is recommended to transfer words from/to PSS_DR. The default recommended setting balances bandwidth over each port (source port 0 to read from peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performances, the channel allocation can be one of channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory, if no specific requirements.*

3.4.2 Inter-integrated circuit interface (I2C)

The STM32H5 devices embed two I2Cx instances (x = 1, 2) on APB1 bus, and the STM32H573/563/562 devices embed two additional I2Cx instances (x = 3, 4) on APB3 bus.

The STM32U5 devices embed three I2Cx instances (x = 1, 2, 4) on APB1 bus, and one I2C3 instance on APB3 bus. The STM32U59x/5Ax/5Fx/5Gx devices embed two additional I2Cx instances (x = 5, 6) on APB1.

The I2C is not burst capable when doing GPDMA transfer. All transfers must be handled in single data mode by the GPDMA controller, since the I2C sends a new GPDMA request each time a byte data needs to be written/read, or a control word is needed to be written, depending on the considered GPDMA request signal.
The GPDMA performs a 32-bit aligned access to the related data or control register.

The 8-bit data receive register must be read with a programmed 8-bit source data width, with a fixed addressing, and a 32-bit aligned access.

The 8-bit data transmit register must be written with a programmed 8-bit destination data width, with a fixed addressing, and a 32-bit aligned access.

The 32-bit control word must be written with a programmed 32-bit destination data width, and with a fixed addressing (if consecutive updates are required).

One of the channels (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others can be allocated (as well as one of the channels 4 to 7 (only for STM32H5) or 12 to 15, especially if the memory to be written is an external memory).

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, the peripheral bus on which the peripheral is connected, and the corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 17. I2C peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
i2c_rx_dma	APB1 for I2C1/2 3 for STM32H7Rx, 4/5/6 for STM32U5 APB3 for I2C3 for STM32H5 and STM32U5, 4 for STM32H5	None	Read 8-bit I2C_RXDR	Request for a new received data byte to be read by the GPDMA. Must be read with a 32-bit aligned address.
i2c_tx_dma			Write 8-bit I2C_TXDR	Request for a new data byte to be written by the GPDMA for transmission. Must be read with a 32-bit aligned address.
i2c_evc_dma			Write 32-bit I2C_CR2	Request for a new command word to be written by the GPDMA.

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 18. GPDMA recommendation for I2C

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	i2c_tx, i2c_rx: 8-bit i2c_evc: 32-bit	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	GPDMA1 port 0 is optimized to connect the APB1 I2Cx since there is a direct connection (at architecture level) from the GPDMA to the APB bridge on which the I2C is connected. There is less latency and fast response time for a given priority level The default recommended setting also applies to APB3 because it balances bandwidth over each port (port 0 for peripheral access, port 1 for memory access). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performances, the channel allocation can be one of the channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *Port 0 is selected for reading/writing APB1 peripheral register (direct connection), and port 1 for accessing to memory. By default, and if no specific requirements, it can be the same for APB3 (balanced traffic over the two ports).*

3.4.3 Improved inter-integrated circuit interface (I3C)

The STM32H5/H7Rx/H7Sx devices embed one I3C instance (I3C1) in APB1. Additionally, the STM32H503 devices embed another I3C instance (I3C2) in APB3.

The I3C interface handles the communication between this device and others: sensors and host processor(s), connected on an I3C bus. The I3C bus is a two-wire, serial single-ended, multidrop bus, intended to improve a legacy I2C bus.

The I3C peripheral is not burst capable when doing GPDMA transfer. All transfers must be handled in single data mode by the GPDMA controller, since the I3C peripheral sends a new GPDMA request each time:

- either a control word is needed to be written,
- or a status word is needed to be read,
- or a data byte/word is needed to be written or read depending on the considered GPDMA request signal.

The GPDMA performs a 32-bit aligned access to the related data or control register.

The 32-bit control word must be written with a programmed 32-bit destination data width, and with a fixed addressing.

The 32-bit status word must be read with a programmed 32-bit source data width, and with a fixed addressing.

The data transmit register can be accessed by bytes (via I3C_TDR) or by words (via I3C_TDWR). It must be written with a programmed respectively 8-bit or 32-bit destination data width, with a fixed addressing, and a 32-bit aligned access.

The allocated DMA channel for the data receive bytes/words must be programmed with PFREQ = 1 in DMA_CxTR2. This is to support that an external I3C device may early complete the data transmission. Since this PFREQ field is present only in channels 0, 1 and 8, 9, one of these channels must be allocated.

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, the peripheral bus on which the peripheral is connected, and the corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 19. I3C peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
i3c_rx_dma	APB1 for I3C1 APB3 for I3C2	8-byte RX-FIFO	Read 8-bit I3C_RDR or 32-bit I3C_RDWR	Received data must be read with a 32-bit aligned address.
i3c_tx_dma	None	8-byte TX-FIFO	Read 8-bit I3C_TDR or 32-bit I3C_TDWR	To be transmitted data must be written with a 32-bit aligned address.
i3c_tc_dma		2-word C-FIFO	Write 32-bit I3C_CR	To be transmitted control word.
i3c_rs_dma		2-word S-FIFO	Read 32-bit I3C_SR	Received status word

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 20. GPDMA recommendation for I3C

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	i3c_tx_dma/ i3c_rx_dma: 8-bit or 32-bit i3c_tc_dma/ i3c_rs_dma: 32-bit	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	GPDMA port 0 is optimized to connect the APB1 I3C1 since there is a direct connection (at architecture level) from the GPDMA to the APB bridge on which the I3C is connected. There is less latency and fast response time for a given priority level. The default recommended setting also applies to APB3 because it balances bandwidth over each port (port 0 for peripheral access, port 1 for memory access). The GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performances, the channel allocation can be one of the channels 4 to 7 (only for STM32H5) or 8 to 15 (with 8-word FIFO).

1. *Port 0 is selected for reading/writing APB1 peripheral register (direct connection), and port 1 for accessing to memory. By default, and if no specific requirements, it can be the same for APB3 (balanced traffic over the two ports).*

3.4.4 Universal synchronous/asynchronous/low-power receiver transmitter (USART/UART/LPUART)

The STM32 devices embed multiple USART/UART instances. Each instance can have a GPDMA channel linked to the transmit or receive queue.

This peripheral is not burst capable when doing GPDMA transfer. All transfers must be handled in single data mode by the GPDMA controller, since the USART/UART sends a new GPDMA request each time a data needs to be written/read, depending on the considered GPDMA request signal. The GPDMA performs a 32-bit aligned access to the related data register.

The 9-bit data receive register must be read with a programmed 8-, 16-, or 32-bit source data width, with a fixed addressing, and with a 32-bit aligned access, depending on the format.

The 9-bit data transmit register must be written with a programmed 8-, 16-, or 32-bit destination data width, with a fixed addressing, and with a 32-bit aligned access, depending on the format.

One of the channels (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others can be allocated (as well one of the channels 4 to 7 (only for STM32H5) or 12 to 15, especially if the memory to be written is an external memory).

The USART/UART/LPUART have transmit and receive FIFO in order to avoid as much as possible overrun/ underrun situation which may break the data transfer flow. The table below summarizes the main peripheral characteristics: the different hardware request signals to the GPDMA, the peripheral bus on which the peripheral is connected, and the corresponding peripheral register from/to which the GPDMA must be programmed to read/ write.

Table 21. U(S)ART and LPUART peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
(lp)u(s)art_rx_dma	APB1 for U(S)ART2/3/4/5/6/7/8/9/10/11/12 for STM32H5, 2/3/4/5/7/8 for STM32H7Rx/7Sx, 1/3/4/5/6 for STM32U5 APB2 for USART1 for STM32H5 and STM32H7Rx/7Sx, USART2 for STM32U5 APB3 for LPUART1 for STM32H5 and STM32U5. APB4 LPUART1 for STM32H7Rx/7Sx	8x 12-bit width RXFIFO (9 bits for Rx data + 3 bits for errors) 8x 9-bit width TXFIFO(1)	Read USART_RDR or LPUART_RDR	Request for a new received data to be read by the GPDMA. Must be read with a 32-bit aligned address, whatever is the data width.
(lp)u(s)art_tx_dma			Write USART_TDR or LPUART_TDRR	Request for a new data to be written by the GPDMA for transmission. Must be written with a 32-bit aligned address, whatever is the data width.

1. *FIFO can be enabled/disabled by software.*

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 22. GPDMA recommendation for U(S)ART and LPUART

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0(1)	16- or 32-bit read/write in 9-bit mode and 8-, 16-, or 32-bit read/write in 8-bit and 7-bit modes	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	GPDMA1 port 0 is optimized to connect the APB1/APB2 U(S)ART since there is a direct connection (at architecture level) from the GPDMA to the APB bridge on which the U(S)ART is connected. There is less latency and fast response time for a given priority level. The default recommended setting also applies to APB3 because it balances bandwidth over each port (port 0 for peripheral access, port 1 for memory access). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performances, the channel allocation can be one of channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *Port 0 is selected for accessing APB1/APB2 peripheral register (direct connection), and port 1 for accessing to memory. By default, and if no specific requirements, it can be the same for APB3 (balanced traffic over the two ports).*

3.4.5 Serial peripheral interface (SPI)

In STM32 devices embed multiple USART/UART instances. All of them can have a GPDMA channel linked to the transmit or receive queue.

The peripheral is not burst capable when doing GPDMA transfer. All transfers must be handled in single data mode by the GPDMA controller, since the peripheral sends a new GPDMA request each time a data is needed to be written/read, depending on the considered GPDMA request signal. Then GPDMA performs a 32-bit aligned access to the related data register.

The data receive register must be read with a programmed 8-, 16-, or 32-bit source data width, with a fixed addressing, and a 32-bit aligned access.

The data transmit register must be written with a programmed 8-, 16-, or 32-bit destination data width, with a fixed addressing, and a 32-bit aligned access.

The SPI has a transmit and a receive FIFO in order to avoid as much as possible overrun/ underrun situation which may break the data transfer flow.

The SPI can pack/unpack the received/to-be-transmitted byte-stream versus the written/read 32-bit word stream at the data register level. It is recommended to optimize the data transfer to reduce the overall system latency, and to decrease the bus bandwidth on the GPDMA master port (recommended port 0), which is allocated to access the SPI_TXDR/SPI_RXDR register. Decrease the load on the APB peripheral bus is also recommended when applicable. It is recommended to perform 32-bit data width access to the peripheral register to decrease by a ratio of 4 the bus traffic vs a byte-stream access on the peripheral side.

A channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others can be allocated (as well a channel 4 to 7 (only for STM32H5) or 12 to 15, especially if the memory to be written is an external memory).

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, peripheral bus on which the peripheral is connected, and the corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 23. SPI peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
spi_rx_dma	STM32H5 (APB1 for SPI1/4/6 APB2 for SPI2/3 APB3 for SPI5) STM32U5 (APB1 for SPI1/2 APB3 for SPI3) STM32H7Rx/7Sx (APB1 for SPI2/3 APB2 for SPI1/4/5 APB4 for SPI6)	16-byte RXFIFO and TXFIFO for SPI1 and SPI2 8-byte RXFIFO and TXFIFO for SPI3(1)	Read SPI_RXDR	Request for a new received data to be read by the GPDMA. Must be read with a 32-bit aligned address, whatever is the data width.
spi_tx_dma			Write SPI_TXDR	Request for a new data to be written by the GPDMA for transmission. Must be written with a 32-bit aligned address, whatever is the data width.

1. *FIFO can be enabled/disabled by software.*

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 24. GPDMA recommendation for SPI

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	32-bit recommended. If not applicable, 8-, or 16-bit.	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	GPDMA1 port 0 is optimized to connect the APB1 SPI since there is a direct connection (at architecture level) from the GPDMA to the APB bridge on which the SPI is connected. There is less latency and fast response time for a given priority level. The default recommended setting also applies to APB3 because it balances bandwidth over each port (port 0 for peripheral access, port 1 for memory access). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performances, the channel allocation can be one of channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *Port 0 is selected for accessing APB1 peripheral register (direct connection), and port 1 for accessing to memory. By default, and if no specific requirements, it can be the same for APB3 (balanced traffic over the two ports).*

3.4.6 Serial audio interface (SAI)

The STM32H573/563/562 devices embed two SAI instances (SAI1 and SAI2) connected to APB2. Each SAI has two subblocks A and B. Each subblock has a dedicated GPMDA request, its own FIFO, and can have one GPDMA channel.

The peripheral is not burst capable when doing GPDMA transfer. All transfers must be handled in single data mode by the GPDMA controller, since the peripheral sends a new GPDMA request each time an audio data is needed to be written/read. Then GPDMA performs a read/write access to the related 32-bit data register, with a programmed source/destination data width of 32-bit and a fixed addressing.

One of the channels (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others can be allocated (as well one of the channels 4 to 7 (only for STM32H5) or 12 to 15, especially if the memory to be written is an external memory).

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, peripheral bus on which the peripheral is connected, and the corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 25. SAI peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
sai_a_dma	APB2	8-word FIFO	Read/write 32-bit SAI_ADR	Subblock A requests for a new 32-bit data to be read/written by the GPDMA.
sai_b_dma			Read/write 32-bit SAI_BDR	Subblock B requests for a new 32-bit data to be read/written by the GPDMA.

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 26. GPDMA recommendation for SAI

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	32-bit	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	GPDMA1 port 0 is optimized to connect the APB2 SAI since there is a direct connection (at architecture level) from the GPDMA to the APB bridge on which the SAI is connected. There is less latency and fast response time for a given priority level. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performances, the channel allocation can be one of channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *Port 0 is selected for accessing APB2 peripheral register (direct connection), and port 1 for accessing to memory.*

3.4.7 SPDIF receiver interface (SPDIFRX)

There is one instance of the SPDIFRX in the STM32H7Rx/7Sx devices.

The SPDIFRX peripheral is designed to receive an S/PDIF flow compliant with IEC-60958 and IEC-61937. These standards support simple stereo streams up to high sample rate, and compressed multichannel surround sound, such as those defined by Dolby or DTS.

This peripheral can handle two DMA requests, handled over two DMA channels:

- A DMA channel dedicated to the transfer of audio samples
- A DMA channel dedicated to the transfer of IEC60958 channel status and user Information

The DMA mode for the data can be enabled for reception by setting the RXDMAEN bit in the SPDIFRX_CR register. In this case, as soon as the SPDIFRX_FMTx_DR is not empty, the SPDIFRX interface sends a transfer request to the DMA. The DMA reads the data received through the SPDIFRX_FMTx_DR register.

For the channel status and user information, the DMA read he SPDIFRX channel status register SPDIFRX_CSR.

The peripheral is not burst capable, and DMA must be programmed for reading 32-bit single words.

The table below summarizes the main peripheral characteristics: hardware request signal to the GPDMA, peripheral bus on which the SPDIFRX is connected, and peripheral register from/to which the GPDMA must be programmed to read or write.

Table 27. SPDIFRX peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
spdifrx_dat_dma	APB1	None	32-bit read to SPDIFRX_FMTx_DR	Request to read the 32-bit data register.
spdifrx_ctrl_dma			32-bit read to SPDIFRX_CSR	Request to read the 32-bit channel status register.

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 28. GPDMA recommendation for SPDIFRX

GPDMA source port	GPDMA source burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	32-bit	1	0 to 11 typically	The default recommended setting also balances bandwidth over each port (source port 0 to read from peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel 0 to 11 to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations.

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory if no specific requirements.*

3.4.8 USB Type-C®/USB Power Delivery interface (UCPD)

The UCPD has one GPDMA request for the transmission, and another one for the reception.

The peripheral is not burst capable when doing GPDMA transfer and does not embed any FIFO. All transfers must be handled in single data mode by the GPDMA controller, since the peripheral sends a new GPDMA request each time a data is needed to be written/read. Then GPDMA performs a read/write access to the related 32-bit data register, with a programmed source/destination data width of 8-bit and a fixed addressing at the 32-bit aligned register address.

One of the channels (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others can be allocated (as well one of the channels 4 to 7 (only for STM32H5) or 12 to 15, especially if the memory to be written is an external memory).

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, the peripheral bus on which the peripheral is connected, and the corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 29. UCPD peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
ucpd_rx_dma	APB1	8-word FIFO	Read 8-bit UCPD_RXDR	Request for a new received 8-bit data to be read by the GPDMA. Must be read with a 32-bit aligned address.
ucpd_tx_dma			Write 8-bit UCPD_TXDR	Request for a new 8-bit data to be written by the GPDMA for transmission. Must be written with a 32-bit aligned address.

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 30. GPDMA recommendation for UCPD

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	8-bit	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	GPDMA1 port 0 is optimized to connect the APB1 UCPD since there is a direct connection (at architecture level) from the GPDMA to the APB bridge on which the UCPD is connected. There is less latency and fast response time for a given priority level. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performances, the channel allocation can be one of channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *Port 0 is selected for accessing APB1 peripheral register (direct connection), and port 1 for accessing to memory.*

3.4.9 Audio/multi-function digital filter (ADF/MDF)

The STM32U5 devices have both ADF and MDF, while the STM32H7Rx/7Sx devices have ADF only.

ADF and MDF have DMA capabilities to unload the software for the data transfer read from the peripheral and written to the memory. The MDF is an AHB1 peripheral, and the ADF an AHB3 peripheral for STM32U5 devices and AHB1 peripheral for STM32H7Rx/7Sx devices. There is one GPDMA request per audio digital filter: one for ADF, six for MDF.

The ADF/MDF are not burst capable when doing GPDMA transfers, even if an embedded FIFO for each filter is present. All transfers need to be handled in single data mode by the GPDMA controller since the ADF/MDF sends a DMA request each time one 32-bit register data (containing one 24-bit sample) is available, or two 32-bit register data (containing two 32-bit samples) are available, depending on the programmed FIFO threshold. Then GPDMA reads the 32-bit ADF_DFLT0DR / MDF_DFLTyDR (y = 0 to 7) digital filter 0 data register. This 32-bit data register must be accessed with a programmed source data width being 32-bit and with a fixed addressing.

The embedded FIFO is in the audio kernel clock, hence a few extra latencies is added for resynchronization with the AHB clock domain.

A channel 0 to 11 can be allocated. A channel 12 to 15 may also be allocated if the accessed memory is external.

The table below summarizes the main peripheral characteristics: hardware request signals to the GPDMA, peripheral bus on which the peripheral is connected, and peripheral register from/to which the GPDMA must be programmed to read.

Table 31. ADF/MDF peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
adf1_flt0_dma	AHB3 for STM32U5 AHB3 for STM32H7Rx/7Sx	8x 24-bit width FIFO	Read 32-bit ADF_DFLT0DR	Request for a new received audio data from filter 0 to be read by the GPDMA
mdf_flt0_dma mdf_flt1_dma mdf_flt2_dma mdf_flt3_dma mdf_flt4_dma mdf_flt5_dma	AHB2	8x 24-bit width FIFO for each filter	Read corresponding 32-bit MDF_DFLTyDR (y = 0 to 5)	Request for a new received audio data from filter y (y = 0 to 5) to be read by the GPDMA

The table below shows the supported and recommended GPDMA settings to be used for serving the ADF/MDF peripheral request: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 32. GPDMA recommendation for ADF/MDF

GPDMA source port	GPDMA source burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0 (1)	32-bit	1	0 to 11 typically	The default recommended setting also balances bandwidth over each port (source port 0 to read from peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel 0 to 11 to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations For improved burst performances, the channel allocation can be one of channels 12 to 15 (with 8-word FIFO).

1. *If no specific requirements, by default, port 0 is selected for reading peripheral register, and port 1 for writing to memory.*

3.4.10 Filter math accelerator (FMAC)

The STM32H573/563/562 and STM32U5 devices embed one FMAC instance.

The FMAC performs 16-bit fixed-point arithmetic operations on vectors. It is GPDMA capable for read/write channels. The memory embedded in the FMAC is used to split, by user configuration, the size and the allocation of the 256 x 16-bit local memory for each buffer (X1, X2, and Y). The GPDMA requests are based on the X1 buffer free-space availability (not full) in order to ask for vector preloading to get the data when required. This avoids stalling the calculation if the data is not yet present into the local memory.

The FMAC is not burst capable when doing GPDMA transfer, even if an embedded FIFO is present. All transfers must be handled in single data mode by the GPDMA controller, since the FMAC sends a new GPDMA request each time a 16-bit data is needed to be read/written. Then GPDMA performs a 32-bit aligned access to read/write from/to the 16-bit FMAC_RDATA/FMAC_WDATA register. This data register must be accessed with a programmed source/destination data width of 16-bit and with a fixed addressing.

No burst performance can be achieved for the read/write access to the FMAC. A channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others can be allocated, unless a channel 4 to 7 (only for STM32H5) or 12 to 15 is free/unallocated and the required performances in terms of bandwidth are significant for this AHB peripheral. A channel 4 to 7 (only for STM32H5) or 12 to 15 may be also allocated if the accessed memory is external.

The table below summarizes the main peripheral characteristics: hardware request signal to the GPDMA, peripheral bus on which the peripheral is connected, and peripheral register from/to which the GPDMA must be programmed to read/write.

Table 33. FMAC peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
fmac_read_dma	AHB1	256x 16-bit local memory shareable with X1, X2, and Y buffers(1)	Read 16-bit FMAC_RDATA	Request for a new computed 16-bit output data to be read by the GPDMA. Must be read with a 32-bit aligned address.
fmac_write_dma			Write 16-bit FMAC_WDATA	Request for a new 16-bit input data to be written by the GPDMA for computation. Must be written with a 32-bit aligned address.

1. *Each partitioning is user configurable.*

The table below shows the supported and recommended GPDMA settings to be used for serving the FMAC peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 34. GPDMA recommendation for FMAC

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	16-bit	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others 4 to 7 (only for STM32H5) or 12 to 15 especially if remains unallocated/free channels for such AHB peripherals, which may benefit from higher FIFO	The default recommended setting also balances bandwidth over each port (source port 0 to read from peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performances, the channel allocation can be one of channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory, if no specific requirements.*

3.4.11 CORDIC coprocessor

The STM32 devices embed one CORDIC instance.

The CORDIC provides an hardware acceleration of mathematical functions commonly used in motor control, metering, signal processing, and many other tasks, with 16- or 32-bit fixed-point. The GPDMA transfers can take place in read and write mode.

The CORDIC allows multiple register read/write by the GPDMA:

- ARG1 and possibly ARG2 as input arguments (depending on NARGS in CORDIC_CSR)
  If both arguments need to be input for computation, they are input thanks to two single GPDMA requests.
- RES1 and RES2 as respectively primary and secondary results (depending on NRES in CORDIC_CSR)
  If both computed results need to be read by the GPDMA, the CORDIC asserts two GPDMA requests in single mode.

The data width can be one of the following:

- 16-bit format
  The primary argument ARG1 is written to the least significant half-word whereas the secondary argument ARG2 is written to the most significant half-word. Thanks to this data packing, the argument is sent within a single GPDMA data transfer, which improves the bus bandwidth by minimizing the amount of DMA transfers to perform.
- 32-bit format
  There are two GPDMA data accesses to fill the two arguments (if CORDIC configuration requires two arguments).

The table below summarizes the main peripheral characteristics: hardware request signal to the GPDMA, peripheral bus on which the peripheral is connected, and peripheral register from/to which the GPDMA must be programmed to read/write.

Table 35. CORDIC peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
cordic_read_dma/ cordic_rd_dma	AHB1 and AHB2 for STM32H7Rx/7Sx	None	Read CORDIC_RDATA	Request for a new computed output data to be read by the GPDMA. Must be read with a 32-bit aligned address.
cordic_write_dma/ cordic_wr_dma			Write CORDIC_WDATA	Request for a new input data to be written by the GPDMA for computation. Must be written with a 32-bit aligned address.

The table below shows the supported and recommended GPDMA settings to be used for serving the FMAC peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 36. GPDMA recommendation for CORDIC

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	16- or 32-bit	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others 4 to 7 (only for STM32H5) or 12 to 15 especially if remains unallocated/free channels for such AHB peripherals, which may benefit from higher FIFO	The default recommended setting also balances bandwidth over each port (source port 0 to read from peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performances, the channel allocation can be one of channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory, if no specific requirements.*

3.5 GPDMA configuration for timers

3.5.1 Low-power timers (LPTIM)

The STM32H5 devices embed the LPTIM1 in APB3 and LPTIM2 in APB1. Additionally, the STM32H573/563/562 devices embed four other LPTIM instances (LPTIM3/4/5/6) in APB3 while STM32U5 devices embed LPTIM1/3 in APB3 and LPTIM2 in APB1. Additionally, STM32H7Rx/7Sx devices embed LPTIM2/3 in APB4 and LPTIM1 in APB1.

The low-power timers are 16-bit timers that benefit from the ultimate developments in power-consumption reduction. Thanks to their clock source diversity, the LPTIMx are able to keep running in all power modes except Standby mode.

The LPTIM is not burst capable when doing GPDMA transfer. All transfers are in single data mode.

The LPTIM4, LPTIM5 (only on STM32H7Rx/7Sx) does not provide any GPDMA request.

The LPTIM is GPDMA capable with two types of requests:

- Input capture
  - from timer channel 1, lptimx_ic1_dma
  - from timer channel 2, lptimx_ic2_dma
    Then the GPDMA reads the 16-bit LPTIMx_CCRy (y = 1, 2) registers to get the new latched counter value in input capture mode.
- Update event
  - An lptimx_ue_dma request is a GPDMA block-type request, and requires a specific hardware connection between the timer and the GPDMA to be activated by programming BREQ = 1 (and SWREQ = 0) in GPDMA_CyTR2 (y is the allocated GPDMA channel).
  - The block consists of up to three single writes to the LPTIMx to update up to two of the following registers, in order to update the generation of a PWM signal: LPTIMx_CCR1 (capture compare register 1 for the duty cycle), LPTIMx_CCR2 (capture compare register 2 for the duty cycle), and LPTIMx_RCR (repetition counter register), followed by a mandatory update of the LPTIMx_ARR for the period.

Port 0 is typically allocated for the read/write timer registers access, and port 1 for the memory access to balance traffic over the two ports. One of the channels (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others is allocated with a high priority to get a transfer latency as less as possible sensitive to other traffic. See Section 2.4 for more details, especially for time-sensitive use of the timer.

Any LPTIMx generates two output signals (lptimx_ch1 and lptimx_ch2), which are used as trigger for any GPDMA transfer and/or for a DAC conversion.

The LPTIM4 and LPTIM5 for STM32H7Rx/7Sx generates an output signal, lptim4_out, lptim5_out which are used as trigger for any GPDMA transfer and/or DAC conversion.

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, the peripheral bus on which the peripheral is connected, and the corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 37. LPTIM1/2/3/5/6 peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
lptimx_icy_dma (x = 1, 2, 3, 5, 6 and y = 1, 2)	APB4 for LPTIM2/3 for STM32H7Rx/7Sx APB3 for LPTIM1/3/5/6 for STM32H5 and LPTIM1/3 for STM32U5 APB1 for LPTIM2 for STM32H5 and STM32U5 and LPTIM1 for STM32H7Rx/7Sx	None	Read 16-bit/32-bit LPTIMx_CCRy	Request to read the capture compare register on an input capture event (connected input signal). Must be a 32-bit aligned address access.
lptimx_ue_dma (x = 1, 2, 3, 5, 6)			Write up to 3x 16-bit or 3x 32-bit registers among LPTIMx_CCR1, LPTIMx_CCR2, LPTIMx_RCR and mandated LPTIMx_ARR	Request to up to three writes of the LPTIMx registers on an update event. Must be a 32-bit aligned address access.

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 38. GPDMA recommendation for LPTIM1/2/3/5/6

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	16- or 32-bit	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	GPDMA1 port 0 is optimized to connect the APB1 LPTIM2 since there is a direct connection (at architecture level) from the GPDMA to the APB bridge on which the TIMx is connected. There is less latency and faster response time for a given priority level. The default recommended setting also balances bandwidth over each port (source port 0 to read from peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. Priority 3 is recommended if the application requires a minimized latency whatever is other concurrent traffic. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations.

1. *Port 0 is selected for accessing APB1 peripheral register (direct connection), and port 1 for accessing to memory. If no specific requirements, by default, port 0 can be selected for accessing APB3 peripheral register, and port 1 for accessing to memory.*

3.5.2 Basic timers (TIM6/7)

The basic timers consist of a 16-bit auto-reload upcounter driven by a programmable prescaler. They can be used as generic timers for time-base generation.

These timers are not burst capable when doing GPDMA transfer.

The timer update request to the GPDMA, (timx_upd_dma with x = 6, 7), occurs when the counter overflows (when a programmed period of time elapses). The GPDMA performs a 32-bit write to TIMx_ARR (x = 6, 7 and including 4 dithering bits), in order to update the timer output period and waveform for the next counter overflow.

It is recommended to allocate:

- the port 0 for peripheral register access (direct connection to AHB/APB bridge)
- the port 1 for memory access (to balance traffic), a channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others, and a high priority (to get a transfer latency as less as possible sensitive to other traffic)

See Section 2.4 for more details, especially for time-sensitive usage of the timer.

Both TIM6 and TIM7 generate a trigger signal, respectively tim6_trgo and tim7_trgo, which may be used as trigger for any GPDMA transfer and/or for a DAC conversion.

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, peripheral bus on which the peripheral is connected, and corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 39. TIM6/7 peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
timx_upd_dma (x = 6, 7)	APB1	None	Write 32-bit TIMx_ARR	Request on an update event (on a counter overflow)

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 40. GPDMA recommendation for TIM6/7

GPDMA destination port	GPDMA destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	32-bit	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	GPDMA1 port 0 is optimized to connect the APB1/APB2 TIMx since there is a direct connection (at architecture level) from the GPDMA to the APB bridge on which the TIMx is connected. There is less latency and faster response time for a given priority level. Priority 3 is recommended if the application requires a minimized latency whatever is other concurrent traffic. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations.

1. *Port 0 is selected for writing APB1 peripheral register (direct connection), and port 1 for accessing to memory.*

3.5.3 General-purpose timers

The STM32H5, STM32U5 and STM32H7Rx/7Sx devices embed the general-purpose timers TIM2/3 in APB1. Additionally, the STM32H573/563/562 devices embed five other general-purpose timers : TIM4/5 in APB1 and TIM15/16/17 in APB2 that have DMA mode.

The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler. TIM2/3/4/5/15/16/17 can be used for various purposes such as measuring the pulse lengths of input signals (input capture), or generating output waveforms (output compare, PWM, possibly complementary PWM with dead-time insertion).

These timers are not burst capable when doing GPDMA transfer. All read/write transfers must be handled in single data mode by the GPDMA controller, since the timer sends a new GPDMA request each time a 32-bit or 16-bit data/control is needed to be written/read.

A timer request can be used for the following actions:

- a 32-bit write to TIMx_ARR in basic mode, to update the output waveform and period when counter over/underflows
- a 32-bit read to TIMx_CCRy (y = 1 to 4), to get the latched counter value in input capture mode
- a 32-bit write to TIMx_CRRy (y = 1 to 4), to update the output compare register in output compare mode (duty cycle)
- a 16-bit/32-bit write to TIMx_RCR (x = 15, 16, 17), in PWM output/modulation mode, to update the repetition counter while the period is given by TIMx_ARR, and duty cycle is given by TIMx_CCR
- a 32-bit read to TIMx_CCR1, and a 32-bit read to TIMx_CCR2 in input PWM mode to get the latched counter for respectively the period and the pulse width
- a series of 32-bit writes to the intermediate TIMx_DMAR register in timer 'burst' mode, or a series of 32-bit reads from the intermediate TIMx_DMAR register in timer 'burst' mode
  The timer supports multiple single accesses to redirect the value to/from the timer registers through one entry point. This timer capability to write multiple timer registers through a single entry-point is called 'burst'. The GPDMA can perform the data transfers in single mode multiple times upon any programmed hardware event/request (among timx_chy_dma, timx_upd_dma, timx_trg_dma, timx_com_dma possible events). This timer 'burst' capability is different and must not be confused with a GPDMA burst that is defined in Section 2.5 and widely used in this document (as well as in the GPDMA section of the reference manual).

A timer request to GPDMA is quite programmable and flexible: it is related to the occurrence of a programmed event versus the real-time value of the timer counters. A timer request is not tightly coupled to write/read a specific timer register. Any timer register may be read/written depending on the application (the contrary of the other peripherals).

It is recommended to allocate:

- the port 0 for peripheral register access (direct connection to AHB/APB bridge)
- the port 1 for memory access (to balance traffic), a channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others, and a high priority (to get a transfer latency as less as possible sensitive to other traffic)

See Section 2.4 for more details, especially for time-sensitive usage of the timer.

For some application needs, a timer request to GPDMA can be used to initiate a data transfer not involving timer register read/write, but other memory-mapped location, like for memory-to-memory transfers. In this case, the GPDMA port allocation strategy must not be considered with respect to the timer register location, but with the source/destination locations.

TIM2 and TIM15 generate a trigger signal (respectively tim2_trgo and tim15_trgo) which can be used as trigger for any GPDMA transfer.

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, peripheral bus on which the peripheral is connected, and corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 41. TIM2/3/4/5/15/16/17 peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
timx_chy_dma (x = 2 to 5 and y = 1 to 4) or (x = 15 to 17 and y = 1)	APB1 for TIM2/3/4/5 APB2 for TIM15/16/17	None	32-bit write TIMx_ARR in basic mode or 32-bit write TIMx_CCRy in output compare/PWM mode or 32-bit read TIMx_CCRy in input capture mode or a series of 32-bit writes to TIMx_DMAR or a series of 32-bit writes to TIMx_DMAR	Request on an input capture event or an output compare event
timx_upd_dma (x = 2 to 5, 15 to 17)				Request on an update event
timx_trg_dma (x = 3, 5, 15)				Request on a trigger event
tim15_com_dma				Request on a commutation event

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 42. GPDMA recommendation for TIM2/3/4/5/15/16/17

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	32-bit (or 16-bit for TIMx_RCR)	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	GPDMA1 port 0 is optimized to connect the APB1/APB2 TIMx since there is a direct connection (at architecture level) from the GPDMA to the APB bridge on which the TIMx is connected. There is less latency and faster response time for a given priority level. Priority 3 is recommended if the application requires a minimized latency whatever is other concurrent traffic. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations.

1. *Port 0 is selected for reading/writing APB1/APB2 peripheral register (direct connection), and port 1 for accessing to memory. If the memory location to be accessed is not a timer register, this recommendation can be ignored.*

3.5.4 Advanced-control timers (TIM1/8)

The advanced-control timers consist of a 16-bit auto-reload counter driven by a programmable prescaler. TIM1/8 (STM32H7Rx/7Sx has TIM1 only) can be used for various purposes such as measuring the pulse lengths of input signals (input capture), or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

These timers are not burst capable when doing GPDMA transfer. All read/write transfers must be handled in single data mode by the GPDMA controller, since the timer sends a new GPDMA request each time a 32-bit or 16-bit data/control is needed to be written/read.

A timer request can be used for the following actions:

- a 32-bit write to TIMx_ARR in basic mode, to update the output waveform and period when counter over/underflows
- a 32-bit read to TIMx_CCRy (x = 1, 8 and y = 1 to 4), to get the latched counter value in input capture mode
- a 32-bit write to TIMx_CRRy (x = 1, 8 and y = 1 to 4), to update the output compare register and duty cycle in output compare mode
- a 16-bit/32-bit write to TIMx_RCR (x = 1, 8), in PWM output/modulation mode, to update the repetition counter while period is given by TIMx_ARR, and duty cycle is given by TIMx_CCR
- a 32-bit read to TIMx_CCR1 and a 32-bit read to TIMx_CCR2, in input PWM mode, to get the latched counter for respectively the period and the pulse width
- a series of 32-bit writes to the intermediate TIMx_DMAR register in timer ‘burst’ mode, or a series of 32-bit reads from the intermediate TIMx_DMAR register in timer ‘burst’ mode. The timer supports multiple single accesses to redirect the value to/from the timer registers through one entry point. This timer capability to write multiple timer registers through a single entry-point is called 'burst'. The GPDMA can perform the data transfers in single mode multiple times upon any programmed hardware event/request (among timx_chy_dma, timx_upd_dma, timx_trg_dma, timx_com_dma possible events). This timer ‘burst’ capability is different and must not be confused with a GPDMA burst that is defined in Section 2.5 and widely used in this document (as well as in the GPDMA section of the product reference manual).

A timer request to GPDMA is quite programmable and flexible: it is related to the occurrence of a programmed event versus the real-time value of the timer counters. A timer request is not tightly coupled to write/read a specific timer register, and any timer register can be read/written, depending on the application (the contrary of the other peripherals).

It is recommended to allocate:

- the port 0 for peripheral register access (direct connection to AHB/APB bridge)
- the port 1 for memory access (to balance traffic), a channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others, and a high priority (to get a transfer latency as less as possible sensitive to other traffic

See Section 2.4 for more details, especially for time-sensitive usage of the timer.

For some application needs, a timer request to GPDMA can be used to initiate a data transfer not involving timer registers read/write, but other memory-mapped location, like for memory-to-memory transfers. In this case, the GPDMA port allocation strategy must be not be considered with respect to the timer registers location, but with the source and destination locations.

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, peripheral bus on which the peripheral is connected, and corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 43. TIM1/8 peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
timx_chy_dma (x = 1, 8 and y = 1 to 4)	APB2	None	No tightly coupled timer register read/write Typically a 32-bit read/write to TIMx_CCRy or a 16-/32-bit read/write to TIMx_RCR or a series of 32-bit read/ writes to TIMx_DMAR (x = 1, 8 and y = 1 to 4)	Request on an input capture event or an output compare event
timx_upd_dma (x = 1, 8)				Request on an update event
timx_trg_dma (x = 1, 8)				Request on a trigger event
timx_com_dma (x = 1, 8)				Request on a commutation event

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 44. GPDMA recommendation for TIM1/8

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	32-bit (or 16-bit for TIMx_RCR)	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	GPDMA1 port 0 is optimized to connect the APB2 TIM1/8 since there is a direct connection (at architecture level) from the GPDMA to the APB bridge on which the TIM1/8 is connected. There is less latency and faster response time for a given priority level. Priority 3 is recommended if the application requires a minimized latency whatever is other concurrent traffic. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations.

1. *Port 0 is selected for reading/writing APB2 peripheral register (direct connection), and port 1 for accessing to memory. If the memory location to be accessed is not a timer register, this recommendation can be ignored.*

3.6 GPDMA configuration for cryptographic peripherals

3.6.1 Hash processor (HASH)

The HASH is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-224, SHA-256), the MD5 (message-digest algorithm 5) hash algorithm, and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message.

The HASH implementation proposes a DMA mode. The HASH generates GPDMA requests allowing the user to perform single or burst GPDMA write transfers. The GPDMA request signal is used by the hardware to request a new input to be written by the GPDMA whenever the peripheral needs it. The request may be a request for either a 32-bit single data to be written, or a burst of 4 (32-bit) words to be written. The GPDMA writes the HASH_IN register. This HASH register must be accessed by with a destination source data width being 32-bit with a fixed addressing, and a burst length of 1 or 4.

The HASH implementation uses a 16-word FIFO for the HASH_IN register.

For best GPDMA performances, it is recommended to allocate one of the channels 4 to 7 (only for STM32H5) or 12 to 15 with an 8-word FIFO. If such a channel is allocated, 4-word bursts for writing to the memory can also be programmed and implemented.

If no channel 4 to 7 (only for STM32H5) or 12 to 15 is free, a channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others is allocated, and programming a fixed 4-word burst write is also recommended for higher performances. This allows the request signal hash_in_dma to be handled in hardware between the GPDMA and the HASH as a hardware burst request, and not as a single. Knowing that at least 4 AHB clock cycles must occur between two occurrences of a HASH request, the transfer rate can be higher via programmed burst. On the memory side, transfers are then performed by 32-bit single reads (due to FIFO size).

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, peripheral bus on which the peripheral is connected, and corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 45. HASH peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
hash_dma	AHB2	16-word FIFO	Write 32-bit HASH_IN	HASH input data register (HASH_IN) can be written by a single word or by a 4-word burst.

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 46. GPDMA recommendation for HASH

GPDMA destination port	GPDMA destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	32-bit	1 or 4	4 to 7 (only for STM32H5) or 12 to 15 for burst performances if available else (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others operating with single mode only	The default recommended setting also balances bandwidth over each port (source port 0 to write to peripheral, destination port 1 to read from memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. For improved burst performances, the channel allocation can be one of the channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory, if no specific requirements.*

3.6.2 AES hardware accelerator (AES)

The AES is not available on STM32H7Rx/7Sx devices.

The AES hardware accelerator encrypts/decrypts data, using an algorithm and implementation fully compliant with the advanced encryption standard (AES) defined in federal information processing standard (FIPS) publication 197.

The AES supports CTR, GCM, GMAC, CCM, ECB, and CBC chaining modes for 128- or 256-bit key size.

The AES performs 128-bit block cryptographic processing, and implements a 128-bit input buffer/FIFO and a 128-bit output buffer/FIFO.

This peripheral is not burst capable when doing GPDMA transfer. All transfers must be handled in single data mode by the GPDMA controller, since the peripheral sends a new GPDMA request each time an input/output 32-bit data is needed to be written/read. Then GPDMA performs a read/write access to the related 32-bit data register, with a programmed source/destination data width of 32-bit and a fixed addressing.

One of the channels (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others can be allocated (as well one of the channels 4 to 7 (only for STM32H5) or 12 to 15, especially if the memory to be written is an external memory).

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, the peripheral bus on which the peripheral is connected, and the corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 47. AES peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
aes_in_dma	AHB2	128-bit/4-word input buffer	Write 32-bit AES_DINR	Request for a new 32-bit input data to be written by the GPDMA
aes_out_dma		128-bit/4-word output buffer	Read 32-bit AES_DOUTR	Request for a new 32-bit output data to be read by the GPDMA

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 48. GPDMA recommendation for AES

GPDMA source/ destination port	GPDMA source/ destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0(1)	32-bit	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	The default recommended setting also balances bandwidth over each port (source port 0 to write to peripheral, destination port 1 to read from memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. It is better to use channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others to let the other channels free for 2D operations, memory copies, or for peripherals requesting higher internal DMA FIFO size to manage burst operations. For improved burst performances, the channel allocation can be one of the channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory, if no specific requirements.*

3.6.3 Secure AES coprocessor (SAES)

The SAES is the secure AES version. All data described in Section 3.6.2 are valid. The only difference is the GPDMA request signal names (saes_in_dma and saes_out_dma).

3.6.4 Cryptographic processor (CRYP)

The cryptographic processor is available only on STM32H7Rx/7Sx devices.

The cryptographic processor (CRYP) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST.

CRYP supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128, 192, or 256 bits. CRYP has the possibility to load by hardware the key stored in SAES peripheral, under SAES control.

The peripheral supports DMA transfers for incoming and outgoing data (two DMA channels are required). CRYP also includes input and output 8-word FIFOs for better performance.

The output FIFO supports both single and burst transfers, while the input FIFO supports only burst transfers.

The table below summarizes the main peripheral characteristics: different hardware request signals to the GPDMA, the peripheral bus on which the peripheral is connected, and the corresponding peripheral register from/to which the GPDMA must be programmed to read/write.

Table 49. CRYP peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
cryp_in_dma	AHB3	8-word input FIFO	Write 32-bit CRYP_DIN	Request for a 4-word (fixed) burst data to be written to CRYP input FIFO.
cryp_out_dma		8-word output FIFO	Write 32-bit CRYP_DOUT	Request for a 4-word (fixed) burst data or a single word data to be read from CRYP output FIFO.

The tables below show the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 50. GPDMA recommendation for writing to CRYP input data

GPDMA source port	GPDMA source burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	32-bit	4	12 to 15 typically	On write: peripheral is burst capable and burst length must be 4. 4-word burst must be DMA programmed, for any channel. DMA may issue a single word or a 2, 3 or 4-word burst. For improved burst performances, one GPDMA channel from 12 to 15 must be allocated. The default recommended setting balances bandwidth over each port (source port 0 to write to peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs.

1. *By default, port 0 is recommended to access peripheral register, and port 1 to access memory if no specific requirements.*

Table 51. GPDMA recommendation for writing to CRYP output data

GPDMA source port	GPDMA source burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 0[1]	32-bit	1 or 4	0 to 11 typically	On read: peripheral is burst capable and burst length must be 1 or 4. 1-word single or 4-word burst must be DMA programmed, for any channel. DMA may issue a single word or a 2, 3 or 4-word burst.

GPDMA source port	GPDMA source burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
				For improved burst performances, one GPDMA channel from 12 to 15 must be allocated. The default recommended setting balances bandwidth over each port (source port 0 to write to peripheral, destination port 1 to write to memory). GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs.

3.7 GPDMA configuration for general-purpose I/Os

The general purpose I/Os (GPIO) or low-power general-purpose I/Os (LPGPIO) can also be controlled by the GPDMA, even if there is no specific hardware GPDMA request. The GPDMA in memory-to-memory configuration can control the I/Os (as well as the data in output for instance). For the LPGPIOs, the LPDMA can be functional even down to Stop 2 mode, to control up to 16 I/Os.

3.8 GPDMA configuration for external memories

3.8.1 Octo-SPI interface (OCTOSPI)

There is one instance of the Octo_SPI interface (OCTOSPI1) in STM32H573/563/562 and STM32U535/545 devices. In other STM32U5 devices, there are two instances (OCTOSPI1, OCTOSPI2).

The OCTOSPI supports most external memories, such as serial PSRAMs, serial NAND and serial NOR Flash memories, HyperRAM™ and HyperFlash™ memories.

The OCTOSPI in Indirect mode proposes a DMA mode for read/write data transfers to its peripheral registers. The OCTOSPI is burst capable via its embedded 32-byte FIFO.

The GPDMA request signal, ospi1_dma for STM32H5 and octospix_dma (x = 1, 2) for STM32U5 and no GPDMA request signal for STM32H7Rx/7Sx, is used by the hardware to request a new burst to be read/written by the GPDMA whenever the peripheral needs it. The GPDMA reads/writes the OCTOSPI_DR data register. This 32-bit data register can be accessed by a 32-bit word-aligned read access, with a programmed source data width being typically either 8-, 16-, or 32-bit and with a fixed addressing.

The burst is defined on the OCTOSPI side by programming the FIFO threshold FTHRES[4:0] in OCTOSPI_CR with a 1- to 32-byte burst size.

The allocated GPDMA channel is recommended to be one of channel 4 to 7 (only for STM32H5) or 12 to 15 with a large 32-byte FIFO. It can then perform bursts of a size equal to half of its FIFO size (up to 16-byte, 8-half-word, or 4-word bursts). This means programming a 15 decimal value on the OCTOSPI side (FTHRES[:0] = 15).

Using a 32-bit data width is preferred for minimizing the AHB bus load, unless the application requires a byte-level or half-word data management.

The table below summarizes the main peripheral characteristics: hardware request signal to the GPDMA, peripheral bus on which the OCTOSPI is connected, and peripheral register from/to which the GPDMA must be programmed to read or write.

Table 52. OCTOSPI peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
ospi1_dma/ octospix_dma (x = 1, 2)	AHB_OCTOSPI (dedicated target of bus matrix)	32-byte FIFO	Read/Write OCTOSPI_DR	It is recommended to access OCTOSPI_DR with a programmed 4-word burst, when a 32-bit data width is applicable.

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length (only single data mode).

Table 53. GPDMA recommendation for OCTOSPI

GPDMA source/ destination port	GPDMA source/destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 1[1]	32-bit (preferred) else 8- or 16-bit	4-word (preferred) or 8-half-word or 16-byte	4 to 7 (only for STM32H5) or 12 to 15 for burst performances	The default port allocation (port 0 to peripheral, port 1 to external memory) enables to balance the bandwidth over each port for memory-to-peripheral and peripheral-to-memory transfers. It also avoids bursts to memory from impacting directly the latency of a peripheral access. GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. For improved burst performances, the channel allocation can be one of the channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *By default, port 1 is recommended to access external memory, if no specific requirements.*

3.8.2 Hexa-SPI interface (HSPI)

There is one instance of the Hexa-SPI interface (HSPI1) in the STM32U59x/5Ax/5Fx/5Gx devices.

The HSPI supports most external memories, such as serial PSRAMs, serial NAND/NOR flash memories, HyperRAM™ and HyperFlash™ memories.

The HSPI in indirect mode proposes a DMA mode for read/write data transfers to its peripheral registers. The HSPI is burst capable via its embedded 64-byte FIFO.

The GPDMA request signal (hspi1_dma) is used by the hardware to request a new burst to be read/written by the GPDMA whenever the peripheral needs it. The GPDMA reads/writes the HSPI_DR data register. This 32-bit data register can be accessed by a 32-bit word-aligned read access, with a programmed 8-, 16-, or 32-bit source data width, and with a fixed addressing.

The burst is defined on the HSPI side by programming the FIFO threshold FTHRES[5:0] in HSPI_CR with a 1-to 64-byte burst size.

The allocated GPDMA channel is recommended to be one of the channels 12 to 15 with a large 32-byte FIFO. It can then perform GPDMA bursts of a size equal to half of its FIFO size (up to 16-byte, 8-half-word, or 4-word). This means programming a 15 decimal value on the HSPISPI side (FTHRES[5:0] = 15).

Using a 32-bit data width is preferred for minimizing the AHB bus load, unless the application requires byte-level or half-word data management.

The table below summarizes the main peripheral characteristics: hardware request signal to the GPDMA, peripheral bus on which the HSPI is connected, and peripheral register from/to which the GPDMA must be programmed to read or write.

Table 54. HSPI peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
hspi1_dma	AHB_HSPI1 (dedicated target of bus matrix)	64-byte FIFO	Read/write HSPI_DR	It is recommended to access HSPI_DR with a programmed 4-word burst, when a 32-bit data width is applicable.

The table below shows the supported and recommended GPDMA settings to be used for serving the different peripheral requests: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length.

Table 55. GPDMA recommendation for HSPI

GPDMA source/ destination port	GPDMA source/destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 1[1]	32-bit (preferred) else 8- or 16-bit	4-word (preferred) or 8-half-word or 16-byte	12 to 15 for burst performances	The default port allocation (port 0 to peripheral, port 1 to external memory) enables to balance the bandwidth over each port for memory-to-peripheral and peripheral-to-memory transfers. It also avoids bursts to memory from impacting directly the latency of a peripheral access. GPDMA port selection for the source and the destination transfers can be changed to fit specific application needs. For improved burst performances, the channel allocation can be one of the channels 12 to 15 (with 8-word FIFO).

1. *By default, port 1 is recommended to access external memory, if no specific requirements.*

3.8.3 Flexible static memory controller (FSMC)

The FSMC is not available on STM32H503, STM32U535/545, and STM32H7Rx/7Sx devices.

The FSMC (also named FMC) includes two to three memory controllers:

- synchronous/asynchronous NOR or PSRAM/SRAM/FRAM memory controller
- asynchronous NAND memory controller
- synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) controller

There is no specific request from the FMC to ask for a GPDMA transfer. The GPDMA may be configured to perform data transfer for memory-to-memory transfers, with the FMC being either the source memory or the destination memory.

If the external memory is asynchronous, it is recommended to configure the GPDMA to perform transfers in single data mode only, and with a data width equal to the interface bus width of the memory (8- or 16-bit). Programming bursts must be avoided for not stalling the AHB bus and the allocated GPDMA port during a long time, and for not forcing other GPDMA requests to be served with a high latency.

If the external memory is a synchronous memory, read must also be configured as single transactions of 8- or 16-bit data width, for not stalling the AHB bus.

If the external memory is a synchronous memory, the FSMC must be written by GPDMA bursts, because the FMC implements a 16-word write FIFO. When this FIFO is full, the FSMC stalls the AHB bus and the allocated GPDMA until the burst completion. The memory must then be written by GPDMA blocks up to a size which corresponds to the write FIFO being full. The software must wait for the FSMC to have its FIFO empty before enabling another block transfer. The allocated GPDMA channel must be a channel 4 to 7 (only for STM32H5) or 12 to 15, in order to be able to perform either 4-word, 8-half-word, or 16-byte bursts.

Table 56. FMC peripheral information

Peripheral request to GPDMA	Peripheral bus	Peripheral FIFO (unit, size)	Peripheral register access	Comments
-	AHB_FMC (dedicated target of bus matrix)	16-word write FIFO	-	Read/write access directly to the external FMC memory space

The table below shows the supported and recommended GPDMA settings to be used for serving the FMC: GPDMA port allocation, GPDMA channel allocation, and programmed GPDMA bursts in terms of data width and burst length.

Table 57. GPDMA recommendation for FMC on read

GPDMA source port	GPDMA source burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 1[1]	8- or 16-bit	1	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	Read by single The default port allocation (port 0 to peripheral, port 1 to memory) also balances the bandwidth over each port for memory-to-peripheral and peripheral-to-memory transfers. GPDMA port selection for the FMC source transfer can be changed to fit specific application needs.

1. *By default, port 1 is recommended to access external memory, if no specific requirements.*

Table 58. GPDMA recommendation for FMC on write

GPDMA destination port	GPDMA destination burst		GPDMA channel	Comments
	GPDMA data width	GPDMA burst length		
Port 1[1]	32-bit (preferred) else 8- or 16-bit	4-word (preferred) or 8-half-word or 16-byte	4 to 7 (only for STM32H5) or 12 to 15 for write burst performances	The default port allocation (port 0 to peripheral, port 1 to memory) enables to balance the bandwidth over each port for memory-to-peripheral and peripheral-to-memory transfers. It also avoids write bursts to memory from impacting directly the latency of a peripheral access. GPDMA port selection for the FMC destination transfer can be changed to fit specific application needs. For improved write burst performances, the channel allocation can be one of the channels 4 to 7 (only for STM32H5) or 12 to 15 (with 8-word FIFO).

1. *If no specific requirements, by default, port 1 is assigned to memory access.*

4 System performance

The microcontroller architecture needs to be understood in order to control that the different GPDMA requested transfers can be efficiently performed: transferred in due time (with an acceptable latency and completion time versus the application requirements) while optimizing the bus bandwidth efficiency, and minimizing the resource utilization/load (bus/resource utilization, bus overhead, AHB/APB frequencies).

The previous sections refined the GPDMA characteristics, and provided user guidelines to get the best performances for serving a GPDMA data transfer. The following sections describe other key points which impact this performance.

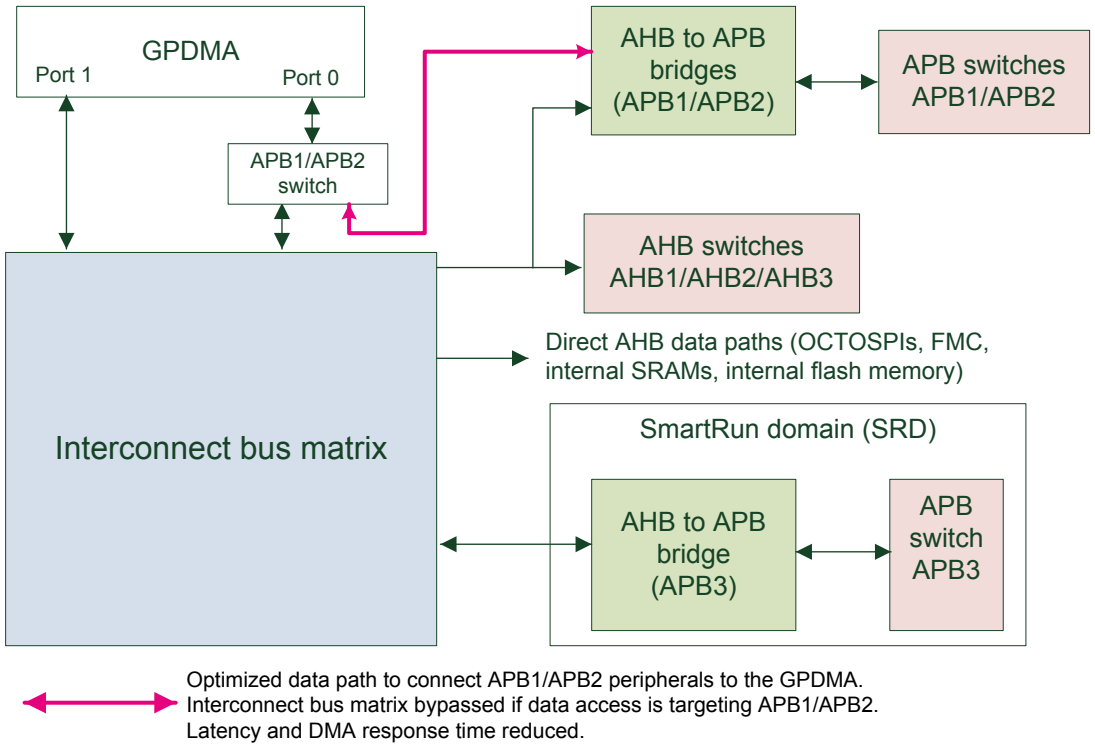
4.1 GPDMA

The GPDMA has two master ports for parallelization of AHB transfers. Simultaneous transfers via the two ports are possible. A GPDMA arbitration occurs for each port as follows:

- a priority-based arbitration (see Figure 1) between the 8 possible requested FIFO-based read bursts
- a priority-based arbitration (see Figure 1) between the 8 possible requested FIFO-based write bursts
- a final round-robin arbitration stage between read and write

The GPDMA port 0 has a direct connection with APB1/APB2 peripherals as shown in the figure below.

Figure 2. Latency optimization (APB1/2 peripherals mapped on GPDMA port0)



The strategy and guidelines to be applied for the GPDMA port allocation and the priority assignment are detailed in Section 2.3 and Section 2.4.

4.2 Bus matrix

The multilayer AHB bus matrix allows parallel access to a number of shared AHB slaves from a number of different AHB masters. The bus matrix improves the transfer parallelism, and contributes to reduce the transfer execution time and to optimize the shared resources utilization.

4.2.1 AHB bus definitions

- AHB master: a bus master initiates read/write AHB single/burst transfer. Only one master can get bus ownership at a given time.
- AHB slave: a bus slave responds to the AHB transfer from a master. The bus slave routes signals back to the master to inform about the access success, failure or waiting states.
- AHB arbiter: a bus arbiter ensures that only one master can access a slave at a given time.
- AHB bus matrix: a switch matrix that interconnects AHB masters to AHB slaves. It implements a round-robin AHB arbiter for each slave.

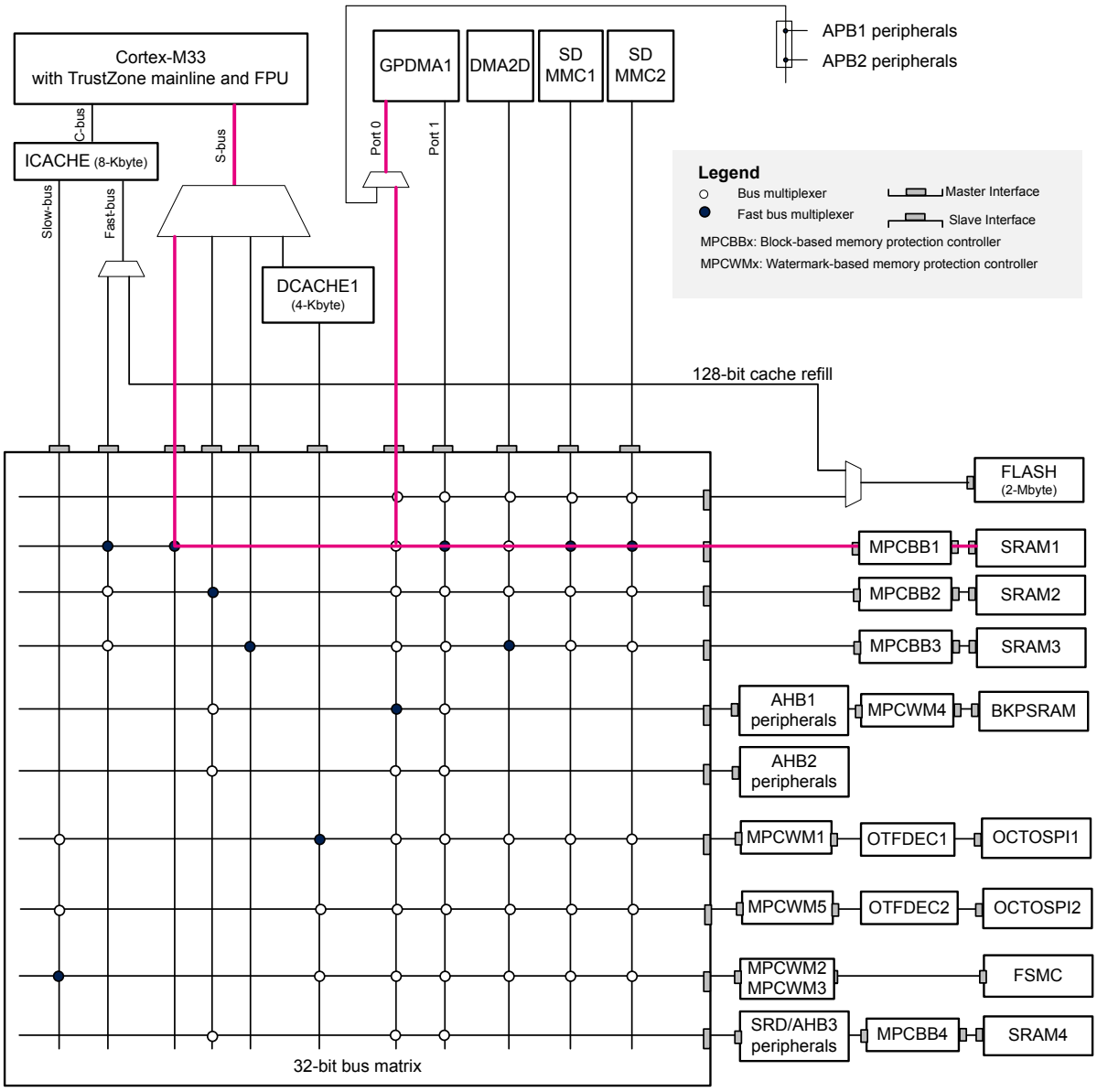
4.2.2 Bus matrix round-robin arbitration

A bus matrix arbitration is implemented for each slave in order to arbitrate its access from concurrent masters. This round-robin arbitration allows a fair distribution of the shared resource over the time (see the example in Figure 3) and a bounded latency. Round robin algorithm is implemented with the following rules:

- A round-robin quantum is one AHB transfer. There is bus switching on every transfer if another master port waits for the access to a slave.
- A master port has a defined default slave.
  - When accessing a default slave, a master port gets:
    - 0-cycle arbitration penalty when it first accesses the default slave (provided that the bus is free/idle)
    - 0-cycle rearbitration penalty, and is immediately granted to reaccess the default slave with a back-to-back transaction, if there is no other master waiting for the shared slave
  - When accessing a non-default slave, a master port gets:
    - 1-cycle arbitration penalty when it first accesses the non-default slave (provided that the bus is free/idle)
    - 1-cycle rearbitration penalty to be granted to reaccess the non-default slave if there is at least one dead clock cycle/wait state between its back-to-back transfers (even if no other master waiting for the shared slave).
    - 0-cycle rearbitration penalty to be granted to reaccess the non-default slave if there is no dead clock cycle/wait state between its back-to-back transfers (if no other master waits for the shared slave).

The figure below shows an example where both the CPU and GPDMA try to access SRAM1.

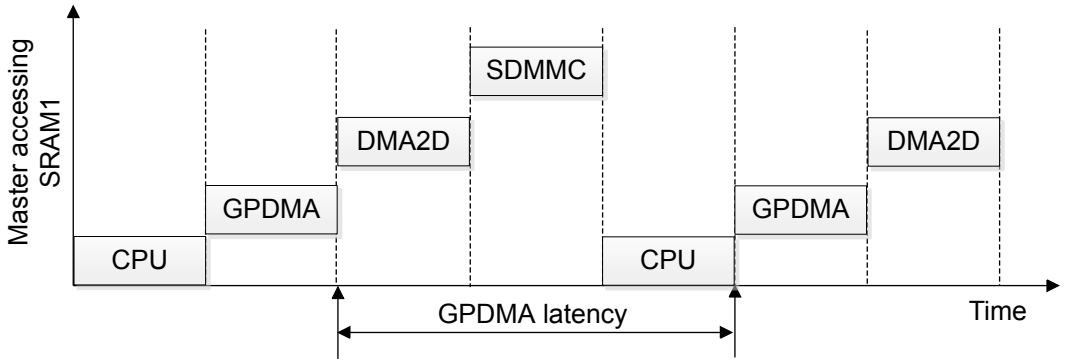
Figure 3. Example of CPU and GPDMA requesting an access to SRAM1



In case of concurrent accesses from the CPU and the GPDMA, the bus matrix arbitration rules the access to the SRAM1. If the last access is from the CPU, during the next access, the GPDMA wins the bus, and accesses SRAM1. After the CPU can again access SRAM1.

The transfer latency initiated by one master to access a slave depends on the number of other pending transfers initiated by other masters to access the same AHB slave. The figure below details another example where four masters try to access SRAM1 simultaneously.

Figure 4. Example of four master requesting an access to SRAM1



The latency associated to the GPDMA (burst) transfer to access the bus and SRAM1 in the example above, is increased by the sum of execution times of all pending requests coming from the other masters. If the DMA2D accesses the SRAM1 in burst mode for example, the GPDMA access latency is increased by the latency of the DMA2D burst.

Note: The latency due to CPU transfers issued by LDM/STM instructions can be reduced by configuring the compiler to split load/store multiple instructions into single load/store instructions.

4.3 Bus (AHB and APB) switches

As depicted in Figure 2, the STM32 devices embed several AHB/APB switches in order to decode/multiplex the AHB/APB signals.

A bus switch adds no cycle penalty to a transfer latency/execution time.

4.4 SRAMs

4.4.1 Transfer latency and execution time

The SRAM access latency is the SRAM execution time for completing a burst/single read/write access, seen from the AHB bus interface.

A GPDMA write is a non-bufferable write. In the STM32 devices, a bufferable write access is restricted to a CPU access to an external memory via the data cache.

An SRAM introduces no additional wait-state for an AHB access. An 8-,16-, or 32-bit GPDMA transfer that accesses an SRAM in read or write, as a single, takes 2 AHB clock cycles so that the access is completed (one for the address phase, one for the data phase).

For completion at the SRAM bus interface:

- A GPDMA single transfer (1x NONSEQ) takes 2 cycles.
- A GPDMA 4-beat incremented burst transfer (1x NONSEQ + 3x SEQ) takes 5 cycles.

At the SRAM bus interface, it takes 8 AHB clock cycles via singles versus 5 cycles via a burst to transfer 4 words in SRAM at consecutive address locations (with transfer not arbitrated and preempted by other concurrent ones in front).

The GPDMA inserts also 1 cycle between two memory accesses. The bus matrix inserts another cycle if the addressed SRAM is not the default slave (if SRAM1 is not accessed by the GPDMA port1). Consequently, for completion:

- A GPDMA single read/write transfer to SRAM takes 3 to 4 cycles.
- A GPDMA 4-word incremented burst read/write transfer to SRAM takes 6 to 7 cycles.

The table below gives the latency for completion of a programmed (half) transfer to the SRAM, depending on the allocated channel and its FIFO size (see Section 2.2 and Section 2.5).

Table 59. GPDMA read/write transfer to SRAM latency (4-beat burst versus single)

Programmed GPDMA (half) transfer (from/to FIFO to/from SRAM)	Channel number	Read/Write latency (cycles)
Single	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	3 to 4
	4 to 7 (only for STM32H5) or 12 to 15	
4-word incremented burst	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others(1)	3 to 4(2)
	4 to 7 (only for STM32H5) or 12 to 15	6 to 7

1. *A channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others has a 2-word FIFO. A programmed 4-word burst is then implemented via 4 singles.*
2. *Latency for an implemented single.*

If the first transfer latency is increased by 3 cycles, the latency for 4 words is decreased of 3 cycles. A buffer transfer is earlier completed by performing bursts versus singles.

4.4.2 Transfer rate/throughput and bandwidth

Beside this SRAM transfer latency, it is important to consider the performance in terms of how much fast (consecutive) transfers can happen (maximum achievable data throughput).

The SRAM throughput is the number of transferred data per second. For a given AHB operating frequency, the SRAM throughput is the number of (recommended 32-bit) data transferred per AHB clock cycles.

The 32-bit AHB bus architecture consists of:

- a dedicated address bus
- a separated bus for read and write data
- 1-cycle pipelined transfers in terms of address and data (on a same clock cycle, the address of a transfer occurs meanwhile the data phase of the previous transfer)

The SRAM introduces no penalty. A transfer rate of one 32-bit read or write per cycle can then be sustained. The GPDMA and the AHB bus cannot achieve such a high throughput.

The table below provides the achieved transfer rate for copying a buffer via a GPDMA channel in memory-to-memory mode, when both the source buffer and the destination buffer are located in the SRAM, depending on the allocated channel and the programmed burst length.

Table 60. GPDMA in memory-to-memory mode, SRAM (read+write) transfer rate (4-beat burst versus single)

Programmed GPDMA (half) transfer to/from SRAM	Channel number	Number of cycles (per transferred data/beat)	Transfer rate with a 100 MHz AHB clock (Mtransfer/s)
single	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others	3.5 to 4	25 to 28
	4 to 7 (only for STM32H5) or 12 to 15	2 to 4(1)	25 to 50
4-word incremented burst	(0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others(2)	3.5 to 4	25 to 28
	4 to 7 (only for STM32H5) or 12 to 15	1.6 to 2.2	45 to 62

1. *The improvement between a channel 4 and a channel 0 on single transfers (2 cycles versus 3.5) is because the 8-word FIFO is earlier ready than a 2-word FIFO to perform a next single.*
2. *A channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others has a 2-word FIFO. A programmed 4-word burst is then implemented via 4 singles. Throughput is the same as with single.*

The previous table highlights that a channel 4 to 7 (only for STM32H5) or 12 to 15 that is programmed with a 4-word burst can consume only 1.6 cycles in average per transferred word/beat, versus a minimum of 3.5 cycles in average on an allocated channel (0 to 3 or 8 to 11) for STM32H5 and 0 to 11 for others and single-word transfers. Programming bursts on an allocated channel, which is burst-capable is around twice more efficient than performing singles:

- For a given buffer copy, the required GPDMA traffic consumes half less the bus.
- For a given bandwidth budget, the GPDMA transfer rate performance is twice higher.

The advantage of performing transfers to/from SRAM by burst transfers versus single is two-fold:

- The achievable transfer rate is higher, enabling to sustain more performance.
- The bus load/utilization is lower, freeing the bus (as well as bus matrix and other master ports in front) for other concurrent transfers to operate at the given frequency, or/and enabling to operate at a lower frequency for a given traffic.

For memory-to-memory GPDMA transfers to SRAM, it is recommended to allocate a GPDMA channel 4 to 7 (only for STM32H5) or 12 to 15, and to program bursts (of length 4 and of 32-bit data width) as much as possible. The performances are not then limited by the peripheral side, as it is the case for peripheral-to-memory and memory-to-peripheral transfers.

4.5 AHB-to-APB bridges

An APB peripheral generates a hardware request to the GPDMA. The allocated and configured GPDMA channel generates either a read transfer from the corresponding APB register (in memory-to-peripheral mode), or a write transfer to the APB register (in peripheral-to-memory mode).

Section 3 lists the APB peripherals working in DMA mode, and their corresponding GPDMA programming. It is recommended that the GPDMA generates only single transfers (GPDMA burst of length 1) as a read or as a non-bufferable write to an APB register.

4.5.1 AHB-to-APB bridge arbitration

The APB1 and APB2 peripherals can be accessed concurrently either from the GPDMA master port 0 via the direct path, or from the CPU or the GPDMA master port 1 via the AHB switch on the AHB1 output path of the bus matrix.

A concurrent access is resolved at AHB-to-APB bridge level, by a round-robin arbitration with a quantum of one APB transfer.

There is no arbitration inside the AHB-to-APB bridge in front of the APB3.

4.5.2 AHB-to-APB clock ratio

The transfer latency to an APB peripheral register is impacted by the clock ratio between APB and AHB clocks. The best case is achieved with a ratio of 1. When the APB clock runs lower than the AHB clock, the latency is increased.

4.5.3 AHB-to-APB bridge transfer latency

From the AHB interface of the AHB-to-APB bridge, a GPDMA transfer execution time/latency, as a single, to the APB peripheral register by the GPDMA is shown in the table below (when there is no concurrent/pending transfer from another master to an APB1 or APB2 peripheral).

Table 61. AHB-to-APB bridge read/write (single) transfer latency to APB register

AHB/APB clock ratio	Read latency (AHB clock cycles)	Write latency (AHB clock cycles)
1:1	4	4 to 5
1:2	5 to 6	
1:4	9 to 12	
1:8	17 to 24	

The GPDMA adds one clock-cycle latency, and the bus arbitration may add one clock cycle. The table below details the GPDMA transfer latency for completion of a single read/write to an APB peripheral register, provided that:

- neither another GPDMA channel, nor the CPU generates a concurrent traffic to the addressed APB1/2/3 bus
- the APB peripheral does not insert any wait state

Table 62. GPDMA read/write transfer latency to APB register

AHB/APB clock ratio	Read latency (AHB clock cycles)	Write latency (AHB clock cycles)
1:1	5 to 6	
1:2	6 to 8	
1:4	10 to 14	
1:8	18 to 26	

Revision history

Table 63. **Document revision history**

Date	Version	Changes
24-Sept-2021	1	Initial release.
3-Dec-2021	2	Updated the whole content of this document.
13-Dec-2021	3	Updated Section 3.5.1 Low-power timers (LPTIM1/2/3/4).
22-Aug-2022	4	Document scope enlarged to all devices of the STM32U5 Series Updated: • Section 3.2.1 Analog-to-digital converter ADC1/ADC12 • New Section 3.3.2 JPEG codec • Section 3.4.2 Inter-integrated circuit interface (I2C) • Section 3.4.3 Universal synchronous/asynchronous/low-power receiver transmitter (USART/UART/LPUART) • Section 3.8.1 Octo-SPI interface (OCTOSPI) • New Section 3.8.2 Hexa-SPI interface (HSPI) • Section 3.8.3 Flexible static memory controller (FSMC) • Section 4.1 System bus
14-Mar-2024	5	Document scope enlarged to add STM32H5 series and STM32H7R3/7S3 and STM32H7R7/7S7 lines. Updated document title and all topics. Added: • Section 3.2.2: Analog-to-digital converter ADC4 • Section 3.4.3: Improved inter-integrated circuit interface (I3C) • Section 3.4.7: SPDIF receiver interface (SPDIFRX) • Section 3.6.4: Cryptographic processor (CRYP) Removed Section System bus.

Contents

List of tables

List of figures