PIC32CM6408PL10028/032/048/064 Data Sheet

PIC32CM6408PL10028/032/048/064

Product Page Links

5 Volt, 64 KB Flash, 8 KB SRAM with Advanced Analog

Operating Conditions

- 1.8–5.5V, -40°C to +85°C, DC to 24 MHz
- 1.8–5.5V, -40°C to +125°C, DC to 24 MHz

Qualification

- AEC-Q100 Grade 1 (-40°C to +125°C)

Core: Arm® Cortex®-M0+ CPU Running at up to 24 MHz

- Single-cycle hardware multiplier

Memories

- 64 KB in-system self-programmable Flash
- 8 KB SRAM

System

- Power-On Reset (POR) and Brown-Out Detection (BOD)
- Internal and external clock options
- External Interrupt Controller (EIC)
- One non-maskable interrupt

Low Power

- Idle and Standby sleep modes
- SleepWalking peripherals

Input/Output (I/O)

- Up to 55 programmable I/O pins
- Up to 16 external interrupts
- Multi-Voltage I/O (MVIO)

Clock System

- Main Clock Controller (MCLK)
- Generic Clock Controller (GCLK)
- Clock sources:
  - Internal 1-24 MHz High Frequency Oscillator (OSCHF)
  - Internal 32.768 kHz Oscillator (OSC32K)
  - External 32.768 kHz Oscillator input (XOSC32K)

Debugger Development Support

- In-circuit and in-application programming
- Two-Wire Serial Wire Debug Port Interface
- Four hardware breakpoints and one data watchpoint
- Micro Trace Buffer (MTB) for instruction trace in SRAM
- Programming and Debugging Interface Disable (PDID) functionality

Advanced Analog

- One 12-bit, 800 ksps Analog-to-Digital Converter (ADC)
  - Differential and single-ended input
  - Automatic offset and gain error compensation
  - Oversampling and decimation in hardware to support up to 16-bit resolution
- Two Analog Comparators (AC) with window compare function
- Peripheral Touch Controller (PTC) with up to 29 self-capacitance touch channels
- Internal and external voltage reference options

Peripherals

- 2-channel Direct Memory Access Controller (DMAC) with CRC Generator
- 4-channel Event System
- Four 16-bit Timer/Counters:
  - Three 16-bit basic timers (TC0-2)
  - One 16-bit Timer/Counter for Control with four PWM channels (TCC0)
- 32-bit Real Time Counter (RTC) with clock/calendar function
- Watchdog Timer (WDT)
- Two Serial Communication Interfaces (SERCOM), configurable to operate as:

  - USART with full-duplex and single-wire half-duplex configuration
  - ISO7816 UART
  - I^{2}C up to 1 MHz (SERCOM0)
  - SPI
  - LIN Host/Client
  - RS-485
- Configurable Custom Logic (CCL) with four Look-Up Tables (LUT)
- Integrated Temperature Sensor

Available Packages

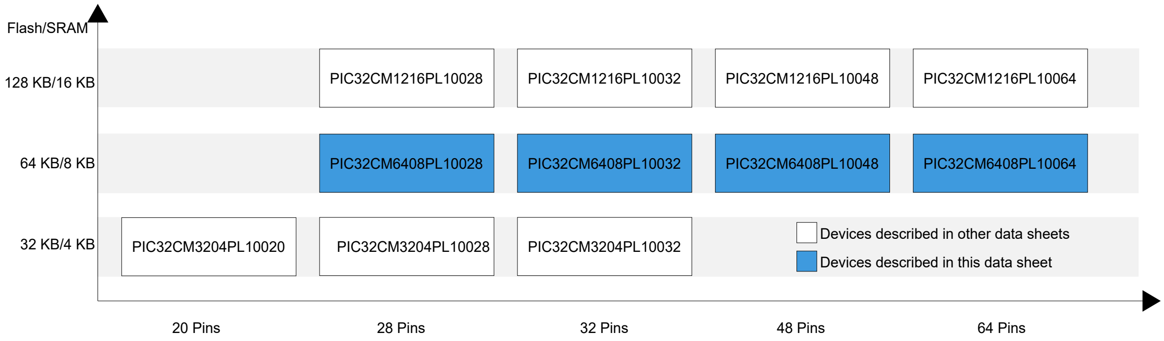
- 28-pin VQFN with Wettable Flanks (WF), SSOP, and SPDIP
- 32-pin TQFP and VQFN WF
- 48-pin TQFP and VQFN WF
- 64-pin TQFP and VQFN WF

Family Overview

The figure below shows the PIC32CM PL10 family devices, illustrating pin count variants and memory sizes:

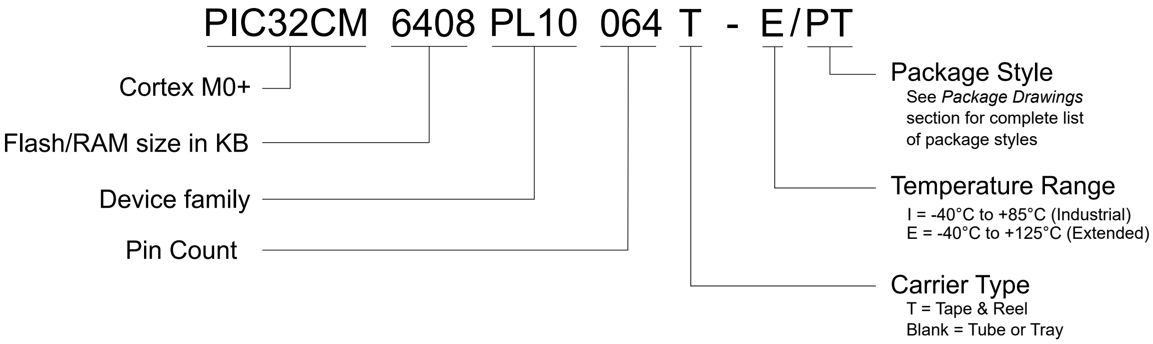
- Vertical migration is possible without code modification, as these devices are fully pin- and feature-compatible
- Horizontal migration to the left reduces the pin count and, therefore, the available features

Figure 1. PIC32CM PL10 family Overview



The name of a device in the PIC32CM PL10 family is decoded as follows:

Figure 2. PIC32CM PL10 family Family Device Designations


PIC32CM 6408 PL10 064 T - E / PT
Cortex M0+
Flash/RAM size in KB
Device family
Pin Count
Package Style
See *Package Drawings* section for complete list of package styles
Temperature Range
I = -40°C to +85°C (Industrial)
E = -40°C to +125°C (Extended)
Carrier Type
T = Tape & Reel
Blank = Tube or Tray

Memory Overview

Table 1. Memory Overview

Devices	CM3204-020 CM3204-028 CM3204-032	CM6408-028 CM6408-032 CM6408-048 CM6408-064	CM1216-028 CM1216-032 CM1216-048 CM1216-064
Flash memory	32 KB	64 KB	128 KB
SRAM	4 KB	8 KB	16 KB
Boot ROM (BOOTROM)	8 KB	8 KB	8 KB
User row (USERROW)	512B	512B	512B

Peripheral Overview

Table 2. Peripheral Overview

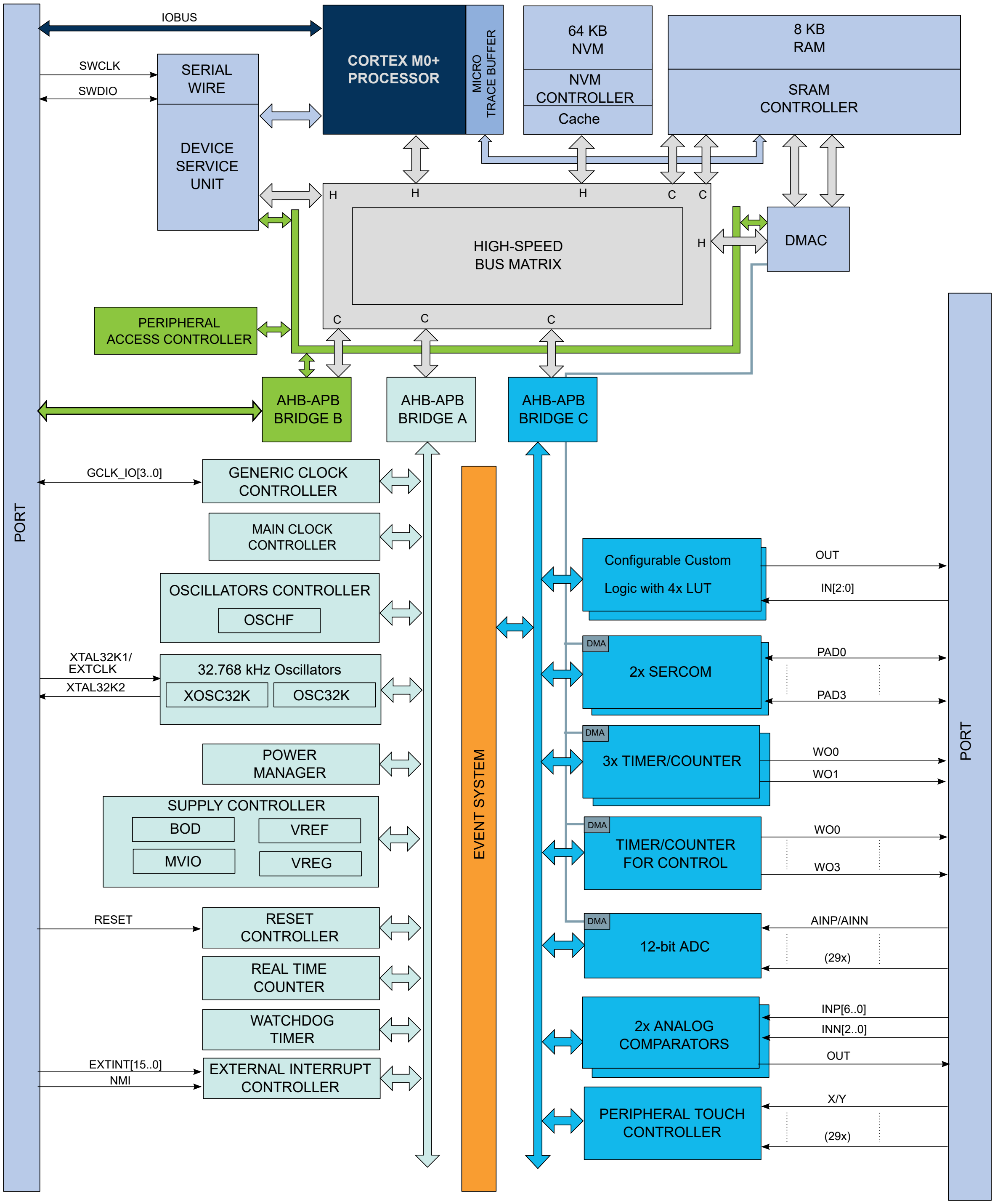
Feature	CM3204-020	CM3204-028 CM6408-028 CM1216-028	CM3204-032 CM6408-032 CM1216-032	CM6408-048 CM1216-048	CM6408-064 CM1216-064
Pins	20	28	32	48	64
Max. frequency (MHz)	24	24	24	24	24
Generic Clock Controller (GCLK)	4	4	4	4	4
External 32.768 kHz Oscillator	1	1	1	1	1
Timer/Counter (TC)	3	3	3	3	3
Timer/Counter for Control Applications (TCC)	1	1	1	1	1
Real-Time Counter (RTC)	1	1	1	1	1
Serial Communication Interface (SERCOM)	2	2	2	2	2
ADC (channels)	1 (11)	1 (16)	1 (20)	1 (25)	1 (29)
Analog Comparator (AC)	2	2	2	2	2
Peripheral Touch Controller (PTC) and (channels)	1 (11)	1 (16)	1 (20)	1 (25)	1 (29)
Configurable Custom Logic Look-up Table (CCL LUT)	4	4	4	4	4
Watchdog Timer (WDT)	1	1	1	1	1
Direct Memory Access Channels (DMA)	2	2	2	2	2
Event System channels (EVSYS)	4	4	4	4	4
General Purpose I/O[1]	17/16	23/22	27/26	42/41	56/55
Multi-Voltage I/O (MVIO)	3	4	4	8	8
External Interrupts	11	13	15	16	16

Note:

1. The PA30/$\overline{\text{RESET}}$ pin is input only.

Table of Contents

1. Block Diagram



2. Pinout

Each pin is controlled by the I/O Pin Controller (PORT) as a general purpose I/O and can alternatively be assigned to one of the peripheral functions A-P.

The following table describes the peripheral signals multiplexed to the PORT I/O pins for each package.

Refer to the *Schematic Checklist* and the *Electrical Characteristics* chapters for the hardware requirements and other considerations for the pin connections.

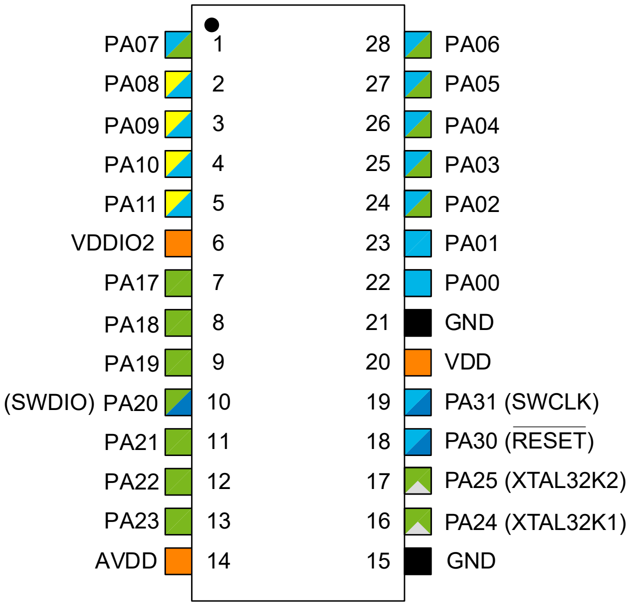
2.1. Configuration Summary

Table 2-1. Configurations and Peripherals of PIC32CM6408PL10

PIC32CM6408PL10				
Peripheral	64-Pin Package	48-Pin Package	32-Pin Package	28-Pin Package
Maximum CPU frequency	24 MHz			
SysTick timer	1			
Serial Wire Debug Interface	Yes			
Oscillators	32.768 kHz crystal oscillator (XOSC32K)			
	32.768 kHz internal oscillator (OSC32K)			
	24 MHz high-accuracy internal oscillator (OSCHF)			
Generic Clock - GCLK	4			
DMA channels	2			
Event System channels	4			
I/O Pins	55	41	26	22
External Interrupt lines	16 and one non-maskable interrupt (NMI)			
Serial Communication Interfaces - SERCOM/of which I^{2}C is supported	2 / 1	2 / 1	2 / 1	2 / 1
Timer Counter - TC	3			
Waveform outputs/Capture inputs channels per TC instance	2			
TC Maximum and Minimum Capture	Yes			
Timer Counter for Control - TCC (number of waveform output channels)	1(4)			
Configurable Custom Logic - CCL (number of LUTs)	1(4)			
Analog-to-Digital Converter - ADC (channels)	1(29)	1(25)	1(20)	1(16)
Analog Comparators - AC	2			
Real-Time Counter - RTC	1			
RTC alarms	1			
RTC compare values	One 32-bit value or two 16-bit values			
Watchdog Timer - WDT	Yes			
Memories CRC32 computation in DMAC	Yes (SRAM, Flash)			
Brown-Out Detection - BOD	VDD, VDDCORE			
Peripheral Touch Controller - PTC (channels)	1(29)	1(25)	1(20)	1(16)

2.2. Pinout Diagrams

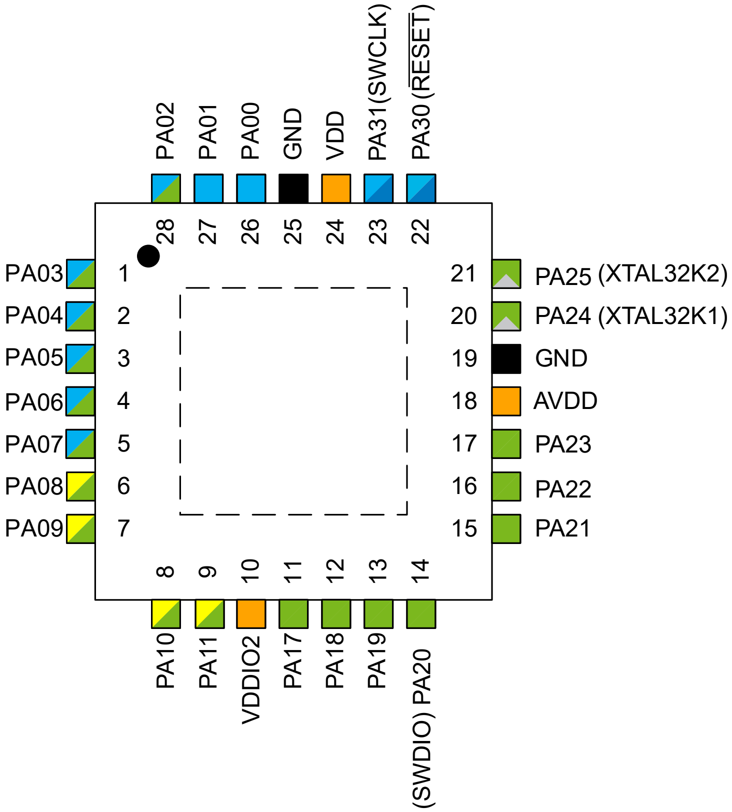
2.2.1. 28-pin SSOP and SPDIP



Note: AVDD is internally connected to VDD (not separate power domains).

Power	Functionality
🟧 Power Supply	◩ Programming/Debug
⬛ Ground	◩ Clock/Crystal
◩ Pin on VDD Power Domain	◩ Digital Function Only
◩ Pin on AVDD Power Domain	◩ Analog Function
◩ Pin on VDDIO2 Power Domain	

2.2.2. 28-pin VQFN



Note: AVDD is internally connected to VDD (not separate power domains).

Power

🟧 Power Supply

⬛ Ground

◩ Pin on VDD Power Domain

◩ Pin on AVDD Power Domain

◩ Pin on VDDIO2 Power Domain

Functionality

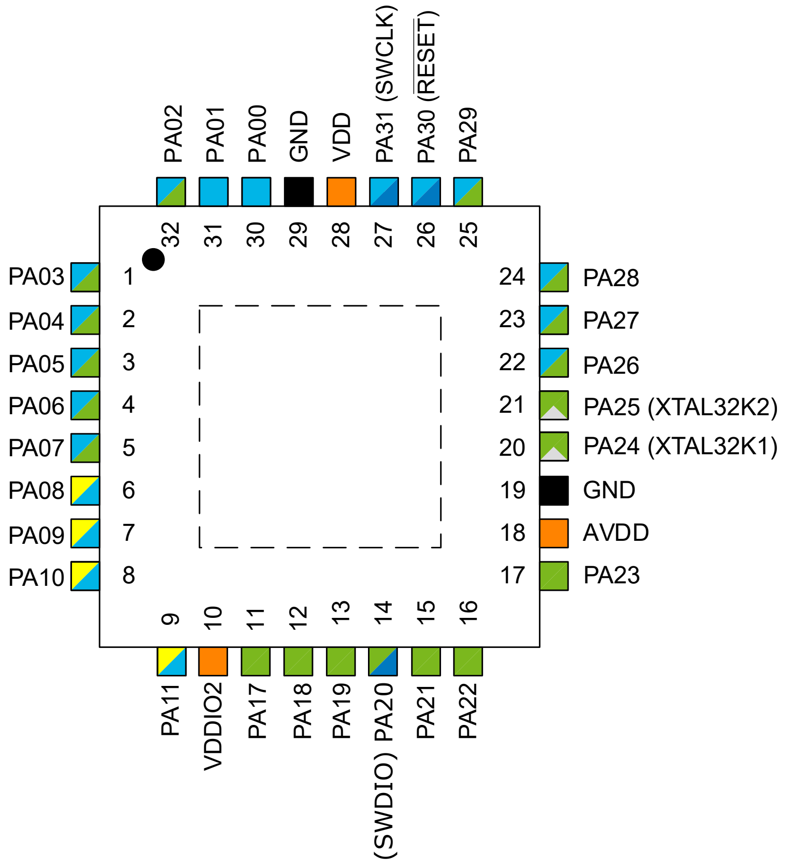
◩ Programming/Debug

◩ Clock/Crystal

◩ Digital Function Only

◩ Analog Function

2.2.3. 32-pin VQFN and TQFP



Note: AVDD is internally connected to VDD (not separate power domains).

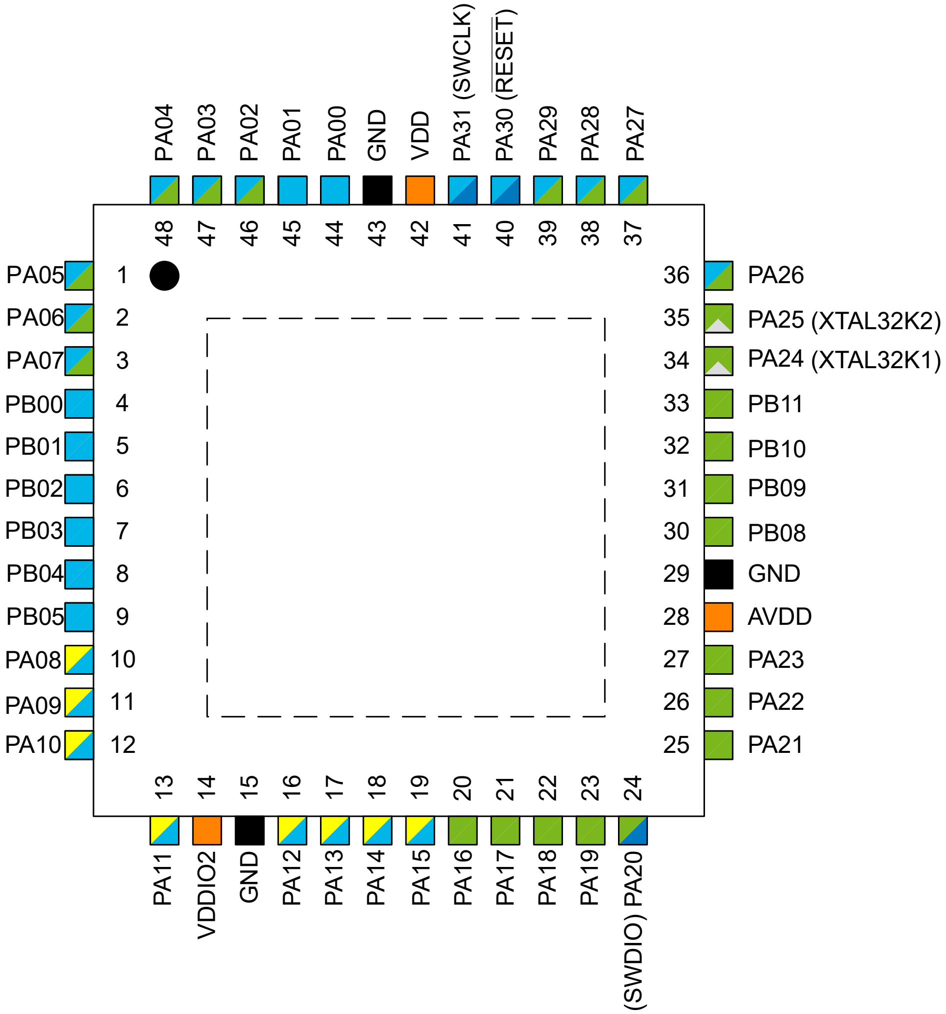
Power

- 🟧 Power Supply
- ⬛ Ground
- ◩ Pin on VDD Power Domain
- ◩ Pin on AVDD Power Domain
- ◩ Pin on VDDIO2 Power Domain

Functionality

- ◪ Programming/Debug
- ◪ Clock/Crystal
- ◪ Digital Function Only
- ◪ Analog Function

2.2.4. 48-pin VQFN and TQFP



Note: AVDD is internally connected to VDD (not separate power domains).

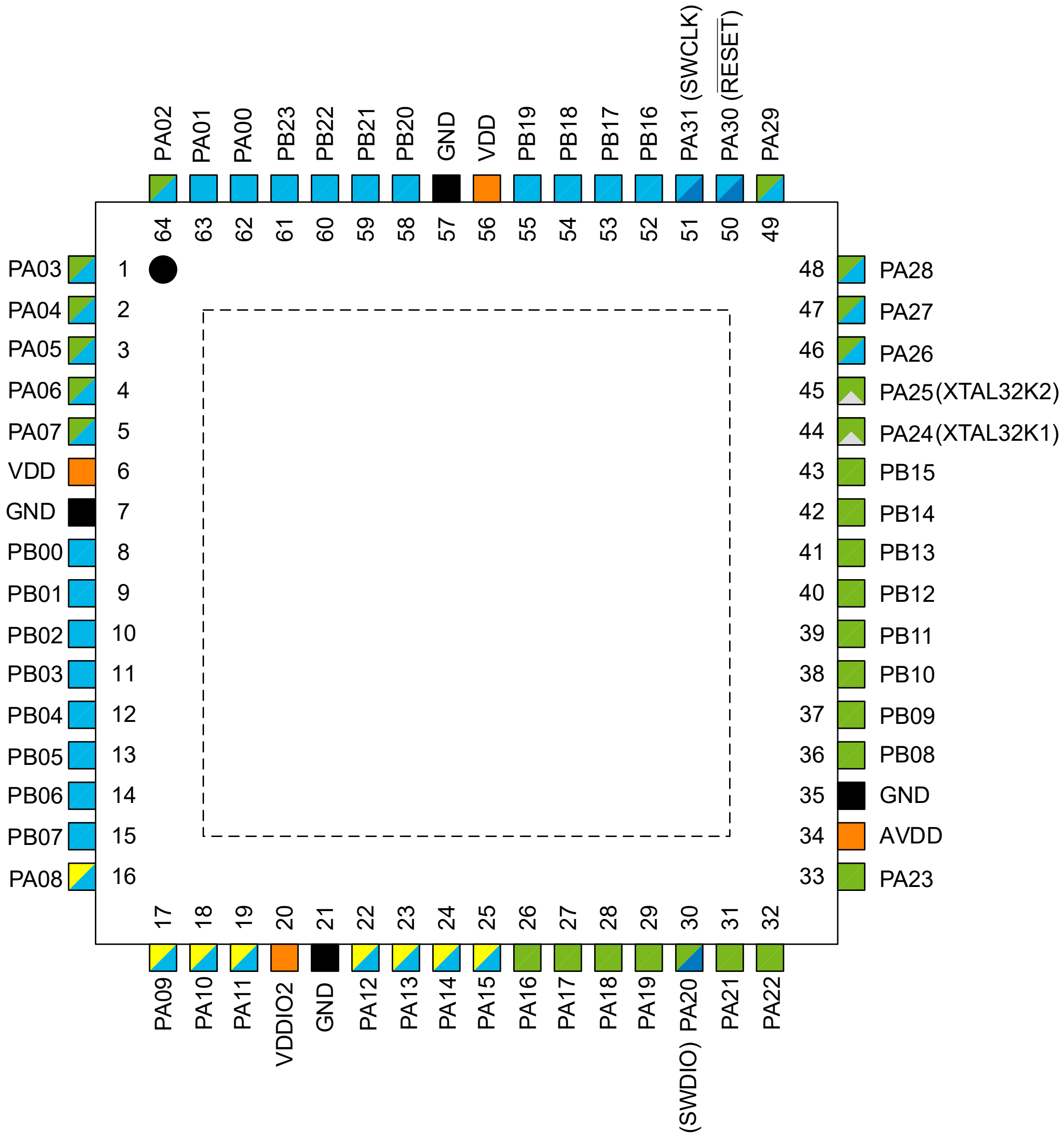
Power

- 🟧 Power Supply
- ■ Ground
- ◩ Pin on VDD Power Domain
- ◩ Pin on AVDD Power Domain
- ◩ Pin on VDDIO2 Power Domain

Functionality

- ◪ Programming/Debug
- ◪ Clock/Crystal
- ◪ Digital Function Only
- ◪ Analog Function

2.2.5. 64-pin VQFN and TQFP



Note: AVDD is internally connected to VDD (not separate power domains).

Power

- 🟧 Power Supply
- ⬛ Ground
- ◩ Pin on VDD Power Domain
- ◩ Pin on AVDD Power Domain
- ◩ Pin on VDDIO2 Power Domain

Functionality

- ◩ Programming/Debug
- ◩ Clock/Crystal
- ◩ Digital Function Only
- ◩ Analog Function

2.3. Pinout and Multiplexing

Package/Pin Number					Pin Name	Special	A	B				C	D	E	F	G	H	I	P
VQFN TQFP			VQFN	SSOP SPDIP			EXTINT	ADC0		PTC	ACn	SERCOMn		TCn	TCC0	SWCLK	CCLn, ACn OUT	GCLKn	PTC
64-pin	48-pin	32-pin	28-pin	28-pin				Pos	Neg	EXTCINT									X/Y
62	44	30	26	22	PA00		EXTINT[0]					0,PAD[0] [2]	1,PAD[0]	0,WO0	WO0		0,IN0	1,IO	
63	45	31	27	23	PA01		NMI					0,PAD[1] [2]	1,PAD[1]	0,WO1	WO1		0,IN1	2,IO	
64	46	32	28	24	PA02		EXTINT[2]	AINP2	AINN2			0,PAD[2]	1,PAD[2]	1,WO0	WO2		0,IN2	3,IO	X2/Y2
1	47	1	1	25	PA03		EXTINT[3]	AINP3	AINN3			0,PAD[3]	1,PAD[3]	1,WO1	WO3		0,OUT	0,IO	X3/Y3
2	48	2	2	26	PA04		EXTINT[4]	AINP4	AINN4			0,PAD[0] [2]						1,IO	X4/Y4
3	1	3	3	27	PA05		EXTINT[5]	AINP5	AINN5			0,PAD[1] [2]					AC1[4]	2,IO	X5/Y5
4	2	4	4	28	PA06		EXTINT[6]	AINP6	AINN6			0,PAD[2]					0,OUT	3, IO	X6/Y6
5	3	5	5	1	PA07		NMI	AINP7	AINN7			0,PAD[3]					AC0[4]	0,IO	X7/Y7
6					VDD														
7					GND														
8	4				PB00		EXTINT[0]						1,PAD[0]		WO0				
9	5				PB01		EXTINT[1]						1,PAD[1]		WO1				
10	6				PB02		EXTINT[2]						1,PAD[2]		WO2				
11	7				PB03		EXTINT[3]						1,PAD[3]		WO3				
12	8				PB04		EXTINT[4]												
13	9				PB05		EXTINT[5]												
14					PB06		EXTINT[6]												
15					PB07		EXTINT[7]												
16	10	6	6	2	PA08	VDDIO2[3]	EXTINT[8]					1,PAD[2]	0,PAD[2]	1,WO0	WO0		1,IN0	1.IO	
17	11	7	7	3	PA09	VDDIO2[3]	EXTINT[9]					1,PAD[3]	0,PAD[3]	1,WO1	WO1		1,IN1	2,IO	
18	12	8	8	4	PA10	VDDIO2[3]	EXTINT[10]					1,PAD[0]	0,PAD[0] [2]	0,WO0	WO2		1,IN2	3,IO	
19	13	9	9	5	PA11	VDDIO2[3]	EXTINT[11]					1,PAD[1]	0,PAD[1] [2]	0,WO1	WO3		1,OUT	1,IO	
20	14	10	10	6	VDDIO2														
21	15				GND														
22	16				PA12	VDDIO2[3]	EXTINT[12]					1,PAD[2]							
23	17				PA13	VDDIO2[3]	EXTINT[13]					1,PAD[3]					AC1[4]		
24	18				PA14	VDDIO2[3]	EXTINT[14]					1,PAD[0]					1,OUT		
25	19				PA15	VDDIO2[3]	EXTINT[15]					1,PAD[1]					AC0[4]		
26	20				PA16		EXTINT[0]	AINP16	AINN16		0,N1 1,N1				WO0		2,IN0		X16/Y16
27	21	11	11	7	PA17		EXTINT[1]	AINP17	AINN17						WO1		2,IN1		X17/Y17
28	22	12	12	8	PA18		EXTINT[2]	AINP18	AINN18		0,P0 1,P0				WO2		2,IN2		X18/Y18
29	23	13	13	9	PA19		EXTINT[3]	AINP19	AINN19		0,N0 1,P1				WO3		2,OUT		X19/Y19
30	24	14	14	10	PA20	SWDIO	EXTINT[4]	AINP20	AINN20		0,P5 1,P2	1,PAD[0]		2,WO0				2,IO	X20/Y20
31	25	15	15	11	PA21		EXTINT[5]	AINP21	AINN21	EXTCINT0	0,P6 1,N0/P5	1,PAD[1]		2,WO1				3,IO	X21/Y21
32	26	16	16	12	PA22		EXTINT[6]	AINP22	AINN22	EXTCINT1	0,P3 1,P3	1,PAD[2]					2,OUT	1, IO	X22/Y22
33	27	17	17	13	PA23	VREFA	EXTINT[7]	AINP23	AINN23		0,N2 1,N2	1,PAD[3]						0,IO	X23/Y23
34	28	18	18	14	AVDD														
35	29	19	19	15	AGND														
36	30				PB08		EXTINT[8]	AINP40	AINN40		0,P1				WO0				X40/Y40
37	31				PB09		EXTINT[9]	AINP41	AINN41						WO1				X41/Y41
38	32				PB10		EXTINT[10]	AINP42	AINN42		0,P2				WO2				X42/Y42
39	33				PB11		EXTINT[11]	AINP43	AINN43						WO3				X43/Y43
40					PB12		EXTINT[12]	AINP44	AINN44										X44/Y44
41					PB13		EXTINT[13]	AINP45	AINN45										X45/Y45
42					PB14		EXTINT[14]	AINP46	AINN46										X46/Y46
43					PB15		EXTINT[15]	AINP47	AINN47										X47/Y47
44	34	20	20	16	PA24	XTAL32K 1	EXTINT[8]	AINP24	AINN24			1,PAD[2]			WO0		3,IN0	1,IO	X24/Y24
45	35	21	21	17	PA25	XTAL32K 2	EXTINT[9]	AINP25	AINN25			1,PAD[3]			WO1		3,IN1	2,IO	X25/Y25
46	36	22			PA26		EXTINT[10]	AINP26	AINN26			1,PAD[0]			WO2		3,IN2	3,IO	X26/Y26
47	37	23			PA27		EXTINT[11]	AINP27	AINN27			1,PAD[1]			WO3		3,OUT		X27/Y27
48	38	24			PA28		EXTINT[12]	AINP28	AINN28										X28/Y28
49	39	25			PA29		EXTINT[13]	AINP29	AINN29									1,IO	X29/Y29
50	40	26	22	18	PA30	RESET	NMI												
51	41	27	23	19	PA31	SWCLK	EXTINT[15]	AINP31	AINN31			1,PAD[0]	0,PAD[2]			SWCLK		0,IO	X31/Y31
52					PB16		EXTINT[0]								WO0				
53					PB17		EXTINT[1]								WO1				
54					PB18		EXTINT[2]								WO2				
55					PB19		EXTINT[3]								WO3				
56	42	28	24	20	VDD														
57	43	29	25	21	GND														
58					PB20		EXTINT[4]							2,WO0					
59					PB21		EXTINT[5]							2,WO1					
60					PB22		EXTINT[6]												
61					PB23		EXTINT[7]												

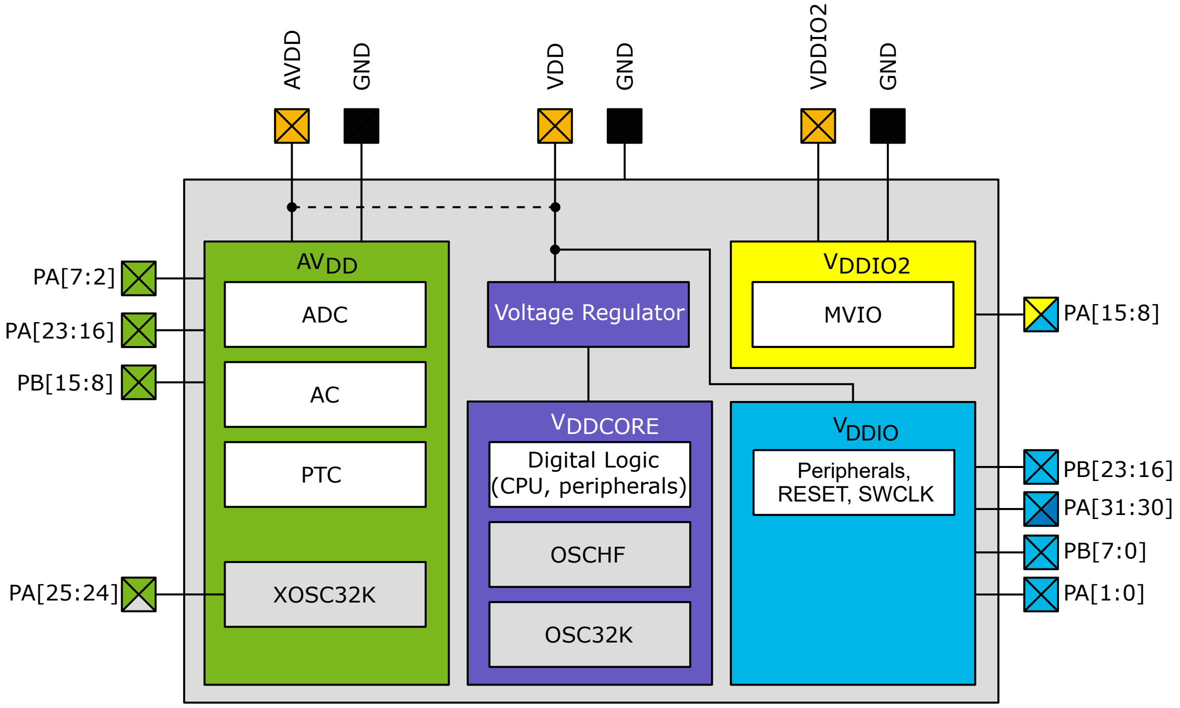
1. To select pin positions, refer to the *PORT* chapter.
2. This pin supports I^{2}C with SERCOM0.

3. This pin is on the VDDIO2 power domain. For usage and properties, refer to the MVIO description in the *SUPC - Supply Controller* and the *Electrical Characteristics* sections.
4. The output of the Analog Comparator to the pin can be used as an input to the CCL. Refer also to the *CCL - Configurable Custom Logic* section.

3. Power Supply and Start-Up Considerations

3.1. Power Domain Overview

Figure 3-1. Power Domain Overview



3.2. Power Supply Considerations

3.2.1. Power Supplies

The PIC32CM6408PL10 has the following power supply pins:

- VDD: Powers the I/O lines (VDDIO) and the VREG
- AVDD: Powers the analog peripherals: AC, ADC, and PTC
- VDDIO2: Powers the VDDIO2 I/O lines

The same voltage must be applied to both the VDD and AVDD pins. This common voltage is referred to as V_{DD} in the data sheet.

The ground pins, GND, are common to all supplies.

For decoupling recommendations for the different power supplies, refer to the *Schematic Checklist* chapter.

3.2.2. Voltage Regulator

The PIC32CM6408PL10 voltage regulators ensure correct supply voltage to the CPU, digital peripherals, internal oscillators, and other digital logic. They have the following modes:

- Normal mode: The CPU and peripherals are running
- Low Power (LP) mode: This mode is used when the device is in Standby sleep mode

3.2.3. Power-Up Sequence

If VDDIO2 is configured in single supply mode, VDD and VDDIO2 must have the same supply sequence. It is recommended to connect them together outside of the device. See also the *SUPC - Supply Controller* chapter. Observe the minimum and maximum rise rates for V_{DD}, as described in the *Electrical Characteristics* chapter.

3.3. Start-Up

After a successful power-up, the device behavior is controlled by the Power Manager (PM) and the Reset Controller (RSTC).

3.3.1. Starting of Clocks

After power-up, the device is set to its initial state and kept in reset until the power has stabilized throughout the device. Once the power has stabilized, the Boot ROM will initialize the device and select the internal OSCHF divided to a 4 MHz clock signal as the default clock for the device. This is also used as the clock source for Generic Clock Generator 0 (GCLK0). In turn, GCLK0 provides the main clock, GCLK_MAIN, which is used by the Main Clock (MCLK) peripheral. Any further configuration of the clock system is done by the application.

Some synchronous system clocks are active after startup, allowing software execution.

Refer to the Clock Mask Register in the *MCLK - Main Clock* section for the list of default peripheral clocks that are running.

3.3.2. I/O Pins

After power-up, the I/O pins are tri-stated except for RESETn, SWDIO and SWCLK, which are pull-up enabled and configured as input.

3.3.3. Fetching of Initial Instructions

After the device is released from the initial reset, the CPU starts fetching the Program Counter (PC) and Stack Pointer (SP) values from the reset address, which is 0x00000000. This address points to the first executable address in the internal Flash. The code read from the internal Flash is free to configure the clock system and clock sources. Refer to the Arm Architecture Reference Manual for additional information on CPU startup (http://www.arm.com).

3.4. Power-On Reset (POR) and Brown-Out Detectors (BOD)

The following features are used to monitor, warn, and reset the device:

- POR: Power-On Reset on V_{DD} and V_{DDIO2}, handled by the Reset Controller (RSTC) peripheral. The POR is always activated and monitors the voltages at startup and during all sleep modes:
  - If V_{DD} goes below the threshold voltage, the entire device is reset
  - If V_{DDIO2} goes below the threshold voltage, all I/Os supplied by V_{DDIO2} are reset
- BODVDD: Brown-Out Detector on V_{DD}, controlled by the Supply Controller (SUPC) peripheral
- BODVDDIO2: Brown-Out Detector on V_{DDIO2}, controlled by the SUPC peripheral

4. CPU and Architecture

4.1. Cortex M0+ Processor

The PIC32CM6408PL10 implements the Arm Cortex-M0+ processor, based on the ARMv6 Architecture and Thumb®-2 ISA. The Cortex M0+ is fully instruction set compatible with its predecessor, the Cortex-M0 core, and upward compatible with the Cortex-M3 and M4 cores. The implemented Arm Cortex-M0+ is revision r0p1. For more information refer to www.arm.com.

4.1.1. Cortex M0+ Configuration

Table 4-1. Cortex M0+ Configuration

Features	PIC32CM6408PL10 Configuration
Interrupt lines	16
Data endianness	Little-endian
SysTick timer	Present
Number of watchpoint comparators	1
Number of breakpoint comparators	4
Halting debug support	Present
Multiplier	Fast (single cycle)
Single-cycle I/O port	Present
Wake-up interrupt controller	Not supported
Vector Table Offset Register	Present
Unprivileged/Privileged support	All accesses are privileged
Memory Protection Unit	None
Reset all registers	Absent
Instruction fetch width	32-bit

The Arm Cortex-M0+ core has the following bus interfaces:

- A single 32-bit AMBA-3 AHB-Lite system interface provides connections to peripherals and all system memory, including Flash and RAM
- A single 32-bit I/O port bus interfaces to the PORT with 1-cycle loads and stores

4.1.2. Cortex-M0+ Peripherals

- System Control Space (SCS)
  - The processor provides debug through registers in the SCS. Refer to the *Cortex-M0+ Technical Reference Manual* for details (www.arm.com).
- Nested Vectored Interrupt Controller (NVIC)
  - External interrupt signals connect to the NVIC, which prioritizes the interrupts. Software can set the priority of each interrupt. The NVIC and the Cortex-M0+ processor core are closely coupled, providing low-latency interrupt handling and efficient processing of late-arriving interrupts. Refer to NVIC – Nested Vectored Interrupt Controller and *Cortex-M0+ Technical Reference Manual* for details (www.arm.com).
- System Timer (SysTick)
  - The System Timer is a 24-bit timer clocked by CLK_CPU that extends the functionality of both the processor and the NVIC. Refer to *Cortex-M0+ Technical Reference Manual* for details (www.arm.com). When the SysTick Overflow Interrupt is enabled, the RAM Back Bias Control must be disabled (PM->STDBYCFG.bit.BBIASHS = 0) before entering Standby sleep mode.
- System Control Block (SCB)

  - The System Control Block provides system implementation information and system control. This includes configuration, control, and reporting of system exceptions. Refer to *Cortex-M0+ Devices Generic User Guide* for details (www.arm.com).
- Micro Trace Buffer (MTB)
  - The CoreSight MTB-M0+ (MTB) provides a simple execution trace capability to the Cortex-M0+ processor. Refer to the section Micro Trace Buffer and the *CoreSight MTB-M0+ Technical Reference Manual* for details (www.arm.com).

4.1.3. Cortex-M0+ Address Map

Table 4-2. Cortex-M0+ Address Map

Address	Peripheral
0xE000E000	System Control Space (SCS)
0xE000E010	System Timer (SysTick)
0xE000E100	Nested Vectored Interrupt Controller (NVIC)
0xE000ED00	System Control Block (SCB)
0x41008000	Micro Trace Buffer (MTB)

4.1.4. I/O Interface

The AMBA® AHB-Lite™ interface offers direct access from the CPU to the PORT registers.

Accesses to the AHB-Lite™ and the single cycle I/O interfaces can be made concurrently: The Cortex-M0+ processor can fetch the next instructions while accessing the I/Os, sustaining single cycle I/O accesses for as long as needed.

4.2. NVIC – Nested Vectored Interrupt Controller

4.2.1. Overview

The Nested Vectored Interrupt Controller (NVIC) in the PIC32CM6408PL10 supports 16 interrupt lines with four different priority levels. For more details, refer to the Cortex-M0+ Technical Reference Manual (www.arm.com).

4.2.2. Interrupt Line Mapping

Each of the interrupt lines is connected to one peripheral instance, as shown in the table below. Each peripheral can have one or more interrupt flags, which are located in the peripheral's Interrupt Flag Status and Clear (INTFLAG) register.

The interrupt flag is set when the interrupt condition occurs. Each interrupt in the peripheral can be individually enabled by writing a one to the corresponding bit in the peripheral's Interrupt Enable Set (INTENSET) register, and disabled by writing a one to the corresponding bit in the peripheral's Interrupt Enable Clear (INTENCLR) register.

An interrupt request is generated by the peripheral when the interrupt flag is set and the corresponding interrupt is enabled.

The interrupt requests for one peripheral are ORed together at the system level, generating one interrupt request for each peripheral. An interrupt request will set the corresponding interrupt pending bit in the NVIC interrupt pending registers (SETPEND/CLRPEND bits in ISPR/ICPR).

For the NVIC to activate the interrupt, it must be enabled in the NVIC interrupt enable register (SETENA/CLRENA bits in ISER/ICER). The NVIC interrupt priority registers, IPR0–IPR7, provide a priority field for each interrupt.

Table 4-3. Interrupt Line Mapping

Vector Number	Interrupt Vector	Interrupt Source	Description
NMI	NMI	NMI	Non-Maskable Interrupt from the EIC
0	SYSTEM	CKRDY	Clock Ready interrupt from MCLK
		OSCHFRDY	OSCHF is ready interrupt from OSCCTRL
		XOSC32KRDY	XOSC32K ready interrupt from OSCCTRL
		CLKFAIL	XOSC32K has failed interrupt from OSCCTRL
		OSC32KRDY	OSC32K ready interrupt from OSCCTRL
		VDDIO2LPMPOR	V_{DDIO2} Low-Power Mode POR interrupt from SUPC
		VDDIO2OK	V_{DDIO2} OK interrupt from SUPC
		VLM	Voltage Level Monitor interrupt from SUPC
		BODVDDRDY	V_{DD} Brown-Out interrupt Detector Ready interrupt from SUPC
		FLASH	Access Error is detected by client Flash
		HSRAMCM0P	Access Error is detected by client SRAMCM0P
		HSRAMDSU	Access Error is detected by client SRAMDSU
		APBB	Access Error is detected by client APBB
		APBA	Access Error is detected by client APBA
		APBC	Access Error is detected by client APBC
		SRAMDMAC	Access Error is detected by client SRAMDMAC
		BROM	Access Error is detected by client BROM
		PAC	Peripheral Access Error occurs while accessing PAC
		PM	Peripheral Access Error occurs while accessing PM
		MCLK	Peripheral Access Error occurs while accessing MCLK
		RSTC	Peripheral Access Error occurs while accessing RSTC
		OSCCTRL	Peripheral Access Error occurs while accessing OSCCTRL
		SUPC	Peripheral Access Error occurs while accessing SUPC
		GCLK	Peripheral Access Error occurs while accessing GCLK
		WDT	Peripheral Access Error occurs while accessing WDT
		RTC	Peripheral Access Error occurs while accessing RTC
		EIC	Peripheral Access Error occurs while accessing EIC
		PORT	Peripheral Access Error occurs while accessing PORT
		DSU	Peripheral Access Error occurs while accessing DSU
		NVMCTRL	Peripheral Access Error occurs while accessing NVMCTRL
		DMAC	Peripheral Access Error occurs while accessing DMAC
		MTB	Peripheral Access Error occurs while accessing MTB
		HMATRIXHS	Peripheral Access Error occurs while accessing HMATRIXHS
		EVSYS	Peripheral Access Error occurs while accessing EVSYS
		SERCOMn	Peripheral Access Error occurs while accessing SERCOMn
		TCn	Peripheral Access Error occurs while accessing TCn
		TCCn	Peripheral Access Error occurs while accessing TCCn
		ADC0	Peripheral Access Error occurs while accessing ADC0
		AC	Peripheral Access Error occurs while accessing AC
		CCL	Peripheral Access Error occurs while accessing CCL
		PTC	Peripheral Access Error occurs while accessing PTC
		SYSCTRL	Peripheral Access Error occurs while accessing SYSCTRL
1	WDT	EW	Early Warning interrupt from WDT

Table 4-3. Interrupt Line Mapping (continued)

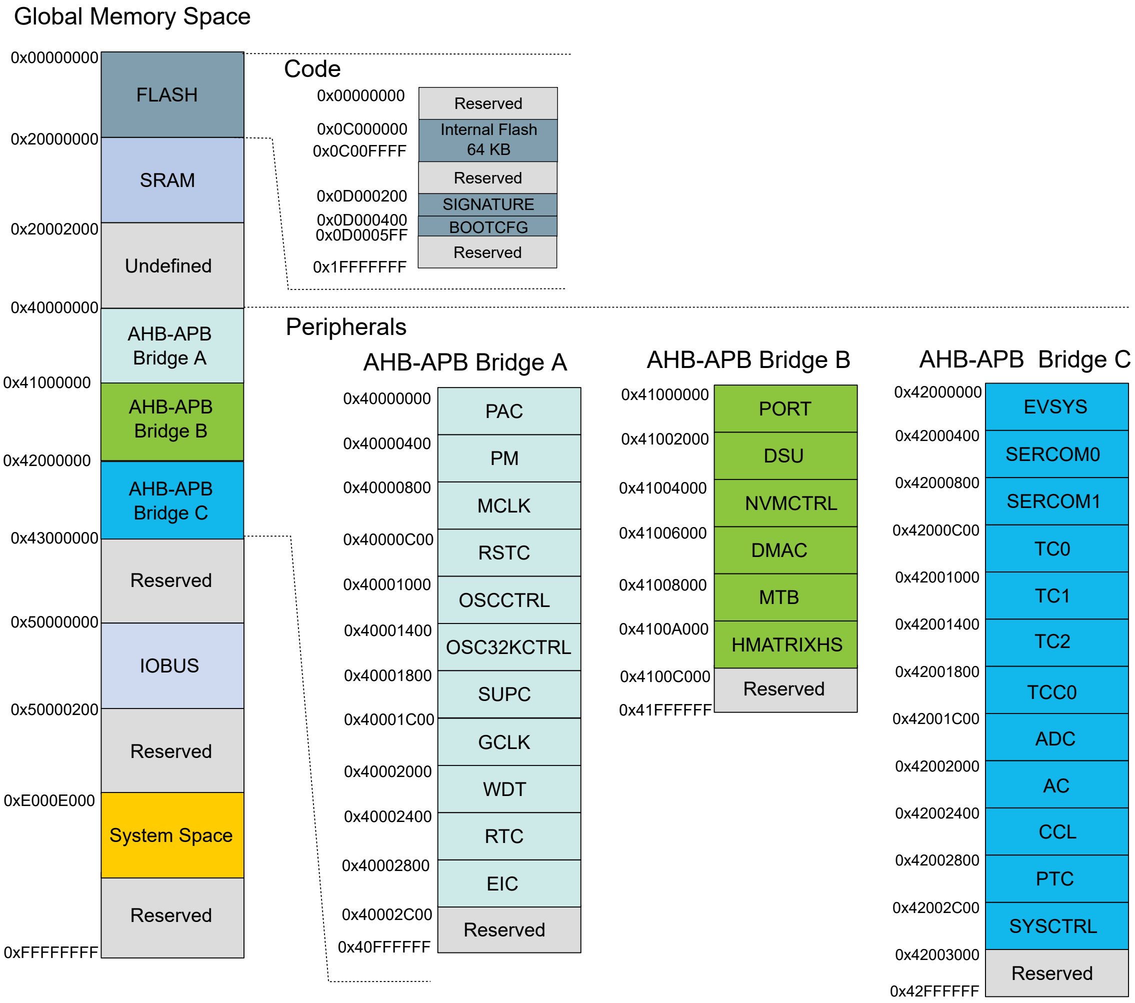
Vector Number	Interrupt Vector	Interrupt Source	Description
2	RTC	OVF	Overflow interrupt from RTC
		CMPn/ALARMn	Compare n interrupt (Mode 0 and 1)/Alarm n interrupt (Mode 2) from RTC
		PERn	Periodic Interval n interrupt from RTC
3	EIC	EXTINTn	External Interrupt n from the EIC
4	NVMCTRL	ERROR	Error interrupt from NVMCTRL
		READY	Flash ready interrupt from NVMCTRL
5	DMAC	TERR	Transfer error interrupt from DMAC
		TCMPL	Transfer complete interrupt from DMAC
		SUSP	Suspend interrupt from DMAC
6	EVSYS	OVRn	Overrun Channel n interrupt from EVSYS
		EVDn	Event Detected Channel n interrupt from EVSYS
7	SERCOM0	DRE	Data Register Empty from SERCOM0
		RXC	Receive Complete from SERCOM0
		TXC	Transmit Complete from SERCOM0
		RXS	Receive Start from SERCOM0
		CTSIC	Clear to Send Input Change from SERCOM0
		RXBRK	Received Break from SERCOM0
		SSL	SPI Select Low from SERCOM0
		DRDY	Data Ready from SERCOM0
		AMATCH	Address Match from SERCOM0
		PREC	Stop Received from SERCOM0
		ERROR	Error from SERCOM0
8	SERCOM1	DRE	Data Register Empty from SERCOM1
		RXC	Receive Complete from SERCOM1
		TXC	Transmit Complete from SERCOM1
		RXS	Receive Start from SERCOM1
		CTSIC	Clear to Send Input Change from SERCOM1
		RXBRK	Received Break from SERCOM1
		SSL	SPI Select Low from SERCOM1
		DRDY	Data Ready from SERCOM1
		AMATCH	Address Match from SERCOM1
		PREC	Stop Received from SERCOM1
		ERROR	Error from SERCOM1
9	TC0	OVF	Overflow interrupt from TC0
		ERR	Error interrupt from TC0
		MCn	Match or Capture Channel n=(0,1) interrupt from TC0
10	TC1	OVF	Overflow interrupt from TC1
		ERR	Error interrupt from TC1
		MCn	Match or Capture Channel n=(0,1) interrupt from TC1
11	TC2	OVF	Overflow interrupt from TC2
		ERR	Error interrupt from TC2
		MCn	Match or Capture Channel n=(0,1) interrupt from TC2

Table 4-3. Interrupt Line Mapping (continued)

Vector Number	Interrupt Vector	Interrupt Source	Description
12	TCC0	OVF	Overflow interrupt from TCC0
		TRG	Retrigger interrupt from TCC0
		CNT	Counter interrupt from TCC0
		ERR	Error interrupt from TCC0
		UFS	Non-Recoverable Update Fault interrupt from TCC0
		DFS	Non-Recoverable Debug Fault interrupt from TCC0
		FAULTx	Recoverable Fault x interrupt from TCC0
		FAULTn	Non-Recoverable Fault n interrupt from TCC0
		MCn	Match or Capture Channel n interrupt from TCC0
13	ADC0	TRIGOVR	Trigger Overrun interrupt from ADC0
		SAMPOVR	Sample Overwrite interrupt from ADC0
		RESOVR	Result Overwrite interrupt from ADC0
		RESRDY	Result Ready interrupt from ADC0
		WCMP	Window Comparator interrupt from ADC0
		SAMPRDY	Sample Ready interrupt from ADC0
14	AC	COMPn	Comparator n interrupt from AC
		WINn	Window n interrupt from AC
15-31	-	-	Reserved

4.3. Product Mapping

Figure 4-1. PIC32CM6408PL10 Product Mapping



The section labelled *System Space* maps the core peripherals of the Cortex-M0+ CPU. Refer to the *Cortex-M0+ Address Map* table in the *CPU and Architecture* chapter.

4.4. HMATRIXHS - High-Speed Bus System

4.4.1. Features

The High-Speed Bus System (HMATRIXHS) has the following features:

- Symmetric crossbar bus switch implementation
- Allows concurrent accesses from different hosts to different clients
- 32-bit data bus
- Operation at a 1-to-1 clock frequency with the bus hosts

4.4.2. Configuration

Figure 4-2. Host/Client Relation High-Speed Bus Matrix

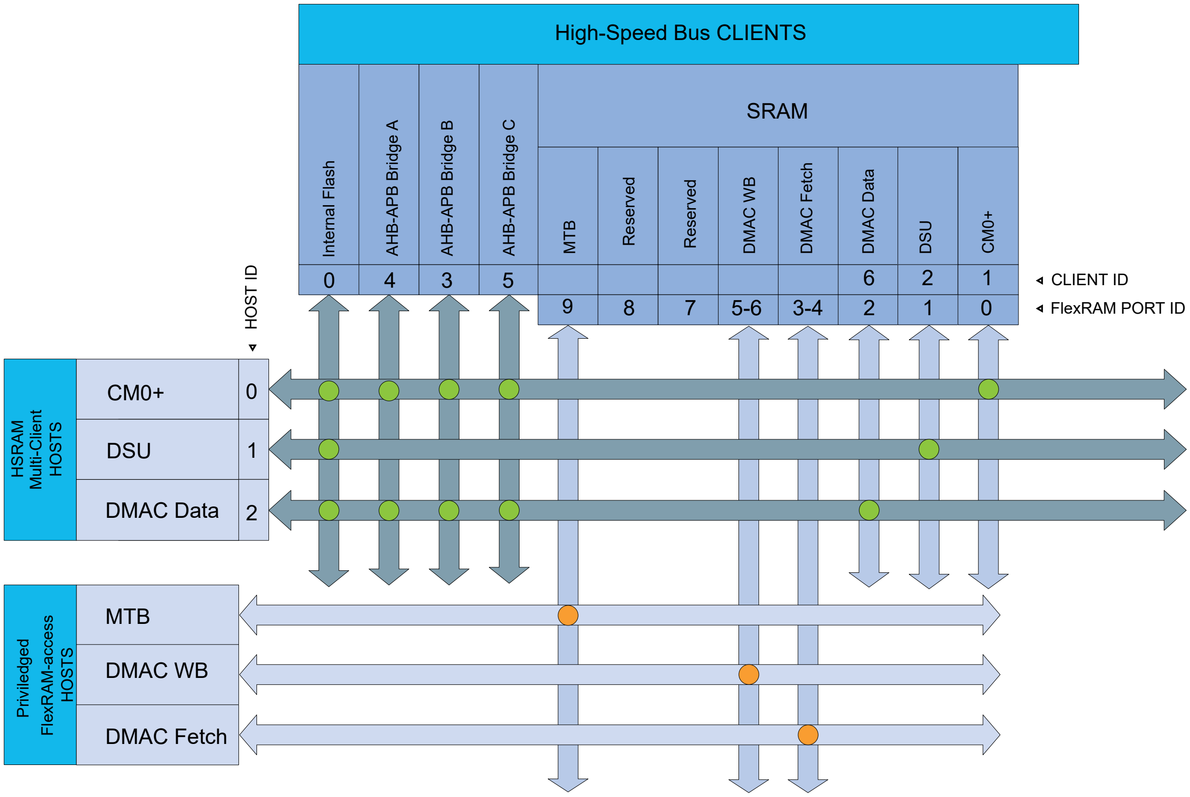


Table 4-4. Bus Matrix Hosts

Hosts (HSRAM)
Cortex M0+ Processor (CM0+)
Device Service Unit (DSU)
Direct Memory Access Controller/Data Access (DMAC)

Table 4-5. Bus Matrix Clients

Bus Matrix Clients
Internal Flash Memory
SRAM - CM0+ Access
SRAM - DSU Access
AHB-APB Bridge A
AHB-APB Bridge B
AHB-APB Bridge C
SRAM - DMAC Data Access

Table 4-6. SRAM Port Connections

SRAM Port Connection	Port ID	Connection Type
Cortex M0+ (CM0+) Processor	0	Bus Matrix
Device Service Unit (DSU)	1	Bus Matrix
Direct Memory Access Controller (DMAC) - Data Access	2	Bus Matrix
Direct Memory Access Controller (DMAC) - Fetch Access 0	3	Direct
Direct Memory Access Controller (DMAC) - Fetch Access 1	4	Direct

Table 4-6. SRAM Port Connections (continued)

SRAM Port Connection	Port ID	Connection Type
Direct Memory Access Controller (DMAC) - Write-Back Access 0	5	Direct
Direct Memory Access Controller (DMAC) - Write-Back Access 1	6	Direct
Reserved	7	
Reserved	8	
Micro Trace Buffer (MTB)	9	Direct

4.4.3. SRAM Quality of Service

To ensure that hosts with latency requirements receive sufficient priority when accessing SRAM, the different hosts can be configured to have a given priority for different types of access.

The Quality of Service (QoS) level is independently selected for each host accessing the SRAM. For any access to the SRAM, the SRAM also receives the QoS level. The QoS levels and their corresponding bit values for the QoS level configuration is shown in the following table.

Table 4-7. Quality of Service Level Configuration

Value	Name	Description
0x0	DISABLE	Background (no sensitive operation)
0x1	LOW	Sensitive Bandwidth
0x2	MEDIUM	Sensitive Latency
0x3	HIGH	Critical Latency

If a host is configured with QoS level DISABLE (0x0) or LOW (0x1), there will be a minimum latency of one cycle for the SRAM access.

The priority order for concurrent accesses is determined by two factors: First, the QoS level for the host, and second, a static priority given by the SRAM Port ID, as defined in SRAM Port Connections. The lowest port ID has the highest static priority.

The MTB has a fixed QoS level of HIGH (0x3).

The CPU QoS level can be written to or read from address 0x4100A110, bits [1:0]. Its reset value is 0x0.

Refer to the different host registers for configuring their QoS.

4.5. Micro Trace Buffer

4.5.1. Features

- Program flow tracing for the Cortex-M0+ processor
- MTB SRAM can be used for both trace and general purpose storage by the processor
- The position and size of the trace buffer in SRAM are configurable by software
- CoreSight compliant

4.5.2. Overview

When enabled, the MTB records changes in the program flow, reported by the Cortex-M0+ processor over the execution trace interface shared between the Cortex-M0+ processor and the CoreSight MTB-M0+. This information is stored as trace packets in the SRAM by the MTB. An off-chip debugger can extract the trace information using the Debug Access Port to read the trace information from the SRAM. The debugger can then reconstruct the program flow from this information.

The MTB simultaneously stores trace information in the SRAM and provides the processor with access to the SRAM. The MTB ensures that trace write accesses have priority over processor accesses.

The execution trace packet consists of a pair of 32-bit words that the MTB generates when it detects that the processor PC value changes non-sequentially. A non-sequential PC change can occur during branch instructions or exception entry. Refer to the *CoreSight MTB-M0+ Technical Reference Manual* for details on the MTB execution trace packet format.

Tracing is enabled when the MASTER.EN bit in the Host Trace Control Register is 1. There are various ways to set the bit to '`1`' to start tracing, or to '`0`' to stop tracing. Refer to the *CoreSight Cortex-M0+ Technical Reference Manual* for details on the Trace start and stop, and for a detailed description of the MTB's MASTER register. The MTB can be programmed to stop tracing automatically when the memory fills to a specified watermark level, or to start or stop tracing by writing directly to the MASTER.EN bit. If the watermark mechanism is not used and the trace buffer overflows, then the buffer wraps around, overwriting previous trace packets.

The base address of the MTB registers is 0x41008000, and this address is also listed in the CoreSight ROM table. The offset of each register from the base address is fixed, as defined by the *CoreSight MTB-M0+ Technical Reference Manual*. The MTB has four programmable registers to control the behavior of the trace features:

- **POSITION:** Contains the trace write pointer and the wrap bit
- **MASTER:** Contains the main trace enable bit and other trace control fields
- **FLOW:** Contains the WATERMARK address and the AUTOSTOP and AUTOHALT control bits
- **BASE:** Indicates where the SRAM is located in the processor memory map. This register is provided to enable auto-discovery of the MTB SRAM location, by a debug agent.

Refer to the *CoreSight MTB-M0+ Technical Reference Manual* for a detailed description of these registers.

4.6. Reset Sources

Generally, PIC32CM devices can distinguish between these types of reset:

1. Power Resets
   - POR: Power On Reset
   - BOR: Brownout Detector Reset
2. User Resets
   - WDT: Watchdog Timer
   - $\overline{\text{RESET}}$ pin
   - CPU - both `sysreset` request and core lockup

Important: On devices of the PIC32CM PL10 family, all User Reset sources are treated equally like Power Reset sources.

5. Memories

5.1. Embedded Memories

- Internal high-speed Flash
- Internal high-speed SRAM with single-cycle access at full speed

5.2. Physical Memory

Table 5-1. Memory Properties

Memory		StartAddress	PIC32CM6408PL10
Embedded Flash	Size	0x00000000	64 KB
	Page number		128
	Page size		512B
Embedded high-speed SRAM		0x20000000	8 KB

The high-speed bus is implemented as a bus matrix. All high-speed bus addresses are fixed and are never remapped in any way, even during boot. See the Product Mapping chapter for the logical organization and addresses.

5.3. Register Properties

Registers can be 8, 16, or 32 bits wide. Atomic 8-bit, 16-bit, and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register, as well as the 8-bit halves of a 16-bit register, can be accessed directly.

Registers can have one or multiple properties, as indicated in the register description:

PAC Write-Protection

Some registers can be write-protected by the Peripheral Access Controller (PAC).

PAC write protection is indicated by the “PAC Write-Protection” property in the register description. For more details, refer to the *PAC - Peripheral Access Controller* chapter.
Note: PAC Write-Protection is optional.

Local Write-Protection

Many peripherals offer a local, key-based write-protection mechanism for registers with write access. These peripherals have a Write Protection Control (WPCTRL) register, and the registers that can be protected bear the property “Local Write-Protection”.

Follow these steps to configure the optional local write-protection:

- When writing to the WPCTRL register, the Write Protection Key (WPKEY) bit field must contain the specific KEY value.
- The local write-protection is enabled by writing a `'1'` to the Write Protection Enable (WPEN) bit in the WPCTRL register.
- The WPCTRL register itself can be protected by writing a `'1'` to the Write Protection Lock (WPLCK) bit. This bit can be cleared by a reset, but not by the application.

Note: The optional local write-protection mechanism is not preventing debugger access.

Enable-Protected

Some registers or bit fields can only be written when the peripheral is disabled, denoted by the “Enable-Protected” property in the register description.

Note: Enable-Protection is not optional.

Read-Synchronized, Write-Synchronized

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers or bit fields need to be synchronized when being written or read. Required write-synchronization is indicated by the “Write-Synchronized” property in the register description. For more details, refer to the *Register Synchronization* section in the *CS - Clock System* chapter.

5.4. SIGNATURE

Each device has a unique 128-bit serial number, which is a concatenation of four 32-bit words from the Serial Number fuses SERNUM[3:0]. The uniqueness of the serial number is ensured only when all 128 bits are used.

For general device selection purposes, such as by a debugger, the pin/memory configuration of the device is also stored in the Device ID register of the Device Service Unit (DSU).

5.4.1. Register Summary - SIGNATURE

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00 ... 0x1F	Reserved									
0x20	SERNUM0	7:0	SERNUM0[7:0]							
		15:8	SERNUM0[15:8]							
		23:16	SERNUM0[23:16]							
		31:24	SERNUM0[31:24]							
0x24 ... 0x27	Reserved									
0x28	SERNUM1	7:0	SERNUM1[7:0]							
		15:8	SERNUM1[15:8]							
		23:16	SERNUM1[23:16]							
		31:24	SERNUM1[31:24]							
0x2C ... 0x2F	Reserved									
0x30	SERNUM2	7:0	SERNUM2[7:0]							
		15:8	SERNUM2[15:8]							
		23:16	SERNUM2[23:16]							
		31:24	SERNUM2[31:24]							
0x34 ... 0x37	Reserved									
0x38	SERNUM3	7:0	SERNUM3[7:0]							
		15:8	SERNUM3[15:8]							
		23:16	SERNUM3[23:16]							
		31:24	SERNUM3[31:24]							

5.4.1.1. Serial Number 0

Name: SERNUM0
Offset: 0x20
Reset: 0xXXXXXXXXXXXXXXXX
Property: R

Each device has a unique 128-bit serial number, which is a concatenation of four 32-bit words from the Serial Number fuses SERNUM[3:0]. The uniqueness of the serial number is ensured only when all 128 bits are used.

Bit	31	30	29	28	27	26	25	24
	SERNUM0[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	23	22	21	20	19	18	17	16
	SERNUM0[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	15	14	13	12	11	10	9	8
	SERNUM0[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	7	6	5	4	3	2	1	0
	SERNUM0[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 31:0 – SERNUM0[31:0] Serial Number 0

5.4.1.2. Serial Number 1

Name: SERNUM1
Offset: 0x28
Reset: 0xXXXXXXXXXXXXXXXX
Property: R

Each device has a unique 128-bit serial number, which is a concatenation of four 32-bit words from the Serial Number fuses SERNUM[3:0]. The uniqueness of the serial number is ensured only when all 128 bits are used.

Bit	31	30	29	28	27	26	25	24
	SERNUM1[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x
Bit	23	22	21	20	19	18	17	16
	SERNUM1[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x
Bit	15	14	13	12	11	10	9	8
	SERNUM1[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x
Bit	7	6	5	4	3	2	1	0
	SERNUM1[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 31:0 – SERNUM1[31:0] Serial Number 1

5.4.1.3. Serial Number 2

Name: SERNUM2
Offset: 0x30
Reset: 0xXXXXXXXXXXXXXXXX
Property: R

Each device has a unique 128-bit serial number, which is a concatenation of four 32-bit words from the Serial Number fuses SERNUM[3:0]. The uniqueness of the serial number is ensured only when all 128 bits are used.

Bit	31	30	29	28	27	26	25	24
	SERNUM2[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x
Bit	23	22	21	20	19	18	17	16
	SERNUM2[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x
Bit	15	14	13	12	11	10	9	8
	SERNUM2[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x
Bit	7	6	5	4	3	2	1	0
	SERNUM2[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 31:0 – SERNUM2[31:0] Serial Number 2

5.4.1.4. Serial Number 3

Name: SERNUM3
Offset: 0x38
Reset: 0xXXXXXXXXXXXXXXXX
Property: R

Each device has a unique 128-bit serial number, which is a concatenation of four 32-bit words from the Serial Number fuses SERNUM[3:0]. The uniqueness of the serial number is ensured only when all 128 bits are used.

Bit	31	30	29	28	27	26	25	24
	SERNUM3[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	23	22	21	20	19	18	17	16
	SERNUM3[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	15	14	13	12	11	10	9	8
	SERNUM3[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	7	6	5	4	3	2	1	0
	SERNUM3[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 31:0 – SERNUM3[31:0] Serial Number 3

5.5. BOOTCFG - Boot Configuration Fuses

The BOOTCFG fuses contain configuration details for the device and essential peripherals, such as BOD or WDT, that are applied after a device reset and during start-up.

When the Sequence Number (SEQNUM) fuse is not 0xFFFFFFFF at start-up and after a Reset, the contents of the other BOOTCFG fuses are copied to the respective peripheral registers and bit fields, as described in the following fuse descriptions. Follow these steps to alter the device configuration:

1. Write the desired values to the fuses.
2. Reset the device. The values are copied from the fuses to the peripheral registers.
3. (Optional) To read the actual configuration, read the respective registers of the SYSCTRL, WDT, and SUPC.

Attention: When the SEQNUM fuse is not 0xFFFFFFFF, *all* of the BOOTCFG fuse values are copied to the peripheral registers. For this reason, all BOOTCFG fuses must be written to desired values—the factory values of these fuses have no practical meaning.
When the value of the Sequence Number (SEQNUM) fuse is equal to 0xFFFFFFFF, *none* of the BOOTCFG fuse values are copied to any of the peripheral registers at startup/reset. Instead, the respective reset values given in the peripheral registers' descriptions are used.

Note: The entire BOOTCFG section is unaffected by the Chip Erase command. For this reason, the residual space of the BOOTCFG section (after the Fuses described below) can be used to store persistent application data, such as serial numbers or application keys.

5.5.1. Register Summary - BOOTCFG

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	SEQNUM	7:0	SEQNUM[7:0]							
		15:8	SEQNUM[15:8]							
		23:16	SEQNUM[23:16]							
		31:24	SEQNUM[31:24]							
0x04 ... 0x07	Reserved									
0x08	BOOTPROT	7:0						BOOTPROT[2:0]		
		15:8								
		23:16								
		31:24								
0x0C ... 0x0F	Reserved									
0x10	WDTCFG	7:0					ALWAYSON	WEN	ENABLE	
		15:8	WINDOW[3:0]				PER[3:0]			
		23:16					EWOFFSET[3:0]			
		31:24								
0x14 ... 0x17	Reserved									
0x18	BODCFG	7:0		RUNSTDBY	STDBYCFG				ENABLE	
		15:8				SAMPFREQ				ACTCFG
		23:16							LEVEL[1:0]	
		31:24	WRTLOCK				VLMCFG[1:0]		VLMLVL[1:0]	
0x1C ... 0x1F	Reserved									
0x20	USERCFG	7:0	CRCBOOT	CRCSEL						MVIOMODE
		15:8						SUT[2:0]		
		23:16								
		31:24								
0x24 ... 0x27	Reserved									
0x28	BOOT_GPIOSEL	7:0	ENABLE			GPIOPINSEL[4:0]				
		15:8					GPIOPORTSEL[3:0]			
		23:16				SLEWLIM	ODRAIN		POL	
		31:24								

5.5.1.1. Sequence Number Fuse

Name: SEQNUM
Offset: 0x0000
Default: 0xFFFFFFFF
Property: R/W

The default value given in this fuse description is the factory-programmed value and must not be mistaken for the Reset value.

Bit	31	30	29	28	27	26	25	24
	SEQNUM[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	SEQNUM[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	SEQNUM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	SEQNUM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	1	1	1	1	1	1	1

Bits 31:0 – SEQNUM[31:0] Sequence Number

When the value of this bit field is equal to `0xFFFFFFFF` (factory default) at start-up, none of the BOOTCFG fuse values is copied to any of the peripheral registers at startup or reset. Instead, the respective reset values given in the peripheral registers' descriptions are used.

When this bit field is **not** `0xFFFFFFFF` at start-up, the user-defined Boot Configuration (BOOTCFG) fuse values are applied.

The value of this fuse can be used by the bootloader or application for version tracking or similar purposes, as long as it is not `0xFFFFFFFF`.

5.5.1.2. Boot Protection Fuse

Name:	BOOTPROT
Offset:	0x0008
Default:	0xFFFFFFFF
Property:	R/W

The bit group values of this fuse are written to the corresponding bit groups of the NVM Parameters (PARAM) register in the Non-Volatile Memory Controller peripheral (NVMCTRL) at start-up. The default value given in this fuse description is the factory-programmed value and must not be mistaken for the Reset value.

Bit	31	30	29	28	27	26	25	24
Access								
Default								

Bit	23	22	21	20	19	18	17	16
Access								
Default								

Bit	15	14	13	12	11	10	9	8
Access								
Default								

Bit	7	6	5	4	3	2	1	0
						BOOTPROT[2:0]		
Access						R/W	R/W	R/W
Default						1	1	1

Bits 2:0 – BOOTPROT[2:0] Boot Protection

This bit field value is loaded into the Boot Loader Size (BOOTPROT) bit field of the NVMCTRL.PARAM register during Reset. Refer to the *NVMCTRL - Non-Volatile Memory Controller* section for further details.

Value	Name	Description
0x0	SIZE_32768BYTES	32768 bytes are protected
0x1	SIZE_16384BYTES	16384 bytes are protected
0x2	SIZE_8192BYTES	8192 bytes are protected
0x3	SIZE_4096BYTES	4096 bytes are protected
0x4	SIZE_2048BYTES	2048 bytes are protected
0x5	SIZE_1024BYTES	1024 bytes are protected
0x6	SIZE_512BYTES	512 bytes are protected
0x7	SIZE_0BYTES	0 bytes are protected (default)

5.5.1.3. Watchdog Timer Configuration Fuse

Name: WDTCFG
Offset: 0x0010
Default: 0xFFFFFFFF
Property: R/W

The bit group values of this fuse are written to the corresponding bit groups of the Control A (CTRLA), Configuration (CONFIG), and Early Warning Control (EWCTRL) registers in the Watchdog Timer peripheral (WDT) at start-up.

The default value given in this fuse description is the factory-programmed value and must not be mistaken for the Reset value.

Bit	31	30	29	28	27	26	25	24
Access								
Default								

Bit	23	22	21	20	19	18	17	16
					EWOFFSET[3:0]			
Access					R/W	R/W	R/W	R/W
Default					1	1	1	1

Bit	15	14	13	12	11	10	9	8
	WINDOW[3:0]				PER[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
					ALWAYSON	WEN	ENABLE	
Access					R/W	R/W	R/W	
Default					1	1	1	

Bits 19:16 – EWOFFSET[3:0] Early Warning Interrupt Time Offset Fuse

This bit field value is loaded into the Early Warning Interrupt Time Offset (EWOFFSET) bit field of the WDT.EWCTRL register during Reset. Refer to the *WDT - Watchdog Timer* section for further details.

Value	Name	Description
`0x0`	CYC8	8 clock cycles
`0x1`	CYC16	16 clock cycles
`0x2`	CYC32	32 clock cycles
`0x3`	CYC64	64 clock cycles
`0x4`	CYC128	128 clock cycles
`0x5`	CYC256	256 clock cycles
`0x6`	CYC512	512 clock cycles
`0x7`	CYC1024	1024 clock cycles
`0x8`	CYC2048	2048 clock cycles
`0x9`	CYC4096	4096 clock cycles
`0xA`	CYC8192	8192 clock cycles
`0xB - 0xF`	—	Reserved

Bits 15:12 – WINDOW[3:0] Window

This bit field value is loaded into the WINDOW bit field of the WDT.CONFIG register during Reset. Refer to the *WDT - Watchdog Timer* section for further details.

Value	Name	Description
0x0	CYC8	8 clock cycles
0x1	CYC16	16 clock cycles
0x2	CYC32	32 clock cycles
0x3	CYC64	64 clock cycles
0x4	CYC128	128 clock cycles
0x5	CYC256	256 clock cycles
0x6	CYC512	512 clock cycles
0x7	CYC1024	1024 clock cycles
0x8	CYC2048	2048 clock cycles
0x9	CYC4096	4096 clock cycles
0xA	CYC8192	8192 clock cycles
0xB	CYC16384	16384 clock cycles
Other	—	Reserved

Bits 11:8 – PER[3:0] Time-Out Period

This bit field value is loaded into the PER bit field of the Watchdog Configuration (WDT.CONFIG) register during Reset. Refer to the *WDT - Watchdog Timer* section for further details.

Value	Name	Description
0x0	CYC8	8 clock cycles
0x1	CYC16	16 clock cycles
0x2	CYC32	32 clock cycles
0x3	CYC64	64 clock cycles
0x4	CYC128	128 clock cycles
0x5	CYC256	256 clock cycles
0x6	CYC512	512 clock cycles
0x7	CYC1024	1024 clock cycles
0x8	CYC2048	2048 clock cycles
0x9	CYC4096	4096 clock cycles
0xA	CYC8192	8192 clock cycles
0xB	CYC16384	16384 clock cycles
Other	—	Reserved

Bit 3 – ALWAYSON Always On

This bit value is loaded into the ALWAYSON bit field of the WDT.CTRLA register during Reset. Refer to the *WDT - Watchdog Timer* section for further details.

Value	Description
0	The WDT can be enabled and disabled through the ENABLE bit
1	The WDT is enabled and can only be disabled by a Power-on Reset (POR)

Bit 2 – WEN Window Mode Enable

This bit value is loaded into the WEN bit field of the WDT.CTRLA register during Reset. Refer to the *WDT - Watchdog Timer* section for further details.

Value	Description
0	Window mode is disabled
1	Window mode is enabled

Bit 1 – ENABLE Enable

This bit value is loaded into the ENABLE bit in the WDT.CTRLA register during Reset. Refer to the *WDT - Watchdog Timer* section for further details.

Value	Description
0	The WDT is disabled
1	The WDT is enabled

5.5.1.4. BOD Configuration Fuse

Name: BODCFG
Offset: 0x018
Default: 0xFFFFFFFF
Property: R/W

The bit group values of this fuse are written to the corresponding bit groups of the BOD VDD Control (BODVDD) register in the Supply Controller (SUPC) at start-up.

The default value given in this fuse description is the factory-programmed value and must not be mistaken for the Reset value.

Bit	31	30	29	28	27	26	25	24
	WRTLOCK				VLMCFG[1:0]		VLMLVL[1:0]	
Access	R/W				R/W	R/W	R/W	R/W
Default	1				1	1	1	1

Bit	23	22	21	20	19	18	17	16
							LEVEL[1:0]	
Access							R/W	R/W
Default							1	1

Bit	15	14	13	12	11	10	9	8
				SAMPFREQ				ACTCFG
Access				R/W				R/W
Default				1				1

Bit	7	6	5	4	3	2	1	0
		RUNSTDBY	STDBYCFG				ENABLE	
Access		R/W	R/W				R/W	
Default		1	1				1	

Bit 31 – WRTLOCK Write Lock

This bit value is loaded into the WRTLOCK bit in the SUPC.BODVDD register during Reset. Refer to the *SUPC - Supply Controller* chapter for further details.

Value	Description
0x0	The BODVDD configuration is not locked
0x1	The BODVDD configuration is locked

Bits 27:26 – VLMCFG[1:0] Voltage Level Monitor Interrupt Configuration

This bit value is loaded into the VLMCFG bit in the SUPC.BODVDD register during Reset. Refer to the *SUPC - Supply Controller* chapter for further details.

Value	Name	Description
0x0	FALLING	VDD falls below VLM threshold
0x1	RISING	VDD rises above VLM threshold
0x2	BOTH	VDD crosses VLM threshold

Bits 25:24 – VLMLVL[1:0] Voltage Level Monitor Level

This bit field value is loaded into the VLMLVL bit in the SUPC.BODVDD register during Reset. Refer to the *SUPC - Supply Controller* chapter for further details.

Value	Name	Description
0x0	OFF	VLM disabled
0x1	5ABOVE	VLM threshold is 5% above BOD threshold

Value	Name	Description
0x2	15ABOVE	VLM threshold is 15% above BOD threshold
0x3	25ABOVE	VLM threshold is 25% above BOD threshold

Bits 17:16 – LEVEL[1:0] BOD Level

This bit field value is loaded into the LEVEL bit field in the SUPC.BODVDD register during Reset.

Notes:

1. Refer to the *BOD and POR Characteristics* in the *Electrical Characteristics* section for further details.
2. Values in the description are typical values.

Value	Name	Description
0x0	BODLEVEL0	1.90V
0x1	BODLEVEL1	2.45V
0x2	BODLEVEL2	2.70V
0x3	BODLEVEL3	2.85V

Bit 12 – SAMPFREQ BOD Sampling Frequency

This bit value is loaded into the SAMPFREQ bit in the SUPC.BODVDD register during Reset. Refer to the *SUPC - Supply Controller* chapter for further details.

Value	Name	Description
0	128HZ	The sample frequency is 128 Hz
1	32HZ	The sample frequency is 32 Hz

Bit 8 – ACTCFG BOD Operation Mode in Active Mode

This bit value is loaded into the ACTCFG bit in the SUPC.BODVDD register during Reset. Refer to the *SUPC - Supply Controller* chapter for further details.

Value	Name	Description
0	ENABLE	In active mode, the BOD operates in continuous mode
1	SAMPLE	In active mode, the BOD operates in sampling mode

Bit 6 – RUNSTDBY Run in Standby

This bit value is loaded into the RUNSTDBY bit in the SUPC.BODVDD register during Reset. Refer to the *SUPC - Supply Controller* chapter for further details.

Value	Description
0x0	The BOD is not running in Standby sleep mode
0x1	The BOD is running in Standby sleep mode if enabled by the ENABLE bit

Bit 5 – STDBYCFG BOD Configuration in Standby Sleep Mode

This bit value is loaded into the STDBYCFG bit in the SUPC.BODVDD register during Reset. Refer to the *SUPC - Supply Controller* chapter for further details.

Value	Name	Description
0x0	CONT	In Standby sleep mode, the BOD is configured in continuous mode
0x1	SAMP	In Standby sleep mode, the BOD is configured in sampling mode

Bit 1 – ENABLE BOD Enable

This bit value is loaded into the ENABLE bit in the SUPC.BODVDD register during Reset. Refer to the *SUPC - Supply Controller* chapter for further details.

Value	Name	Description
0	DISABLE	BOD is disabled
1	ENABLE	BOD is enabled

5.5.1.5. User Configuration Fuse

Name: USERCFG
Offset: 0x0020
Default: 0xFFFFFFFF
Property: R/W

The bit field values of this fuse are written to the corresponding bit fields of the User Configuration register (SYSCTRL.USERCFG) in the System Controller and the Multivoltage I/O register (SUPC.MVIO) in the Supply Controller at start-up. The default value given in this fuse description is the factory-programmed value and must not be mistaken for the Reset value.

Bit	31	30	29	28	27	26	25	24
Access								
Default								

Bit	23	22	21	20	19	18	17	16
Access								
Default								

Bit	15	14	13	12	11	10	9	8
						SUT[2:0]		
Access						R/W	R/W	R/W
Default						1	1	1

Bit	7	6	5	4	3	2	1	0
	CRCBOOT	CRCSEL						MVIOMODE
Access	R/W	R/W						R/W
Default	1	1						1

Bits 10:8 – SUT[2:0] Start-up Time
This bit field value is loaded into the SUT bit field in the USERCFG register of the System Controller (SYSCTRL.USERCFG) during Reset. Refer to the *SYSCTRL - System Controller* section for further details.

Value	Name	Description
0x0	0MS	0 ms
0x1	1MS	1 ms
0x2	2MS	2 ms
0x3	4MS	4 ms
0x4	8MS	8 ms
0x5	16MS	16 ms
0x6	32MS	32 ms
0x7	64MS	64 ms

Bit 7 – CRCBOOT CRC Boot
This bit value is loaded into the CRCBOOT bit in the USERCFG register of the System Controller (SYSCTRL.USERCFG) during Reset. Refer to the *SYSCTRL - System Controller* section for further details.

Value	Name	Description
0	DISABLE	No CRC
1	ENABLE	CRC of the Boot section

Bit 6 – CRCSEL CRC Polynomial Selection

This bit value is loaded into the CRCSEL bit in the USERCFG register of the System Controller (SYSCTRL.USERCFG) during Reset. Refer to the *SYSCTRL - System Controller* section for further details.

Value	Name	Description
0	CRC16	CRC-16-CCITT
1	CRC32	CRC-32 (IEEE 802.3)

Bit 0 – MVIOMODE MVIO Operation Mode

This bit value is loaded into the MODE bit in the MVIO register of the Supply Controller (SUPC.MVIO) during Reset. Refer to the *SUPC - Supply Controller* section for further details.

Value	Name	Description
0	DUAL	MVIO operating in dual supply mode
1	SINGLE	MVIO operating in single supply mode

5.5.1.6. Boot External Notification I/O Pin Fuse

Name: BOOT_GPIOSEL
Offset: 0x0028
Default: 0xFFFFFFFF
Property: R/W

The bit fields in this fuse select the pin and its signal properties for the Boot External Notification signal at start-up.

The default value given in this fuse description is the factory-programmed value and must not be mistaken for the Reset value.

Bit	31	30	29	28	27	26	25	24
Access								
Default								

Bit	23	22	21	20	19	18	17	16
				SLEWLIM	ODRAIN		POL	
Access				R/W	R/W		R/W	
Default				1	1		1	

Bit	15	14	13	12	11	10	9	8
					GPIOPORTSEL[3:0]			
Access					R/W	R/W	R/W	R/W
Default					1	1	1	1

Bit	7	6	5	4	3	2	1	0
	ENABLE			GPIOPINSEL[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Default	1			1	1	1	1	1

Bit 20 – SLEWLIM Slew Rate Control for Boot External Notification Signal

This bit controls whether the slew rate is limited.

Value	Name	Description
0	FALSE	The slew rate is not limited (fast rise/fall time operation)
1	TRUE	The slew rate is limited.

Bit 19 – ODRAIN Open Drain Control for Boot External Notification Signal

This bit controls whether the fault source is of normal drive type or open-drain drive type. Together with the Pin Output Level for Boot External Notification Signal (POL) bit, the resulting behavior of the pin is as follows:

	ODRAIN = 0	**ODRAIN = 1**
POL = 0	Tristate	Drive high
POL = 1	Drive low	Drive low

Value	Name	Description
0	FALSE	The selected fault source is open drain drive
1	TRUE	The selected fault source is normal drive

Bit 17 – POL Select Polarity in Normal Drive Mode for Boot External Notification Signal

This bit selects whether the Boot External Notification signal is active-low or active-high. Refer also to the Open Drain Control for Boot External Notification Signal (ODRAIN) bit description, because ODRAIN and POL are interdependent properties for the pin.

Value	Name	Description
0	FALSE	The selected fault source is active-high
1	TRUE	The selected fault source is active-low

Bits 11:8 – GPIOPORTSEL[3:0] GPIO Port Select for Boot External Notification Signal

This bit field selects the I/O pin port used for the Boot External Notification signal.

Note: Select only a port that exists on the device.

Value	Name	Description
0x0	PORTA	Port A
0x1	PORTB	Port B
0x2	PORTC	Port C
0x3	PORTD	Port D
Other	-	No Boot External Notification signal used

Bit 7 – ENABLE Enable Boot External Notification Signal

This bit enables the Boot External Notification signal.

Value	Name	Description
0	DISABLE	The Boot External Notification signal is disabled
1	ENABLE	The Boot External Notification signal is enabled

Bits 4:0 – GPIOPINSEL[4:0] GPIO Pin Select for Boot External Notification Signal

This bit field selects the I/O pin used for the Boot External Notification signal.

Note: Select only a pin that exists on the device.

Value	Name	Description
0x00	PIN00	Pin 0
0x01	PIN01	Pin 1
...	...	...
0x1E	PIN30	Pin 30
0x1F	PIN31	Pin 31

6. BootROM - Boot ROM

6.1. Features

- Device Integrity Check to Verify Authenticity and Configuration
- Device Lock to Prevent any External Read/Write Access (Programming and Debugging Interface Disable - PDID)
- Boot Failure Indication via I/O Pin
- Boot Interactive Mode (IMODE) for Programming and Maintenance Operation With a Connected Debugger
- Deterministic Boot Process: ROM Code Executes Without Debugger Access Using a Dedicated Clock Source

6.2. Overview

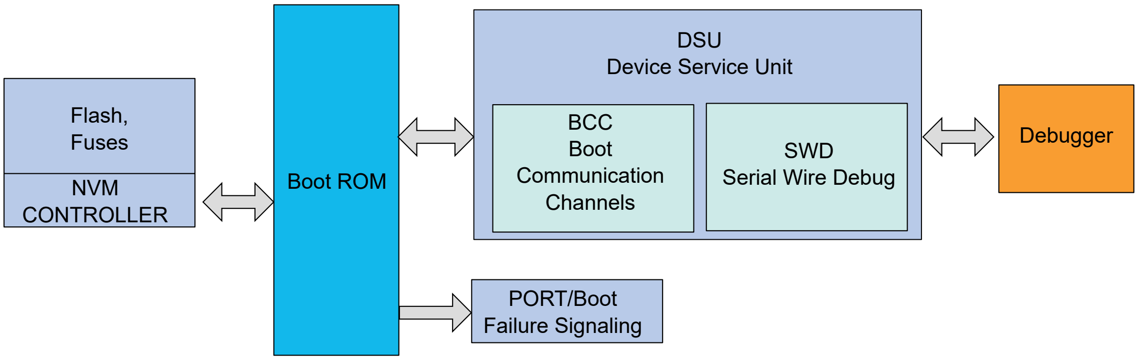
The Boot ROM is the entry point of the device following any type of reset. It performs a series of vital checks, including image validation and configuration verification. If these checks pass, the CPU loads the initial Program Counter (PC) and Stack Pointer (SP) values from Flash memory and begins executing the user code.

The Boot ROM also provides the Interactive mode, allowing both hot-plugging and cold-plugging of a debugger and programming.

Note:
Throughout this document, *Debugger* shall be interpreted as SWD-Host (and vice-versa) that is supporting this device.

6.3. Block Diagram

Figure 6-1. Boot ROM Block Diagram



6.4. Functional Description

6.4.1. Boot ROM Operation at Startup

The Boot ROM operation at startup can be logically divided into the following functional stages:

1. **Integrity Check of the Program Flash Memory (Optional)**
If the CRCBOOT bit in the User Configuration (BOOTCFG.USERCFG) fuse is set, the Boot ROM will perform a CRC-32 scan on the Program Flash Memory. If any integrity violation is detected, the Boot ROM halts further execution, reports the error in the DSU.BCC registers[(1)], and places the device into a known safe state called Interactive mode (IMODE)[(2)]. Otherwise, the Boot ROM proceeds to the next stage.

2. **Debugger Probe Detection**
   The Boot ROM is designed to asynchronously accept a cold-plug-in request from the debugger (i.e. SWD host) at any time during the device's operating life cycle. The Boot ROM automatically detects debugger attachement to the device as soon as the debugger follows the cold-plug-in sequence. Refer to the *DSU - Device Service Unit* chapter and the *Programming Specifications* section in this chapter for details on debugger operation.
3. **Application of User Settings and Finalization**
   Applies user-selected fuse configurations to the respective peripherals. Refer to the *BOOTCFG - Boot Configuration Fuses* chapter for more details.

   At the end of the boot process, the OSCHF at 4 MHz is selected as the default clock for GCLK0 and, consequently, MCLK. See the chapters on the *Clock System, GCLK, MCLK* for details.
4. **Transition to First Mutable Application Executable**
   Transfers control to the first mutable application executable by loading the PC and SP values only if the boot operation is successful. Refer to the *Product Mapping* and *Memories* chapters for details.

Notes:

1. If the Boot External Notification I/O Pin (BOOT_GPIOSEL) fuse in the BOOTCFG memory section is written, a fault signal is asserted on the configured pin.
2. When no debugger probe is detected after entering IMODE, the device will reset.

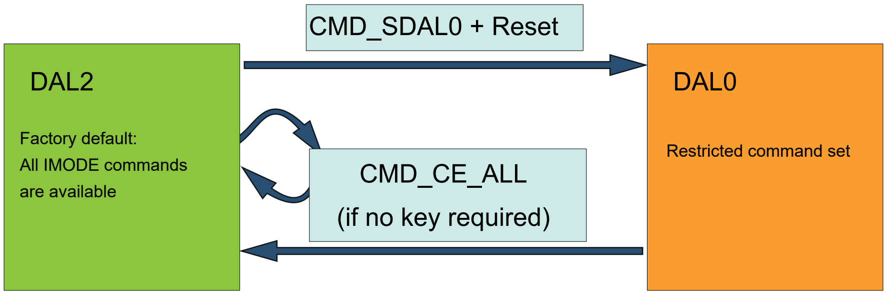
6.4.2. Debugger Access Level (DAL)

The Boot ROM supports two levels of debugger access, known as **Device Access Levels (DAL)**. These levels define the access a debugger has to the device during and after boot:

- **DAL2**: Full access. The device is completely open, allowing all operations supported by the device. This level is typically used during development and debugging.
- **DAL0**: Locked state. The device is highly restricted. Debug access is limited and defined by the user prior to setting DAL to 0.

The active DAL is determined by the value programmed into an internal memory location. Once configured, the DAL governs the debugger's ability to interact with the device, including access to memory, peripherals, and Boot ROM features:

Figure 6-2. Changing Between The Debugger Access Levels



1. The device is factory-programmed with **DAL2** to allow the user to configure the device according to end application requirements. All commands of the Interactive mode IMODE are accessible, allowing full debugging and programming access.
2. Once application development is finished, the device can be configured to **DAL0** by issuing the Select DAL 0 command `CMD_SDAL0` in IMODE and resetting the device. Now, only a restricted set of commands is available in IMODE.
   Note: Refer to the sections *Programming and Debug Interface Disable (PDID)* and *Programming Specification* for more information on how to restrict commands and access.

3. Issuing the Chip Erase All command `CMD_CE_ALL` in IMODE will clear all Flash and volatile memories (but not the BOOTCFG and calibration fuses) and, eventually, set the DAL back to DAL2.

⚠ CAUTION It is possible to lock the `CMD_CE_ALL` command. When the device is set to DAL0 and the `CMD_CE_ALL` command is locked, the device may become **permanently unrecoverable** if the user does not retain access to the Key Value fuse required to unlock `CMD_CE_ALL`. Refer to the *Programming Specification* section for more information.

⚠ WARNING Carefully plan and validate the fuse configurations before transitioning the device to DAL0, especially in production environments where recovery options are limited.

6.4.3. Operating Modes

The Boot ROM operation is logically divided into the following four modes, each serving a distinct purpose during the device startup and debug process:

1. **Boot Mode**

   This mode is entered immediately after any type of Reset when no debugger probe is present. In this mode, the Boot ROM performs system initialization, fuse integrity checks, and attempts to boot the device using the user-configured settings.

   Under normal operation, the device will then transition to *Mission mode*.

   When a boot error occurs, a boot failure signal can be transmitted (see the *Boot Failure Signal* section), and the device resets.

   When a boot error occurs and a debugger is connected, *Interactive mode* is invoked.

2. **Interactive Mode (IMODE)**

   The Boot ROM enters IMODE under either of the following conditions:

   – A boot failure occurs and a debugger is connected

   – A debugger explicitly requests entry into IMODE to perform advanced operations

   The IMODE allows to perform tasks such as:

   - Full chip erase
   - Device locking (setting DAL = 0)
   - Configuration Flash Memory (CFM) integrity checks

   Some tasks may be prohibited due to deliberate locking or protection measures.

   IMODE is always exited by a Reset. The cause of the Reset can be an IMODE command, a hard fault, the absence of a connected debugger, or any regular Reset source. The device will then enter *Boot mode*.

3. **Mission Mode**

   This mode allows the device to begin execution from the first mutable application executable location in Flash.

   The device remains in *Mission mode* until it is reset for any reason. After a Reset, the device will enter *Boot mode*.

4. **Park Mode**

This mode is invoked under certain conditions after a cold-plugging sequence. *Park mode* is specifically designed to enable the debugger to offer additional debug features. Refer to the *Programming Specifications* section for information on invoking and exiting this mode.

6.4.4. Boot Failure Signal

The Boot ROM provides a mechanism to indicate boot failure through a user-configurable Boot Failure Signal, enabled via the Boot External Notification I/O Pin (BOOT_GPIOSEL) fuse located in the Boot Configuration (BOOTCFG) fuse.

Users can select a specific I/O pin to serve as the fault indicator for their application. If a boot failure occurs (e.g., due to a fuse integrity check failure or invalid configuration), the Boot ROM asserts the selected I/O pin to signal the fault condition.

Note: By default, the Boot Failure Signal is **disabled** (BOOT_GPIOSEL = NONE). To enable this feature, users must program the BOOT_GPIOSEL fuse with a valid I/O selection. Refer to the *BOOTCFG - Boot Configuration Fuse* and *Pinout* chapters for details on available I/O options.

➔ **Important:** Once the Boot ROM transfers control to the first mutable application code, it no longer manages or controls the fault signal pin. The application is responsible for any further de-assertion of the signal and configuration of the pin.

6.4.5. Sequence Number Fuse Check

The Boot ROM reads the Sequence Number (SEQNUM) fuse in the Boot Configuration (BOOTCFG) memory section automatically upon any type of reset to determine the preferred boot configuration:

1. When BOOTCFG.SEQNUM is *not* 0xFFFFFFFF, the Boot ROM copies the contents of the BOOTCFG fuses to the respective peripherals registers.
2. When BOOTCFG.SEQNUM reads 0xFFFFFFFF, the device is booted using the factory defaults of the peripheral registers, and no pin is asserted for the Boot Failure Signal.

When the value of this fuse is not all-F, it can be used by the bootloader or application for version tracking, as a time stamp, or for similar purposes.

6.4.6. Debugger Usage

The Interactive mode (IMODE) serves for debugger interaction. In IMODE, the debugger can send commands and receive status messages and error codes.

Involuntary Entry to IMODE

IMODE is only entered involuntarily when a debugger is connected and the Boot ROM experiences any error and/or Fault during the boot operation that prevents the device from executing the first mutable application code. The device may encounter unsafe operating conditions due to various reasons, including but not limited to:

- Invalid or inconsistent device configuration
- Memory Faults
- Unintended or undefined device states

In such scenarios, the Boot ROM proactively halts the boot process and transitions the device into IMODE as a known safe state. See also the *Programming Specifications* section and the *DSU - Device Service Unit* chapter for details.
Note: When the Boot ROM encounters unsafe operating conditions and no debugger is connected, it will signal a Fault on a pin (if configured; see the *Boot Failure Signal* section) and wait for 100 ms before resetting the system.

Voluntary Entry to IMODE

This denotes when users have connected a debugger to the device and the device is ready to accept one of the supported Interactive mode commands. Voluntary entry into IMODE is typically done during product development or when debugging issues with a released product. Often, an IDE such as MPLAB® X serves as the user interface for issuing commands.

Debugger Interaction

Both entry methods require a cold-plugging sequence. See the *DSU - Device Service Unit* chapter for details. The debugger or programmer used must support IMODE and be compatible with this device. Compatible debuggers can be Microchip-designed, such as PICkit™ 5, ICE 4, ICD 5, or from a third party.

All compatible debuggers support commands to enter and exit IMODE, to manipulate the Device Access Level (DAL), and perform a full device erase.

The IMODE commands to read out configuration fuses and to run CRC on specific memory regions are considered advanced debugging features. As such, support for these commands is optional in debugger implementations, since they are not not intended for post-production use.

Refer to the *Programming Specifications* section for further details on the commands.

⚠ CAUTION

It is possible to lock commands, i.e., prohibit their execution on the device. The device may become **permanently unrecoverable** if the user does not retain access to recover paths. Refer to the *Programming Specification* section for more information.

6.4.7. Program and Debug interface Disable (PDID)

The Program and Debug Interface Disable (PDID) comprises of a number of measures and dependencies that prevent access to the device's reprogrammable Flash memory via programmer or debugger. After activating the PDID measures, a programmer or debugger is prevented from making any changes to the device through the DSU, but can still read out a restricted set of device information.

Follow these steps to inhibit Flash manipulation via a debugger or programmer:

1. Enter the Interactive Mode (IMODE) of the Boot ROM.
2. Disable the IMODE commands by writing their respective key value in the ROM Configuration (CFM::ROMCFG) to all-zeroes. This change will take effect only after the next device reset.
3. Issue the command `CMD_SDAL0` to set the Debugger Access Level to DAL0. This change will take effect only after the next device reset.
4. Reset the device so the previous measures take effect.

Now the following rules apply:

- A debugger can only access DSU registers mapped in the DSU external space (at offsets between 0x0100–0x01FF) and the DSU CoreSight™ ROM table. This allows device identification but inhibits read/write access to the PFM sections.
- The Chip Erase command (`CMD_CE_ALL`) is disabled, so it cannot force the factory default for neither Debugger Access Level (factory default: DAL2) nor command key values (factory default: all valid).
- Read or write operations on the application code can only be performed by code located in the Boot Code section (bootloader).

Note:
The bootloader software must be able to receive new data and to program the Application Code section. The bootloader cannot alter code stored in the Boot Code section, and the access to the Boot Code section by the DSU is restricted by the DAL0 setting.

The application authors must ensure that the bootloader implementation fulfills the security requirements.

⚠ CAUTION The device may become **permanently unrecoverable** if the user does not retain access to recover paths. Devices with PDID invoked will have extremely limited failure analysis capabilities.

6.4.8. Status and Error Codes

During the boot process or while interacting with a debugger, the device may emit a 32-bit code indicating either a status code (starting with `0x0000nnnn`) or an error code (`0xEEEEnnnn`). These codes are primarily intended to assist the host debugger in interpreting the device's state, especially during Interactive mode, and to confirm successful boot operation to the application.

6.4.8.1. Status Codes

Status codes report the current state or progress of the boot process. Most status codes are internal and/or used by the debugger to support IMODE commands.

Under normal operation, the code STATUS_BOOTOK (`0x00000004`) indicates that the entire boot sequence has completed successfully. If applicable, the Boot ROM writes the STATUS_BOOTOK code to the Boot ROM Channel (BCC) register in the DSU. The DSU.BCC register can be read by the first mutable application executable code to confirm that the device has booted correctly.

Note: If the boot operation fails, the device will not transfer control to the first mutable executable application code, and the STATUS_BOOTOK code will not be written. The device may remain in a recovery or Interactive mode.

6.4.8.2. Error Codes

During the boot process, the device may encounter error conditions that prevent it from completing initialization. In such cases, the Boot ROM issues a 32-bit error code, which is written to the DSU.BCC register. The DSU.BCC register retains the *last value written* by the device, allowing the user or debugger to inspect the cause of the boot failure.

Note: Error code values start with `0xEEEEnnnnn`. The presence of an error code indicates that the boot operation was interrupted and the first mutable application executable code was not executed.

6.5. Programming Specifications

The following sections provide the necessary information for programming and debugging the device without using the software and hardware tools provided by Microchip.

6.5.1. Reference Documents

Aside from the data sheet, refer also to the following documents:

- ARM®v6-M Architecture Reference Manual - ARM DDI 0419E
- AMBA 3 AHB-Lite Protocol Specification v1.0 - ARM IHI 0033A
- Debug Interface v5.2 Architecture Specification - IHI0031C
- ARM® CoreSight™ Architecture Specification v3.0 - ARM IHI 0029E
- CoreSight™ MTB-M0+ Technical Reference Manual - ARM DDI 0486B

6.5.2. Memory Map

Refer to the *Memory Map* chapter or use the `CMSIS *.SVD` files provided for each part when implementing your tools. In addition, these registers from the ROM Configuration (ROMCFG) fuse are relevant for programming and debugging the device:

Table 6-1. ROMCFG Key Values

Address	Name	Disables/Allows	Bit Field Values	Value after Chip Erase
`0x0D00_0018`	KEYVAL_CRC	`CMD_CRC` (check memory integrity)	`KEYVAL[31:0]`	`0xFFFF_FFFF`
`0x0D00_0020`	KEYVAL_CE_ALL	`CMD_CE_ALL` (Chip Erase)	`KEYVAL[31:0]`	`0xFFFF_FFFF`
`0x0D00_0028`	KEYVAL_SDAL0	`CMD_SDAL0` (Set Debugger Access Level to 0)	`KEYVAL[31:0]`	`0xFFFF_FFFF`

Notes: Decoding of `KEYVAL` bit field values:

- All bytes are 0xFF: Command is allowed
- All bytes are 0x00: Command is diabled

6.5.3. Device ID

The device is identified by reading the Device Identification (DID) register in the Device Service Unit (DSU):

Table 6-2. Device ID

Part Number	DSU.DID[31:0]		
	DSU.DID[31:28] (Silicon Revision ID)	DSU.DID[27:12] (Part Number)	DSU.DID[11:0] (Fixed)
CM6408PL10028	0x0	0xBA00	0x053
CM6408PL10032	0x0	0xBA01	0x053
CM6408PL10048	0x0	0xBA02	0x053
CM6408PL10064	0x0	0xBA03	0x053

Note: Refer to the register description in the *DSU - Device Service Unit* chapter for more details.

6.5.4. Device Reset

The device only supports Reset sources defined by the Arm® implementation. No additional vendor-specific Reset controllers are implemented.

6.5.4.1. System Reset

Writing `0x05FA_0004` to the AIRCR core register triggers SYSRESETREQ, which resets the entire chip, including all peripherals.

6.5.4.2. Vector Catch

The vector catch feature allows the CPU to halt after a Reset or other exceptions. Note that, since the PIC32CM PL10 family has a Boot ROM, the Reset vector catch cannot be used to prevent the CPU from executing the first instruction in the First Mutable Executable (FME), as the vector catch would halt the CPU on the first instruction in the Boot ROM while the device is locked (DAL = `0`). Instead, Park mode is used to halt the CPU and prevent it from executing the FME.

Refer to the *ARM®v6-M Architecture Reference Manual* for more details.

6.5.4.3. External Reset

A debugger can pull the external RESET pin low to reset the device. The application software can reconfigure the RESET pin to the General Purpose Input/Output (GPIO) function, in which case the Reset function is lost. This reconfiguration is performed by the application code, i.e., after the Boot ROM has completed, so Interactive or Park modes can be entered to prevent such RESET pin reconfiguration.

6.5.4.4. CPU Reset Extension

The DSU peripheral has CPU Reset and Boot ROM extension mechanisms that allow a debugger to be connected while preventing the CPU from starting execution of the customer application. This is also known as debugger cold-plugging.

This feature requires the debugger to control the SWCLK and $\overline{\text{RESET}}$ pins. The debugger forces a CPU Reset extension by driving three pulses on SWCLK when the $\overline{\text{RESET}}$ pin is low.

The main purpose of the CPU Reset extension is to avoid executing potentially corrupted code at the boot and to provide a safe way to connect to a device without making any assumptions about its current state. Combined with the Boot ROM Reset extension, the debugger controls the behavior of the Boot ROM Interactive and Park modes. This allows reading and writing of the memory and peripherals without CPU interference.

After the initial connection sequence, the debugger must control the CPU Reset and Boot ROM extension to maneuver the Boot ROM into Interactive or Park modes. Typically, Park mode is used to program the device using the ARM DAP AHB-AP port.

The CPU Reset Extension (bit 8) and Boot ROM Extension (bit 16) are located in the DSU Status A (DSU.STATUSA) register @0x4100_2104. Each of these bits can be cleared by writing a '`1`' to them. To release the CPU, write a '`1`' to the CRSTEXT0 bit in the STAUSA (STATUSA.CRSTEXT0) register. The BREXT0 bit in the STATUSA (STATUSA.BREXT0) register is typically written or left untouched within the same write operation, depending on whether entry into Interactive mode or Park mode is required. Writing a '`0`' to these bits has no effect.

6.5.4.5. CPU Boot

After leaving the Boot ROM, the Cortex-M0+ processor boots from a vector table at address offset '`0`' in the Program Flash Memory (PFM). The processor reads the Main Stack Pointer value from the address offset `0x0` and the code entry point (Reset vector) from address offset `0x4`.

6.5.5. SW-DP – Serial Wire Debug Port

This section describes how to access the Serial Wire Debug Port (SW-DP).

6.5.5.1. Overview

The Serial Wire Debug (SWD) protocol is described in Chapter 5 of *Arm® Debug Interface v5 Architecture Specification (ARM IHI 0031 Revision x*).

The exact revision of the document is not specified by Arm.

6.5.5.2. Operation Sequences

To initialize or uninitialize the SWD, follow the Debug Interface specification (ARM® IHI 0031).

Refer also to the *Programming* and *In-Circuit Debugging* sections for more details.

6.5.5.3. Debugger Features

The Device Service Unit (DSU) supports debugger features required by the Arm CoreSight specification, namely a CoreSight ROM table, as previously described. It also extends the debugger functionality by supporting:

- The debugging and testing of a protected or protectable device
- Debugger interface pin allocation
- Other debug and test features that add value to the device

The debugger interfaces with the device through the Debug Access Port (DAP). In addition to its use in Arm's CoreSight debug architecture, the Serial Wire Interface can also be used for production testing, validation and internal debugging.

6.5.5.3.1. Debugger Routing

The debugger explicitly selects the target AHB-AP in its request. AHB-AP0 is connected to the CPU's (the Cortex M0+) AHB bus.

All debugger transactions addressed to a Private Peripheral Bus (PPB) peripheral remain within the CPU. The PPB is where the ARM embedded debug components reside. All debugger transactions are routed to the CPU's main bus, where they can be forwarded to other peripherals of the system via the AHB bus matrix.

6.5.5.4. Debugger Security

The DSU implements security features to prevent regular access to a protected device through the debug interface, thereby protecting sensitive data from being read and user code from reverse engineering.

6.5.5.4.1. DAL – Debug Access Level

The current Debug Access Level (DAL) is indicated in the CPU0 bit field of the DSU.DAL register.

The DAL setting controls debugger access to everything in the system. The lower the DAL value, the more restricted the debugger access is.

Table 6-3. DAL Settings

DAL[1:0]	Access
0x3	Reserved, do not use
0x2	The debugger can access all addresses accessible from CPU0
0x1	Reserved, do not use
0x0	The debugger can only access the DSU addresses in the external address range, i.e., the DSU External Space Address Registers (offset 0x0100–0x01FF) and the DSU CoreSight™ ROM table (offset 0x1000–0x1FFF)

It is possible to set DAL < 0x2 and also prevent this from being changed by disabling the interactive mode command CMD_CE_ALL. See also the *Program and Debug interface Disable (PDID)* section of the Boot ROM chapter.

6.5.5.4.2. DSU Register Access

The DSU registers can be accessed through its APB bus interface.

When DAL < 0x2, only DSU registers mapped in the DSU external space (at offsets between 0x100 and 0xFFF, and the DSU ROM table) are accessible.

6.5.5.4.3. Debugger Detection

The DSU detects the presence of the debugger and supports both hot plugging and cold plugging:

- Cold plugging detection occurs if at least three SWCLK pulses are detected before the $\overline{\text{RESET}}$ pin is de-asserted. The device must be out of Power-on Reset (POR) before these pulses can be detected.
- Hot plugging detection occurs on a falling edge of SWCLK when the part is not in Reset. An internal pull-up on SWCLK prevents false detection if SWCLK is unconnected. Hot plugging is not available when DAL = 0x0 or if the pin that SWCLK is on has been switched to a different function.

Refer to the *CPU Reset Extension* section for more information on hot- and cold-plugging. Once the debugger is detected, the SWDIO function takes control of the pin to which it is assigned and can only be released back for other usage by a Power-on Reset (POR) in the case of cold plugging, or by either a POR or external reset in the case of hot plugging.

6.5.5.5. DSU Register Write Protection

Several DSU registers with write access, except for DCC0/1 and BCC0/1, are write-protected by the WPCTRL mechanism.

6.5.5.6. Serial Wire Interface

Table 6-4. Serial Wire Signals

DIR	Signal	Description
I	SWCLK	Clock input
I/O	SWDIO	Bidirectional serial data

The devices in the PIC32CM PL10 family support Arm's CoreSight Serial Wire Interface. This interface consists of two pins: SWCLK, a clock signal that is always an input to the device, and SWDIO, a bidirectional data pin. This interface provides access to all of Arm's customer-visible debug features.

This section summarizes the critical details of this interface for production test and validation.

Refer to the *Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2 (IHI 0031)* for more details.

6.5.5.6.1. Architecture

As defined by Arm, the Serial Wire Interface connects from external pins to a Debug Port (DP). The Debug Port handles communication and synchronization between the external host, debugger, or tester and the target/device. The Debug Port then connects to one or more Access Ports (AP) to communicate with other resources within the device.

6.5.5.6.2. Debug Port

The SW-DP handles communication and synchronization between the external host, debugger, or tester and the target/device using the Serial Wire Interface. Each transaction can access either a DP or AP register. The devices in the PIC32CM PL10 family implement a DPv2-compliant DP.

6.5.5.6.3. Memory Access Port

Essentially, a MEM-AP is a bus host on a bus interface. It is accessed and controlled by reading and writing to its register interface using the DP. Any memory-mapped resource that is accessible from the bus driven by this MEM-AP can be written or read from, following the normal access permissions of that resource.

The devices in the PIC32CM PL10 family contain one MEM-AP, which is an AHB-AP.

The AHB-AP connects through the CPU's AP and to the AHB bus matrix. The MEM-AP can access all subsystem memory-mapped resources in the design. The only exception to this is when the device is protected (DAL < `0x2`); in this case, the DSU performs security filtering as described in the *Debugger Security* section.

6.5.6. Boot ROM for Programming and Debugging

The Boot ROM plays an essential part during device programming and debugging, as illustrated below.

Figure 6-3. Boot Flow Overview

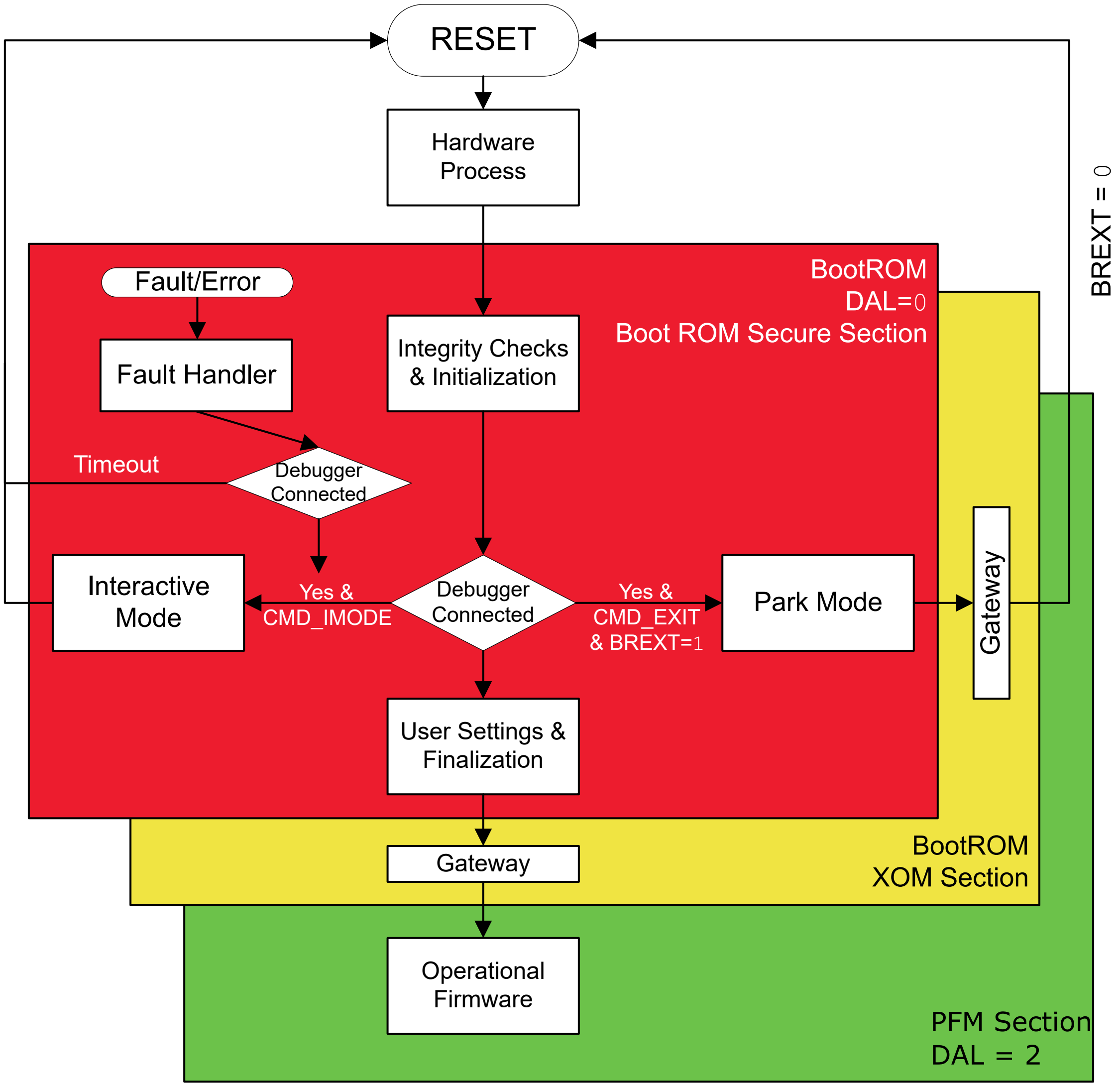
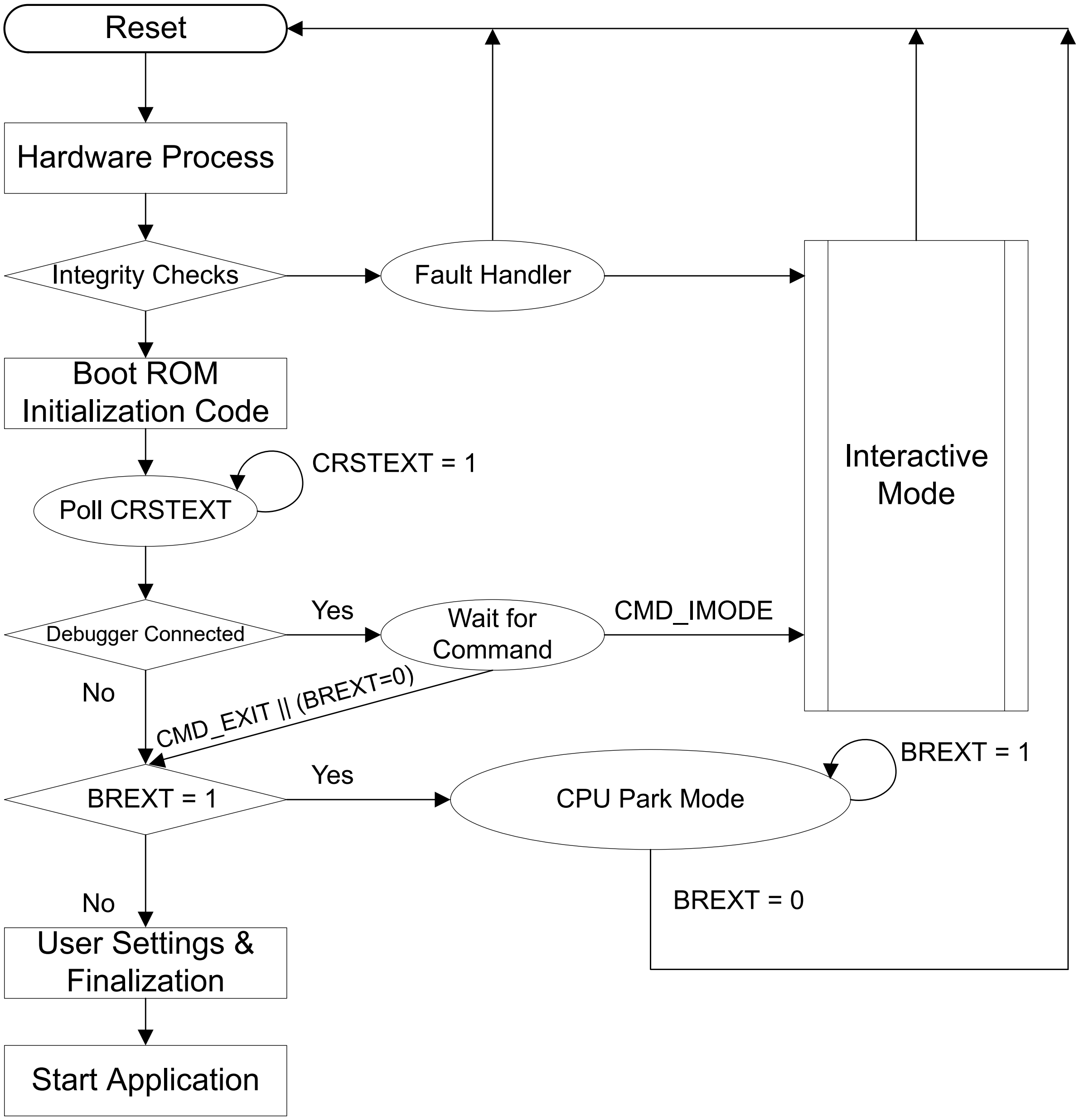


Figure 6-4. Boot ROM Detail



6.5.6.1. Integrity Checks and Initialization

The Boot ROM initializes the device, applies calibration data, and starts clocks.

Detected errors are handled as noted in the *Error Handling* section.

6.5.6.1.1. Hardware CRC of PFM on Boot

The CRC table is listed in Table 6-13. This table is located at `ENDADDR(CFM::BOOTCFG.BOOTPROT.BOOTPROT)-16`. If `BOOTPROT==7` (0 bytes), then the CRC table is located at `ENDADDR(PFM)-16`.

6.5.6.1.2. Error Handling

When the Boot ROM detects an error in the boot flow or a hardware fault, it writes a code as indicated in Table 6-10 to BCC1 and jumps to a fault handler.

As part of Power-On Self Test (POST), the Boot ROM checks for faults in the hardware and memories it uses. If a fault is detected, the Boot ROM enters the fault handler.

- If a debugger is not connected and the user has selected a GPIO configuration in `CFM::BOOTCFG.BOOT_GPIOSEL`, the Boot ROM attempts to apply those settings. It then waits 100 ms (so the error status can be observed) before resetting the device to check if the error is transient.
- If a debugger is connected, the Boot ROM does not apply GPIO settings. It waits indefinitely for the debugger to read status and send commands:
  - If the command is `CMD_IMODE`, the Boot ROM enters Interactive mode
  - If the command is `CMD_EXIT`, the Boot ROM resets the device
- If the Boot ROM is already in Interactive mode when it detects an error, it puts the status in BCC1 and returns to the Wait-for-Command state. The GPIO selections are not applied in this case.
- If POST finds no faults, the Boot ROM continues normal operation (Boot, Interactive, and Park modes). Therefore, a hardware or memory error occurring during normal operation is a failure condition and causes a hard fault. A hard fault is an unrecoverable error that triggers a device Reset, regardless of whether a debugger is connected. Interactive mode is not available in this situation.

6.5.6.2. Debugger Communication Channels

The DSU provides two pairs of registers for debugger and device communication as detailed in the following subsections.

6.5.6.2.1. Debug Communication Channels

The DSU provides two registers: Debug Communication Channel x (DCC0 and DCC1) for communication and synchronization between the debugger and the CPU. Each register has a corresponding dirty bit, DCCD0 and DCCD1, respectively. These two dirty bits are set on a write and cleared on a read by hardware, and they reside in DSU.STATUSB register.

Customers can use these registers to build their own communication protocol.

6.5.6.2.2. Boot Communication Channels

The DSU also provides two additional registers, BCC0 and BCC1, for communication between the debugger and CPU while the Boot ROM is executing. These registers map to the DCC0/1 registers but have their own dirty bits. These dirty bits also reside in the DSU.STATUSB register and are called BCCD0 and BCCD1. These bits were added to ensure that the usage of the Boot ROM does not interfere with DCCx communication.

6.5.6.3. Boot ROM Operation During Programming and Debugging

The Boot ROM supports two modes. Refer to Figure 6-4:

1. **CPU Park mode:** In this mode, the core is halted in sleep mode, so it does not access any peripheral registers or memories. Therefore, the debugger can safely program the device.
2. **Interactive mode:** In this mode, the Boot ROM waits for commands and executes them.

The core never needs to be halted through the standard Cortex-Mx (DHCSR, DEMCR, AIRCR) registers. Park mode is used instead.

To run the Boot ROM in Interactive mode, a special cold-plug Reset sequence is used (refer to the *CPU Reset Extension* section). This process pauses the Boot ROM execution after the Boot ROM has initialized the device. To end the Reset extension, the debugger must clear the CRSTEXT bit in the DSU.STATUS register. The bit is cleared by writing a ‘`1`’ to it.

After some time (5 ms is recommended), the debugger must check if the Boot ROM has flagged any errors by reading BCC1. At this point, the debugger may exit the application by clearing the BREXT bit and sending the `CMD_EXIT` command. If `CMD_EXIT` is issued while the BREXT bit is set, the Boot ROM will enter Park mode and the response code will be set to `STATUS_BOOTOK`.

Alternatively, by leaving the BREXT bit set and waiting for the Boot ROM to set STATUS_INITCHECK_OK, the debugger may issue `CMD_IMODE` to enter the Interactive mode. Once the Boot ROM reports `STATUS_OK`, the device is in Interactive mode. In this mode, the Boot ROM is ready to accept other commands.

6.5.6.4. Interactive Mode

The Boot ROM provides interactive debugging capabilities to facilitate chip erase operations on protected devices (DAL < `0x2`). This functionality is also available on unprotected devices.

It also allows the debugger to request a Reset. During interactive debugging, the host, debugger, and the target/device utilize the Boot Communication Channels to send a limited and predetermined set of commands and responses between each other. Refer to Boot Communication Channels for more information.

By convention, BCC0 is always used to send commands from the host to the target while BCC1 is always used to send responses from the target to the host.

6.5.6.4.1. Entering Interactive Mode

The procedure in Table 6-5 must be performed to enter Interactive debug mode unless an integrity check fails. In that case, the device will automatically enter Interactive debug mode. If there is a cold-plug detection of the debugger, the procedure in Table 6-5 must be followed, regardless of whether the user wants to enter Interactive debug mode.

Table 6-5. Debugger Cold Plug Start-Up Procedure

Step	Host/Debugger	Target/Device
1	Assert $\overline{\text{RESET}}$ pin	Device is held in Reset
2	Drive at least three pulses on the SWCLK pin; de-assert $\overline{\text{RESET}}$ pin	Cold-plugging: The debugger is detected, and the Boot ROM starts the system start-up sequence
3	Wait enough time to guarantee that system initialization has completed	The CPU performs mandatory integrity checks and, if any fail, carries out error handling as described in the *Error Handling* section
4	SW: Set DSU.STATUSA.CRSTEXT to `1'b1` to clear the CPU Reset Extension	The CPU continues executing from the Boot ROM
5	Wait for DSU.STATUSB.BCCD1 = `1'b1`	
6	SW: Read DSU.BCC1 for the response code to check if any errors were logged	CPU polls DSU.STATUSB.BCCD0 = `1'b1`

After this start-up procedure, the device waits for a command from the host. If no errors are reported, there are only two valid procedures to continue the process.. The host can follow the procedure indicated in Table 6-6 to enter Interactive debug mode or follow the procedure in Table 6-7 to skip it. If any other command is written to DSU.BCC0, the Boot ROM will simply ignore it and continue waiting for one of these commands.

Table 6-6. Interactive Debug Mode Entry Procedure

Step	Host/Debugger/Tester	Target/Device
7	SW: Write DSU.BCC0 with `CMD_IMODE`	The CPU polls DSU.STATUSB.BCCD0 = `1'b1` (continued from Step 6 in Table 6-5)
8	Wait for DSU.STATUSB.BCCD1 = `1'b1`	The CPU reads DSU.BCC0 for the command code and acknowledges `CMD_IMODE` by writing STATUS_CMD_VALID to DSU.BCC1
9	SW: Read DSU.BCC1 for the response code to confirm entry into Interactive debug mode	Device enters Interactive debug mode

Table 6-7. Interactive Debug Mode Skip Procedure

Step	Host/Debugger/Tester	Target/Device
7	SW: Write DSU.BCC0 with `CMD_EXIT`	The CPU polls DSU.STATUSB.BCCD0 for the command code (continued from Step 6 in Table 6-5)
8	Wait for DSU.STATUSB.BCCD1 = `1'b1`	The CPU reads DSU.BCC0 for the command code and acknowledges `CMD_EXIT` by writing STATUS_BOOTOK to DSU.BCC1
9	SW: Read DSU.BCC1 for the response code to confirm Boot ROM exit	The CPU completes Boot ROM execution and enters Park mode

If Interactive debug mode is skipped, the normal operation of the Boot ROM continues until it enters Park mode. Refer to the *CPU Reset Extension* section for details on Boot Entry.

Once in Interactive debug mode, an external Reset or the `CMD_EXIT` command can be used to exit this mode.

6.5.6.4.2. Interactive Mode Command Access Controls

The Command Access controls for non-secure devices do not provide secure, authenticated use of the command, even though they use the same protocol as secure devices. The protocol uses a sequence to support command lock selection based on a KeyVal field associated with each command. Refer to the *Memory Map* section in this chapter for available KeyVal registers.

6.5.6.4.3. Challenge-Response

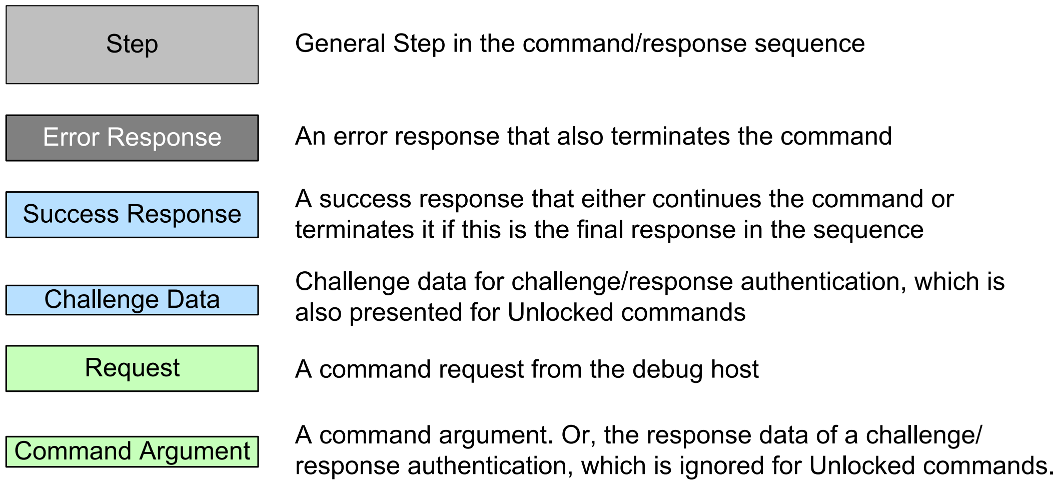
The standard protocol for the PIC32CM PL10 devices uses a challenge-and-response mechanism that supports both unlocked and authenticated commands. In this case, the host sends a command, and the device responds with STATUS_CHALLENGE. The device then sends 128 bits of challenge data and waits for the 256 bits of response data. The host then performs an HMAC to generate the response.

⚠ CAUTION This challenge-response protocol (i.e., KeyVal = Authenticated) is not supported by the PIC32CM PL10 family: These devices do not check the response and always return a status of STATUS_CMD_VALID. They will send any challenge, and the host can therefore send any response.

When the command is set to Locked, the device responds with STATUS_ARG_INVALID. The device does not process the command any further, and it waits for another command.

6.5.6.4.4. Command Diagram and Definition Legend

Figure 6-5. Command Message Sequence Legend



Step	General Step in the command/response sequence
Error Response	An error response that also terminates the command
Success Response	A success response that either continues the command or terminates it if this is the final response in the sequence
Challenge Data	Challenge data for challenge/response authentication, which is also presented for Unlocked commands
Request	A command request from the debug host
Command Argument	A command argument. Or, the response data of a challenge/ response authentication, which is ignored for Unlocked commands.

Command Definition tables use the following column headers:

- Type: The type of information that flows during a part of the command sequence
- Input: Information from the debugger. This precedes the output, except in the case of challenge and response.
- Size: The size of non-status information passed between the debugger and the device
- Range: The valid range of the input information
- Output: Either STATUS or the information requested by the COMMAND

6.5.6.4.5. Interactive Mode Commands

The PIC32CM PL10 devices support the debug commands listed in Table 6-8. All commands, except for `CMD_IMODE` and `CMD_EXIT`, can only be used once the device is in Interactive debug mode. The `CMD_IMODE` and `CMD_EXIT` commands are used to enter or exit the Interactive debug mode.

All the commands listed in Table 6-8, except for the `CMD_IMODE` and CMD_EXIT, initiate an Interactive debug operation. The descriptions and procedures for these commands are provided below. These procedures assume that the device has already entered Interactive debug mode. Upon completion of any of these commands, the Boot ROM will wait for the next command. If, for any reason, the Boot ROM does not properly recognize a command, it will return STATUS_CMD_INVALID instead of STATUS_CMD_VALID.

Table 6-8. Interactive Mode Commands

Name	Description	Value
`CMD_IMODE`	Enter Interactive mode	0x444247 55
`CMD_EXIT`	Jump to exit section	0x444247 AA
`CMD_READCFM`	Read a configuration memory word	0x444247 4C
`CMD_CRC`	Check the integrity of memory using the Cyclic Redundancy Check (CRC) table	0x444247 C0
`CMD_CE_ALL`	Erase Flash memory, wipe volatile memories, and set the device to DAL2	0x444247 E3
`CMD_SDAL0`	Set the device to DAL0	0x444247 10

Table 6-9. Interactive Mode Completion Status

Name	Definition	Code
STATUS_CRC_FAIL	The CRC of memory area doesn’t match the expected value in the table	0x00000001
STATUS_CRC_OK	The CRC of memory area matches the expected value in the table	0x00000002
STATUS_INITCHECK_OK	Boot initialization complete, Interactive mode may be called	0x00000003
STATUS_BOOTOK	Entire boot sequence complete, DAL is set to the final value	0x00000004
STATUS_CMD_VALID	The previously transmitted command is valid and command flow continues	0x00000005
STATUS_ARG_INVALID	Unexpected or incoherent argument submitted to command or a locked command	0x00000006
STATUS_DATA_VALID	Boot ROM has valid data to output to debugger	0x00000008
STATUS_OK	Operation or command sub-sequence succeeded	0x00000009
STATUS_CHALLENGE	Prepare to accept an authentication challenge	0x0000000B

Table 6-10. Interactive Mode Error Status

Name	Definition	Interactive Mode Available	Code
STATUS_ERR_HARDFAULT	Hard fault occurred	No	0xEEEE0001
STATUS_ERR_CALOTP	Cal-OTP page integrity check failed	Yes	0xEEEE0002
STATUS_ERR_BOOTCFG	BOOTCFG page integrity check failed	Yes	0xEEEE0004
STATUS_ERR_EXEC	Boot ROM execution error, program flow diverted	No	0xEEEE0006
STATUS_ERR_PROG	Flash programming error	Yes	0xEEEE0007
STATUS_ERR_ECC	Double error detected on Flash while running the command	Yes	0xEEEE0008
STATUS_ERR_BUS	Bus error occurred on access to a system address	Yes	0xEEEE000C
STATUS_ERR_CRCTABLE	Incorrect CRC table format/values	Yes	0xEEEE000D
STATUS_ERR_ERASE	Erasing of Flash memory failed	Yes	0xEEEE000F
STATUS_ERR_FUSEUP	Hardware fuse update compare check failed	Yes	0xEEEE0011
STATUS_ERR_ROMCFG	ROMCFG page integrity check failed (Only `CMD_READCFM` is guaranteed to be available. Other commands may be disabled or their KEYVAL is in error.)	Yes	0xEEEE0019
STATUS_ERR_PFMCRC	Program Flash Memory integrity check failed	Yes	0xEEEE001A

Command `IMODE`

This command is used to enter Interactive mode, as noted above.

Type	Input	Size	Range	Output	Comment
Command	`CMD_IMODE`	32-bits		STATUS_CMD_VALID	Refer to Table 6-9

Command `EXIT`

This command is used to proceed to the Exit section, where the selection of BREXT determines entry into either Mission Debug mode or Park mode.

Type	Input	Size	Range	Output	Comment
Command	`CMD_EXIT`	32-bits		STATUS_CMD_VALID	Refer to Table 6-9

Command `READCFM`

This command reads an address in the `NVMCTRL::CFM` space and returns the data. If the address is read-protected or out of range, the device returns an invalid argument status and waits for another command. If the address is within range, the command succeeds with data valid status, followed by the data.

This procedure allows one 32-bit value to be read at a time, but continues accepting target addresses until an address outside the allowable range is requested. By convention, write a target address of `32’h0000_0000` when you intentionally want to exit this procedure.

The Table 6-11 defines the command structure, and Figure 6-6 shows the communication sequence.

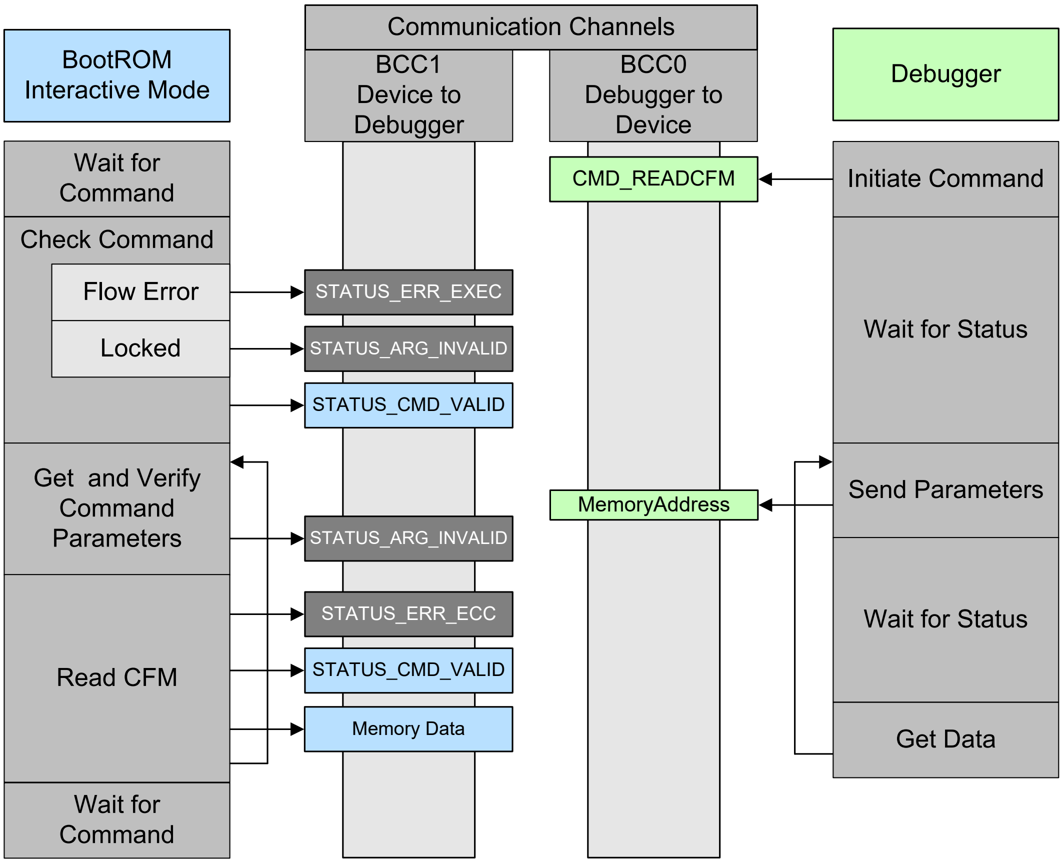
Table 6-11. CMD_READCFM Definition

Type	Input	Size	Range	Output	Comment
Command	`CMD_READCFM`			STATUS_ERR_EXEC STATUS_ARG_INVALID STATUS_CMD_VALID	Refer to Table 6-9 and Table 6-10
Argument1	Address	32 bits	Refer to Table 6-12	STATUS_ARG_INVALID	The command is aborted on an address error. This is how the command is intentionally ended. Refer to Table 6-9
Result				STATUS_ERR_ECC STATUS_DATA_VALID	Refer to Table 6-9 and Table 6-10
Returns		32 bits		Data	The command awaits a new Address after a valid read

Table 6-12. READ_CFM Valid Addresses

Region	Description
`CFM::SIGNATURE`	Firmware metadata
`CFM::BOOTCFG`	User configuration

Figure 6-6. CMD_READCFM Message Sequence



Command CRC

This command performs a CRC32 on a range of Flash memory as specified by a CRC table that exists in that region. Applicable Flash regions are those listed in the device's "System Address Map" that start with `NVMC::`. The CRC tables are user-programmable, and multiple tables can exist in a region. Tables must be aligned to a 16-byte address boundary.

These restrictions apply as security measures:

- The CRC command will not return the result of the CRC; it will only indicate whether the value matched or mismatched the expected value
- The region of memory to check and its expected CRC must be stored internally in a table. Therefore, the regions that may be tested must be planned in advance, and their expected CRCs must be calculated beforehand The debugger does not provide the expected CRC value; instead, it provides the starting location of one of these tables. As a result, the user must know where these tables are stored. Refer to Table 6-13 for more information on the CRC tables.
- There is a 128-bit fuse value (KeyVal*) used as a key to unlock the CRC command. If KeyVal is set to all '1's, no key is required; if it is set to all '0's, this functionality is disabled.

Table 6-13. General CRC Table Format

Word	Offset	Name	Value/Format	Description
0	0x0	HDR	0x43524349	Constant, denotes the CRC header
1	0x4	ADDR	0x0C000000	Start Address, must be word-aligned (four bytes)
2	0x8	SIZE	0x100	Size in bytes, but must be a multiple of four. The maximum size is the largest Flash region.