Customer Training Workshop
Traveo™ II Fault Subsystem

Q4 2020
## Target Products

### Target product families 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/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>
Fault Subsystem Overview

› Stores fault information
› Supports signaling interface; reset, interrupt, trigger, or output
› Fault sources:
  – MPU/SMPU/PPU violations
  – Peripheral-specific errors
    – For example: WDT, CSV, and BOD on VDDA, OVD on VDDA and LVD
  – Memory-controller-specific errors
    – For example: SRAM ECC errors and Flash ECC errors
› Communicated as either a bus error or a fault in the fault report structure
› Functionality is available only in Active/Sleep power modes
Fault Subsystem Block Diagram

 Fault subsystem components

Fault Source

... Up to 96 ...

Fault Source

Fault Source

fault_req

fault_data

Fault Report Structure

Central Structure (Pending Faults)

Fault Report Structure i

FAULT_STRUCTx_CTL

FAULT_STRUCTx_STATUS

FAULT_STRUCTx_DATA0

... 

FAULT_STRUCTx_DATA3

FAULT_STRUCTx_PENDING0

FAULT_STRUCTx_PENDING1

FAULT_STRUCTx_PENDING2

FAULT_STRUCTx_MASK0

FAULT_STRUCTx_MASK1

FAULT_STRUCTx_MASK2

FAULT_STRUCTx_INTR

FAULT_STRUCTx_INTR_SET

FAULT_STRUCTx_INTR_MASK

FAULT_STRUCTx_INTR_MASKED

Fault_Report [FAULT_NR]

FAULT_TR_OUT [FAULT_NR]

FAULT_OUT_X

fault_reset_req [FAULT_NR]

Review TRM section 15.1 for additional details

See each device datasheet for Fault Sources

Hint Bar
Fault Sources

› MPU/SMPU/PPU violations
› Peripheral-specific errors
  – Example: WDT, CSV, and BOD on VDDA, OVD on VDDA, LVD
› Memory-controller-specific errors
  – Examples: SRAM ECC errors, Flash and Flash ECC errors

Review TRM chapter 15 for additional details
See each device datasheet for Fault Sources
Pending faults are shared by all fault report structures.

Central structure keeps track of all the pending faults:
- \texttt{FAULT\_STRUCTx\_PENDINGy}\(^1\) registers reflect which of the fault sources are pending.
- When a pending fault is captured by a fault structure, the associated pending bit is cleared to ‘0’ by hardware.

\(^1\) \texttt{FAULT\_STRUCTx}, ‘x’ signifies the fault structure instance and ‘y’ in \texttt{FAULT\_STRUCTx\_PENDINGy} varies from 0 through 2.
Fault Report Structure

› Captures fault information

› Signaling interface
  - Interrupt
  - Trigger
  - Output signal
  - Reset request

1 See each device datasheet for the number of the fault report structures
2 Fault, Software, MCWDT, Debug, CSV

Traveo II has fault report structures (FAULT_NR\(^1\)) to group the fault sources

Each fault report structure has a dedicated set of control and status registers, and captures a single fault

Review TRM section 15.1 and Register TRM for additional details

1 \(^1\) See each device datasheet for the number of the fault report structures
2 \(^2\) Fault, Software, MCWDT, Debug, CSV
Captured Fault Information

- Each fault report structure has a dedicated set of control and status registers and captures a single fault.

- The captured fault information includes:
  - A validity bit field that indicates that a fault is captured.
  - A fault index that identifies the fault source.
  - Additional fault information that changes depending on the fault source.

### Fault Sources

<table>
<thead>
<tr>
<th>Fault Sources</th>
<th>Additional Fault Information</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fault Block</td>
</tr>
<tr>
<td>MPU/SMPU/PPU/CAN ECC</td>
<td>✔ ✔ ✔</td>
</tr>
<tr>
<td>SRAM/Flash ECC</td>
<td>✔ ✔</td>
</tr>
<tr>
<td>Flash Bus Error</td>
<td>✔</td>
</tr>
<tr>
<td>WDT/CSV/BOD/OVD/LVD</td>
<td>✔</td>
</tr>
</tbody>
</table>
Use Case:

Fault and Protection

- Use a Fault Interrupt when an SMPU violation is detected
  - Enable Fault Source 0 (CM0+ MPU/SMPU violation) of MASK register and INTR_MASK to generate the interrupt
- When CM0+ accesses three memory regions (Region 0, 1, 2), as shown below, a fault interrupt occurs
  - Privileged, Write, Secure, Protection Context (PC) = 4, Access to Region 0, 1, 2

CM0+ CPU
- Software Func1
- Privileged
- Write
- Secure
- PC = 4

CM4, CM7_0, or CM7_1 CPU
- Software Func1
- Unprivileged
- Write
- Non-secure
- PC = 6

Use Case:

- Use a Fault Interrupt when an SMPU violation is detected
  - Enable Fault Source 0 (CM0+ MPU/SMPU violation) of MASK register and INTR_MASK to generate the interrupt
- When CM0+ accesses three memory regions (Region 0, 1, 2), as shown below, a fault interrupt occurs
  - Privileged, Write, Secure, Protection Context (PC) = 4, Access to Region 0, 1, 2

Region 0:
- Attributes:
  - PC = 4
  - Privileged
  - Read/Write
  - Unprivileged
  - No Access
  - Secure

Region 1:
- Attributes:
  - PC = 6
  - Privileged
  - Read/Write
  - Unprivileged
  - Read/Write
  - Non-secure

Region 2:
- Attributes:
  - PC = 4, 6
  - Privileged
  - Read Only
  - Unprivileged
  - Read/Write
  - Non-secure

Memory:
- Fault Interrupt
- Safe Operation for User Applications
- Return Interrupt

Safe Operation for User Applications

- Check the following from the STATUS Register:
  - MPU, PPU, ECC BUS_ERROR, or SRSS Violation
- Check the following from the DATA Register:
  - Violating Address
  - Master ID
  - Protection Context ID
  - Access Information
    - User Read/User Write/User Execute/Privileged
    - Read/Privileged Write/Privileged Execute/Non-secure

Clear the Fault Interrupt Cause Flag
(FAULT_STRUCTx_INTR)
The fault subsystem supports a signaling interface that notifies the CPU or the external system that there is a fault.

The following signals are enabled by software:
- A fault interrupt (“Fault_Interrupt”)
- A trigger (“FAULT_TR_OUT”)  
- A chip output pin (“FAULT_OUT_X”)  
- A fault reset request (“fault_reset_req”)

Use Case:
- Advantage
  - Centralizes fault information management and handles each fault according to system requirements

<table>
<thead>
<tr>
<th>Fault</th>
<th>Signaling Interface</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPU/SMPU/PPU Violation</td>
<td>Interrupt</td>
<td>Check the master and address</td>
</tr>
<tr>
<td>ECC Correctable Error (SEC)</td>
<td>Trigger connects to TCPWM</td>
<td>Log the error counts</td>
</tr>
<tr>
<td>LVD/CSV Violation</td>
<td>Output pin</td>
<td>Send alarm to system and stop the MCU</td>
</tr>
<tr>
<td>MCWDT Timeout</td>
<td>Reset request</td>
<td>Recover from abnormal operation</td>
</tr>
</tbody>
</table>

Review TRM section 15.1 and Register TRM for additional details.

Review the Trigger Multiplexer training section for additional trigger details.
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>6129736</td>
<td>04/10/2018</td>
<td>Initial release</td>
</tr>
<tr>
<td>*A</td>
<td>6321436</td>
<td>09/26/2018</td>
<td>Added page 2, 6, and the note descriptions of all pages. Updated page 3, 4, 7, 9. Removed the Appendix.</td>
</tr>
<tr>
<td>*B</td>
<td>6595227</td>
<td>06/14/2019</td>
<td>Updated page 2, 4 to 7, 9, 10.</td>
</tr>
<tr>
<td>*C</td>
<td>7013756</td>
<td>10/29/2020</td>
<td>Updated page 2, 6.</td>
</tr>
</tbody>
</table>