Customer Training Workshop
Traveo™ II Direct Memory Access (DMA)

Q4 2020
## 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™ II Automotive Body Controller Entry</td>
<td>CYT2B6</td>
<td>Up to 576KB</td>
</tr>
<tr>
<td>Traveo II Automotive Body Controller Entry</td>
<td>CYT2B7</td>
<td>Up to 1088KB</td>
</tr>
<tr>
<td>Traveo II Automotive Body Controller Entry</td>
<td>CYT2B9</td>
<td>Up to 2112KB</td>
</tr>
<tr>
<td>Traveo II Automotive Body Controller Entry</td>
<td>CYT2BL</td>
<td>Up to 4160KB</td>
</tr>
<tr>
<td>Traveo II Automotive Body Controller High</td>
<td>CYT3BB/CYT4BB</td>
<td>Up to 4160KB</td>
</tr>
<tr>
<td>Traveo II Automotive Body Controller High</td>
<td>CYT4BF</td>
<td>Up to 8384KB</td>
</tr>
<tr>
<td>Traveo II Automotive Cluster</td>
<td>CYT3DL</td>
<td>Up to 4160KB</td>
</tr>
<tr>
<td>Traveo II Automotive Cluster</td>
<td>CYT4DN</td>
<td>Up to 6336KB</td>
</tr>
</tbody>
</table>
Introduction to Traveo II Body Controller Entry

DMA is part of the CPU subsystem
Introduction to Traveo II Body Controller High

DMA is part of the CPU subsystem
Introduction to Traveo II Cluster

» DMA is part of the CPU subsystem

CPU Subsystem
- Cortex M7 320 MHz
- SRAM0 256 KB
- SRAM1 256 KB
- SRAM2 128 KB
- MIFIO8
- CRYPTO AES, SHA, CRC, TINLIN/SA, ECC
- SMIF
- Cortex M0+ 100 MHz
- ROM 64 KB

System Interconnect (Multi Layer AXI/AHB, IPC, MPU/SMPU)
- System Resources
  - Power
    - POR, BOD, LVD
    - PWRSYS HT
    - REF
    - POR
    - XRES
    - LVD
    - BOD
    - Active/Sleep
    - Low Power Active/Sleep
    - Power Modes
    - IM2O
    - SRAM3 128 KB
    - SRAM Controller
    - SRAM2 64 KB
    - SRAM1 64 KB
    - SRAM0 32 KB
    - SRAM
    - Controller
    - SRAM 256 KB
    - SMIF
    - Cortex M0+ 100 MHz
    - ROM 64 KB

Peripheral Interconnect (MMIO, PPU)
- eCT FLASH 6336 KB Code flash + 128 KB Work flash
- IOSS GPIO
- PCLK
- 52x GPIO_STD, 8x GPIO_ENH, 26x GPIO_SMC, 70x HSIO_STD, 22x HSIO_ENH, 4x HSIO_ENG_DIFF

GFX Subsystem
- GFX Interconnect (AXI)
- 1x RGB/MIPI Input
- 2x RGB/LVDS Output
- 2.5D Engine
- GFX Subsystem
  - GFX Interconnect (AXI)
  - 1x RGB/MIPI Input
  - 2x RGB/LVDS Output
  - 2x Smart IO
  - Vector Gfx
  - 1x SMIF
  - Vector Gfx

IO Subsystem
- SARMUX 48 ch
- SAR ADC (12-bit)
- Audio DAC
- 2x PCM-PWM
- 5x SG
- 2x Mixer
- Audio DAC
- 4x TDM
- LPECO
- SRAM2 128 KB
- SRAM Controller

Review TRM chapter 7 for additional details

Copyright © Infineon Technologies AG 2020. All rights reserved.
DMA Overview

› Traveo II 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>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>- Interrupt type (4 types)</td>
<td>- Interrupt type (4 types)</td>
</tr>
<tr>
<td></td>
<td>- Descriptor chaining</td>
<td>- Descriptor chaining</td>
</tr>
<tr>
<td>Access-control</td>
<td>- Privileged/Unprivileged</td>
<td>- Privileged/Unprivileged</td>
</tr>
<tr>
<td>attributes³</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 blue are the main difference between P-DMA and M-DMA.

Refer to the Protection Units training section for additional details
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.
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

Copyright © Infineon Technologies AG 2020. All rights reserved.
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

*Hint Bar*
Review TRM section 7.1.6 for additional details
P-DMA Block Diagram

› Descrptor
  - 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 2020. 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-word descriptor
   - Transfer example:
     \[ \text{DST_ADDR} = \text{(DATA\_SIZE)} \times \text{SRC\_ADDR} \]

› 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 SRC\_ADDR to DST\_ADDR
   4. Generate interrupt to CPU

Review TRM chapter 7.1.3 and Register TRM for additional details
Descriptor Type (2/4)

1D Transfer
- One-dimensional “for loop” transfer
- Uses five-word descriptor
- Transfer example:
  ```
  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

Hint Bar
Review TRM section 7.1.3 and Register TRM for additional details

Copyright © Infineon Technologies AG 2020. All rights reserved.
Descriptor Type (3/4)

- 2D Transfer
  - Two-dimensional “for loop” transfer
  - Uses six-word descriptor
  - Transfer example:
    ```
    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-word descriptor
  - Transfer example:
    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

Copyright © Infineon Technologies AG 2020. All rights reserved.
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

Hint Bar
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
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

![Diagram showing trigger actions and transfer status](image-url)
Descriptor Trigger Types (4/5)

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

Transfer Status

- 1\D ( =Descriptor 1)
- 2\D ( =Descriptor 0)
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 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 This is different from P-DMA.

Hint Bar

Review TRM section 7.2.6 and chapter 29 for additional details
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\(^1\)
  - Has four priority levels
  - Performs round-robin arbitration within the same priority group

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

Review TRM section 7.2.6 for additional details
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.

Review TRM section 7.2.6 for additional details
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 control
  - Next descriptor pointer
  - The descriptor pointer position for each channel is stored in the register

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.

Review TRM section 7.2.6 and Register TRM for additional details.

This is different from P-DMA.
Descriptor Type (1/3)

› Memory copy
  - One-dimensional “for loop” transfer
  - Uses five-word descriptor\(^1\)
  - Transfer example:
    ```c
    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.
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 descriptors
- Transfer example:
  ```c
  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
Part of your life. Part of tomorrow.
## 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>7/22/2019</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>
</tbody>
</table>