Customer training workshop
TRAVEO™ T2G interrupts

Q4 2021

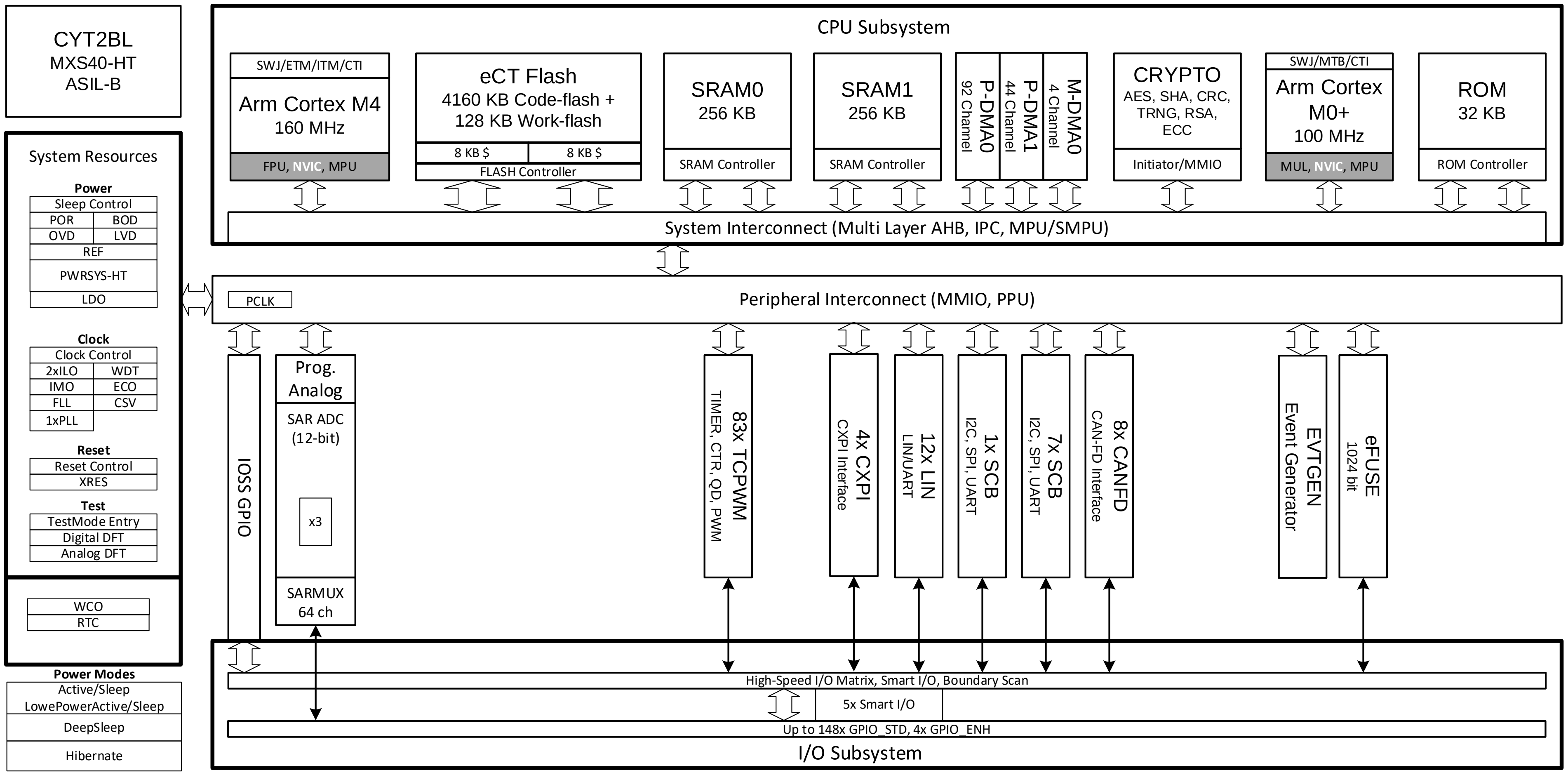
Target products

› Target product list for this training material:

Family category	Series	Code flash memory size
TRAVEO™ T2G Automotive Body Controller Entry	CYT2B6	Up to 576 KB
TRAVEO™ T2G Automotive Body Controller Entry	CYT2B7	Up to 1088 KB
TRAVEO™ T2G Automotive Body Controller Entry	CYT2B9	Up to 2112 KB
TRAVEO™ T2G Automotive Body Controller Entry	CYT2BL	Up to 4160 KB
TRAVEO™ T2G Automotive Body Controller High	CYT3BB/ CYT4BB	Up to 4160 KB
TRAVEO™ T2G Automotive Body Controller High	CYT4BF	Up to 8384 KB
TRAVEO™ T2G Automotive Cluster	CYT2CL	Up to 4160 KB
TRAVEO™ T2G Automotive Cluster	CYT3DL	Up to 4160 KB
TRAVEO™ T2G Automotive Cluster	CYT4DN	Up to 6336 KB

Introduction to TRAVEO™ T2G Body Controller Entry

› The interrupt controller is in the CPUSS block

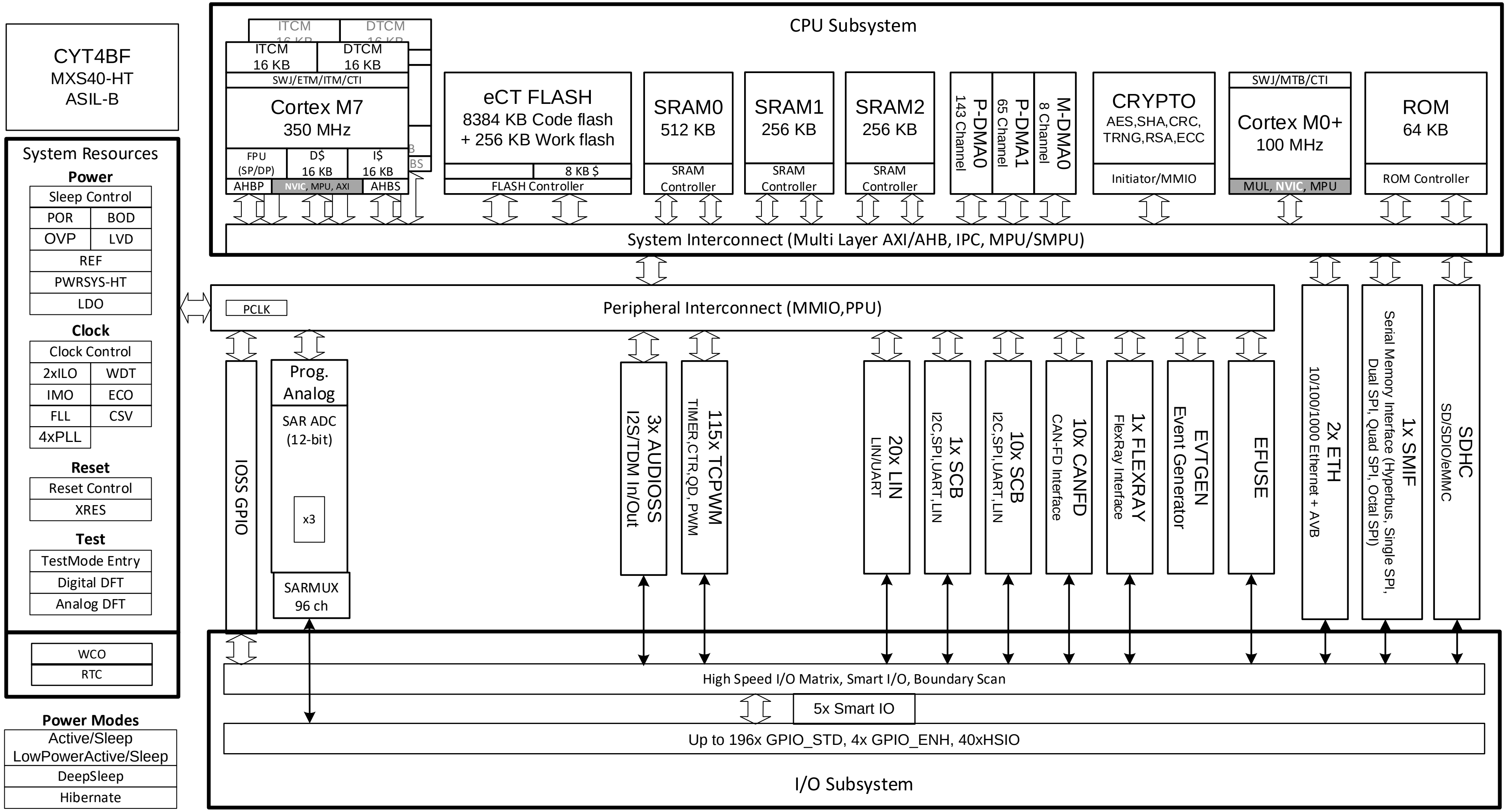


Hint Bar

Review TRM chapter 12 for additional details

Introduction to TRAVEO™ T2G Body Controller High

› The interrupt controller is in the CPUSS block

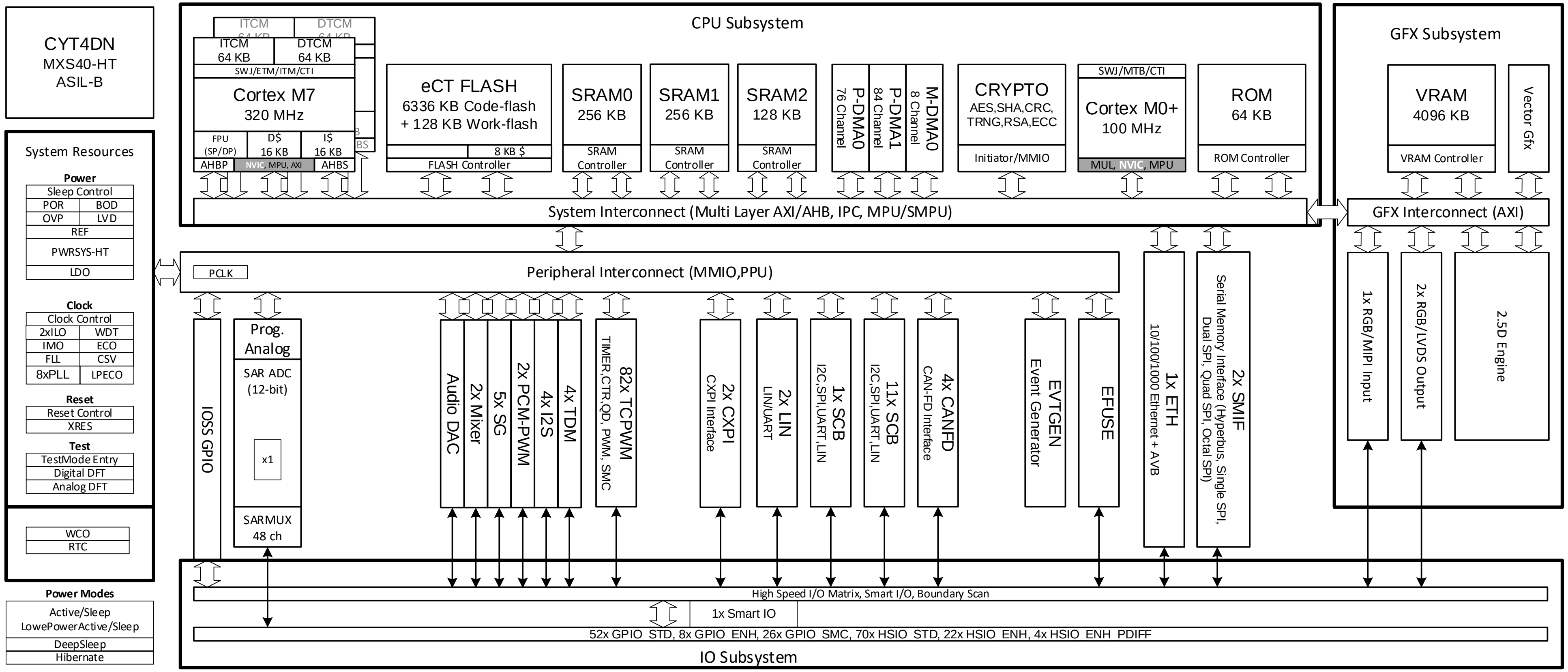


Hint Bar

Review TRM chapter 12 for additional details

Introduction to TRAVEO™ T2G Cluster

› The interrupt controller is in the CPUSS block



Hint Bar

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

Hint Bar

Review TRM section 12.1 for additional details specific to Interrupts

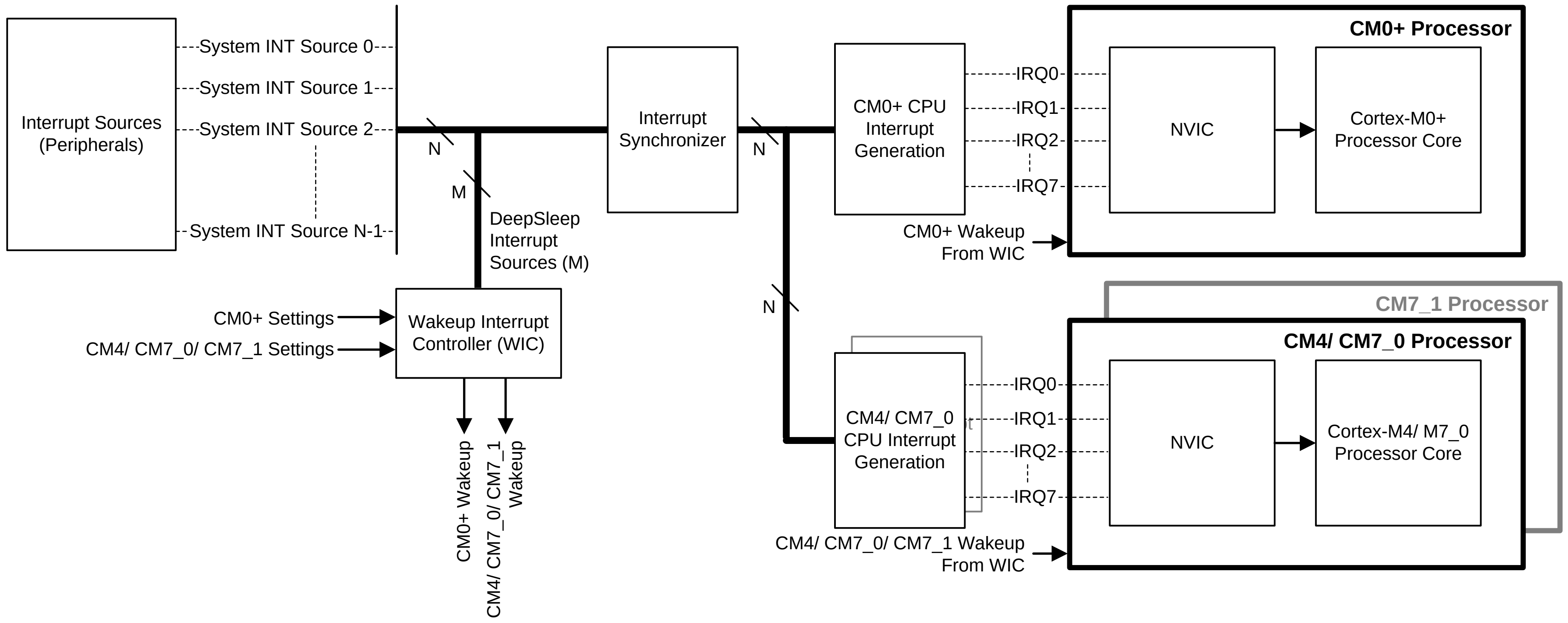
Refer to each device datasheet for the list of system interrupts

Refer to the CPUSS Training Section for additional Vector Table Relocation details

[1] The total number of system interrupts varies depending on the device

Components in interrupt architecture

› The interrupt architecture consists of the following components.



Hint Bar

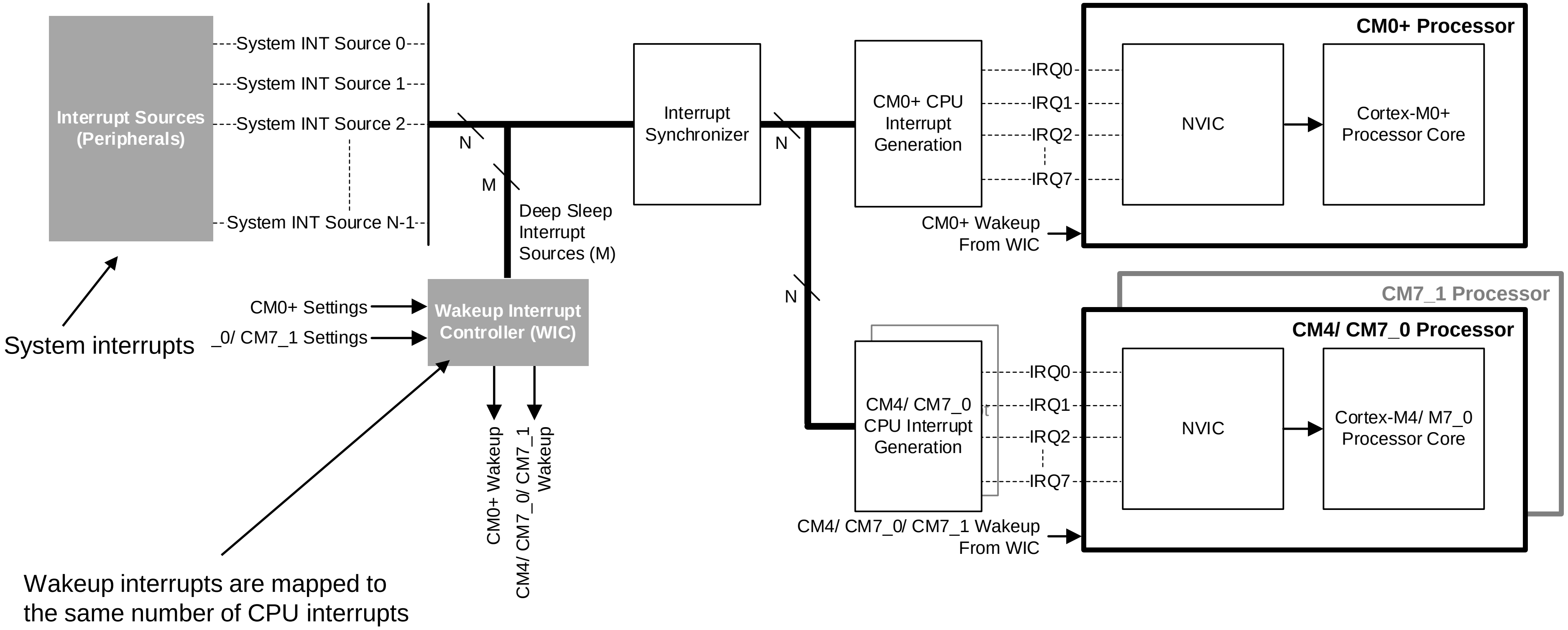
Arm provides additional reference material on their webpage at: infocenter.arm.com

Nested Vectored Interrupt Controller (NVIC)

Interrupt request (IRQ)

Interrupt architecture block diagram

- Interrupt architecture components
  - Interrupt sources
  - System interrupts
  - Wakeup interrupts



Hint Bar

Number of system interrupts (N)

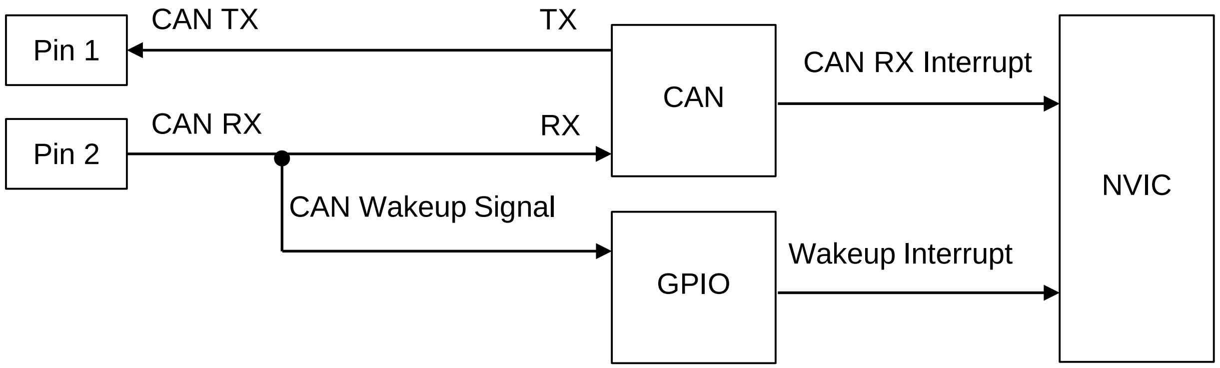
Number of wakeup interrupts (M)

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

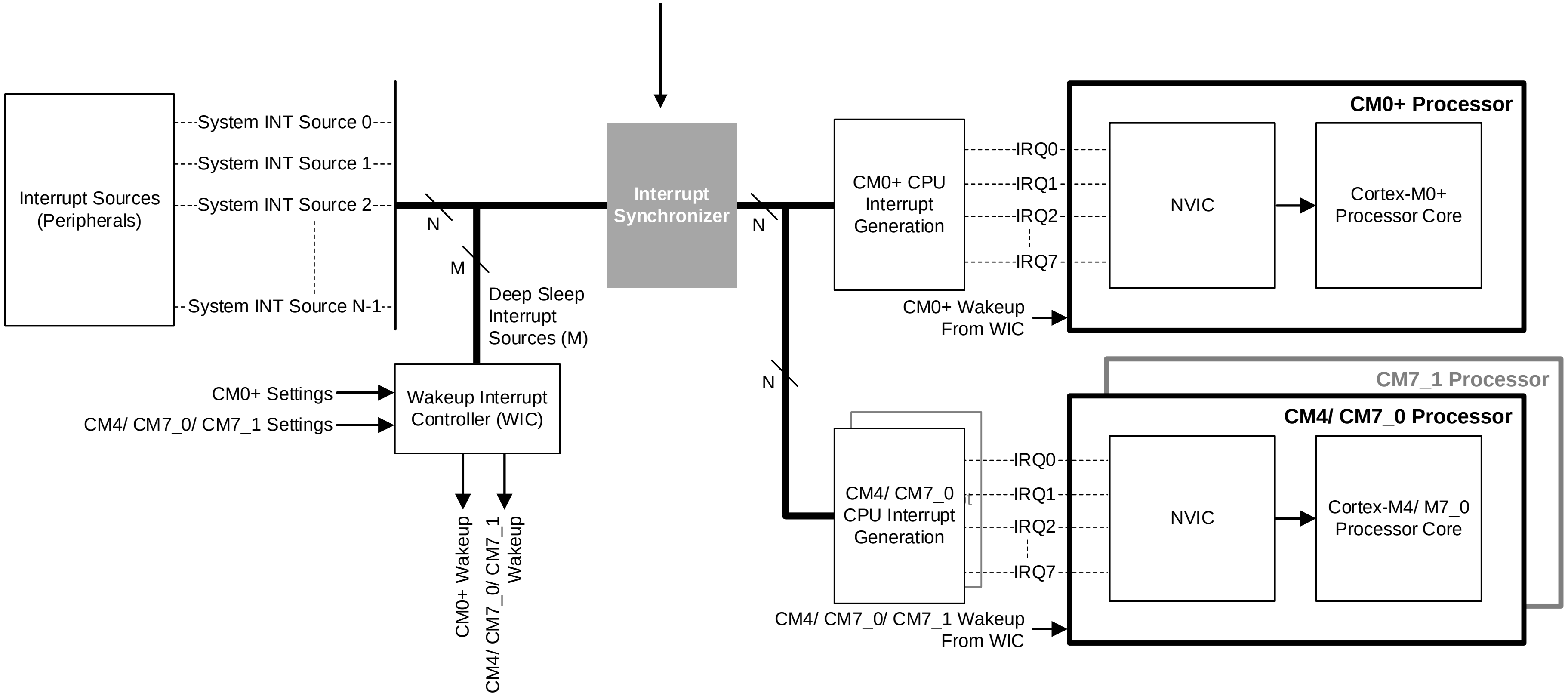
Review TRM section 12.10 for additional details specific to WIC

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

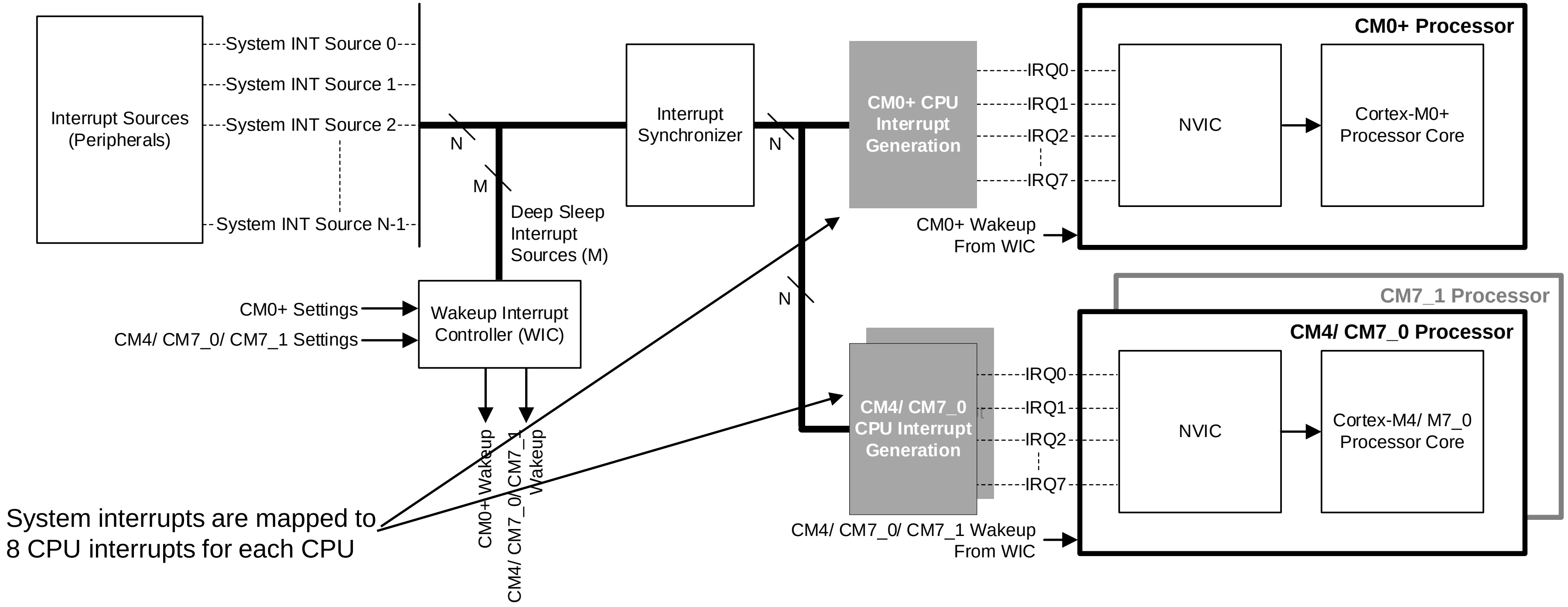
Wakeup from DeepSleep mode involves the WIC

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



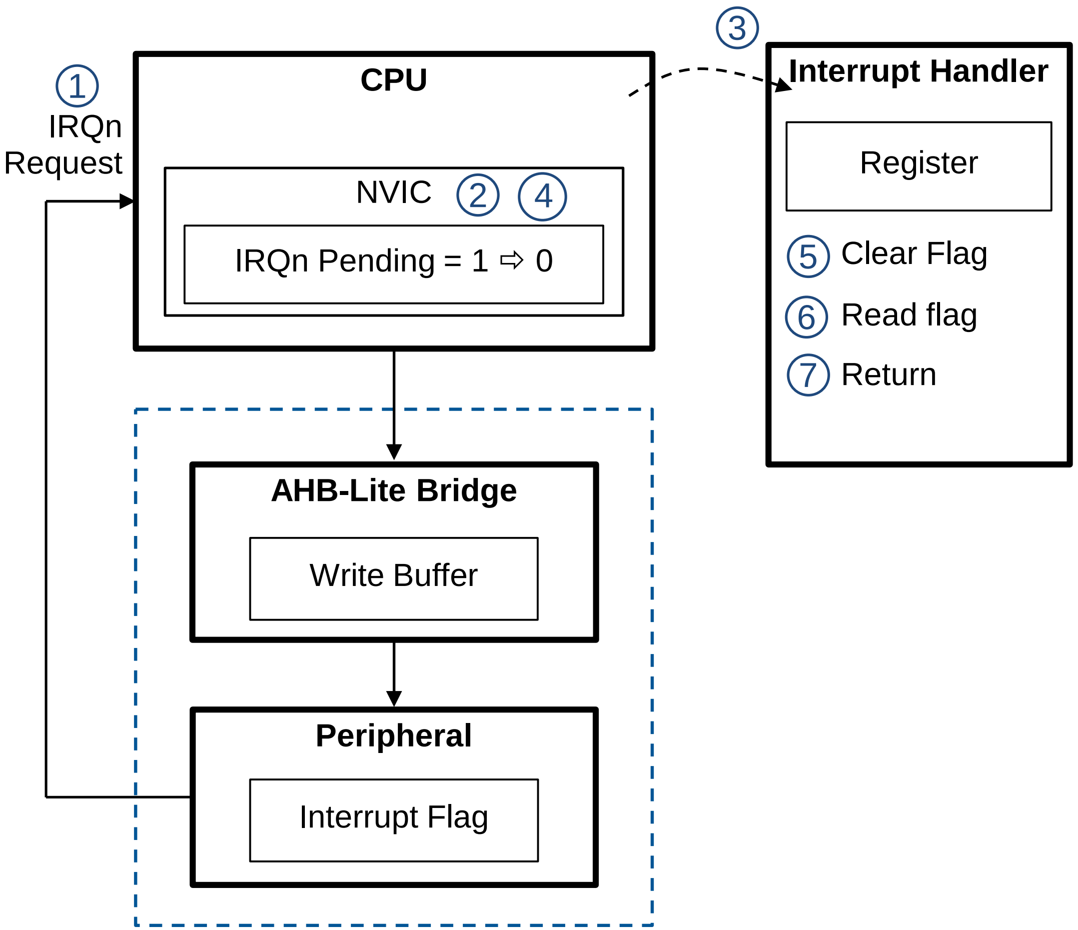
Hint Bar

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 CYT4 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



Interrupt handler processing (2/3)

- 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

Hint Bar

Review TRM section 12.5 for additional details

The lookup table is usually located in one of the system memories

Interrupt handler processing (3/3)

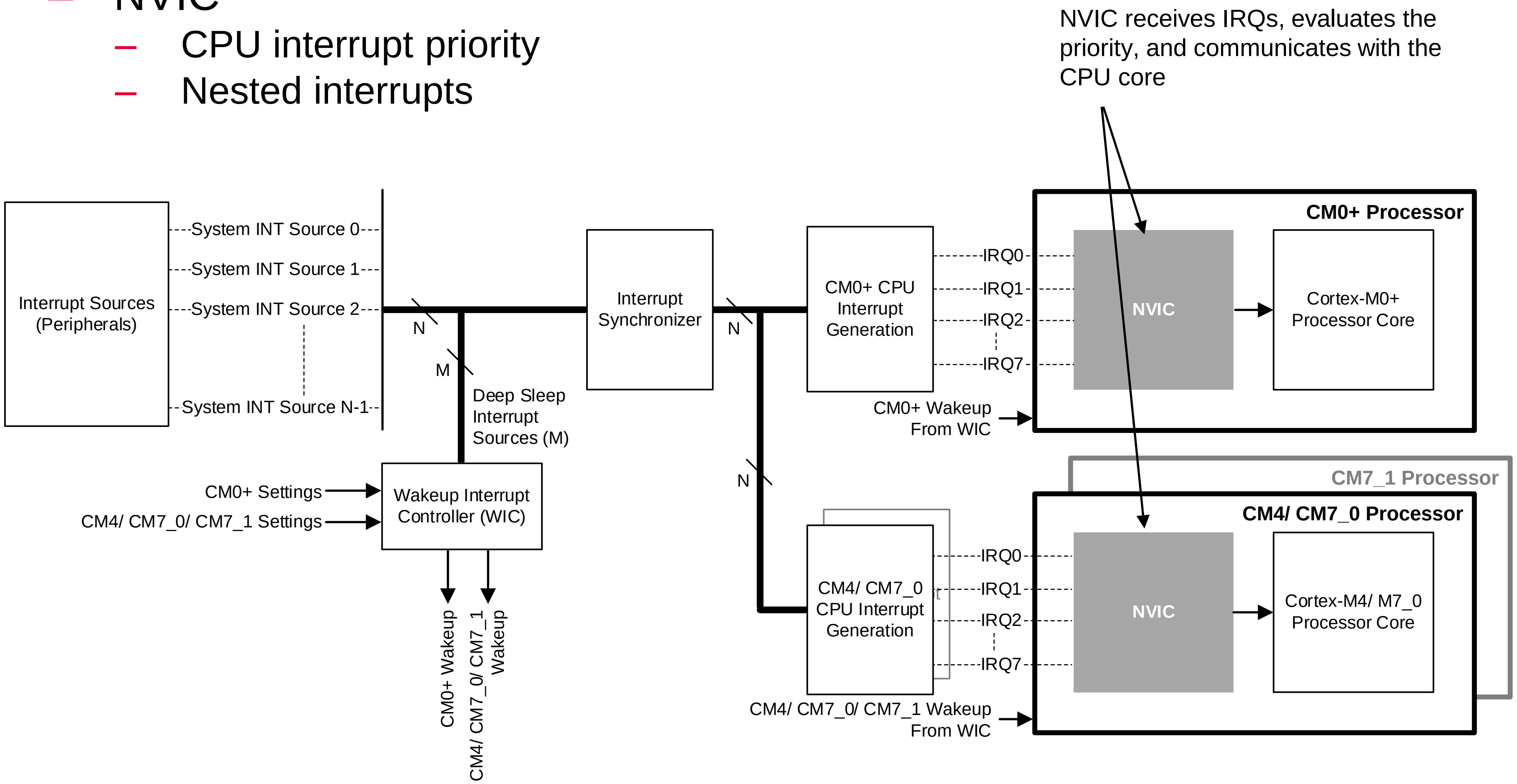
- The following code illustrates the sequence:

```
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.
```

NVIC

- Interrupt architecture components
  - NVIC
    - CPU interrupt priority
    - Nested interrupts



Hint Bar

Arm provides additional reference material on their webpage at: infocenter.arm.com

Exception vector table (1/2)

- 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

Exception #	Exception	Exception Priority	Vector Address
–	Initial stack pointer value	Not applicable (N/A)	Start_Address = 0x0000 or CM0P_SCS_VTOR[1]
1	Reset	–3, highest priority	Start_Address + 0x04
2	Non-maskable Interrupt (NMI)	–2	Start_Address + 0x08
3	Hard fault	–1	Start_Address + 0x0C
4–10	Reserved	N/A	Start_Address + 0x10 to Start_Address + 0x28
11	Supervisory call (SVCall)	Configurable (0–3)	Start_Address + 0x2C
12–13	Reserved	N/A	Start_Address + 0x30 to Start_Address + 0x34
14	Pend supervisory (PendSV)	Configurable (0–3)	Start_Address + 0x38
15	System tick timer (SysTick)	Configurable (0–3)	Start_Address + 0x3C
16	External interrupt (IRQ0)	Configurable (0–3)	Start_Address + 0x40
…	…	…	…
23	External interrupt (IRQ7)	Configurable (0–3)	Start_Address + 0x5C
24	Internal (SW only) interrupt (IRQ8)	Configurable (0-3)	Start_Address + 0x60
…	…	…	…
31	Internal (SW only) interrupt (IRQ15)	Configurable (0-3)	Start_Address + 0x7C

[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

Exception #	Exception	Exception Priority	Vector Address
–	Initial stack pointer value	–	Start_Address = 0x0000 or CM4/CM7_0/CM7_1_SCS_VTOR[1]
1	Reset	–3, highest priority	Start _Address + 0x0004
2	Non-maskable Interrupt (NMI)	–2	Start _Address + 0x0008
3	Hard fault	–1	Start _Address + 0x000C
4	Memory management fault	Configurable (0–7)	Start _Address + 0x0010
5	Bus fault	Configurable (0–7)	Start _Address + 0x0014
6	Usage fault	Configurable (0–7)	Start _Address + 0x0018
7–10	Reserved	–	–
11	Supervisory call (SVCall)	Configurable (0–7)	Start _Address + 0x002C
12–13	Reserved	–	–
14	Pend supervisory (PendSV)	Configurable (0–7)	Start _Address + 0x0038
15	System tick timer (SysTick)	Configurable (0–7)	Start _Address + 0x003C
16	External interrupt (IRQ0)	Configurable (0–7)	Start _Address + 0x0040
…	…	…	…
23	External interrupt (IRQ7)	Configurable (0–7)	Start _Address + 0x005C
24	Internal (SW only) interrupt (IRQ8)	Configurable (0–7)	Start _Address + 0x60
…	…	…	…
31	Internal (SW only) interrupt (IRQ15)	Configurable (0-7)	Start _Address + 0x7C

[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

CPU interrupt priority

- 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

Priority Level (High → Low)

-2	**:**	Fault	NMI exception, protection violation
0	**:**	GPIO	Ignition detection, wakeup use
1	**:**	CAN	Communication with other ECU
2	**:**	LIN	Communication with other ECU
3	**:**	SCB	Communication with external IC
4	**:**	TCPWM	Task management
5	**:**	Event Generator	Wakeup use
6	**:**	ADC	Sensor data acquisition
7	**:**	RTC	Real-time clock alarm
Main routine			

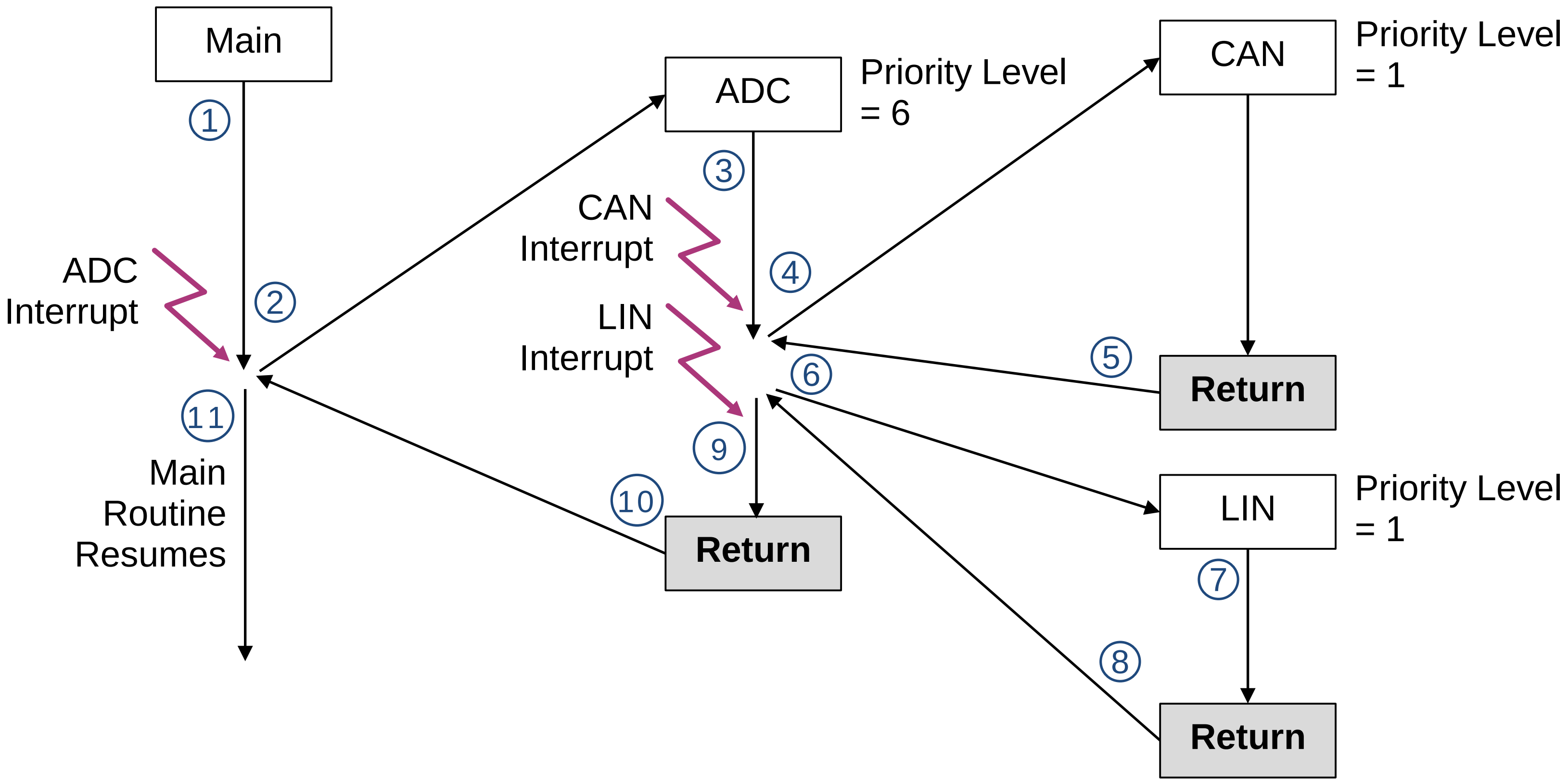
Hint Bar

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



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

Hint Bar

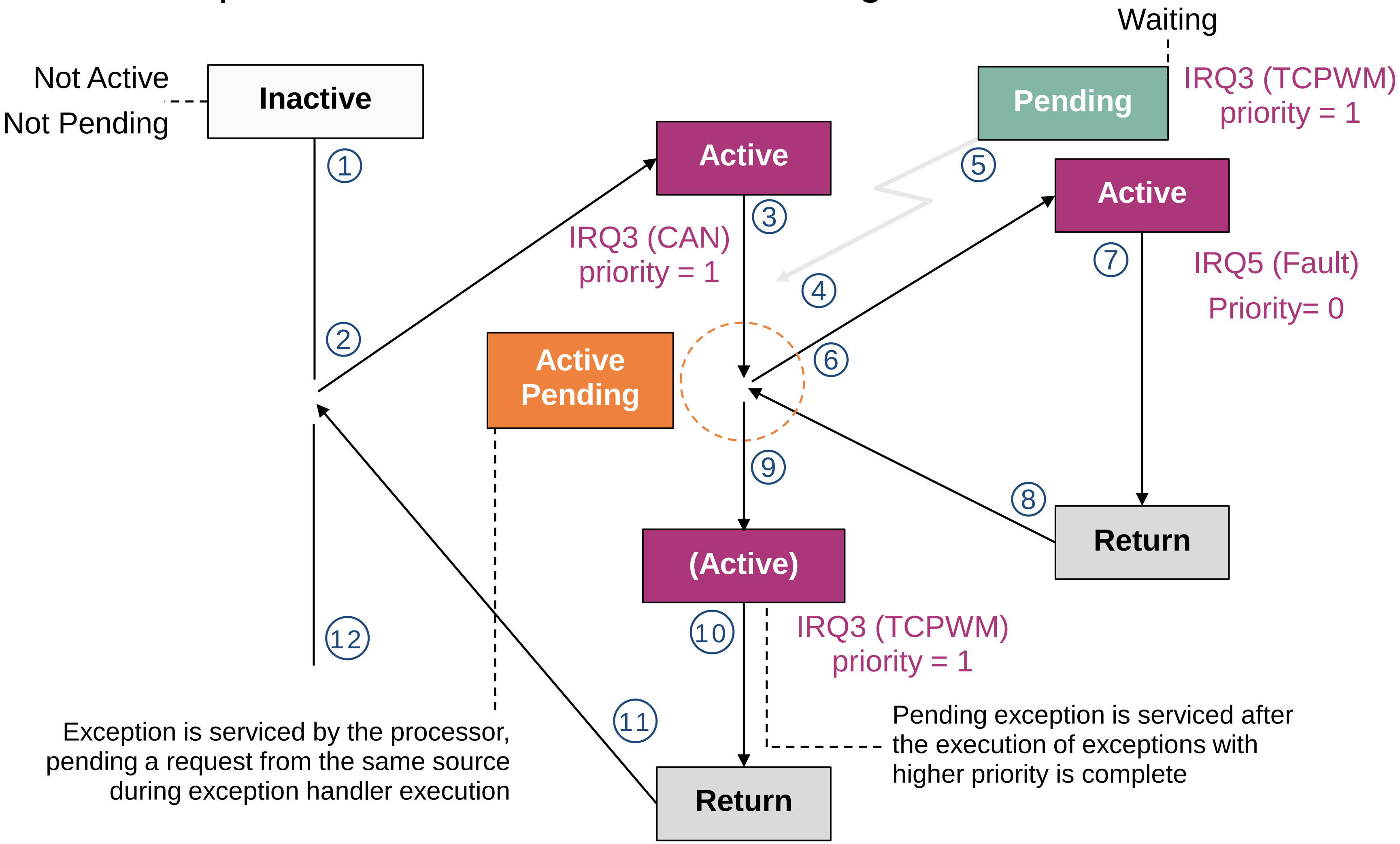
Review TRM section 12.7 for additional details

Interrupt Set-Enable Register (ISER)

Interrupt Clear-Enable Register (ICER)

Exception states

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



Hint Bar

Review TRM section 12.8 for additional details.

Appendix

Number of interrupts

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

Series	Maximum number of system interrupts	Maximum number of wakeup interrupts
CYT2B6	228	38
CYT2B7	353	45
CYT2B9	383	45
CYT2BL	383	45
CYT3BB/CYT4BB	443	51
CYT4BF	567	51
CYT2CL	786	44
CYT3DL	795	34
CYT4DN	795	38

Hint Bar

Refer to each device datasheet for the list of system interrupts

Part of your life. Part of tomorrow.

Revision history

Revision	ECN	Submission date	Description of change
**	6154903	04/29/2018	Initial release
*A	6396762	11/29/2018	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
*B	6612968	07/04/2019	Updated note descriptions for pages 3, 4, 5, 23, and 24 Updated pages 2, 5, 14, and 23
*C	7042917	12/11/2020	Updated page 2, 3, 9 and 23
*D	7400452	10/15/2021	Updated page 1 to 5, 11, 14, 23