Customer Training Workshop
Traveo™ II Body High and Cluster 2D SRAM Interface

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/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 High

The SRAM interface is in the CPU subsystem

CYT4BF series has two options of RAM size: 768KB and 1024KB
Introduction for Traveo II Cluster

The SRAM interface is in the CPU subsystem.

Hint Bar
- Review TRM chapter 10 for additional details
- Training section reference: SRAM Interface for Traveo II Body Entry
SRAM Interface Overview

› SRAM controller for SRAM memory interface
› Features
  - One AXI bus interface in the fast clock domain
  - One AHB-Lite bus interface in the slow clock domain
  - Programmable wait states from 0 to 3
    - Wait states for slow clock domain
      - In 0 wait cycle, up to 100 MHz of CLK_MEM
      - In 1 wait cycle, from 100 MHz to 200 MHz of CLK_MEM
    - Wait states for fast clock domain
      - In 0 wait cycle, up to 200 MHz of CLK_MEM
  - 64-bit wide interface to SRAM memory
  - Error-correcting code (ECC)
  - Optional retention of SRAM contents in DeepSleep mode

1 Divided version of high-frequency root clock (CLK_HF0)
TCM Interface Overview

- Used by the Arm® Cortex®-M7 core to enable low-latency access to the external memories
- Features
  - Programmable 0 wait states
    - Wait states for fast clock domain
      - In 0 wait cycle, up to 350 MHz of CLK_FAST\(^1\)
  - TCM interfaces at CM7 core
    - ITCM\(^2\) (64-bit data)
    - DTCM\(^3\) (32-bit data)
  - Error-correcting code (ECC)

---

\(^1\) Divided version of high-frequency root clock (CLK_HF1)
\(^2\) Instruction TCM
\(^3\) Data TCM
The SRAM controller consists of:
- AHB-Lite interface
- Synchronization
- AXI interface
- Port arbitration
- Error-correcting code (ECC)
- SRAM memory
- Write buffer
- Fault reporting

Review TRM section 10.2 for additional details.
AHB-Lite Interface and AXI Interface

- The AHB-Lite interface and AXI interface includes:
  - SRAM access
  - Arbitration priority

Review TRM section 10.1 for additional details
SRAM Access

› SRAM has a 64-bit wide bus. The write access differs depending on the data size and ECC state

› Data read from SRAM
  – AHB-Lite and AXI read transfer
    – Translated into an SRAM read access and can be done by a single read access to SRAM

› Data write to SRAM without ECC (Disabled)
  – AHB-Lite and AXI write transfer
    – Translated into an SRAM write access and can be done by a single write access to SRAM

› Data write to SRAM with ECC (Enabled)
  – 64-bit AXI write transfers
    – Translated into an SRAM write access and can be done by a single write access to SRAM
  – 8-bit, 16-bit, 32-bit AXI, and AHB-Lite write transfer
    – Translated into an SRAM read access and an SRAM write access
    – Requires two accesses to SRAM

Review TRM section 10.2.3 for additional details
Review the Error Correcting Code (ECC) section for additional details about ECC
Review the Write Buffer section for additional details about the write buffer
Arbitration Priority

› The arbiter component performs priority-based arbitration on the AHB-Lite interface and AXI interface ports

› Arbitration priority is set according to the PROT_SMPU_MS0_CTL.PRIO register of SMPU (0 is highest, 3 is lowest)

› If Masters have the same priority, round-robin arbitration is performed according to the bus master identifier (starting from 15 to a lower identifier number)

<table>
<thead>
<tr>
<th>Master Identifier</th>
<th>Bus Master (CYT4BF)</th>
<th>Bus Master (CYT4DN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Arm® Cortex®-M0+</td>
<td>Arm Cortex-M0+</td>
</tr>
<tr>
<td>1</td>
<td>Crypto</td>
<td>Crypto</td>
</tr>
<tr>
<td>2</td>
<td>P-DMA0</td>
<td>P-DMA0</td>
</tr>
<tr>
<td>3</td>
<td>P-DMA1</td>
<td>P-DMA1</td>
</tr>
<tr>
<td>4</td>
<td>M-DMA</td>
<td>M-DMA</td>
</tr>
<tr>
<td>5</td>
<td>SDHC</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>Ethernet 0</td>
<td>Ethernet 0</td>
</tr>
<tr>
<td>10</td>
<td>Ethernet 1</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>Video subsystem</td>
</tr>
<tr>
<td>13</td>
<td>Cortex-M7_1</td>
<td>Cortex-M7_1</td>
</tr>
<tr>
<td>14</td>
<td>Cortex-M7_0</td>
<td>Cortex-M7_0</td>
</tr>
<tr>
<td>15</td>
<td>Test controller</td>
<td>Test controller</td>
</tr>
</tbody>
</table>
Error-Correcting Code (ECC)

- ECC includes:
  - SEC/DED\(^1\)
  - Fault reporting
  - Error injection

1 Single-error correction/double-error detection
ECC

› ECC supports SEC/DED, which:
  – Detects and corrects single-bit error and detects double-bit errors
  – Reports error to the fault reporting structure
  – Includes 8-bit ECC per 64 bits of data

› Fault reporting
  – Correctable and non-correctable ECC errors are reported to the fault structure in the same way
  – Use case
    – SEC report: Logging single-bit error count via the trigger that connects to TCPWM
    – DED report: Use for fail-safe operation, such as stop system control

› Error injection
  – Providing an error injection address and injection parity\(^1\) can generate an ECC error
  – Use case
    – Use as initial diagnosis of ECC before running the application

\(^1\) Any access size can be used to inject parity

Review TRM section 10.3 for additional details
Write Buffer

- Write buffer request

```
<table>
<thead>
<tr>
<th>AHB-Lite interface</th>
<th>AXI interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronization</td>
<td></td>
</tr>
<tr>
<td>Port arbitration</td>
<td></td>
</tr>
<tr>
<td>ECC</td>
<td></td>
</tr>
<tr>
<td>Write buffer</td>
<td></td>
</tr>
<tr>
<td>SRAM</td>
<td></td>
</tr>
</tbody>
</table>
```

- This path includes ECC parity injection
- Partial writes path

**Hint Bar**
Review TRM section 10.2.4 for additional details
The SRAM controller write buffer is used only when ECC is enabled.

Sequence of data merging involving write buffer:
- Requested read data is merged with partial write data to provide a complete 64-bit data word.
- Address and merged write data are written to the write buffer.
- A future write buffer request results in an SRAM write access with the merged write data.

For 8-bit, 16-bit, and 32-bit data size write transfers, an additional SRAM read access precedes the SRAM write access to retrieve the “missing” data bytes.

The write buffer is not retained in DeepSleep mode and should be emptied before entering the mode.

The state of the write buffer is shown by WB_EMPTY.

Review TRM section 10.2.4 for additional details.
Write 64 Bits

- Only a single SRAM write access is required
Data from AXI interface (or AHB-Lite interface) is stored to the write buffer.
Write 8/16/32 Bits, Partial Write with ECC (2/3)

1️⃣ Data from the AXI interface (or AHB-Lite interface) is stored to the write buffer

2️⃣ Missing data are read from SRAM and stored to the write buffer to complete the 64-bit word
Write 8/16/32 Bits, Partial Write with ECC (3/3)

1. Data from the AXI interface (or AHB-Lite interface) is stored to the write buffer.
2. Missing data are read from SRAM and stored to the write buffer to complete the 64-bit word.
3. Complete 64-bit word with ECC is written to the SRAM.
Read With ECC with 1-bit Correctable Error (1/3)

① 1-bit error is detected in AHB-Lite (or AXI interface) bus read data
Read With ECC with 1-bit Correctable Error (2/3)

1. 1-bit error is detected in AHB-Lite (or AXI interface) bus read data

2. The error is notified to fault reporting and the corrected data is sent to the AHB-Lite master requester
Read With ECC with 1-bit Correctable Error (3/3)

1. 1-bit error is detected in AHB-Lite (or AXI interface) bus read data
2. The error is notified to fault reporting and the corrected data is sent to the AHB-Lite master requester
3. If ECC_AUTO_CORRECT is enabled, the corrected data is written to the SRAM
Non-correctable error is detected in AHB-Lite (or AXI interface) bus read data.
Read With ECC with Non-Correctable Error (2/2)

1. Non-correctable error is detected in AHB-Lite (or AXI interface) bus read data

2. The error is notified to fault reporting
   - Fault reporting captures the non-correctable error
     - Single-bit error detected in the word address
     - Double-bit error detected in the data access
   - No bus error is generated while a fault is generated
SRAM retention in DeepSleep

› In DeepSleep mode, SRAM0 can be fully retained or retained in increments of 32-KB sectors

› Advantage
  – By setting the size of backup RAM according to the system, it is possible to optimize the current consumption during DeepSleep mode

Hint Bar

SRAM units, other than SRAM0, can be fully retained

Review TRM section 10.1 for additional details on SRAM region that is not guaranteed to be retained across resets
TCM Interface (1/2)

› Used by the CM7 core to enable low-latency access to the external memories

› Functionality
  - No read and write wait states
  - Can fetch instructions from any TCM interface
  - ECC with SECDED and fault reporting
    - ITCM with 64-bit data needs 8 ECC bits, which require 72-bit wide SRAMs
    - DTCMs with 32-bit data need 7 ECC bits, which require 39-bit wide SRAMs
    - Supports read-modify-write for smaller\(^1\) byte write

› Advantage
  - Can achieve high-performance operation by placing code in TCM RAM

\(^1\) 8, 16, 32-bit in ITCM
8, 16-bit in D0/1TCM
TCM Interface (2/2)

A CM7 core can access its own TCMs and other CM7’s TCMs via the following specific address:

<table>
<thead>
<tr>
<th>Address</th>
<th>Target TCM</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000:0000</td>
<td>CM7_0_ITCM/CM7_1_ITCM</td>
<td>CM7_0 can only access to CM7_0_I/DTCM through this space</td>
</tr>
<tr>
<td>0x2000:0000</td>
<td>CM7_0_DTCM/CM7_1_DTCM</td>
<td>CM7_1 can only access to CM7_1_I/DTCM</td>
</tr>
<tr>
<td>0xA000:0000</td>
<td>CM7_0_ITCM</td>
<td>CM7_0 can access to CM7_1_I/DTCM through this space</td>
</tr>
<tr>
<td>0xA001:0000</td>
<td>CM7_0_DTCM</td>
<td>CM7_1 can access to CM7_0_I/DTCM</td>
</tr>
<tr>
<td>0xA010:0000</td>
<td>CM7_1_ITCM</td>
<td>The CM7 cannot access their own TCM using this space; it will result in an</td>
</tr>
<tr>
<td>0xA011:0000</td>
<td>CM7_1_DTCM</td>
<td>address decode failure and will be returned as a bus error</td>
</tr>
</tbody>
</table>

No other device should be mapped between the ITCM and DTCM of a particular CM7.
Appendix
# Comparison between CYT2BL, CYT4BF, and CYT4DN

<table>
<thead>
<tr>
<th>Features</th>
<th>CYT2BL</th>
<th>CYT4BF</th>
<th>CYT4DN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory size option</td>
<td>512KB</td>
<td>1024KB</td>
<td>640KB</td>
</tr>
<tr>
<td>Interface width to SRAM memory</td>
<td>32-bit</td>
<td>64-bit</td>
<td></td>
</tr>
<tr>
<td>Bus interface</td>
<td>AHB-Lite</td>
<td>AXI, AHB-Lite</td>
<td></td>
</tr>
<tr>
<td>Wait states</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slow clock domain:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 wait cycle for CLK_HF &lt;= 100 MHz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 wait cycle for 100 MHz &lt; CLK_HF &lt;= 160 MHz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast clock domain:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 wait cycle for CLK_HF &lt;= 160 MHz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus master priority of arbiter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: Cortex-M0+ CPU</td>
<td></td>
<td>0: Cortex-M0+ CPU</td>
<td></td>
</tr>
<tr>
<td>1: Cryptography Component</td>
<td></td>
<td>1: Cryptography Component</td>
<td></td>
</tr>
<tr>
<td>2: P-DMA0</td>
<td></td>
<td>2: P-DMA0</td>
<td></td>
</tr>
<tr>
<td>3: P-DMA1</td>
<td></td>
<td>3: P-DMA1</td>
<td></td>
</tr>
<tr>
<td>4: M-DMA</td>
<td></td>
<td>4: M-DMA</td>
<td></td>
</tr>
<tr>
<td>5: SDHC</td>
<td></td>
<td>5: SDHC</td>
<td></td>
</tr>
<tr>
<td>9: Ethernet 0</td>
<td></td>
<td>9: Ethernet 0</td>
<td></td>
</tr>
<tr>
<td>10: Ethernet 1</td>
<td></td>
<td>10: Ethernet 1</td>
<td></td>
</tr>
<tr>
<td>13: Cortex-M7_1 CPU</td>
<td></td>
<td>13: Cortex-M7_1 CPU</td>
<td></td>
</tr>
<tr>
<td>14: Cortex-M7_0 CPU</td>
<td></td>
<td>14: Cortex-M7_0 CPU</td>
<td></td>
</tr>
<tr>
<td>15: Test Controller</td>
<td></td>
<td>15: Test Controller</td>
<td></td>
</tr>
<tr>
<td>ECC (SED/DED)</td>
<td>Same</td>
<td>Same</td>
<td>Same</td>
</tr>
<tr>
<td>RAM retention in DeepSleep</td>
<td>Same</td>
<td>Same</td>
<td>Same</td>
</tr>
</tbody>
</table>
Comparison between CYT2BL, CYT4BF, and CYT4DN

<table>
<thead>
<tr>
<th>Features</th>
<th>CYT2BL</th>
<th>CYT4BF</th>
<th>CYT4DN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory size</td>
<td>N/A</td>
<td>16KB ITCM, 16KB DTCM</td>
<td>64KB ITCM, 64KB DTCM</td>
</tr>
<tr>
<td>Interface width</td>
<td>ITCM: 64-bit</td>
<td>D0TCM: 32-bit</td>
<td></td>
</tr>
<tr>
<td>Bus interface</td>
<td>TCM interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait states</td>
<td>Fast clock domain: 0 wait cycle for CLK_FAST &lt;= 350 MHz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority</td>
<td>Cortex-M7 CPU only</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECC (SED/DED)</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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>6402537</td>
<td>12/05/2018</td>
<td>Initial release</td>
</tr>
<tr>
<td>*A</td>
<td>6649384</td>
<td>08/07/2019</td>
<td>Updated page 2, 3, 4, and 28.</td>
</tr>
<tr>
<td>*B</td>
<td>7073093</td>
<td>01/22/2021</td>
<td>Updated page 1, 2, 3, 10, 24, and 28.</td>
</tr>
</tbody>
</table>