Customer training workshop
TRAVEO™ T2G Direct Memory Access (DMA)
## Target products

- **Target product list for this training material**

<table>
<thead>
<tr>
<th>Family category</th>
<th>Series</th>
<th>Code flash memory size</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAVEO™ T2G Automotive Body Controller Entry</td>
<td>CYT2B6</td>
<td>Up to 576KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Body Controller Entry</td>
<td>CYT2B7</td>
<td>Up to 1088KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Body Controller Entry</td>
<td>CYT2B9</td>
<td>Up to 2112KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Body Controller Entry</td>
<td>CYT2BL</td>
<td>Up to 4160KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Body Controller High</td>
<td>CYT3BB/CYT4BB</td>
<td>Up to 4160KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Body Controller High</td>
<td>CYT4BF</td>
<td>Up to 8384KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Cluster Entry</td>
<td>CYT2CL</td>
<td>Up to 4160 KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Cluster 2D</td>
<td>CYT3DL</td>
<td>Up to 4160KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Cluster 2D</td>
<td>CYT4DN</td>
<td>Up to 6336KB</td>
</tr>
<tr>
<td>TRAVEO™ T2G Automotive Cluster 2D</td>
<td>CYT4EN</td>
<td>Up to 6336 KB</td>
</tr>
</tbody>
</table>
Introduction to TRAVEO™ T2G Body Controller Entry

- DMA is part of the CPU subsystem

**CPU subsystem**

- Arm® Cortex® M4 160 MHz
- 4160 KB Code-flash + 128 KB Work-flash
- SRAM0 256 KB
- SRAM1 256 KB
- Crypto AES, SHA, CRC, TRNG, RSA, ECC

**System interconnect**

- Multi Layer AHB, IPC, MPU/SMPU

**Peripheral interconnect**

- IOSS GPIO
- PCLK
- I²C, SPI, UART
- ARM® Cortex® M0+ 100 MHz
- MUL, NVIC, MPU

**Power modes**

- Active/Deep Sleep
- Lower Power/Active/Deep Sleep
- Hibernate

**Power resources**

- SWJ/MTB/CTI
- SWJ/ETM/ITM/CTI
- SWJ/ETM/ITM/CTI

**System resources**

- Power
- Clock
- Reset
- Test
- Digital DFT
- Analog DFT

**System interconnect**

- System interconnect (Multi Layer AHB, IPC, MPU/SMPU)

**Peripheral interconnect**

- Peripheral interconnect (MMIO, PPU)

**Power modes**

- High-speed I/O Matrix, Smart I/O, Boundary Scan
- 5x Smart I/O
- 8x GPIO_STD
- 4x GPIO_ENH

**Power modes**

- DeepSleep
- Hibernate

**System resources**

- System resources
- Power
- Clock
- Reset
- Test
- Digital DFT
- Analog DFT

**Copyright © Infineon Technologies AG 2022. All rights reserved.**

002-22194 *F 2022-08-17
Introduction to TRAVEO™ T2G Body Controller High

DMA is part of the CPU subsystem
Introduction to TRAVEO™ T2G Cluster

- DMA is part of the CPU subsystem
P-DMA / M-DMA
DMA overview

TRAVEO™ T2G has two types of DMA
- Peripheral DMA (P-DMA)
- Memory DMA (M-DMA)

<table>
<thead>
<tr>
<th>Feature</th>
<th>P-DMA</th>
<th>M-DMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focus</td>
<td>Low latency</td>
<td>High memory bandwidth</td>
</tr>
<tr>
<td>Used for</td>
<td>Transfer between peripheral and memory¹</td>
<td>Transfer between memories²</td>
</tr>
<tr>
<td>Transfer engine</td>
<td>Shared between all channels</td>
<td>Dedicated for each channel</td>
</tr>
<tr>
<td>Transfer size</td>
<td>8-bit/16-bit/32-bit</td>
<td>8-bit/16-bit/32-bit</td>
</tr>
<tr>
<td>Channel priority</td>
<td>Four levels</td>
<td>Four levels</td>
</tr>
<tr>
<td>Priority</td>
<td>Preemptable</td>
<td></td>
</tr>
<tr>
<td>Transfer mode</td>
<td>- Single</td>
<td>- Single</td>
</tr>
<tr>
<td></td>
<td>- 1D/2D</td>
<td>- 1D/2D</td>
</tr>
<tr>
<td></td>
<td>- CRC transfer</td>
<td>- Memory copy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Scatter</td>
</tr>
<tr>
<td>Descriptor</td>
<td>- Source and destination address</td>
<td>- Source and destination address</td>
</tr>
<tr>
<td></td>
<td>- Transfer size</td>
<td>- Channel action</td>
</tr>
<tr>
<td></td>
<td>- Channel action</td>
<td>- Data transfer mode</td>
</tr>
<tr>
<td></td>
<td>- Data transfer mode</td>
<td>- Activation trigger type (4 types)</td>
</tr>
<tr>
<td></td>
<td>- Activation trigger type (4 types)</td>
<td>- Output trigger type (4 types)</td>
</tr>
<tr>
<td></td>
<td>- Output trigger type (4 types)</td>
<td>- Interrupt type (4 types)</td>
</tr>
<tr>
<td></td>
<td>- Interrupt type (4 types)</td>
<td>- Descriptor chaining</td>
</tr>
<tr>
<td></td>
<td>- Descriptor chaining</td>
<td></td>
</tr>
<tr>
<td>Access-control attributes³</td>
<td>- Privileged/Unprivileged</td>
<td>- Privileged/Unprivileged</td>
</tr>
<tr>
<td></td>
<td>- Secure/Non-secure</td>
<td>- Secure/Non-secure</td>
</tr>
<tr>
<td></td>
<td>- Protection contexts</td>
<td>- Protection contexts</td>
</tr>
</tbody>
</table>

¹ P-DMA can also be used to transfer data between memories.
² M-DMA can also be used to transfer data between peripheral and memory.
³ P-DMA and M-DMA channels inherit the access attributes of the bus transfer that programmed the channel.

Review TRM chapter 7 for additional details

Features highlighted in bold are the main difference between P-DMA and M-DMA.

Refer to the Protection Units training section for additional details.
P-DMA block diagram

- Trigger multiplexers
  - Connect each channel to one specific system trigger
  - Include hardware (HW) and software (SW) system triggers
  - Trigger output (tr_out)
    - Each trigger output can be used as its own active trigger using trigger multiplexers
    - They can be used to trigger different transfers as input triggers for other channels

1 Refer to the device datasheet for available hardware triggers.
2 tr_out can execute a chain transfer. Descriptor chaining can also execute a chain transfer.
Trigger output (tr_out)

Use case: CRC calculation of serial communication data
- Combined operation of Descriptors 0 and 1
- P-DMA:
  1. Is triggered by SCB when data reception is complete
  2. Loads Descriptor 0 from memory
  3. Transfers from FIFO to memory
  4. Outputs tr_out after completing Descriptor 0 transfer
  5. Loads Descriptor 1 by tr_out
  6. Activates CRC Transfer mode
  7. Runs CRC calculation
  8. Transfers CRC result to destination address
  9. Generates interrupt to CPU by end of descriptor

Advantage
- Multiple data transfers can be executed continuously by one trigger
P-DMA block diagram

› Pending triggers
  – Keeps track of activated triggers by storing channel triggers
  – Manages multiple pending channel triggers
› Available pre-emptive setting
  – If set, the higher-priority pending channel can pre-empt the current channel between single transfers

› Priority decoder
  – Determines the highest-priority channel of pending triggers
  – Has four priority levels
  – Performs round-robin arbitration within the same priority group

Review TRM section 7.1.6 for additional details
P-DMA block diagram

- Data transfer engine
  - Shared by each channel
  - Transfers data from source to destination according to descriptor
  - Reads channel descriptor from memory
  - Generates the trigger out to trigger multiplexers
P-DMA block diagram

› Descriptor
- Stored in memory
- Supports descriptor chaining
- Descriptor types
  - Single transfer
  - 1D transfer
  - 2D transfer
  - CRC transfer
- Descriptor structure
  - Descriptor control: Trigger type, transfer size, descriptor type
  - Source address/destination address
  - X(Inner) loop/Y(Outer) loop control
  - Next descriptor pointer
- The descriptor pointer position for each channel is stored in the register

Review TRM chapter 7.1.3 and Register TRM for additional details
System RAM is used store the descriptor

Copyright © Infineon Technologies AG 2022. All rights reserved.
Descriptor chaining

- Descriptors are chained by storing the next descriptor pointer in the current descriptor
- "0" (NULL pointer) is stored at the end of the descriptor list
- It is possible to have a circular list. It will stop when:
  - A transfer error occurs
  - The controller or channel is disabled by software

- Use case: Double buffer storing using two descriptors
  - Descriptors 0 and 1 are chained together
    1. P-DMA is triggered by SCB
    2. Descriptor 0: Data transfer from SCB to Buffer0
       ⇒ Descriptor chaining to Descriptor 1
    3. P-DMA is triggered by SCB
    4. Descriptor 1: Data transfer from SCB to Buffer1
       ⇒ Descriptor chaining to Descriptor 0

A descriptor chaining group is also referred to as a descriptor list
The address of the current descriptor is indicated in the CH_CURR_PTR register
The next descriptor address is updated after the execution of the current descriptor
M-DMA also has the same register
Descriptor type (1/4)

- **Single transfer**
  - Single data element (8-bit, 16-bit, or 32-bit) transfer
  - Uses four-words descriptor
  - Transfer example:
    \[
    \text{DSTADDR} = (\text{DATA_SIZE}) \times \text{SRCADDR}
    \]

- **Use case: Transfer A/D conversion results with A/D trigger**
  1. Move stored conversion data to data register
  2. Activate P-DMA with Channel Done trigger
  3. Transfer data from \(\text{SRCADDR}\) to \(\text{DSTADDR}\)
  4. Generate interrupt to CPU
Descriptor type (2/4)

› 1D transfer
   - One-dimensional “for loop” transfer
   - Uses five-words descriptor
   - Transfer example:
     ```c
     for (X_IDX =0; X_IDX <= COUNT; X_IDX++){
       DST_ADDR[DST_INCR] = (DATA_SIZE) SRC_ADDR[SRC_INCR]
     }
     ```
     *DST_INCR/SRC_INCR depend on X_INCR

› Use case: Transfer SCB reception data in FIFO
   1. Activate P-DMA by receiving a completion trigger
   2. Repeat data transfer from SRC_ADDR to DST_ADDR by COUNT
   3. Generate interrupt to CPU

Review TRM section 7.1.3 and Register TRM for additional details.
Descriptor type (3/4)

› 2D transfer
  - Two-dimensional “for loop” transfer
  - Uses six-words descriptor
  - Transfer example:
    ```c
    for (Y_IDX =0; Y_IDX <= Y_COUNT; Y_IDX++){
      for (X_IDX =0; X_IDX <= X_COUNT; X_IDX++){
        DST_ADDR[DST_INCR] = (DATA_SIZE) SRC_ADDR[SRC_INCR]
      }
    }
    ```
    *DST_INCR/SRC_INCR depend on X/Y_INCR

› Use case: Transfer SCB reception data in FIFO (1/2)
  - Handling multiple FIFO data as one data
    1. Activate P-DMA by receiving a completion trigger
    2. Repeat data transfer from SRC_ADDR to DST_ADDR by X_COUNT

Review TRM section 7.1.3 and Register TRM for additional details
Descriptor type (3/4)

- **2D transfer**
  - Two-dimensional “for loop” transfer
  - Uses six-words descriptor
  - Transfer example:
    ```c
    for (Y_IDX =0; Y_IDX <= Y_COUNT; Y_IDX++){
        for (X_IDX =0; X_IDX <= X_COUNT; X_IDX++){
            DST_ADDR[DST_INCR] = (DATA_SIZE) SRC_ADDR[SRC_INCR]
        }
    }
    ```
  *DST_INCR/SRC_INCR depend on X/Y_INCR

- **Use case: Transfer SCB reception data in FIFO (2/2)**
  - Handling multiple FIFO data as one data
    1. Activate P-DMA by receiving a completion trigger
    2. Repeat data transfer from SRC_ADDR to DST_ADDR by X_COUNT
    3. Activate P-DMA by receiving a completion trigger, when the second data is received
    4. Repeat data transfer from SRC_ADDR to DST_ADDR by X_COUNT

Review TRM section 7.1.3 and Register TRM for additional details
Descriptor type (4/4)

› CRC transfer
  - Execute the CRC calculation of the specified area
    - CRC-32, CRC-16, CRC-16-CCITT
    - Byte ordering/remainder bit reverse
    - Set CRC seed value
    - CRC result is stored to the destination address
  - This mode does not transfer data

› Use case: Initial CRC check for program data in Flash
  1. Activate P-DMA with software trigger
  2. Transfer data from SRC_ADDR
  3. CRC calculation (code area)
  4. Transfer CRC result to DST_ADDR
  5. Generate interrupt to CPU
  6. The CPU checks the CRC result with the expected value

› Advantage: Performs high-speed CRC calculation without a CPU

Review TRM section 7.1.3 and Register TRM for additional details.
Descriptor trigger types (1/5)

› Input trigger
  – Four types of input trigger action
    – Type 0: Executed by a single transfer
    – Type 1: Executed by a single 1D transfer
    – Type 2: Executed by the current descriptor
    – Type 3: Executed by a descriptor list

› Output trigger/interrupt
  – Four types of output trigger/interrupt generation
    – Type 0: Generated by a single transfer
    – Type 1: Generated by single 1D transfer
    – Type 2: Generated by the current descriptor
    – Type 3: Generated by descriptor list

› Input trigger, output trigger, and interrupt can be set individually

Review TRM sections 7.1.3 and 7.2.3 and Register TRM for additional details.
Descriptor trigger types (2/5)

Example of trigger action

- Descriptor input trigger Type 0: Executes a single transfer
- Descriptor output trigger/interrupt Type 0: Generated by a single transfer

![Diagram showing trigger types and their actions](image-url)
Descriptor trigger types (3/5)

- Example of trigger action
  - Descriptor input trigger Type 1: Executed by a single 1D transfer
  - Descriptor output trigger/interrupt Type 1: Generated by a single 1D transfer
Example of trigger action

- Descriptor input trigger Type 2: Executed by the current descriptor
- Descriptor output trigger/interrupt Type 2: Generated by the current descriptor
Example of trigger action

- Descriptor input trigger Type 3: Executed by a descriptor list
- Descriptor output trigger/interrupt Type 3: Generated by a descriptor list
M-DMA block diagram

› Trigger multiplexers
  - Connect each channel to one specific system trigger
  - Software (SW) system triggers¹
  - Trigger output (tr_out)
    - Each trigger output can be used as its own active trigger using trigger multiplexers
    - They can be used to trigger different transfers as input triggers for other channels

¹ This is different from P-DMA.
M-DMA block diagram

- Pending triggers
  - Keeps track of activated triggers by storing channel triggers
  - Manages multiple pending channel triggers

- Priority decoder
  - Determines the highest priority channel of transfer request from each transfer engine
  - Has four priority levels
  - Performs round-robin arbitration within the same priority group

1 This is different from P-DMA.
M-DMA block diagram

Data transfer engine
- Dedicated engine for each channel\(^1\)
- Transfers data from source to destination according to descriptor
- Reads channel descriptor from memory
- Generates the trigger out to trigger multiplexers

\(^1\) This is different from P-DMA.
M-DMA block diagram

Descriptor
- Stored in memory
- Descriptor chaining
- Descriptor types
  - Single transfer
  - 1D transfer
  - 2D transfer
  - Memory copy
  - Scatter
- Descriptor structure
  - Descriptor control: Trigger type, transfer size, descriptor type
  - Source address/destination address
  - X(Inner) loop/Y(Outer) loop size and Increment
  - Next descriptor pointer
- The descriptor pointer position for each channel is stored in the register

1 This is different from P-DMA.

Hint Bar
Review TRM section 7.2.6 and Register TRM for additional details
Descriptor size is different from P-DMA, but the setting items are the same
System RAM is used as a memory to store the descriptor
Descriptor type (1/3)

› Memory copy
  - One-dimensional “for loop” transfer
  - Uses five-word descriptor\(^1\)
  - Transfer example:
    ```
    for (X_IDX =0; X_IDX <= X_COUNT; X_IDX+){
      DST_ADDR[IDX] = SRC_ADDR[IDX]
    }
    ```

› Use case: Interrupt vector copy or program copy for RAM execution
  1. Activate M-DMA by software trigger
  2. Copy the data from Flash to SRAM
  3. Generate interrupt to CPU

\(^1\) SRC_SIZE, DST_SIZE, and DATA_SIZE fields are not used for memory copy. Memory copy transfers the number of bytes specified by X Loop Count in the optimum size. For example, when “Source Address” = 0x08000001, “X Loop Count” = 10, first transfer size is 8-bit, second transfer size is 16-bit, and third transfer size is 32-bit.

Review TRM section 7.2.3 and Register TRM for additional details.
Descriptor type (2/3)

› Scatter
- Writes a set of 32-bit data elements, which has addresses “scattered” around
- Writes the specified data to the specified address
- Uses four-word descriptor
- Transfer example:
  for (X_IDX =0; X_IDX <= X_COUNT; X_IDX +=2){
    address = SRC_ADDR[IDX]
    data = SRC_ADDR[IDX+1]
  }

  *address = data

¹ SRC_SIZE, DST_SIZE must be set to word (32-bit) for scatter.
Use case: Quick initial setting or setting change of peripheral registers without CPU

1. Write the address and data to be set in the peripheral to the memory
2. Activate M-DMA using software trigger
3. Write the data from the memory to peripheral registers
4. Generate an interrupt to CPU when the transfer is complete
AXI M-DMA
AXI M-DMA overview

TRAVEO™ T2G cluster 2D series has AXI M-DMA¹
- AXI master interface
- Data transfer between AXI slaves
- Uses a memory copy transfer as the primitive²
- Support 2D/3D memory copy
- Cannot access the peripheral bus infrastructure
- Buffer size between 64 bytes and 288 bytes per channel³
- Interrupt, input trigger, and output trigger per channel
- A channel is assigned a priority between 0 (high) and 3 (low)
- Round-robin arbitration is applied within same priority group

<table>
<thead>
<tr>
<th>Feature</th>
<th>AXI M-DMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer engine</td>
<td>Dedicated for each channel</td>
</tr>
<tr>
<td>Channel priority</td>
<td>Four levels</td>
</tr>
<tr>
<td>Transfer mode</td>
<td>- Memory copy</td>
</tr>
<tr>
<td></td>
<td>- 2D Memory copy</td>
</tr>
<tr>
<td></td>
<td>- 3D Memory copy</td>
</tr>
<tr>
<td>Descriptor</td>
<td>- Source and destination address</td>
</tr>
<tr>
<td></td>
<td>- Channel action</td>
</tr>
<tr>
<td></td>
<td>- Data transfer mode</td>
</tr>
<tr>
<td></td>
<td>- Activation trigger type (4 types)</td>
</tr>
<tr>
<td></td>
<td>- Output trigger type (4 types)</td>
</tr>
<tr>
<td></td>
<td>- Interrupt type (4 types)</td>
</tr>
<tr>
<td></td>
<td>- Descriptor chaining</td>
</tr>
<tr>
<td>Access-control attributes³</td>
<td>- Privileged/Unprivileged</td>
</tr>
<tr>
<td></td>
<td>- Secure/Non-secure</td>
</tr>
<tr>
<td></td>
<td>- Protection contexts</td>
</tr>
</tbody>
</table>

¹³ See the device specific datasheet to see if the feature is supported.
² Copying of M_COUNT+1 bytes from a source address to a destination address using AXI bursts
³ AXI M-DMA channels inherit the access attributes of the bus transfer that programmed the channel.
A data transfer is initiated by an input trigger via trigger multiplexer
- System trigger
- Software (SW) trigger
- Trigger output (tr_out)
  - Each trigger output can be used as its own active trigger using trigger multiplexers
  - They can be used to trigger different transfers as input triggers for other channels
Channel logic
- Keeps track of the channel’s input trigger
- Maintains the channel state and a data transfer engine
- Reads channel descriptor from memory

AXI arbiter
- Performs arbitration between the channels
- Applies round-robin arbitration within the same priority group

MMIO registers
- AXI M-DMA control / status registers

Bus slave I/F
- AHB-Lite slave I/F for register access
AXI M-DMA block diagram (3/3)

Descriptor

- Stored in memory
- Descriptor chaining
- Descriptor types
  - Memory copy
  - 2D Memory copy
  - 3D Memory copy
- Descriptor structure
  - Descriptor control: Trigger type, descriptor type
  - Source address/destination address
  - Memory copy size
  - X loop/Y loop control
  - Next descriptor pointer
- When fetching the descriptor, always reads 10 words (5 x 64 bits)\(^1\)
  - The descriptor pointer position for each channel is stored in the register

\(^1\) Needs to be considered when setting up descriptors, to avoid AXI bus error responses when reading the descriptor. See the TRM section 7.3.3 for more details.
Rules for generating AXI transactions

- Only incrementing burst is used; wrapping and fixed burst are not supported.
- The data size of AXI transactions is always 64 bits.
- AXI transactions never cross a 32-byte boundary.
- Maximum burst length is four beats.
- Different memory copy operations are never combined to one AXI transaction.
- Within one iteration of a memory copy operation, the transfers within the same aligned 32-byte region are always performed as one AXI transaction.
- First AXI transaction is an unaligned transaction unless the start address is a multiple of 8.
- Last AXI transaction of each memory copy operation starts at an address that is a multiple of 32, and has the minimum burst length for end of the copy address range.
- AXI transactions between the first and the last are full 32-byte bursts.
- For unaligned write transactions at the start and incomplete write transactions at the end of a memory copy operation, only the correct bytes are written.
- For unaligned read transactions at the start and incomplete read transactions at the end of a memory copy operation, reading is always performed in multiples of 8 bytes.

Review TRM 7.3.6 for additional details.
Descriptor type (1/3)

- **Memory copy**
  - One-dimensional “for loop” transfer
  - Uses five-words descriptor
  - Transfer example:
    ```
    for (M_IDX = 0; M_IDX <= M_COUNT; M_IDX++){
      DST_ADDR[M_IDX] = SRC_ADDR[M_IDX]
    }
    ```

- **Use case:**
  - Transfer conditions
    - Source address: A+3
    - Destination address: B+13
    - Transfer count: 57 bytes
  - Read from source address using 2 burst transfers (4 beats + 4 beats)
  - Write to destination address using 3 burst transfers (3 beats + 4 beats + 1 beats)
Descriptor type (2/3)

› 2D memory copy
  - Two-dimensional “for loop” transfer
  - Uses seven-words descriptor
  - Transfer example:
    ```c
    for (X_IDX = 0; X_IDX <= X_COUNT; X_IDX++) {
      for (M_IDX = 0; M_IDX <= M_COUNT; M_IDX++) {
        DST_ADDR[M_IDX + X_IDX * DST_X_INCR] =
          SRC_ADDR[M_IDX + X_IDX * SRC_X_INCR];
      }
    }
    ```

› Use case:
  - 2D memory copy can be used to transfer bitmaps

Review TRM section 7.3.1 and 7.3.3 for additional details
Descriptor type (3/3)

- 3D memory copy
  - Three-dimensional “for loop” transfer
  - Uses nine-word descriptor
  - Transfer example:
    ```c
    for (Y_IDX = 0; Y_IDX <= Y_COUNT; Y_IDX++) {
      for (X_IDX = 0; X_IDX <= X_COUNT; X_IDX++) {
        for (M_IDX = 0; M_IDX <= M_COUNT; M_IDX++) {
          DST_ADDR[M_IDX + X_IDX * DST_X_INCR + Y_IDX * DST_Y_INCR] =
          SRC_ADDR[M_IDX + X_IDX * SRC_X_INCR + Y_IDX * SRC_X_INCR];
        }
      }
    }
    ```

Review TRM section 7.3.3 and Register TRM for additional details
Use case (1)

Load the scene data to the GFX subsystem from external memory to VRAM\(^1\)
- The BLIT engine processes scene data in parallel during AXI M-DMA transfers
- AXI M-DMA transfers data between external memory and GFX memory without CPU involvement

\(^1\) CYT4EN series has LPDDR4 memory interface instead of VRAM. See the specific datasheet for details.
Use case (2)

Transfer PCM stream from external memory to Sound subsystem

1. AXI M-DMA transfers data from external memory to SRAM
2. AXI M-DMA can initiate P-DMA\(^1\) via trigger multiplexer
3. P-DMA transfers stream data from SRAM to Sound subsystem, such as Audio-DAC and PCM-PWM

- AXI M-DMA cannot access to peripheral registers directly

\(^1\) See the specific datasheet for connection between DMAs by trigger multiplexer.
Descriptor trigger types

› Input trigger
  - Four types of input trigger action
    - Type 0: Executed a memory copy
    - Type 1: Executed a 2D memory copy
      - When the descriptor type is memory copy, this type behaves similar to type 0.
    - Type 2: Executed by the current descriptor
    - Type 3: Executed by a descriptor list

› Output trigger/interrupt
  - Four types of output trigger/interrupt generation
    - Type 0: Generated by a memory copy transfer
    - Type 1: Generated by a 2D memory copy transfer
      - When the descriptor type is memory copy, this type behaves similar to type 0.
    - Type 2: Generated by the current descriptor transfer
    - Type 3: Generated by descriptor list transfer

› Input trigger, output trigger, and interrupt can be set individually

Hint Bar
Review TRM sections 7.3.9 and Register TRM for additional details
Descriptor trigger types

Example of each trigger action
- Descriptor 0 is 3D memory copy and has next descriptor pointer address (Descriptor 1)
- Descriptor 1 does not have next descriptor pointer address (= “0”)

Hint Bar
Review TRM sections 7.3.9 and Register TRM for additional details
Revision history

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN</th>
<th>Submission Date</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>6136162</td>
<td>04/17/2018</td>
<td>Initial release</td>
</tr>
<tr>
<td>*A</td>
<td>6351891</td>
<td>16/10/2018</td>
<td>Added slide 2. Updated slides 3, 4, and 5. Added slide 29: Descriptor Type (3/3). Updated figures on slides 7, 9-11, and 23-26</td>
</tr>
<tr>
<td>*B</td>
<td>6633414</td>
<td>07/22/2019</td>
<td>Updated slide 2-4, 9. Added slide 5.</td>
</tr>
<tr>
<td>*C</td>
<td>6825576</td>
<td>03/06/2020</td>
<td>Updated slide 12, 15,16</td>
</tr>
<tr>
<td>*D</td>
<td>6952165</td>
<td>08/20/2020</td>
<td>Update slide 8, 27, 28</td>
</tr>
<tr>
<td>*F</td>
<td>7807792</td>
<td>09/06/2022</td>
<td>Updated slide 2,5 Added slide 31 to 43</td>
</tr>
</tbody>
</table>
Part of your life. Part of tomorrow.