



life.augmented

## STM32U5

DMA: DMA transfers  
hardware and software views

Rev 1.0

Hello, and welcome to this presentation which 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 to/from memory)
  - A hardware request from a peripheral to update its control register(s) from memory
  - A hardware request from a peripheral to read of its status register(s) to memory
  - A software request typically from the CPU for a data transfer from a memory-mapped address to another memory mapped address



To start a DMA transfer, a request is required.

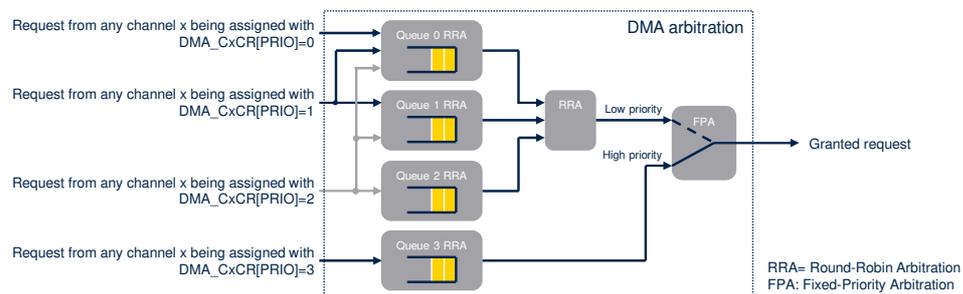
This request can be caused by a hardware event:

- 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 to 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 the 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



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 3, 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.

## LPDMA single

| Programmed LPDMA source/destination single | SDW_LOG2[1:0], DDW_LOG2[1:0] | Data width (bytes)                                | SINC/DINC | Addressing mode | SAR/DAR next single address | Address alignment |
|--|------------------------------|---|-----------|-----------------|-----------------------------|-------------------|
| Fixed byte single                          | 00                           | 1   | 0         | Fixed           | +0                          | 1                 |
| Fixed half-word single                     | 01                           | 2   |           |                 |                             | 2                 |
| Fixed word single                          | 10                           | 4   |           |                 |                             | 4                 |
| Incremented byte single                    | 00                           | 1   | 1         | Incremented     | +1                          | 1                 |
| Incremented half-word single               | 01                           | 2   |           |                 |                             | 2                 |
| Incremented word single                    | 10                           | 4   |           |                 |                             | 4                 |
| Forbidden                                  | 11                           | Causes USEF generation and no single to be issued |           |                 |                             |                   |



This slide and the next two clarify the relationship between the software settings and the hardware transactions for the low power DMA.

A programmed transfer at the lowest level is called a LPDMA single.

A LPDMA single data width is 1, 2 or 4 bytes, as defined by the two LSbs of the SDW\_LOG2 and DDW\_LOG2 fields of the LPDMA\_CxTR1 register. Programming the binary value 11 in these fields causes the user setting error flag to be set.

Note that the width is programmed independently for source and destination transfers.

The addressing mode after each data of a LPDMA single is defined by the SINC and DINC bits of the LPDMA\_CxTR1 register: either a fixed addressing or an

incremented addressing with contiguous data.  
The start and next addresses of a LPDMA source and destination single must be aligned with the respective data width.

# LPDMA data handling

Byte-based padding/truncation, sign extension & left/right alignment

| SDW_LOG2[1:0]                        | Source data              | Source data stream                       | DDW_LOG2[1:0] | Destination data | PAM[0]                                   | Destination data stream                  |                                     |  |                            |
|--------------------------------------|--------------------------|--|---------------|------------------|--|--|-------------------------------------|--|----------------------------|
| 00                                   | Byte                     | $B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0$ | 00            | Byte             | xx                                       | $B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0$ |                                     |  |                            |
|                                      |                          |  | 01            | Half-word        | 0 (Right-Aligned, Zero bit padding)      | $0B_3, 0B_2, 0B_1, 0B_0$                 |                                     |  |                            |
|                                      |                          |  |               |                  | 1 (Right-Aligned, Sign bit extended)     | $SB_3, SB_2, SB_1, SB_0$                 |                                     |  |                            |
|                                      |                          |  | 10            | Word             | 0 (Right-Aligned, Zero bit padding)      | $000B_1, 000B_0$                         |                                     |  |                            |
|                                      |                          |  |               |                  | 1 (Right-Aligned, Sign bit extended)     | $SSSB_1, SSSB_0$                         |                                     |  |                            |
|                                      |                          |  | 01            | Half-word        | $B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0$ | 00                                       | Byte                                | 0 (Right-Aligned, Left Truncated)        | $B_6, B_4, B_2, B_0$       |
| 01                                   | Half-word                | 1 (Left Aligned, Right Truncated)        |               |                  |  | $B_7, B_5, B_3, B_1$                     |                                     |  |                            |
|                                      |                          | 10                                       |               |                  |  | Word                                     | 0 (Right-Aligned, Zero bit padding) | $000B_3, B_2, 000B_1, B_0$               |                            |
| 1 (Right-Aligned, Sign bit extended) | $SSB_3, B_2, SSB_1, B_0$ |  |               |                  |  |  |                                     |  |                            |
| 10                                   | Word                     | $B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0$ |               |                  |  | 00                                       | Byte                                | 0 (Right-Aligned, Left Truncated)        | $B_{12}, B_8, B_4, B_0$    |
|                                      |                          |  |               |                  |  | 01                                       | Half-word                           | 1 (Left Aligned, Right Truncated)        | $B_{15}, B_{11}, B_7, B_3$ |
|                                      |                          |  | 10            | Word             | 0 (Right-Aligned, Left Truncated)        |  |                                     | $B_5, B_4, B, B_0$                       |                            |
|                                      |                          |  |               |                  | 1 (Left Aligned, Right Truncated)        | $B_7, B_6, B_3, B_2$                     |                                     |  |                            |
|                                      |                          |  |               |                  |  |  | xx                                  | $B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0$ |                            |



The user can configure the data handling between a data transferred from the source to the destination. The table lists all possible data handling from the source to the destination.

If the destination data width is equal to the source data width, the source data is copied as is and transferred to the destination.

Otherwise, it depends on the padding and alignment mode control bits:

- If the destination data width is larger than the source data width, the source data can be either right-aligned and padded with 0s, or sign extended up to the destination data width.

- If the destination data width is smaller than the source data width, the source data can be either right-aligned

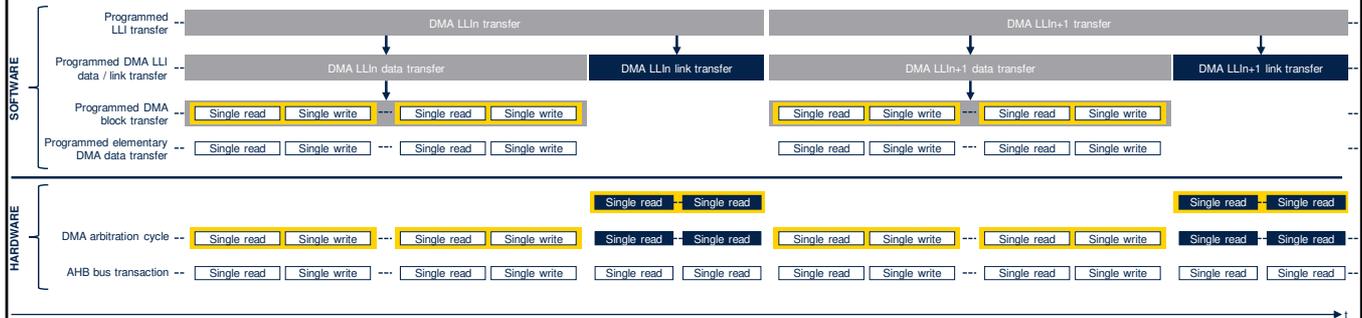
and left-truncated down to the destination data width, or left-aligned and right-truncated down to the destination data width.

There is no data manipulation between two distinct data transferred from the source, before the destination transfer is generated.

In this table, the time-ordering of the data stream is conventionally taken with the first byte of data transferred being the lowest index, B0.

# LPDMA transfers

HW & SW views



This timing diagram highlights the relationship between the software configuration of a channel and the transactions that are generated on the AHB master port. Two consecutive Linked-List Items or LLIs are represented: first LLI number  $n$  and then LLI number  $n+1$ .

An LLI transfer can include a data transfer and a link transfer.

The LPDMA supports a single transfer operation mode: the direct mode.

A data transfer is a single data read immediately followed by a single data write, because the LPDMA does not support FIFOs.

Each single read followed by a write data transfer requires an arbitration cycle, before being transferred

over the AHB master port.

The data transfer is a block transfer, in other words, a series of read followed by single transfers writes.

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

The link transfer updates channel control registers from values read from a structure in memory.

## 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  $\leq$  half of the FIFO size)
  - Destination burst: whenever the FIFO is ready to push one new programmed destination burst
- A burst request of a ready FIFO-based transfer is arbitrated versus other active and simultaneous requests/channels



This slide and the next six 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 bursts can be executed simultaneously.

The requested source burst transfer to the FIFO can be scheduled as soon as possible over the allocated port, depending on when the FIFO is ready to get a new burst from the source.

The requested destination burst transfer from the FIFO can be scheduled as soon as possible over the allocated port, depending on when the FIFO is ready to push a 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],<br>DDW_LOG2[1:0] | Data width<br>(bytes)                             | SINC/<br>DINC                   | SBL_1[5:0],<br>DBL_1[5:0] | Burst length<br>(Data/beats) | Next data /<br>beat address | Next burst<br>address | Burst address<br>alignment |        |    |          |   |
|---------------------------------|---|---------------------------------|---------------------------|------------------------------|-----------------------------|-----------------------|----------------------------|--------|----|----------|---|
| 00                              | 1   | 0 (Fixed)                       | n = 0 to 63               | n+1                          | +0                          | +0                    | 1                          |        |    |          |   |
| 01                              | 2   |                                 |                           |                              |                             |                       | 2                          |        |    |          |   |
| 10                              | 4   |                                 |                           |                              |                             |                       | 4                          |        |    |          |   |
| 00                              | 1   | 1 (Contiguously<br>incremented) |                           |                              | n = 0 to 63                 | n+1                   | +1                         | +(n+1) | 1  |          |   |
| 01                              | 2   |                                 |                           |                              |                             |                       |                            |        | +2 | +2*(n+1) | 2 |
| 10                              | 4   |                                 |                           |                              |                             |                       |                            |        | +4 | +4*(n+1) | 4 |
| 11                              | Causes USEF generation and no single to be issued |                                 |                           |                              |                             |                       |                            |        |    |          |   |

This table lists the main characteristics of a GPDMA burst

A source and destination burst is programmed with a burst length defined by the fields SBL\_1 and DBL\_1 respectively, and a data width defined by the fields SDW\_LOG2 and DDW\_LOG2 respectively, 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 called a single.

- 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 LOG2 divisor 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, 8 and 16-beat incremental bursts
- 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



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 smaller size, each transfer being of a size that is less than or equal to half 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 singles or bursts of shorter length, in order to transfer exactly 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 shorter 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 singles or bursts of shorter length, to be compliant with the AHB protocol.

In all cases, the GPDMA ensures 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 <sub>7</sub> ,B <sub>6</sub> ,B <sub>5</sub> ,B <sub>4</sub> ,<br>B <sub>3</sub> ,B <sub>2</sub> ,B <sub>1</sub> ,B <sub>0</sub> | x   | 00            | Byte  | xx                                    | x   |     | B <sub>7</sub> ,B <sub>6</sub> ,B <sub>5</sub> ,B <sub>4</sub> ,B <sub>3</sub> ,B <sub>2</sub> ,B <sub>1</sub> ,B <sub>0</sub>          |
|               |             |  |     |               |   | 00 (Right-aligned, Zero-bit padding)  | 0   |     | 0B <sub>7</sub> 0B <sub>6</sub> 0B <sub>5</sub> 0B <sub>4</sub> 0B <sub>3</sub> 0B <sub>2</sub> 0B <sub>1</sub> 0B <sub>0</sub>         |
|               |             |  |     |               |   |                                       | 1   |     | B <sub>7</sub> 0 B <sub>6</sub> 0 B <sub>5</sub> 0 B <sub>4</sub> 0 B <sub>3</sub> 0  |
|               |             |  |     |               |   | 00 (Right-aligned, Sign-bit extended) | 0   | x   | SB <sub>7</sub> SB <sub>6</sub> SB <sub>5</sub> SB <sub>4</sub> SB <sub>3</sub> SB <sub>2</sub> SB <sub>1</sub> SB <sub>0</sub>         |
|               |             |  |     |               |   |                                       | 1   |     | B <sub>7</sub> S B <sub>6</sub> S B <sub>5</sub> S B <sub>4</sub> S B <sub>3</sub> S B <sub>2</sub> S B <sub>1</sub> S B <sub>0</sub> S |
|               |             |  |     |               |   | 1x (PACK)                             | 0   |     | B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>                 |
|               |             |  |     |               | 1   |                                       | B <sub>6</sub> B <sub>7</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |     |   |
|               |             |  |     | 10            | Word  | 00 (Right-aligned, Zero-bit padding)  | 0   | 0   | 000B <sub>7</sub> ,000B <sub>6</sub>  |
|               |             |  |     |               |   |                                       |   | 1   | 00B <sub>7</sub> 00B <sub>6</sub> 00  |
|               |             |  |     |               |   |                                       |   | 1   | B <sub>7</sub> 000 B <sub>6</sub> 000   |
|               |             |  |     |               |   |                                       | 0   | 0   | SSSB <sub>7</sub> SSSB <sub>6</sub>   |
|               |             |  |     |               |   |                                       |   | 1   | SSB <sub>7</sub> SSSB <sub>6</sub> S  |
|               | 0           | SSSB <sub>7</sub> SSSB <sub>6</sub> SS   |     |               |   |                                       |   |     |   |
| 1x (PACK)     | 0           | 01 (Right-aligned, Sign-bit extended)  | 1   | 0             | B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |                                       |   |     |   |
|               |             |  |     | 1             | B <sub>6</sub> B <sub>7</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |                                       |   |     |   |
|               |             |  |     | 0             | B <sub>5</sub> B <sub>7</sub> B <sub>6</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |                                       |   |     |   |
|               |             |  |     | 1             | B <sub>5</sub> B <sub>6</sub> B <sub>7</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |                                       |   |     |   |
|               |             |  |     | 0             | B <sub>4</sub> B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |                                       |   |     |   |
|               |             |  |     | 1             | B <sub>4</sub> B <sub>6</sub> B <sub>7</sub> B <sub>5</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |                                       |   |     |   |



This 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 data transferred from the source to the destination.

The data handling that can be performed is:

1. Byte-based source reordering
2. Data width conversion by packing, unpacking, padding or truncation, if the destination data width is different to the source data width, depending on the padding/alignment mode control field
3. Byte-based destination re-ordering.

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

If the destination data width is smaller than the 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  $DBX = 1$  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  $DHX = 1$  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 <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> ,<br>B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> | x   | 00            | Byte             | 00 (Right-aligned, Left-truncated)   | x   | x                                    | B <sub>6</sub> B <sub>4</sub> B <sub>2</sub> B <sub>0</sub>   |   |   |
|               |             |  |     |               |                  | 01 (Left-aligned, Right-truncated)   |   |                                      | B <sub>7</sub> B <sub>5</sub> B <sub>3</sub> B <sub>1</sub>   |   |   |
|               |             |  |     |               |                  | 1x (UNPACK)                          |   |                                      | B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |   |   |
|               |             |  |     | 01            | Half-word        | xx                                   | x   | 00 (Right-aligned, Zero-bit padding) | 0   | 0 | B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |
|               |             |  |     |               |                  |                                      |   |                                      | 1   | 0 | B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>                |
|               |             |  |     |               |                  |                                      |   |                                      | 0   | 1 | 00B <sub>2</sub> 00B <sub>1</sub> B <sub>0</sub>  |
|               |             |  |     | 10            | Word             | 00 (Right-aligned, Zero-bit padding) | 0   | 01 (Right-aligned, Sign-bit extend)  | 0   | 0 | 00B <sub>2</sub> B <sub>3</sub> 00B <sub>1</sub> B <sub>0</sub>   |
|               |             |  |     |               |                  |                                      |   |                                      | 1   | 1 | B <sub>7</sub> B <sub>6</sub> 00B <sub>1</sub> B <sub>0</sub> 00  |
|               |             |  |     |               |                  |                                      |   |                                      | 0   | 0 | SSB <sub>2</sub> B <sub>3</sub> SSB <sub>1</sub> B <sub>0</sub>   |
|               |             |  |     |               |                  |                                      |   |                                      | 1   | 0 | SSB <sub>2</sub> B <sub>3</sub> SSB <sub>1</sub> B <sub>0</sub>   |
|               |             |  |     |               |                  |                                      |   |                                      | 0   | 1 | B <sub>7</sub> B <sub>6</sub> SSB <sub>1</sub> B <sub>0</sub> SS  |
|               |             |  |     |               |                  |                                      |   |                                      | 1   | 1 | B <sub>2</sub> B <sub>3</sub> SSB <sub>1</sub> B <sub>0</sub> SS  |
| 1x (PACK)     | 0           | 0  | 0   | 1x (PACK)     | 0                | 0                                    | B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |                                      |   |   |   |
|               |             |  |     |               | 1                | 0                                    | B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>                |                                      |   |   |   |
|               |             |  |     |               | 0                | 1                                    | B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> B <sub>7</sub> B <sub>6</sub> |                                      |   |   |   |
|               |             |  |     |               | 1                | 1                                    | B <sub>4</sub> B <sub>5</sub> B <sub>6</sub> B <sub>7</sub> B <sub>0</sub> B <sub>1</sub> B <sub>2</sub> B <sub>3</sub> |                                      |   |   |   |



This 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 <sub>15</sub> B <sub>14</sub> B <sub>13</sub> B <sub>12</sub> ,<br>B <sub>11</sub> B <sub>10</sub> B <sub>9</sub> ,<br>B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> ,<br>B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> | 0   | 00   | Byte  | 00 (Right-aligned, Left-truncated) | x  | x   | B <sub>12</sub> B <sub>8</sub> B <sub>4</sub> B <sub>0</sub>  |   |   |
|               |             |   |     |  |   | 01 (Left-aligned, Right-truncated) |  |   | B <sub>15</sub> B <sub>11</sub> B <sub>7</sub> B <sub>3</sub>   |   |   |
|               |             |   |     |  |   | 10 (UNPACK)                        |  |   | B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |   |   |
|               |             |   |     | 01   | Half-word   | 00 (Right-aligned, Left-truncated) | 0  | B <sub>6</sub> B <sub>4</sub> B <sub>0</sub>                |   |   |   |
|               |             |   |     |  |   | 01 (Left-aligned, Right-truncated) | 1  | B <sub>6</sub> B <sub>5</sub> B <sub>3</sub> B <sub>1</sub> |   |   |   |
|               |             |   |     |  |   | 1x (UNPACK)                        | 0  | B <sub>6</sub> B <sub>7</sub> B <sub>2</sub> B <sub>3</sub> |   |   |   |
|               |             |   | 10  | Word   | xx  | 0                                  | B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>                |   |   |   |   |
|               |             |   |     |  |   | 1                                  | B <sub>6</sub> B <sub>7</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>                |   |   |   |   |
|               |             |   |     |  |   | 0                                  | B <sub>6</sub> B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |   |   |   |   |
|               |             |   | 1   | Word   | B <sub>15</sub> B <sub>14</sub> B <sub>13</sub> B <sub>12</sub> ,<br>B <sub>11</sub> B <sub>10</sub> B <sub>9</sub> ,<br>B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> ,<br>B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> | 00                                 | Byte   | 00 (Right-aligned, Left-truncated)                          | x   | x | B <sub>12</sub> B <sub>8</sub> B <sub>4</sub> B <sub>0</sub>  |
|               |             |   |     |  |   |                                    |  | 01 (Left-aligned, Right-truncated)                          |   |   | B <sub>15</sub> B <sub>11</sub> B <sub>7</sub> B <sub>3</sub>   |
|               |             |   |     |  |   |                                    |  | 1x (UNPACK)   |   |   | B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |
| 01            | Half-word   | 00 (Right-aligned, Left-truncated)  |     |  |   | 0                                  | B <sub>6</sub> B <sub>4</sub> B <sub>0</sub>   |   |   |   |   |
|               |             | 01 (Left-aligned, Right-truncated)  |     |  |   | 1                                  | B <sub>6</sub> B <sub>5</sub> B <sub>3</sub> B <sub>1</sub>  |   |   |   |   |
|               |             | 1x (UNPACK)   |     |  |   | 0                                  | B <sub>6</sub> B <sub>7</sub> B <sub>2</sub> B <sub>3</sub>  |   |   |   |   |
| 10            | Word        | xx  | 0   | B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>                |   |                                    |  |   |   |   |   |
|               |             |   | 1   | B <sub>6</sub> B <sub>7</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>                |   |                                    |  |   |   |   |   |
|               |             |   | 0   | B <sub>6</sub> B <sub>7</sub> B <sub>6</sub> B <sub>5</sub> B <sub>4</sub> B <sub>3</sub> B <sub>2</sub> B <sub>1</sub> B <sub>0</sub> |   |                                    |  |   |   |   |   |



This 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 the 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 becomes Bytes 7-5-6-4.

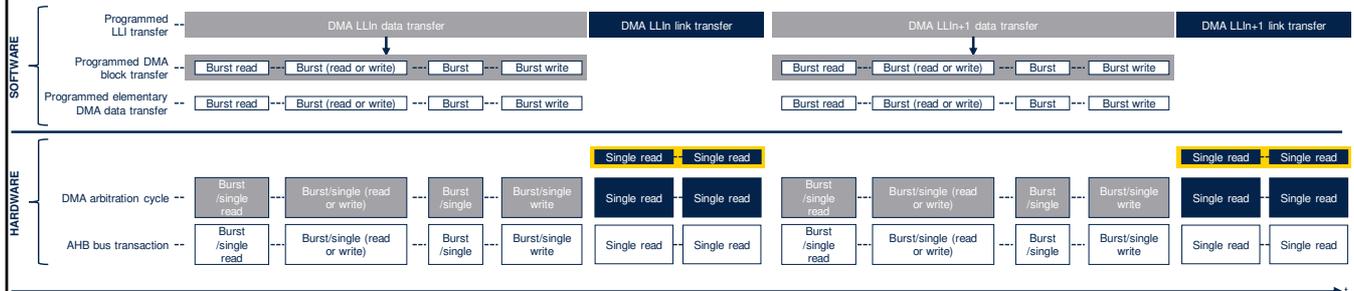
If the destination data width is larger than the source data width, the post SBX source data is either right-aligned and padded with 0s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.

If the destination data width is lower than the 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.

# GPDMA transfers

HW & SW views



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

© STMicroelectronics - All rights reserved.  
ST logo is a trademark or a registered trademark of STMicroelectronics International NV or its affiliates in the EU and/or other countries.  
For additional information about ST trademarks, please refer to [www.st.com/trademarks](http://www.st.com/trademarks).  
All other product or service names are the property of their respective owners.



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

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