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

The interrupt controller is in the CPUSS block

Review TRM chapter 12 for additional details
Introduction to Traveo II Body Controller High

The interrupt controller is in the CPUSS block

Review TRM chapter 12 for additional details
Introduction to Traveo II Cluster

The interrupt controller is in the CPUSS block

Review TRM chapter 12 for additional details
Interrupts Overview

- Interrupts are events generated by peripherals of each CPU
- Exceptions are events generated by each CPU

- Features
  - Up to 1023\(^1\) system interrupts
  - Any of the system interrupts can be mapped to each CPU NMI (up to four)
  - Vector table is placed in either flash or SRAM
  - Configurable priority levels (eight levels for Cortex-M4/M7 and four levels for Cortex-M0+) for each interrupt
  - All the available system interrupt sources are usable in Active power mode and wake up from Sleep power mode
  - Wakeup interrupts are capable of waking the device from DeepSleep power mode

---

\(^1\) The total number of system interrupts varies depending on the device.
Components in Interrupt Architecture

The interrupt architecture consists of the following components.

Arm provides additional reference material on their webpage at: [infocenter.arm.com](http://infocenter.arm.com)
Interrupt Architecture Block Diagram

Interrupt Architecture Components
- Interrupt sources
- System interrupts
- Wakeup interrupts

Wakeup interrupts are mapped to the same number of CPU interrupts

Number of system interrupts (N)
Number of wakeup interrupts (M)

Refer to the Appendix for the number of system interrupts, listed by device
Refer to each device datasheet for the list of system interrupts
Interrupt Sources

› System Interrupts
  – Originate from peripheral interrupts
  – Include wakeup interrupts
› Wakeup Interrupts
  – Wakes CPU up from DeepSleep mode
› Use Case
  – The GPIO can be used as a CAN wakeup interrupt

Hint Bar

Review TRM section 12.5 for additional details about system interrupts
Refer to each device datasheet for the list of system interrupts
Review TRM section 12.10 for additional details about wakeup interrupts
Wakeup from DeepSleep mode involves the WIC
Review TRM section 12.10 for additional details specific to WIC
Review the Device Power Modes training section for additional Wakeup Interrupts details
Interrupt Synchronizer

- Interrupt Architecture Components
  - Interrupt synchronizer

Synchronizes the interrupts to the CPU clock frequency as the peripheral interrupts can be asynchronous to the CPU clock frequency.

**Hint Bar**

Review TRM section 12.5 for additional details about system interrupts

Refer to each device datasheet for the list of system interrupts

Review TRM section 12.10 for additional details about wakeup interrupts

Wakeup from DeepSleep mode involves the WIC

Review TRM section 12.10 for additional details specific to WIC

Review the Device Power Modes training section for additional Wakeup Interrupts details
CPU Interrupt

› Interrupt Architecture Components
  - CPU interrupt
    - Each system interrupt can be mapped to exactly one CPU interrupt of each core
      - Achieved using the register setting
        - CM0/CM4/CM7_0/CM7_1_SYSTEM_INT_CTL.CPU_INT_IDX[2:0]
        - CM0/CM4/CM7_0/CM7_1_SYSTEM_INT_CTL.CPU_INT_VALID

Refer to each device datasheet for the list of system interrupts

CM7_0 and CM7_1 represent the first and the second CM7 core in the TVII-B-H device, respectively
Interrupt Handler Processing (1/3)

Sequence of a normal interrupt request handling

1. Interrupt request asserts IRQn
2. When the IRQn request can be accepted, the NVIC sets the pending status
3. Jumps to the interrupt handler
4. Entering the interrupt handler clears the pending status
5. Clears the interrupt flag of the peripheral register
6. Reads the interrupt flag of the peripheral register to drain the Write Buffer
7. Return
The CPU interrupt handler uses the SYSTEM_INT_IDX field to index a system interrupt lookup table and jump to the system interrupt handler. Read after write (RAW) is important in the interrupt handler processing to ensure completion of the write buffer.

Review TRM section 12.5 for additional details. The lookup table is usually located in one of the system memories.
The following code illustrates the sequence:

```c
void CM4/CM7_0/CM7_1_CpuIntr0_Handler (void)
{
    uint32_t system_int_idx;
    SystemIntr_Handler handler;
    if(CPUSS_CM4/CM7_0/CM7_1_INT_STATUS[0].SYSTEM_INT_VALID)
    {
        system_int_idx = CPUSS_CM4/CM7_0/CM7_1_INT_STATUS[0].SYSTEM_INT_IDX;
        handler = SystemIntr_Table[system_int_idx];
        handler(); // jump to system interrupt handler
    }
    else
    {
        // Triggered by software or due to software cleared a peripheral interrupt flag
        // but did not clear the Pending flag at NVIC
    }
}
...
void CM4/CM7_0/CM7_1_CpuIntr7_Handler (void)
{
    uint32_t    system_int_idx;
    SystemIntr_Handler handler;
    if(CPUSS_CM4/CM7_0/CM7_1_INT_STATUS[7].SYSTEM_INT_VALID)
    {
        system_int_idx = CPUSS_CM4/CM7_0/CM7_1_INT_STATUS[7].SYSTEM_INT_IDX;
        handler = SystemIntr_Table[system_int_idx];
        handler(); // jump to system interrupt handler
    }
    else
    {
        // Triggered by software or due to software cleared a peripheral interrupt flag
        // but did not clear the Pending flag at NVIC
    }
}
```

- void CM4/CM7_0/CM7_1_SystemIntr0_Handler (void)
- {
-     // Clear the peripheral interrupt request flag by register write
-     // Read back the register to ensure completion of register write access
-     // Handle system interrupt 0.
-     }
- ...
- void CM4/CM7_0/CM7_1_SystemIntr1022_Handler (void)
- {
-     // Clear the peripheral interrupt request flag by register write
-     // Read back the register to ensure completion of register write access
-     // Handle system interrupt 1022.
- }
Interrupt Architecture Components

- NVIC
  - CPU interrupt priority
  - Nested interrupts

NVIC receives IRQs, evaluates the priority, and communicates with the CPU core.
The exception vector tables store the entry point addresses for all exception handlers in Cortex-M0+, Cortex-M4, and Cortex M7 cores.

### Cortex-M0+ Exception Vector Table

<table>
<thead>
<tr>
<th>Exception #</th>
<th>Exception</th>
<th>Exception Priority</th>
<th>Vector Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>–</td>
<td>Initial stack pointer value</td>
<td>Not applicable (N/A)</td>
<td>Start_Address = 0x0000 or CM0P_SCS_VTOR¹</td>
</tr>
<tr>
<td>1</td>
<td>Reset</td>
<td>–3, highest priority</td>
<td>Start_Address + 0x04</td>
</tr>
<tr>
<td>2</td>
<td>Non-maskable Interrupt (NMI)</td>
<td>–2</td>
<td>Start_Address + 0x08</td>
</tr>
<tr>
<td>3</td>
<td>Hard fault</td>
<td>–1</td>
<td>Start_Address + 0x0C</td>
</tr>
<tr>
<td>4–10</td>
<td>Reserved</td>
<td>N/A</td>
<td>Start_Address + 0x10 to Start_Address + 0x28</td>
</tr>
<tr>
<td>11</td>
<td>Supervisory call (SVCall)</td>
<td>Configurable (0–3)</td>
<td>Start_Address + 0x2C</td>
</tr>
<tr>
<td>12–13</td>
<td>Reserved</td>
<td>N/A</td>
<td>Start_Address + 0x30 to Start_Address + 0x34</td>
</tr>
<tr>
<td>14</td>
<td>Pend supervisory (PendSV)</td>
<td>Configurable (0–3)</td>
<td>Start_Address + 0x38</td>
</tr>
<tr>
<td>15</td>
<td>System tick timer (SysTick)</td>
<td>Configurable (0–3)</td>
<td>Start_Address + 0x3C</td>
</tr>
<tr>
<td>16</td>
<td>External interrupt (IRQ0)</td>
<td>Configurable (0–3)</td>
<td>Start_Address + 0x40</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>23</td>
<td>External interrupt (IRQ7)</td>
<td>Configurable (0–3)</td>
<td>Start_Address + 0x5C</td>
</tr>
<tr>
<td>24</td>
<td>Internal (SW only) interrupt (IRQ8)</td>
<td>Configurable (0–3)</td>
<td>Start_Address + 0x60</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>31</td>
<td>Internal (SW only) interrupt (IRQ15)</td>
<td>Configurable (0–3)</td>
<td>Start_Address + 0x7C</td>
</tr>
</tbody>
</table>

¹ Start Address = 0x0000 on reset and is later modified in the user code by updating the CM0P_SCS_VTOR register

---

**Hint Bar**

Review TRM section 12.3.3 for additional details.

- IRQ0–IRQ7 are connected to the System Interrupt generation logic.
- IRQ8–IRQ15 can be triggered by software only and are not connected to any peripheral.
## Exception Vector Table (2/2)

### Cortex-M4/M7 Exception Vector Table

<table>
<thead>
<tr>
<th>Exception #</th>
<th>Exception</th>
<th>Exception Priority</th>
<th>Vector Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>–</td>
<td>Initial stack pointer value</td>
<td>–</td>
<td>Start_Address = 0x0000 or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CM4/CM7_0/CM7_1_SCS_VTOR¹</td>
</tr>
<tr>
<td>1</td>
<td>Reset</td>
<td>–3, highest priority</td>
<td>Start_Address + 0x0004</td>
</tr>
<tr>
<td>2</td>
<td>Non-maskable Interrupt (NMI)</td>
<td>–2</td>
<td>Start_Address + 0x0008</td>
</tr>
<tr>
<td>3</td>
<td>Hard fault</td>
<td>–1</td>
<td>Start_Address + 0x000C</td>
</tr>
<tr>
<td>4</td>
<td>Memory management fault</td>
<td>Configurable (0–7)</td>
<td>Start_Address + 0x0010</td>
</tr>
<tr>
<td>5</td>
<td>Bus fault</td>
<td>Configurable (0–7)</td>
<td>Start_Address + 0x0014</td>
</tr>
<tr>
<td>6</td>
<td>Usage fault</td>
<td>Configurable (0–7)</td>
<td>Start_Address + 0x0018</td>
</tr>
<tr>
<td>7–10</td>
<td>Reserved</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>11</td>
<td>Supervisory call (SVCall)</td>
<td>Configurable (0–7)</td>
<td>Start_Address + 0x002C</td>
</tr>
<tr>
<td>12–13</td>
<td>Reserved</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>14</td>
<td>Pend supervisory (PendSV)</td>
<td>Configurable (0–7)</td>
<td>Start_Address + 0x0038</td>
</tr>
<tr>
<td>15</td>
<td>System tick timer (SysTick)</td>
<td>Configurable (0–7)</td>
<td>Start_Address + 0x003C</td>
</tr>
<tr>
<td>16</td>
<td>External interrupt (IRQ0)</td>
<td>Configurable (0–7)</td>
<td>Start_Address + 0x0040</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>23</td>
<td>External interrupt (IRQ7)</td>
<td>Configurable (0–7)</td>
<td>Start_Address + 0x005C</td>
</tr>
<tr>
<td>24</td>
<td>Internal (SW only) interrupt (IRQ8)</td>
<td>Configurable (0–7)</td>
<td>Start_Address + 0x60</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>31</td>
<td>Internal (SW only) interrupt (IRQ15)</td>
<td>Configurable (0–7)</td>
<td>Start_Address + 0x7C</td>
</tr>
</tbody>
</table>

¹ Start Address = 0x0000 on reset and is later modified by the user code by updating the CM4/CM7_0/CM7_1_SCS_VTOR register

Hint Bar

- Review TRM section 12.3.3 for additional details
- IRQ0-IRQ7 are connected to the System Interrupt generation logic
- IRQ8-IRQ15 can be triggered by software only and are not connected to any peripheral
The priority of each interrupt can be configured to eight levels for both Cortex-M4 and M7 and four levels for Cortex-M0+

Use case for CPU interrupt priority
- Example of priority levels and interrupts for body application

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>Interrupt Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Fault</td>
<td>NMI exception, protection violation</td>
</tr>
<tr>
<td>0</td>
<td>GPIO</td>
<td>Ignition detection, wakeup use</td>
</tr>
<tr>
<td>1</td>
<td>CAN</td>
<td>Communication with other ECU</td>
</tr>
<tr>
<td>2</td>
<td>LIN</td>
<td>Communication with other ECU</td>
</tr>
<tr>
<td>3</td>
<td>SCB</td>
<td>Communication with external IC</td>
</tr>
<tr>
<td>4</td>
<td>TCPWM</td>
<td>Task management</td>
</tr>
<tr>
<td>5</td>
<td>Event Generator</td>
<td>Wakeup use</td>
</tr>
<tr>
<td>6</td>
<td>ADC</td>
<td>Sensor data acquisition</td>
</tr>
<tr>
<td>7</td>
<td>RTC</td>
<td>Real-time clock alarm</td>
</tr>
</tbody>
</table>

Review TRM section 12.5 for additional details
Refer to each device datasheet for the list of system interrupts
Nested Interrupts

- Depending on the interrupt priority level, nested interrupts are possible
- Use case for nested interrupts

**Diagram:**
- Main Routine
- ADC Interrupt
- CAN Interrupt
- LIN Interrupt
- Priority Level = 6 for ADC
- Priority Level = 1 for CAN and LIN
- Return

**Hint Bar:**
- Review TRM section 12.5 for additional details
- CAN is serviced before LIN based on the index order of system interrupts
- Refer to each device datasheet for the list of system interrupts
Enabling and Disabling Interrupts

› The NVICs of CM0+, CM4, and CM7 cores provide registers to individually enable and disable the CPU interrupts in software
› CM0+, CM4, and CM7 interrupts are enabled and disabled using the ISER and ICER
› If an interrupt is not enabled, the NVIC will not process the interrupt requests on that interrupt line
› CM0+, CM4, and CM7 provide additional registers to control the activation of exceptions/interrupts based on their priority
  – PRIMASK: Prevent activation of exceptions having configurable priority
  – FAULTMASK: Prevent activation of all exceptions other than NMI
  – BASEPRI: Prevent activation of exceptions having the same or lower priority than the BASEPRI

Review TRM section 12.7 for additional details

Interrupt Set-Enable Register (ISER)
Interrupt Clear-Enable Register (ICER)
Exception States

› Each exception can be in one of the following states:

- **Inactive**
  - Not Active
  - Not Pending
  - \(\rightarrow\) \(\rightarrow\)

- **Active**
  - IRQ3 (CAN) priority = 1
  - \(\rightarrow\) \(\rightarrow\)

- **Pending**
  - IRQ5 (Fault) Priority= 0
  - \(\rightarrow\) \(\rightarrow\)

- **Active Pending**
  - IRQ3 (TCPWM) priority = 1
  - \(\rightarrow\) \(\rightarrow\)

- **Return**
  - Pending exception is serviced after the execution of exceptions with higher priority is complete
  - \(\rightarrow\) \(\rightarrow\)

Exception is serviced by the processor, pending a request from the same source during exception handler execution

Review TRM section 12.8 for additional details.
Appendix
Number of Interrupts

- Maximum number of system interrupts and wakeup interrupts varies by device.

<table>
<thead>
<tr>
<th>Series</th>
<th>Maximum Number of System Interrupts</th>
<th>Maximum Number of Wakeup Interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td>CYT2B6</td>
<td>228</td>
<td>38</td>
</tr>
<tr>
<td>CYT2B7</td>
<td>353</td>
<td>45</td>
</tr>
<tr>
<td>CYT2B9</td>
<td>383</td>
<td>45</td>
</tr>
<tr>
<td>CYT2BL</td>
<td>383</td>
<td>45</td>
</tr>
<tr>
<td>CYT3BB/CYT4BB</td>
<td>443</td>
<td>51</td>
</tr>
<tr>
<td>CYT4BF</td>
<td>567</td>
<td>51</td>
</tr>
<tr>
<td>CYT3DL</td>
<td>798</td>
<td>50</td>
</tr>
<tr>
<td>CYT4DN</td>
<td>798</td>
<td>50</td>
</tr>
</tbody>
</table>

Refer to each device datasheet for the list of system interrupts.
Part of your life. Part of tomorrow.
<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>6154903</td>
<td>04/29/2018</td>
<td>Initial release</td>
</tr>
<tr>
<td>*A</td>
<td>6396762</td>
<td>11/29/2018</td>
<td>Added pages 2, 4, and 5 and note descriptions for all pages Updated pages 3, 6, 7, 8, 10, 11, 14, 15, 16, 17, 18, and 20 Changed the contents of the Appendix section</td>
</tr>
<tr>
<td>*B</td>
<td>6612968</td>
<td>07/04/2019</td>
<td>Updated note descriptions for pages 3, 4, 5, 23, and 24 Updated pages 2, 5, 14, and 23</td>
</tr>
<tr>
<td>*C</td>
<td>7042917</td>
<td>12/11/2020</td>
<td>Updated page 2, 3, 9 and 23</td>
</tr>
</tbody>
</table>