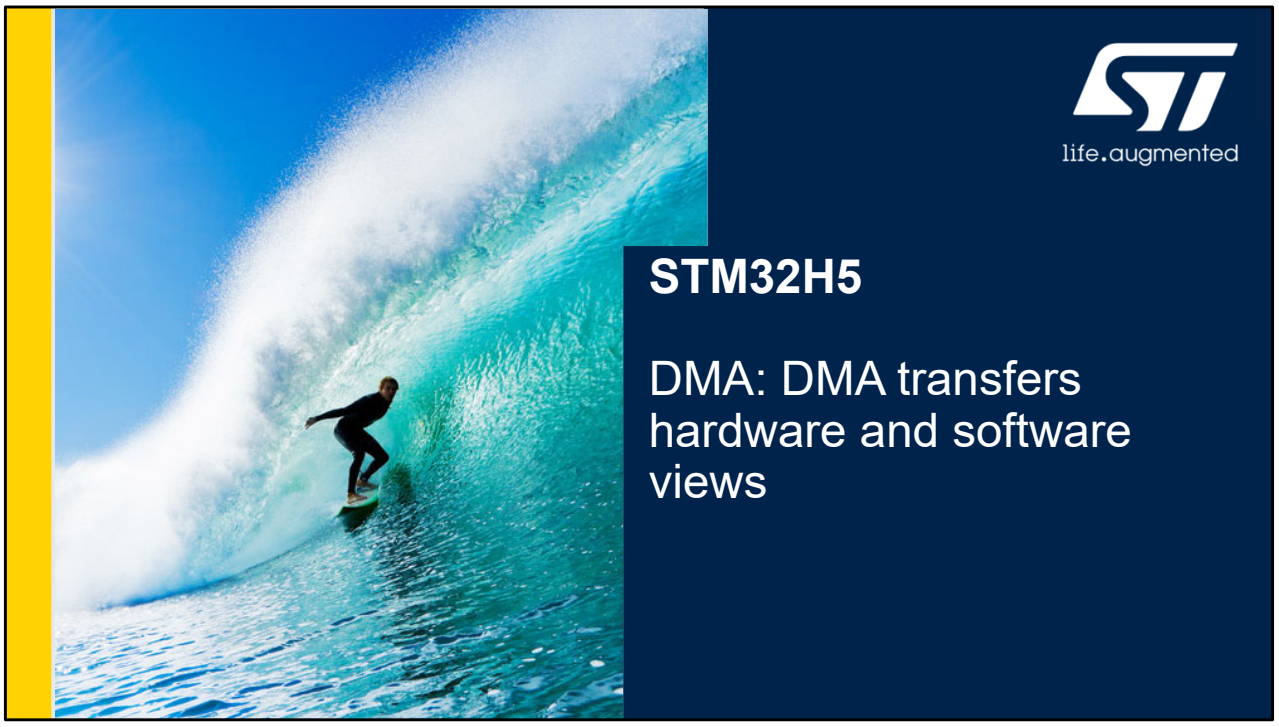


Hello, and welcome to this presentation, that describes the DMA transfers hardware and software views.

DMA requested transfer

- The request that initiates a DMA transfer may be:
  - A hardware request from a peripheral configured in DMA mode (for a transfer from/to the peripheral data register respectively to/from memory)
  - A hardware request from a peripheral for its control register(s) update from memory
  - A hardware request from a peripheral for a read of its status register(s) transferred to memory
  - A software request from typically the CPU for a data transfer from a memory-mapped address to another memory mapped address

life.augmented

2

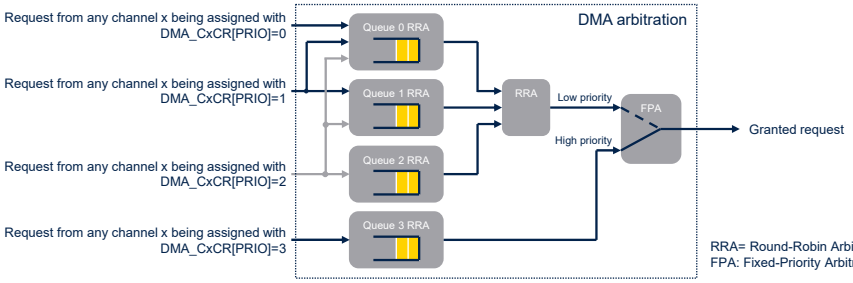
To start a DMA transfer, a request is required.
This request can be caused by a hardware event:

- From a peripheral indicating that data are ready to be transferred
- From a peripheral indicating that an update of its control registers is required
- From a peripheral indicating that a read of its status registers is required.

A software request can also start a DMA transfer. This is typically caused by the CPU writing a control register of the DMA controller in order to transfer data from a memory-mapped address range to another memory-mapped address range.

DMA request arbitration

- DMA arbitration and bandwidth are guaranteed with:
  - An equal maximum bandwidth between requests with a same priority
  - A reserved bandwidth to the time-sensitive requests, i.e. with a highest priority 3
  - A residual weighted bandwidth between different low-priority requests (respectively lowest priority 0 vs. priority 1 vs. priority 2)
    - The different weights are monotonically resulting from the programmed channel priorities



life.augmented

3

The DMA controller implements a programmable arbitration logic, that enables the user to adjust channel bandwidth and latency requirements, according to the following rules.
The priority of the request is programmable from zero to three.
Requests having the same priority are handled with a round-robin arbitration scheme.
Time-sensitive requests should be assigned the priority three, which is handled with a fixed higher priority scheme over the 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, the queue 0 having the lowest weight.

GPDMA burst

- In FIFO-mode, generally, a data transfer consists of two scheduled burst transfers:
  - The read burst from the source to the FIFO over the SAP allocated master port
  - The write burst from the FIFO to the destination over the DAP (typically the other master port)
- ASAP FIFO scheduling :
  - Source burst : whenever the FIFO is ready to get one new programmed source burst (burst size ≤ half of the FIFO size)
  - Destination burst: whenever the FIFO is ready to push one new programmed destination burst
- A ready FIFO-based burst request transfer is arbitrated vs. other active and simultaneous requests/channels

life.augmented

4

This slide and the six next ones will clarify the relationship between the software settings and the hardware transactions for the General Purpose DMA.
The GPDMA supports a single transfer operation mode: the FIFO mode
A programmed transfer at the lowest level is a GPDMA burst.
A GPDMA burst is a burst of data received from the source, or a burst of data sent to the destination.
Since the GPDMA has two AHB master ports, source and destination burst can be performed concurrently.
The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on when the FIFO is ready to get one new burst

from the source.
The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on when the FIFO is ready to push one new burst to the destination.
Based on the channel priority, these ready FIFO-based source and destination transfers are internally arbitrated versus the other requested and active channels.

GPDMA burst

SDW_LOG2[1:0], DDW_LOG2[1:0]	Data width (bytes)	SINC/ DINC	SBL_1[5:0], DBL_1[5:0]	Burst length (Data/beats)	Next data / beat address	Next burst address	Burst address alignment
00	1	0 (Fixed)	n = 0 to 63	n+1	+0	+0	1
01	2						2
10	4						4
00	1	1 (Contiguously incremented)			+1	+(n+1)	1
01	2				+2	+2*(n+1)	2
10	4				+4	+4*(n+1)	4
11	Causes USEF generation and none single to be issued						

life.augmented

5

This table lists the main characteristics of a GPDMA burst
A source and destination burst is programmed with a burst length by the field SBL_1, respectively DBL_1, and with a data width defined by the field SDW_LOG2, respectively DDW_LOG2, in the GPDMA_CxTR1 register.
Programming SDW_LOG2 or DDW_LOG2 with the binary value 11 leads to a user setting error.
The addressing mode after each data, named beat, of a GPDMA burst is defined by SINC and DINC, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.
The start and next addresses of a GPDMA source/destination burst must be aligned with the respective data width.

When the source or destination address increment mode is selected, the address is automatically updated at the end of a burst with the burst size in byte units, which is equal to the burst length + 1 multiplied by the data width. When the burst length is 1, the burst can be also named as single.

GPDMA burst

- A programmed burst may be implemented in hardware:
  - As is (same AHB burst transaction on the master port), or
  - By a series of burst(s) of lower length or/and single(s); possibly due to:
    - Burst size > half of the FIFO size
    - Burst size is not a divisor of the LOG2 of the FIFO size
    - (Source) block size is not a multiple of the source data width
    - AHB constraints
      - 1kB address boundary crossing
      - One of 4-beat, 8-beat, and 16-beat incremental burst
- In any case:
  - Data/addressing integrity is guaranteed without any user constraint
  - Hardware maximises the performance by implementing the largest allowed burst size according to the programmed burst

life.augmented

6

The programmed source and destination GPDMA burst is implemented with an AHB burst as is, unless one of the following conditions is met.
When half of the FIFO size of the channel is lower than the programmed source or destination burst size, the programmed source or destination GPDMA burst is implemented with a series of singles or bursts of a lower size, each transfer being of a size that is lower than or equal to half of the FIFO size.
If the source block size is not a multiple of the source burst size but is a multiple of the data width of the source burst, the GPDMA modifies and shortens bursts into single-data transactions or bursts of lower length, in order to transfer exactly the number of bytes equal to the source block size.

If the source or destination burst transfer crosses the 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.
If the source/destination burst length exceeds 16 on a AHB transfer, the GPDMA modifies and shortens the programmed burst into single-data transactions or bursts of lower length, to be compliant with the AHB protocol.
In any case, the GPDMA keeps ensuring source and destination data and address integrity without any user constraint.

GPDMA data handling

Byte-based **reordering, packing/unpacking**, padding/truncation, sign extension & L/R alignment

SDW_LOG2[1:0]	Source data	Source data stream	SBX	DDW_LOG2[1:0]	Destination data	PAM[1:0]	DBX	DHX	Destination data stream
00	Byte	$B_7,B_6,B_5,B_4,$ B_3,B_2,B_1,B_0	x	00	Byte	xx	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (Right-aligned, Zero-bit padding)	0		$\underline{0B_3},\underline{0B_2},\underline{0B_1},\underline{0B_0}$
							1		$\underline{B_3 0},\underline{B_2 0},\underline{B_1 0},\underline{B_0 0}$
						00 (Right-aligned, Sign-bit extended)	0		$\underline{SB_3},\underline{SB_2},\underline{SB_1},\underline{SB_0}$
							1		$\underline{B_3S},\underline{B_2S},\underline{B_1S},\underline{B_0S}$
						1x (PACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (Right-aligned, Zero-bit padding)	0	0	$\underline{000B_1},\underline{000B_0}$
							1		$\underline{00B_1 0},\underline{00B_0 0}$
							0	1	$\underline{0B_1 00},\underline{0B_0 00}$
							1		$\underline{B_1 000},\underline{B_0 000}$
						01 (Right-aligned, Sign-bit extend)	0	0	$\underline{SSSB_1},\underline{SSSB_0}$
							1		$\underline{SSB_1 S},\underline{SSB_0 S}$
							0	1	$\underline{SB_1 SS},\underline{SB_0 SS}$
							1		$\underline{B_1 SSS},\underline{B_0 SSS}$
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$

life.augmented

7

The table lists the possible data handling from the source to the destination when the source data width is a byte.
The source/destination data width of the programmed burst is byte, half-word or word, as per the SDW_LOG2 and DDW_LOG2 fields.
The user can configure the data handling between transferred data from the source and transfer to the destination.
More specifically, programmed data handling is orderly performed with:
1.Byte-based source reordering
2. Data width conversion by packing, unpacking, padding or truncation, if destination data width is different than the source data width, depending on padding/alignment mode

control field
3. Byte-based destination re-ordering.

If destination data width is larger than source data width, the source data is either right-aligned and padded with 0 s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
If destination data width is lower than source data width, the source data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.

If DBX equals 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM half-words.
For instance, the Byte 7 Byte 6 halfword becomes the Byte 6 Byte 7 halfword.
If DHX equals 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM words.
For instance, the Byte 7- Byte 6- Byte 5- Byte 4 word becomes the Byte 6- Byte 7- Byte 4 – Byte 5 word.

GPDMA data handling

Byte-based **reordering, packing/unpacking**, padding/truncation, sign extension & L/R alignment

SDW_LOG2[1:0]	Source data	Source data stream	SBX	DDW_LOG2[1:0]	Destination data	PAM[1:0]	DBX	DHX	Destination data stream
01	Halfword	$B_7B_6,B_5B_4,$ B_3B_2,B_1B_0	x	00	Byte	00 (Right-aligned, Left-truncated)	x	x	B_6,B_4,B_2,B_0
						01 (Left-aligned, Right-truncated)			B_7,B_5,B_3,B_1
						1x (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	xx	0	x	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (Right-aligned, Zero-bit padding)	0	0	$\underline{00}B_3B_2,\underline{00}B_1B_0$
							1		$\underline{00}B_2B_3,\underline{00}B_0B_1$
							0	1	$B_3B_2\underline{00},B_1B_0\underline{00}$
							1		$B_2B_3\underline{00},B_0B_1\underline{00}$
						01 (Right-aligned, Sign-bit extend)	0	0	$\underline{SS}B_3B_2,\underline{SS}B_1B_0$
							1		$\underline{SS}B_2B_3,\underline{SS}B_0B_1$
							0	1	$B_3B_2\underline{SS},B_1B_0\underline{SS}$
							1		$B_2B_3\underline{SS},B_0B_1\underline{SS}$
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$

life.augmented

8

The table lists the possible data handling from the source to the destination when the source data width is a 16-bit halfword.

GPDMA data handling

Byte-based **reordering, packing/unpacking**, padding/truncation, sign extension & L/R alignment

SDW_LOG2[1:0]	Source data	Source data stream	SBX	DDW_LOG2[1:0]	Destination data	PAM[1:0]	DBX	DHX	Destination data stream
10	Word	$B_{15}B_{14}B_{13}B_{12}$, $B_{11}B_{10}B_9B_8$, $B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	00	Byte	00 (Right-aligned, Left-truncated)	x	x	$B_{12}.B_8.B_4.B_0$
						01 (Left-aligned, Right-truncated)			$B_{15}.B_{11}.B_7.B_3$
						10 (UNPACK)			$B_7.B_6.B_5.B_4.B_3.B_2.B_1.B_0$
				01	Half-word	00 (Right-aligned, Left-truncated)	0		$B_5B_4.B_1B_0$
							1		$B_4B_5.B_0B_1$
						01 (Left-aligned, Right-truncated)	0		$B_7B_6.B_3B_2$
							1		$B_6B_7.B_2B_3$
						1x (UNPACK)	0		$B_7B_6.B_5B_4.B_3B_2.B_1B_0$
							1		$B_6B_7.B_4B_5.B_2B_3.B_0B_1$
				10	Word	xx	0	0	$B_7B_6B_5B_4.B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5.B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6.B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7.B_0B_1B_2B_3$
			1	00	Byte	00 (Right-aligned, Left-truncated)	x	x	$B_{12}.B_8.B_4.B_0$
						01 (Left-aligned, Right-truncated)			$B_{15}.B_{11}.B_7.B_3$
						1x (UNPACK)			$B_7.B_5.B_6.B_4.B_3.B_1.B_2.B_0$
				01	Half-word	00 (Right-aligned, Left-truncated)	0		$B_6B_4.B_2B_0$
							1		$B_4B_6.B_0B_2$
						01 (Left-aligned, Right-truncated)	0		$B_7B_5.B_3B_1$
							1		$B_5B_7.B_1B_3$
						1x (UNPACK)	0		$B_7B_5.B_6B_4.B_3B_1.B_2B_0$
							1		$B_5B_7.B_4B_6.B_1B_3.B_0B_2$
				10	Word	xx	0	0	$B_7B_5B_6B_4.B_3B_1B_2B_0$
							1		$B_5B_7B_4B_6.B_1B_3B_0B_2$
							0	1	$B_6B_4B_7B_5.B_2B_0B_3B_1$
							1		$B_4B_6B_5B_7.B_0B_2B_1B_3$

life.augmented

9

The table lists the possible data handling from the source to the destination when the source data width is a 32-bit word.

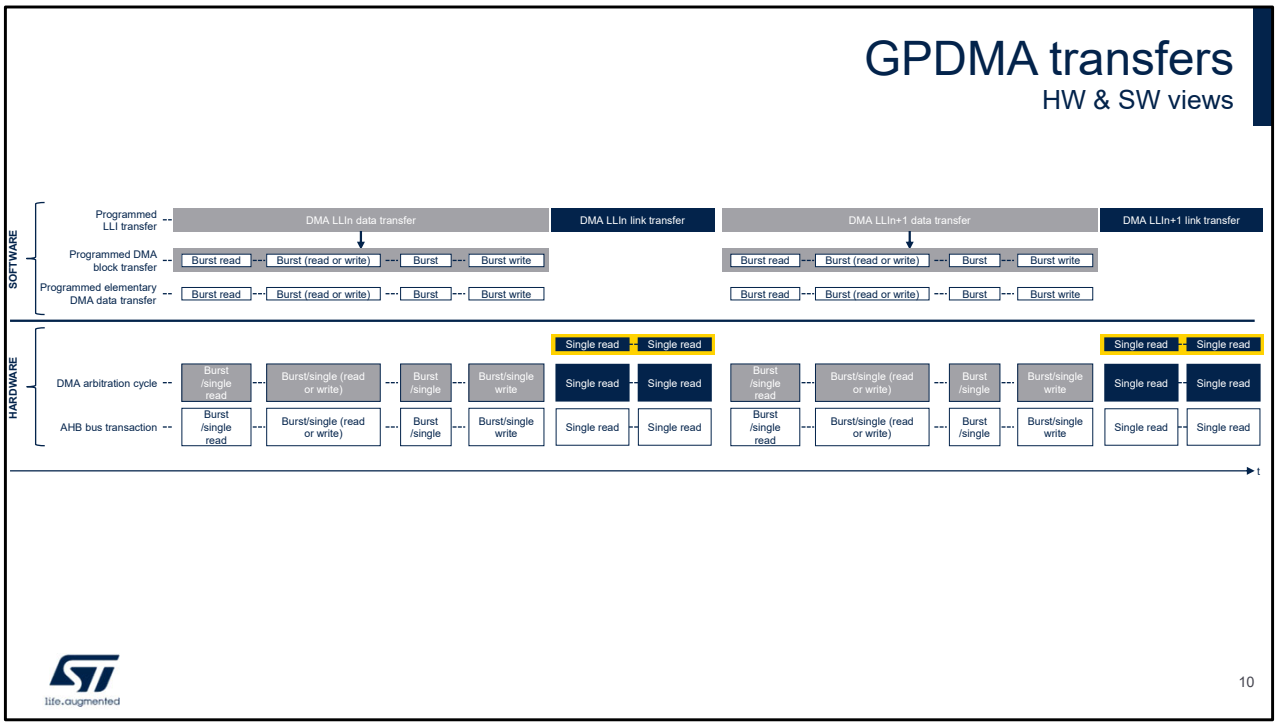
If SBX is equal to 1 and if source data width is a word, the two bytes of the unaligned half-word at the middle of each source data word are exchanged.

For instance, Bytes 7-6-5-4 become Bytes 7-5-6-4.

If destination data width is larger than source data width, the post SBX source data is either right-aligned and padded with 0 s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.

If destination data width is lower than source data width, the post SBX data is either right-aligned and left-truncated

down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.



This timing diagram highlights the relationship between the software configuration of a channel and the transactions that are generated on the AHB master port.
An ongoing GPDMA transfer can be a data transfer, which includes source and destination burst transfers, or a link transfer for the internal update of the linked-list register file from the next linked-list item.
Two consecutive Linked-List Items or LLIs are represented: first LLI number n and then LLI number n+1.
The data transfer is composed of a series of burst data reads followed by burst data writes, because the GPDMA support FIFOs.
The GPDMA may modify and shorten bursts into singles or bursts of lower length, as explained in the previous slides.

Each burst or single data transfer requires an arbitration cycle and a transfer over the AHB master port.
The link transfer is composed of single data reads, each of them requiring an arbitration cycle and a transfer over the AHB master port.

Thank you

life.augmented

In addition to this presentation, you can refer to the other presentations on the GPDMA:

- DMA overview
- Autonomous DMA & low power mode
- DMA linked list
- DMA Circular buffering & double buffering
- DMA 2D addressing
- DMA Register file
- DMA Error reporting
- DMA Input-output LLI control.