Customer Training Workshop
Traveo™ II Serial Memory Interface (SMIF)

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 High</td>
<td>CYT3BB/4BB</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 High

- SMIF is located in the peripheral blocks

Review TRM chapter 32 for additional details
Introduction to Traveo II Cluster

- SMIF is located in the peripheral blocks

Hint Bar

Review TRM chapter 32 for additional details
Serial Memory Interface (SMIF) Overview (1/2)

› SMIF provides an interface to memories with SPI and HyperBus IFs

› Features
  – SPI or HyperBus master functionality only
  – SPI protocol
    – Mode 0 only, with configurable Master-In Slave-Out (MISO) sampling timing
    – Single/Dual/Quad/Octal SPI transfer mode
    – Dual-quad SPI mode (two devices sharing one address range)
    – Single Data Rate (SDR) and Double Data Rate (DDR) transfer
  – Two operation modes
    – Memory Mapped I/O (MMIO) operation mode
    – eXecute-In-Place (XIP) operation mode
      – Read and write access
      – Automatic transition access from dedicated internal memory address to SPI protocol
      – On-the-fly encryption and decryption

Hint Bar

Review TRM section 32.1 for additional details

SPI Mode 0:
Data is driven on a falling edge of SPIHB_CLK
Data is captured on a rising edge of SPIHB_CLK
Serial Memory Interface (SMIF) Overview (2/2)

Features

- Memory Device
  - Device capacity in the range of 64KB to 4GB (memory size: $2^N$)
    - (CYT4BF: 128MB area in XIP mode)
    - (CYT4DN: 512MB x2 area in XIP mode)
  - Support for configurable external device capacities
  - Up to two external memory devices sharing one address range (e.g. Dual-quad mode)
  - Up to four chip selects available
    - (CYT4BF: 2 pcs)
    - (CYT4DN: 2 pcs per channel)

- Memory Interface Logic
  - Support stalling of transfers to address the back pressure on FIFOs
  - Support read-write-data-strobe (RWDS)
  - Support data signal connections between flexible external SPI memory devices
  - Support delay line and data learning pattern (DLP)-based data capture (CYT4DN)

Hint Bar

Review TRM section 32.1 for additional details
Example of Using SMIF

- External memory for code and data space
- Directly mapped to internal memory map (XIP operation mode)
- Encrypted/decrypted code or data in an external space
SMIF Block Diagram

- SMIF block components
  - Bus interface
    - MMIO AHB-Lite interface
    - XIP AXI interface
    - XIP AHB-Lite interface
Bus Interface

› All bus interfaces provide access to external memories
› Any of the three bus interfaces can access external memories at any time
  – MMIO AHB-Lite interface
    – Access to the MMIO registers
    – Supports MMIO operation mode
  – XIP AXI interface
    – AXI interface for CPUSS fast domain\(^1\)
    – Supports XIP operation mode
    – Fixed 512B buffer\(^2\)
  – XIP AHB-Lite interface
    – AHB-Lite interface for CPUSS slow domain\(^3\)
    – Supports XIP operation mode
    – 4KB read-only cache\(^3\)
      – The “hit” read transfers are processed by the cache
      – Four-way set associative with an LRU replacement scheme

\(^1\) The CPUSS fast domain component is CM7 CPUs.
\(^2\) Note that the cache and the AXI interface buffer are not retained in DeepSleep power mode.
\(^3\) The CPUSS slow domain components are CM0+ Crypto and P-DMA.
SMIF Block Diagram

- SMIF block components
  - FIFOs
  - Two operation modes
    - MMIO operation mode
    - XIP operation mode
    - Mode is switched by XIP_MODE
      - Both modes are mutually exclusive
  - Continuous transfer merging

1 Refer to the Register TRM (SMIF_CTL) for additional details.
SMIF has TX command FIFO, TX data FIFO, and RX data FIFO

- Provide an asynchronous clock domain transfer between clk_mem and clk_if_tx/clk_if_rx
- Software\(^1\) controls the FIFOs in MMIO operation mode
- MMIO registers\(^2\) provide access to FIFOs

<table>
<thead>
<tr>
<th>FIFO</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX command FIFO</td>
<td>Transmit memory commands to the memory interface logic for memory transfer. Support five types of commands</td>
</tr>
<tr>
<td>TX data FIFO</td>
<td>Transmit write data to the memory interface logic</td>
</tr>
<tr>
<td>RX data FIFO</td>
<td>Receive read data from the memory interface logic</td>
</tr>
</tbody>
</table>

\(^1\) In XIP operation mode, FIFOs are controlled by hardware.
\(^2\) Refer to the Register TRM for additional details.
FIFOs (1/2)

TX command FIFO supports five command types

- TX command FIFO is controlled by the SMIF_TX_CMD_FIFO_WR.DATA27[26:0] register
- DATA27[26:24] specifies the command and DATA27[23:0] sets the command specification depending on command type

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TX: 0</td>
<td>Width of the data transfer (single, dual, quad, or octal data transfer)</td>
</tr>
<tr>
<td></td>
<td>Data transfer mode (SDR or DDR)</td>
</tr>
<tr>
<td></td>
<td>External device select (multiple devices can be selected simultaneously)</td>
</tr>
<tr>
<td></td>
<td>The command is for the last phase of memory transfer</td>
</tr>
<tr>
<td></td>
<td>Transfer of 1-2 bytes</td>
</tr>
<tr>
<td></td>
<td>A memory transfer must start with a TX command.</td>
</tr>
<tr>
<td>TX_COUNT: 1</td>
<td>This command is used to transmit data from TX data FIFO to external memories</td>
</tr>
<tr>
<td></td>
<td>Value of memory data to be transmitted</td>
</tr>
<tr>
<td></td>
<td>Width of the data transfer</td>
</tr>
<tr>
<td></td>
<td>Data transfer mode (SDR or DDR)</td>
</tr>
<tr>
<td></td>
<td>Specifies if this command is for the last phase of the memory transfer</td>
</tr>
</tbody>
</table>

Refer to the Register TRM for additional details.
TX command FIFO supports five command types

- TX command FIFO is controlled by the SMIF_TX_CMD_FIFO_WR.DATA27[26:0]\(^1\) register
- DATA27[26:24] specifies the command and DATA[23:0] sets the command specification depending on command type

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RX_COUNT: 2</td>
<td>This command is used by RX data FIFO to receive data from external memories</td>
</tr>
<tr>
<td></td>
<td>- Value of the memory data received</td>
</tr>
<tr>
<td></td>
<td>- Width of the data transfer</td>
</tr>
<tr>
<td></td>
<td>- Data transfer mode (SDR or DDR)</td>
</tr>
<tr>
<td></td>
<td>- Specifies if this command is for the last phase of the memory transfer</td>
</tr>
<tr>
<td>DUMMY_COUNT: 3</td>
<td>- Number of dummy cycles (used to implement turnaround time)</td>
</tr>
<tr>
<td></td>
<td>- If the variable latency mode for HyperRAM is enabled causing double the number of dummy cycles, this command never constitutes the last phase of the memory transfer</td>
</tr>
<tr>
<td>DESELECT: 4</td>
<td>Finish a transfer and deselect the memory device</td>
</tr>
<tr>
<td></td>
<td>This command always constitutes the last phase of the memory transfer</td>
</tr>
</tbody>
</table>

\(^1\) Refer to the Register TRM for additional details.
Constructing a Read/Write Command

Read command:

A) Set TX command FIFO in the order of Command, Address, and Mode using the “TX” command
B) Set dummy cycle to the TX command FIFO using the “DUMMY_COUNT” command
C) Set number of reception data to the TX command FIFO using the “RX_COUNT” command
D) Receive the number of data set by RX_COUNT in the RX data FIFO

Write command:

E) Set TX command FIFO in the order of Command, Address, and Mode using the “TX” command
F) Set number of transmission data to the TX command FIFO using the “TX_COUNT” command
G) Transmit the number of data set by TX_COUNT from the TX data FIFO

Mode:

It can eliminate the overhead of SIO instructions when repeating the same type of read command.

Review TRM section 32.1 for additional details.
Operation Mode

- **MMIO Operation Mode**
  - Active by writing “0” to XIP_MODE1
  - Supports access through software using FIFOs
  - Provides the flexibility to implement any SPI device transfer

**Data Read example**

<table>
<thead>
<tr>
<th>MMIO Interface (Command type)</th>
<th>TX/RX FIFOs</th>
<th>Memory Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write to Command byte (TX)</td>
<td>TX command FIFO</td>
<td>TX (command byte)</td>
</tr>
<tr>
<td>Write to Address 0 (TX)</td>
<td>TX command FIFO</td>
<td>TX (address byte 0)</td>
</tr>
<tr>
<td>Write to Address 1 (TX)</td>
<td>TX command FIFO</td>
<td>TX (address byte 1)</td>
</tr>
<tr>
<td>Write to Address 2 (TX)</td>
<td>TX command FIFO</td>
<td>TX (address byte 2)</td>
</tr>
<tr>
<td>Write to Mode Byte (TX)</td>
<td>TX command FIFO</td>
<td>TX (mode byte)</td>
</tr>
<tr>
<td>Write to DUMMY (DUMMY_COUNT)</td>
<td>TX command FIFO</td>
<td>DUMMY_COUNT</td>
</tr>
<tr>
<td>Write to RX_COUNT (RX_COUNT)</td>
<td>TX command FIFO</td>
<td>RX_COUNT</td>
</tr>
<tr>
<td>Read from RX data FIFO</td>
<td>RX data FIFO</td>
<td></td>
</tr>
</tbody>
</table>

**Data Write example**

<table>
<thead>
<tr>
<th>MMIO Interface (Command type)</th>
<th>TX/RX FIFOs</th>
<th>Memory Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write to TX data FIFO</td>
<td>TX data FIFO</td>
<td>TX (command byte)</td>
</tr>
<tr>
<td>Write to TX data FIFO</td>
<td>TX data FIFO</td>
<td>TX (address byte 0)</td>
</tr>
<tr>
<td>Write to TX data FIFO</td>
<td>TX data FIFO</td>
<td>TX (address byte 1)</td>
</tr>
<tr>
<td>Write to TX data FIFO</td>
<td>TX data FIFO</td>
<td>TX (address byte 2)</td>
</tr>
<tr>
<td>Write to TX data FIFO</td>
<td>TX data FIFO</td>
<td>TX (mode byte)</td>
</tr>
<tr>
<td>Write to TX data FIFO</td>
<td>TX data FIFO</td>
<td>TX data FIFO</td>
</tr>
<tr>
<td>Write to TX data FIFO</td>
<td>TX data FIFO</td>
<td></td>
</tr>
</tbody>
</table>

\[^1\) XIP_MODE default value is “0” in the MMIO mode.
Operation Mode

› XIP Operation Mode
  – Allows mapping each external memory into an internal memory address space
  – Active by writing “1” to XIP_MODE
  – Hardware generates memory transfers automatically without software intervention
  – Supports read and write access
  – FIFOs are controlled by hardware
  – Read/write commands are specified by MMIO registers\(^1\) for each device

Data Read example

XIP Interface
(AXI/AHB-Lite)

Memory Interface

Read/write commands specification includes:

- Presence and value of SPI or HyperBus command byte
- Number of address bytes
- Presence and value of the mode byte
- Number of dummy cycles
- Data transfer widths and data transfer mode (SDR or DDR)

1 Refer to the Register TRM (SMIF_DEVICE_RD/WR_CMD_CTL, _RD/WR_ADDR_CTL, _RD/WR_MODE_CTL, RD/WR_DUMMY_CTL, RD/WR_DATA_CTL) for additional details.
Continuous Transfer Merging

The transfer can be merged to a single transfer at the memory interface (MMIO and XIP modes)

- Background: Longer transactions split on the AXI bus due to arbitration and crypto support
- Avoid the overhead of multiple commands, addresses, modes, and dummy (latency) cycles

Advantage
- Improves performance of multiple continuous transfers

1 Refer to the Register TRM (SMIF_CTL.MERGE_EN) for additional details.
Data Learning Pattern\(^1\) (DLP)

› SMIF supports delay line and DLP-based data capture
  - Provides an optimal capture point within the data window
  - Captures input data with different clocks generated by taps of a delay line
  - Finds the delay tap for each data line by comparing the captured DLP with the expected one
  - Memory device provides a known DLP on every data I/O pin before outputting read data
  - No delay line tap resulting in a matching DLP, a data learning failed interrupt (in XIP/MMIO mode), and a bus error response (in XIP mode) being generated

› Advantage
  - Adjust signal delay easily due to wiring length, process, voltage, and temperature for each data line
  - Maximize read data throughput

\(^1\) DLP is available in CYT4DN.
Bus Error Generation

External memory accesses cause bus error under the following conditions

<table>
<thead>
<tr>
<th>Interface</th>
<th>Precondition</th>
<th>Bus Error Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMIO (AHB-Lite)</td>
<td>- SMIF_CTL.BLOCK = 0&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Software attempts to write an entry of a full TX command FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software attempts to write more bytes than the available entries in the TX data FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software attempts to read more bytes than the available entries in the RX data FIFO</td>
</tr>
<tr>
<td></td>
<td>Refer to the Register TRM (SMIF_CTL) for additional details.</td>
<td></td>
</tr>
<tr>
<td>XIP (AXI/AHB-Lite)</td>
<td>- SMIF_CTL.ENABLED = 0&lt;sup&gt;1&lt;/sup&gt; - Write or read access</td>
<td>SMIF is disabled</td>
</tr>
<tr>
<td></td>
<td>- SMIF_CTL.XIP_MODE = 0&lt;sup&gt;1&lt;/sup&gt; - Write or read access</td>
<td>SMIF is not in XIP mode</td>
</tr>
<tr>
<td></td>
<td>- XIP mode (SMIF_CTL.XIP_MODE = 1) - Write or read access</td>
<td>Transfer request is not in a memory region</td>
</tr>
<tr>
<td></td>
<td>- XIP mode (SMIF_CTL.XIP_MODE = 1) - SMIF_DEVICE_CTL.WR_EN = 0&lt;sup&gt;2&lt;/sup&gt;</td>
<td>Write transfer to a region that does not support writes</td>
</tr>
<tr>
<td></td>
<td>- XIP mode (SMIF_CTL.XIP_MODE = 1) - Write access</td>
<td>Transfer address is not a multiple of 2</td>
</tr>
<tr>
<td></td>
<td>- XIP mode (SMIF_CTL.XIP_MODE = 1) - Dual-quad SPI</td>
<td>Transfer size is not a multiple of 2</td>
</tr>
<tr>
<td></td>
<td>- XIP mode (SMIF_CTL.XIP_MODE = 1) - Octal SPI DDR mode or HyperBus mode</td>
<td>Transfer address is not a multiple of 2</td>
</tr>
<tr>
<td></td>
<td>- Memory write byte masking is not supported</td>
<td>Transfer size is not a multiple of 2</td>
</tr>
<tr>
<td></td>
<td>- SMIF_CTL.INT_CLOCK_DL_ENABLED = 1</td>
<td>No delay line tap resulting in a matching data learning pattern</td>
</tr>
</tbody>
</table>

<sup>1</sup> Refer to the Register TRM (SMIF_CTL) for additional details.
<sup>2</sup> Refer to the Register TRM (SMIF_DEVICE_CTL) for additional details.
SMIF Block Diagram

SMIF External Device Connection
- Single, dual, quad, and octal data transfer
- Maximum memory interface clock:
  - Depends on memory and I/O type

- Can connect up to four devices
- Independent SPIHB_CLK_INV outputs
  - Less latency, and no overlapping of SPIHB_CLK

<table>
<thead>
<tr>
<th>Products</th>
<th>I/O Type</th>
<th>SDR</th>
<th>DDR</th>
<th>HyperBus</th>
</tr>
</thead>
<tbody>
<tr>
<td>CYT4BF</td>
<td>GPIO_STD&lt;sup&gt;1&lt;/sup&gt;</td>
<td>32 MHz</td>
<td>32 MHz</td>
<td>32 MHz</td>
</tr>
<tr>
<td></td>
<td>HSIO_STD&lt;sup&gt;2&lt;/sup&gt;</td>
<td>100 MHz</td>
<td>100 MHz</td>
<td>100 MHz</td>
</tr>
<tr>
<td>CYT4DN</td>
<td>HSIO_ENH&lt;sup&gt;3&lt;/sup&gt;</td>
<td>166 MHz</td>
<td>100 MHz</td>
<td>200 MHz</td>
</tr>
</tbody>
</table>

<sup>1</sup> Supports standard automotive GPIO. Refer to the device datasheet for additional details.
<sup>2</sup> Supports high-speed I/O standard for high-speed peripherals. Refer to the device datasheet for additional details.
<sup>3</sup> Supports high-speed I/O enhanced for high-speed peripherals. Refer to the device datasheet for additional details.
**External Device Connection**

**SPI Connection**
- Support for single, dual, quad, and octal SPI protocol
- Data signal type configuration in the MMIO register
- Can be set independently for each memory device (chip select)

**Data Signal Connections**

<table>
<thead>
<tr>
<th>DATA_SEL[1:0]</th>
<th>Single SPI (Full Duplex)</th>
<th>Dual SPI (Half Duplex)</th>
<th>Quad SPI (Half Duplex)</th>
<th>Octal SPI (Half Duplex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SPIHB_DATA [0] = SI</td>
<td>SPIHB_DATA [0] = IO0</td>
<td>SPIHB_DATA [0] = IO0</td>
<td>SPIHB_DATA [0] = IO0</td>
</tr>
</tbody>
</table>

1 Refer to the Register TRM (SMIF_DEVICE_CTL) for additional details.
External Device Connection

› SPI Connection
- The following diagrams show examples of a single SPI connection

› Features
- Data signal connections are configured by DATA_SEL
- Chip select signal setting is configured depending on the operation mode

<table>
<thead>
<tr>
<th>Operation Mode</th>
<th>Setting Register</th>
<th>Chip Select</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMIO</td>
<td>TX command FIFO</td>
<td>DATA[21]: SPIHB0_SEL[1] DATA[20]: SPIHB0_SEL[0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATA[23]: SPIHB1_SEL[1] DATA[22]: SPIHB1_SEL[0]</td>
<td>Supports CYT4DN</td>
<td></td>
</tr>
<tr>
<td>XIP</td>
<td>SMIF0DEVICE0²</td>
<td>SPIHB0_SEL[0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SMIF0DEVICE1²</td>
<td>SPIHB0_SEL[1]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SMIF1DEVICE0²</td>
<td>SPIHB1_SEL[0]</td>
<td>Supports CYT4DN</td>
</tr>
<tr>
<td></td>
<td>SMIF1DEVICE1²</td>
<td>SPIHB1_SEL[1]</td>
<td></td>
</tr>
</tbody>
</table>

1 Refer to the Register TRM (SMIF_TX_CMD_FIFO_WR) for additional details. Multi chip selects can be set at the same time in dual-quad mode.

2 Refer to Register TRM (SMIFDEVICE) for additional details.
External Device Connection

- SPI Connection
  - The following diagrams provide examples of two single SPI device connections:

**Single SPI with Separate Data Signals**

- **Device 0:**
  - SPIHB
  - memory
  - Traveo II
  - SMIF
  - SCK
  - CS#
  - SI
  - SO
  - SPIHB_CLK
  - SPIHB_SEL[0]
  - SPIHB_DATA[0]
  - SPIHB_DATA[1]

- **Device 1:**
  - SPIHB
  - memory
  - SCK
  - CS#
  - SI
  - SO
  - SPIHB_SEL[1]
  - CTL.DATA_SEL[1:0] = 0
  - CTL.DATA_SEL[1:0] = 3

**Single SPI with Shared Data Signals**

- **Device 0:**
  - SPIHB
  - memory
  - Traveo II
  - SMIF
  - SCK
  - CS#
  - SI
  - SO
  - SPIHB_CLK
  - SPIHB_SEL[0]
  - SPIHB_DATA[0]
  - SPIHB_DATA[1]

- **Device 1:**
  - SPIHB
  - memory
  - SCK
  - CS#
  - SI
  - SO
  - SPIHB_SEL[1]
  - CTL.DATA_SEL[1:0] = 0
  - CTL.DATA_SEL[1:0] = 3
  - SPIHB_DATA[6]
  - SPIHB_DATA[7]
External Device Connection

- **SPI Connection**
  - The following diagrams show examples of quad and dual-quad SPI connections

- **Features**
  - Quad SPI can also use SPIHB_DATA [4:7] by the DATA_SEL configuration

---

**Quad SPI Connection**

**Dual-Quad SPI Connection (Pseudo Octal)**
External Device Connection

› SPI Connection
  - The following diagram shows an example of an octal SPI connection

› Features
  - The only valid setting for DATA_SEL is “0”
External Device Connection

› HyperBus Connection
  - The following diagram shows an example of the HyperBus connection

› Features
  - Supports read-write-data-strobe (RWDS)
  - Only valid setting for DATA_SEL is “0”
SMIF Block Diagram

- SMIF block component
  - Cryptography
Cryptography

› Read and write data transfers are available for encryption and decryption
  - Encryption and decryption are based on the AES-128 forward block cipher
    - The key is programmed into the SMIF_CRYPTO_KEY3-0 registers\(^1\)
  - In MMIO mode
    - Supports offline encryption and decryption
    - Provides plaintext in SMIF_CRYPTO_INPUT3-0\(^1\)
    - Provides ciphertext in SMIF_CRYPTO_OUTPUT3-0\(^1\)
  - In XIP mode
    - Supports on-the-fly encryption for write data
    - Supports on-the-fly decryption for read data
    - Uses XIP address for plaintext and extends to SMIF_CRYPTO_INPUT3-0

› Advantage
  - Storing data encrypted in external memory devices prevents leakage of sensitive data

\(^1\) Refer to the Register TRM for additional details.
SMIF Block Diagram

- SMIF block components
  - Trigger and interrupt
    - tr_tx_req
    - tr_rx_req
  - Interrupt
## Trigger and Interrupt

SMIF has two triggers

<table>
<thead>
<tr>
<th>Trigger</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr_tx_req</td>
<td>Active when the number of used TX data FIFO entries is smaller than or equal to the specified number. Activated in MMIO operation mode</td>
</tr>
<tr>
<td>tr_rx_req</td>
<td>Active when the number of used RX data FIFO entries is greater than or equal to the specified number. Activated in MMIO operation mode</td>
</tr>
</tbody>
</table>

1 Refer to the Register TRM (SMIF_TX_DATA_MMIO_FIFO_CTL and SMIF_TX_DATA_MMIO_FIFO_STATUS) for additional details.
2 Refer to the Register TRM (SMIF_RX_DATA_MMIO_FIFO_CTL and SMIF_RX_DATA_MMIO_FIFO_STATUS) for additional details.
## SMIF has one interrupt

<table>
<thead>
<tr>
<th>Interrupt Cause</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TR_TX_REQ</td>
<td>Active when the tr_tx_req trigger is activated.Activated in MMIO operation mode</td>
</tr>
<tr>
<td>TR_RX_REQ</td>
<td>Active when the tr_rx_req trigger is activated.Activated in MMIO operation mode</td>
</tr>
<tr>
<td>XIP_ALIGNMENT_ERROR¹</td>
<td>- The XIP AHB-Lite/AXI bus transfer address is not a multiple of “2” or&lt;br&gt; - The requested XIP AHB-Lite/AXI bus transfer size is not a multiple of “2”&lt;br&gt; In the above conditions&lt;br&gt;  a) a write transfer is requested and&lt;br&gt;  b) Dual-Quad SPI mode is selected or&lt;br&gt;    Octal SPI DDR mode or HyperBus mode is selected without memory write byte masking &lt;br&gt;Activated in XIP operation mode.</td>
</tr>
<tr>
<td>TX_CMD_FIFO_OVERFLOW</td>
<td>Write transfer to TX command FIFO² with not enough free entries available.Activated in MMIO operation mode</td>
</tr>
</tbody>
</table>

¹ Refer to the Register TRM (SMIF_INTR) for additional details.<br>² Refer to the Register TRM (SMIF_TX_CMD_FIFO_WR) for additional details.
### SMIF has one interrupt

<table>
<thead>
<tr>
<th>Interrupt Cause</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX_DATA_FIFO_OVERFLOW</td>
<td>Write transfer to TX data FIFO(^3) with not enough free entries available. Activated in MMIO operation mode</td>
</tr>
<tr>
<td>RX_DATA_FIFO_OVERFLOW</td>
<td>Read transfer from RX data FIFO(^4) with not enough entries available. Activated in MMIO operation mode</td>
</tr>
<tr>
<td>DL_FAIL</td>
<td>Data Learning Failed (no DLP match found on at least one of the input data lines when CTL.INT_CLOCK_DL_ENABLED = 1)</td>
</tr>
<tr>
<td>DL_WARNING</td>
<td>Data Learning Warning (for at least one input data line less then DLP.DL_WARNING_LEVEL delay line taps resulted in a correct DLP capturing when CTL.INT_CLOCK_DL_ENABLED = 1). This interrupt will be suppressed if DL_FAIL also occurs during the same DLP evaluation cycle</td>
</tr>
<tr>
<td>CRC_ERROR</td>
<td>CRC Error. A read transfer data CRC check failed</td>
</tr>
<tr>
<td>FS_STATUS_ERROR</td>
<td>Functional Safety Status Error. A read transfer Functional Safety Status check failed</td>
</tr>
</tbody>
</table>
SMIF Block Diagram

- **SMIF block components**
  - Clock domains
    - CLK_MEM
    - CLK_SLOW
    - CLK_SYS
    - clk_if_tx/rx
The following clocks are used in SMIF

<table>
<thead>
<tr>
<th>Source</th>
<th>Used Block</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk_mem</td>
<td>SMIF Internal Block FIFOs</td>
<td>Used for the SMIF block; it is sourced from the mem clock The frequency range is up to 200 MHz</td>
</tr>
<tr>
<td></td>
<td>AXI Master Interface</td>
<td>Used by the XIP AXI master interface; it is sourced from the mem clock The frequency range is up to 200 MHz</td>
</tr>
<tr>
<td>clk_slow</td>
<td>AHB Slave Interface</td>
<td>Used by the XIP AHB-Lite master interface; it is sourced from the slow clock The frequency range is up to 100 MHz</td>
</tr>
<tr>
<td>clk_sys</td>
<td>AHB Slave Interface</td>
<td>Used by the MMIO AHB-Lite slave interface; it is clocked by the PERI group clock The frequency range is up to 100 MHz</td>
</tr>
<tr>
<td>clk_if_tx/rx</td>
<td>Memory Interface FIFOs</td>
<td>Used for memory interface of transmit and receive; it is clocked by the CLK_HF6 clk_if_tx is the source clock for SPIHB_CLK</td>
</tr>
</tbody>
</table>
Part of your life. Part of tomorrow.
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN</th>
<th>Submission Data</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>6399119</td>
<td>12/3/2018</td>
<td>Initial release</td>
</tr>
<tr>
<td>*A</td>
<td>6633414</td>
<td>7/22/2019</td>
<td>Change from mxsmif_ver2 to mxsmif_ver3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Updated slide 2, 6, 18, 19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Added slide 4, 17, 31</td>
</tr>
<tr>
<td>*B</td>
<td>7039074</td>
<td>12/03/2020</td>
<td>Updated page 2, 3, 4</td>
</tr>
</tbody>
</table>