PIC32CX-BZ2 and WBZ45 Family

High-performance 2.4 GHz Multi-protocol Wireless MCUs and Modules, supporting Bluetooth Low Energy and 802.15.4 protocols with 32-bit ARM® Cortex®-M4F, 2 Msps 12-bit ADC Data Sheet

Introduction

The PIC32CX-BZ2 family is a general purpose, low-cost, 32-bit Microcontroller (MCU) family of devices supporting multi-protocol wireless interfaces (Bluetooth® and Zigbee®), hardware-based security accelerator, transceiver and Power Management Unit (PMU).

The WBZ45 Module is a series of fully RF-certified wireless modules that contain the PIC32CX-BZ2 wireless MCUs with the following antenna options:

- PCB Antenna
- u.FL Connector for External Antenna

In addition to the Bluetooth Low Energy (Bluetooth 5.2) and Zigbee (Zigbee 3.0) wireless protocol support, the PIC32CX-BZ2 devices and the WBZ45 modules also support a rich set of standard MCU peripherals, such as Analog-to-Digital Converter (ADC), Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2C), Quad I/O Serial Peripheral Interface (QSPI), Universal Asynchronous Receiver-Transmitter (UART) and Serial Wire Debug (SWD).

PIC32CX-BZ2 SoC Family Features

The following section lists the PIC32CX-BZ2 SoC related features:

Operating Conditions of MCUs

- 1.9V to 3.6V, -40°C to +125°C, DC to 64 MHz
  - AEC Q100 Grade 1 qualified

Core: 64 MHz ARM® Cortex®-M4

- 3.35 Coremark®/MHz
- 4 KB Combined Instruction Cache and Data Cache
- 8-Zone Memory Protection Unit (MPU)
- Thumb®-2 Instruction Set
- Digital Signal Processing ASE Rev 2
- Nested Vector Interrupt Controller (NVIC)
- Embedded Trace Module (ETM) with Instruction Trace Stream
- Core Sight Embedded Trace Buffer (ETB)
- Trace Port Interface Unit (TPIU)
- IEEE® 754-Compliant Floating Point Unit (FPU)

Memories

- 1 MB On-Chip Self-Programmable Flash with:
  - Error Correction Code (ECC)
  - Prefetch module to speed up Flash accesses
  - 20-years of data retention support
- 32 KB NVR Flash (8 Sectors)
  - Private/public boot code and data
  - NV calibration data
- 128 KB Multi-Port Programmable QoS SRAM Main Memory
  - 64 KB of Error Correction Code (ECC) RAM option
- Up to 4 KB of Tightly Coupled Memory (TCM)
- Up to 8 KB Backup SRAM
- Single 32-bit Backup Register

System

- Power-on Reset (POR) and Brown-out Detect (BOD)
- Internal and External Clock Options
- External Interrupt Controller (EIC)
- Up to four External Interrupts
- One Non-maskable Interrupt
- 2-pin Serial Wire Debug (SWD) Programming, Test and Debugging Interface

Supported Connectivity Standards

- Complies with:
  - Bluetooth v5.2
  - IEEE 802.15.4, Zigbee 3.0
  - ETSI EN 300 328 and EN 300 440 Class 2
  - FCC CFR47 Part 15 and ARIB STD-T66

Power Supply

- Integrated PMU with:
  - Buck (DC-DC/switching) mode; supports High Power (PWM) and Low Power (PSK) mode
  - MLDO (linear) mode
- On-board 1.2V Voltage Regulator (CLDO)
- Power-on Reset (POR) and Brown-out Detect (BOD) on 3.3V and 1.2V Rails
- Run, Idle, Dream, Sleep, Deep Sleep and Extreme Deep Sleep Modes
- Sleep Walking Peripherals
- Embedded Buck/LDO Regulator Supporting On-the-Fly (OTF) Selection

2.4 GHz RF Transceiver

- Integrated 2.4 GHz Ultra-low Power RF Transceiver shared among Bluetooth, Zigbee Modems, Link (MAC) Controllers and Proprietary Modulation Schemes
- Integrated 16 MHz ±20 ppm Crystal Oscillator (External Low Cost Crystal)
- Two PA Design Architecture (LPA (+4 dBm) and MPA (+12 dBm)) to Improve TX Power Efficiency

- Low RBOM Two-port TRX RFFE Architecture
  - Integrated balun (single-ended RF output) and TRX switch
- Hardware Radio Arbiter with Programmable QoS:
  - Resolution per packet level
  - Time-division coexistence between Bluetooth and 802.15.4
  - Based on shared transceiver and antenna
  - Maintains connections of 802.15.4 and Bluetooth simultaneously

Bluetooth

- Bluetooth Low Energy 5.2 Certified
- Up to +12 dBm Programmable Transmit Output Power
- Typical Receiver Power Sensitivity:
  - -95 dBm for Bluetooth Low Energy 1 Mbps
  - -92 dBm for Bluetooth Low Energy 2 Mbps
  - -102 dBm for Bluetooth Low Energy 125 Kbps
  - -99 dBm for Bluetooth Low Energy 500 Kbps
  - Digital RSSI indicator (-50 dBm to -90 dBm)
- Bluetooth Supported Features:
  - 2M uncoded PHY
  - Long range (Coded PHY)
  - Channel selection algorithm #2
  - Advertising extensions, offloads CPU with hardware-based scheduler
  - High duty cycle non-connectible advertising
  - Data length extensions
  - Secure connections
  - Privacy upgrades (with hardware white-list support)
- ECDH P256 Hardware Engine for Link Key Generation when Bluetooth Pairing
- AES128 Hardware Module for Real-Time Bluetooth Payload Data Encryption
- Preprogrammed MAC Address
- Bluetooth Qualification Test Facility (BQTF) Certification
- Bluetooth Low Energy Profiles:
  - Bluetooth Low Energy peripheral and central roles
  - Bluetooth Low Energy APIs for application layer to implement standard or customize GATT based profiles/services
  - Microchip Transparent UART Service
  - Battery Service
  - Device Information Service
  - Custom Service
  - Multi-link and multi-role
- Bluetooth Low Energy Services:
  - Provisioning
  - Over-the-Air (OTA) update (also known as DFU)
  - Advertisement/Beacon
  - Personalized configuration
  - Alert notification service

802.15.4/Zigbee Modulation Scheme

- 802.15.4/Zigbee Physical Layer Service Unit (PSDU) Data Rate: 250 Kbps

- Programmable RX Mode
  - -103 dBm RX sensitivity in the Continuous mode
  - -98 dBm sensitivity in RPC mode
  - RPC mode provides lower power consumption in RX mode to support California Green Energy Specification at the system level
- TX Output Power up to +12 dBm
- Hardware Assisted MAC
  - Auto acknowledge
  - Auto retry
  - Channel access back-off
- Preprogrammed MAC Address
- SFD Detection; Spreading; De-spreading; Framing; CRC-16 Computation
- Independent TX/RX Buffers for Improved CPU Offloading While Handling Zigbee Data
  - 128-byte TX and 128-byte RX frame buffer
- Hardware Security
  - Advanced Encryption Standard (AES)
  - True Random Number Generator (TRNG)
- Zigbee Stack Support
  - Zigbee 3.0 ready
  - Zigbee Pro 2017
  - Zigbee green power support (proxy, sink and multi-sensor)

Proprietary

- Proprietary Data Rates: 500 Kbps, 1 Mbps and 2 Mbps
- TX Output Power up to +12 dBm
- Receiver Sensitivity up to -96 dBm
- Hardware Assisted MAC
  - Auto acknowledge
  - Auto retry
  - Channel access back-off
- SFD Detection; Spreading; De-spreading; Framing; CRC-16 Computation
- Independent TX/RX Buffers for Improved CPU Offloading While Handling Zigbee Data
  - 128-byte TX and 128-byte RX frame buffer
- Hardware Security
  - Advanced Encryption Standard (AES)
  - True Random Number Generator (TRNG)

High Performance Peripherals

- 16-Channel Direct Memory Access Controller (DMAC)
  - Built-in CRC with memory CRC generation/monitor hardware support
- One Quad I/O Serial Peripheral Interface (QSPI)
  - execute-In-Place (xIP) support
  - Dedicated AHB memory zone

System Peripherals

- 32-Channel Event System

  - All channels can be connected to any event generator
  - All channels provide a pure asynchronous path
  - Twelve channels support synchronous and re-synchronous
- Four Serial Communication Interfaces (SERCOM), Each Configurable to Operate as:
  - USART with full-duplex and single-wire half-duplex configuration
  - ISO7816
  - I^2C up to 1 MHz (three SERCOMs support I^2C)
  - LIN Commander/Responder
  - RS485
  - SPI inter-byte space
- Four 16-bit Timers/Counters (TC), Each Configurable as:
  - 16-bit TC with two compare/capture channels
  - 8-bit TC with two compare/capture channels
  - 32-bit TC with two compare/capture channels
- Two 24-bit Timer/Counters for Control (TCC) with Extended Functions:
  - Up to six compare channels with optional complementary output
  - Generation of synchronized Pulse Width Modulation (PWM) pattern across port pins
  - Deterministic fault protection, fast decay and configurable dead-time between complementary output
  - Dithering that increases resolution with up to 5 bits and reduces quantization error
- One 16-bit Timer/Counters for Control (TCC) with Extended Functions:
  - Up to two compare channels with optional complementary output
- 32-bit Real Time Counter (RTCC) with Clock/Calendar Function
- Up to one Wake-up Pin with Tamper Detection and Debouncing Filter
- Watchdog Timer (WDT) with Window Mode
- Deadman Timer (DMT)
- CRC-32 Generator
- Frequency Meter (FREQM)
- Two Configurable Custom Logic (CCL)
- One Analog Comparator (AC) with Window Compare function
- One Temperature Sensor (Die Temperature)

Advanced Analog

- 12-bit ADC SAR Module (ADC):
  - Up to Eight Analog Channels
  - Up to Two MSPS conversion rate
  - Multiple trigger sources
  - Supports die temperature sensor built into RF-Analog (not an external “ambient” temperature sensor)
  - Two Analog Comparator (AC) with Window Compare Function or single Analog Comparator
  - One dedicated AC and second AC is shared with PMU Controller

Security

- AES Engine with Support for 128/192/256-bit Cryptographic Key
- One AES with 256-bit Key Length and up to 2 MB/s Data Rate
  - Five confidential modes of operation (ECB, CBC, CFB, OFB and CTR)
  - Supports counter with CBC-MAC mode
  - Galois Counter Mode (GCM)
- True Random Number Generator (TRNG)

- Public Key Cryptography Controller (PUKCC) and Associated Classical Public Key Cryptography Library (PUKCL)
  - RSA and DSA algorithm
  - Elliptic Curves Cryptography (ECC), ECC GF (2n) and ECCGF (p)
- Integrity Check Module (ICM) Based on Secure Hash Algorithm (SHA1, SHA224 and SHA256), DMA Assisted

Oscillators

- 16 MHz, ±20 PPM Crystal/Resonator Oscillator or External Clock (POSC) for 2.4G RF Transceiver
- Shared System PLL with Bluetooth/Zigbee RF Data Converter PLL
- 32.768 kHz Ultra-low Power Internal Oscillator
- Higher Accuracy 32.768 kHz, ±250 ppm Clock Options
  - POSC derived 32 kHz clock
  - 32.768 kHz crystal/resonator oscillator (SOSC)
  - External 32.768 kHz clock source
- 8 MHz Internal RC Oscillator (FRC)

I/O

- Flexible Peripheral Pin Select (PPS) Support
- High-Current Sink/Source on Most I/O Pins
- Configurable Open-Drain Output on Digital I/O Pins
- Up to 27 Programmable I/O Pins

Package

- PIC32CX1012BZ25048 Package:
  - 48-pin QFN
  - Size – 7 mm x 7 mm x 0.9 mm
- PIC32CX1012BZ25032 Package:
  - 32-pin QFN
  - Size - 5 mm x 5 mm x 1 mm

WBZ45 Module Features

The WBZ45 modules are wireless MCU modules with BLE 5.2 compliant and Zigbee 3.1 Radio.

The following section lists the WBZ45 Module related features, which complement SoC features:

WBZ45 Module Variants

- WBZ451 based on (PIC32CX1012BZ25048 SoC)
  - WBZ451PE (PCB)
  - WBZ451UE (u.FL)
- WBZ450 based on (PIC32CX1012BZ25032 SoC)
  - WBZ450PC (PCB)
  - WBZ450UC (u.FL)
  - WBZ450PE (PCB)
  - WBZ450UE (u.FL)

Antenna

- On-Board PCB Antenna
- External Antenna

Clock Management

- Integrated 16 MHz POSC

System Peripheral, Advanced Analog and Security

- All features of SoC are accessible

Package and Operating Conditions

- WBZ451
  - 39-pin SMD package with Shield CAN
  - Size – 15.5 mm x 20.7 mm x 2.8 mm
- WBZ450
  - 30-pin SMD package with Shield CAN
  - Size – 13.4 mm x 18.7 mm x 2.8 mm
- Operating Conditions
  - 1.9V to 3.6V, -40°C to +85°C, DC to 64 MHz

Certifications

- Certified to FCC, ISED and CE Radio Regulations
- RoHS and REACH Compliant

Note: Traditional LIN documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Commander” and “Responder”, respectively.

Table of Contents

1. Ordering Information

This chapter provides the ordering information of the PIC32CX-BZ2 and WBZ45 family of devices.

1.1 PIC32CXBZ2 SoC and WBZ45 Module Ordering Information

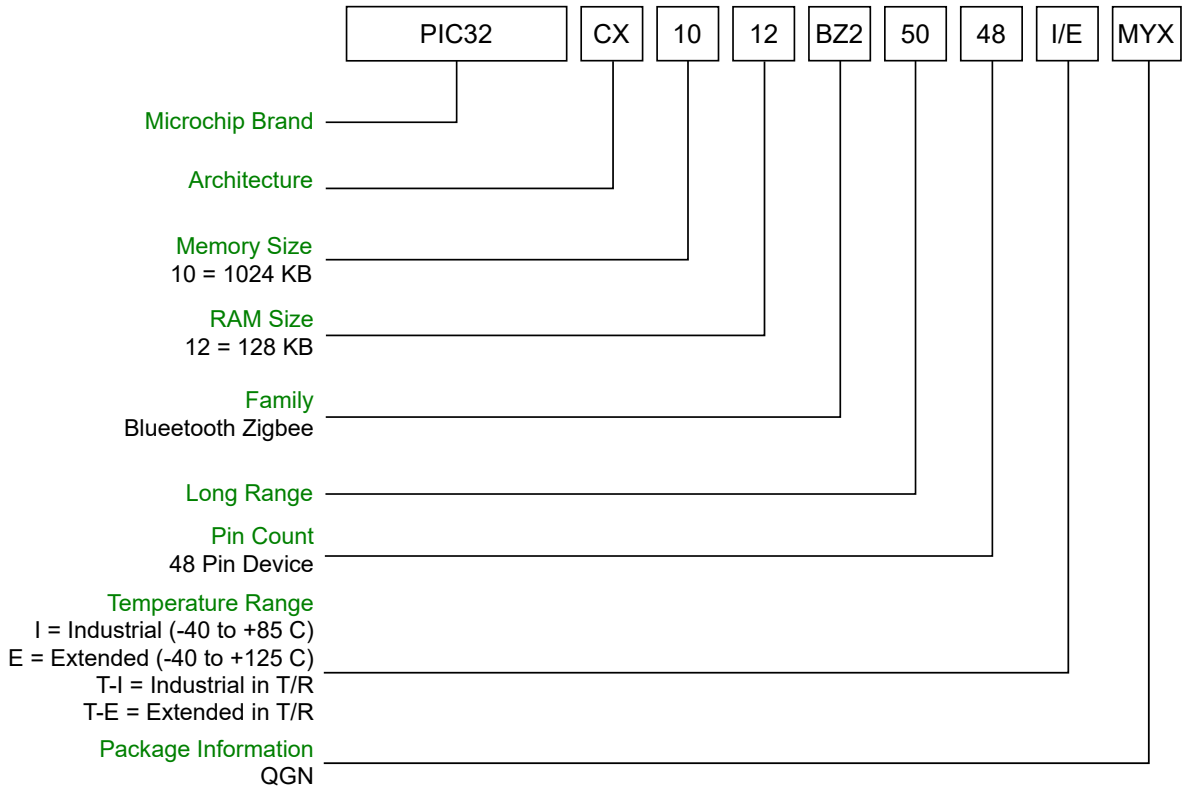
The following table describes the ordering information of the WBZ45 Module.

Table 1-1. WBZ45 Module Ordering Details

Model No.	Module SoC	Description	Regulatory Certification	Ordering Code
WBZ451PE	PIC32CX1012BZ25048-I/MYX	WBZ451 module with PCB antenna	FCC, ISED, CE	WBZ451PE-I
WBZ451UE		WBZ451 module with U.FL connector for external antenna	FCC, ISED, CE	WBZ451UE-I

The following figure illustrates the details of the PIC32CXBZ2 ordering information.

Figure 1-1. PIC32CXBZ2 Ordering Information



2. Configuration Summary

Table 2-1. PIC32CX-BZ2 and WBZ45 Family Features

Device	Program Memory (KB)	Data Memory (KB)	Pins	Package	Peripherals												Analog			Security				Wireless		
					SERCOM	Timer/Counter (TC)	TCC (24-bit/16-bit)	QSPI	DMA Channels	RTC	CCL/LUT	WDT	DMT	Frequency Measurement	Event System (Channels)	External Interrupt Lines	GPIO Pins	Analog Comparators (Channels)	ADC (Channels)	Temperature Sensor	AES	TRNG	Public Key Cryptography (PUKCC)	Integrity Check Monitor	Max TX Power (dBm)	Bluetooth 5.2	802.15.4/Zigbee 3.0
PIC32CX1012BZ25048	1024	128	48	QFN	4	4	2/1	Y	16	Y	1/2	Y	Y	Y	32	4	29	2	8	Y	Y	Y	Y	Y	12	Y	Y
PIC32CX1012BZ24032			32	QFN	2												16	1	6						4		
WBZ451			39	LGA	4												29	2	8						12		
WBZ450			30	LGA	2												16	1	6						4		

3. PIC32CX-BZ2 SoC Description

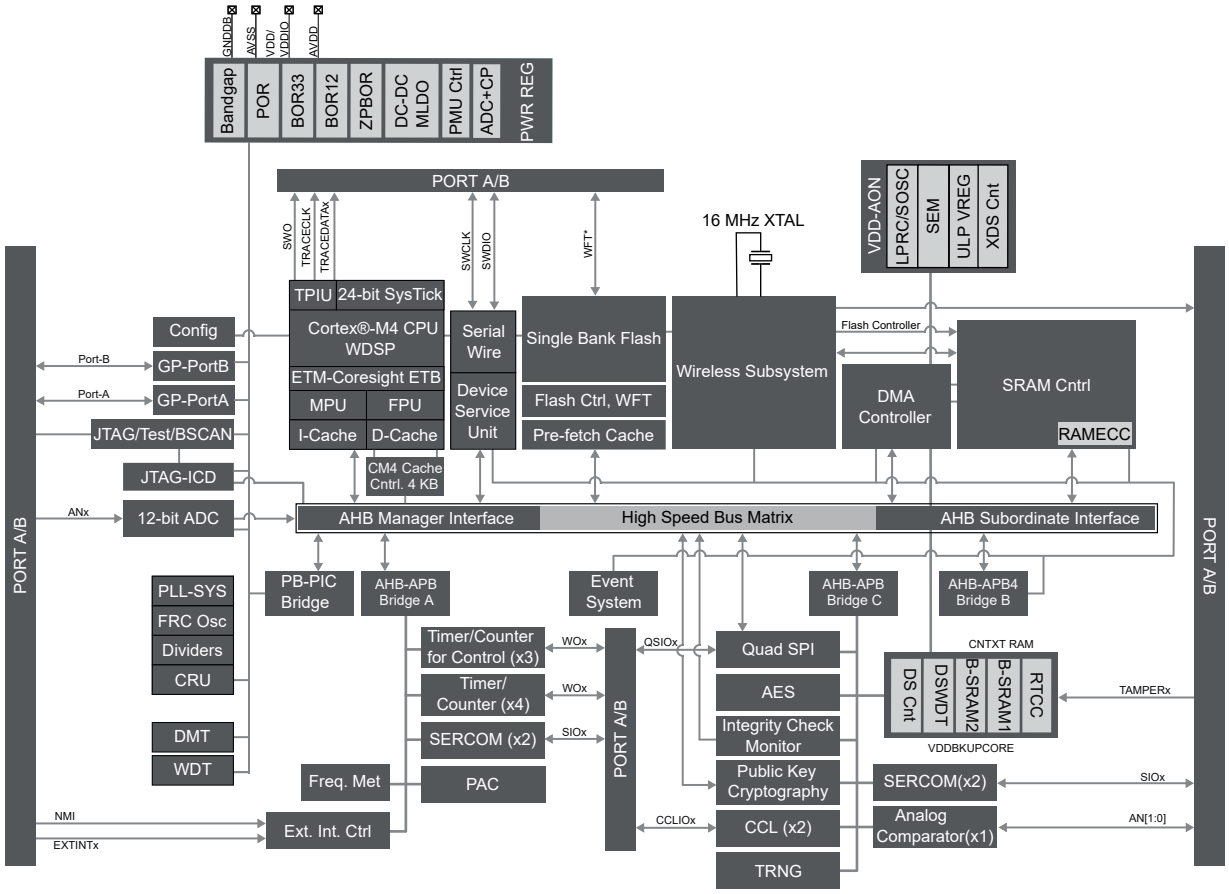
This chapter contains device-specific information for the PIC32CX-BZ2 SoC.

Note: Traditional AHB and APB documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Manager” and “Subordinate”, respectively.

3.1 PIC32CX-BZ2 SoC Block Diagram

The following figure illustrates the block diagram of the core and peripheral modules in the PIC32CX-BZ2 SoC.

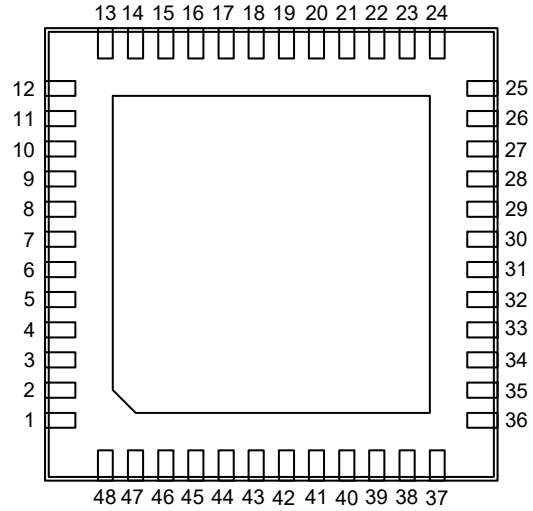
Figure 3-1. PIC32CX-BZ2 SoC Block Diagram



3.2 Pinout Diagram

This section provides details on pin diagrams for each variant of the PIC32CX-BZ2 SoC.

Figure 3-2. PIC32CX1012BZ25048 SoC Pin Diagram (Bottom View)



PIC32CX1012BZ25048

Note:

1. It is required that the exposed paddle on the bottom of the SoC be connected to ground in the PCB.

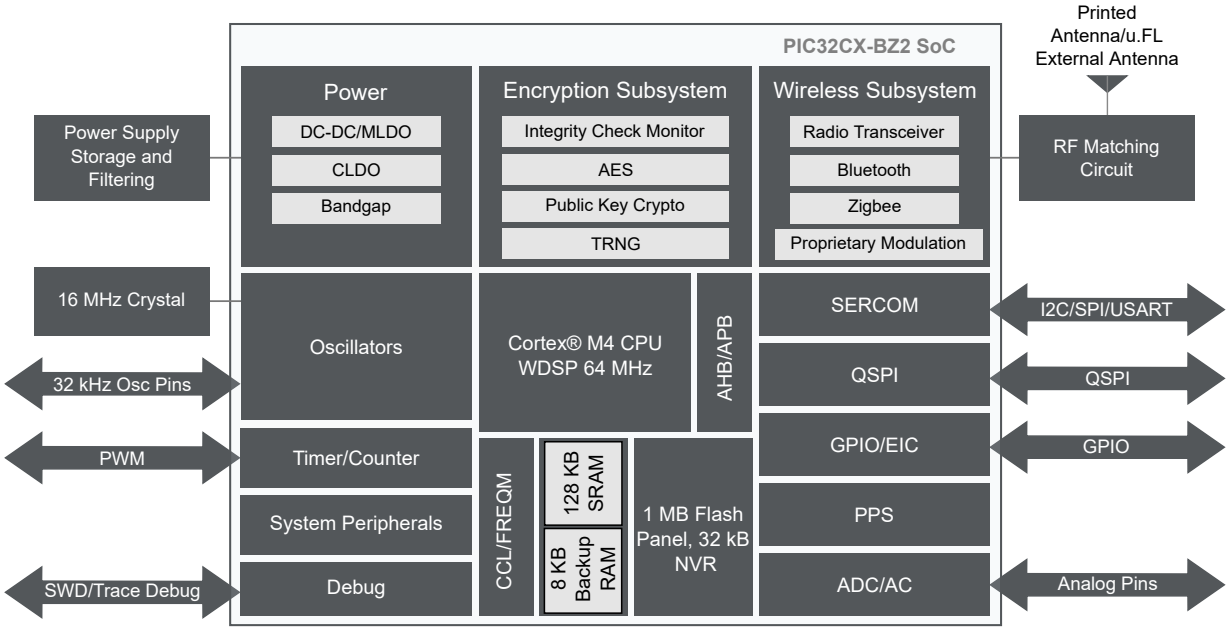
4. WBZ45 Module Description

The WBZ45 contains the PIC32CX-BZ2 SoC with following antenna options:

- PCB antenna
- u.FL connector for external antenna

The following figure represents the WBZ45 module block diagram.

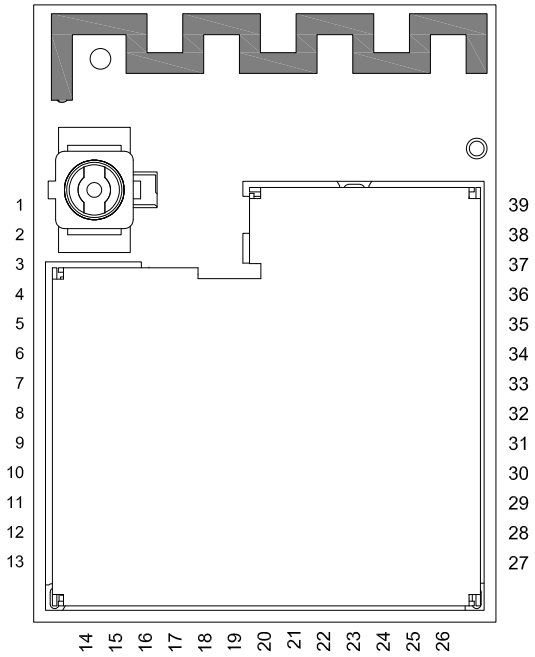
Figure 4-1. WBZ45 Module Block Diagram



4.1 Pinout Diagram

The following figure illustrates the module pinout diagram.

Figure 4-2. WBZ451 Module Pin Diagram (Top View)



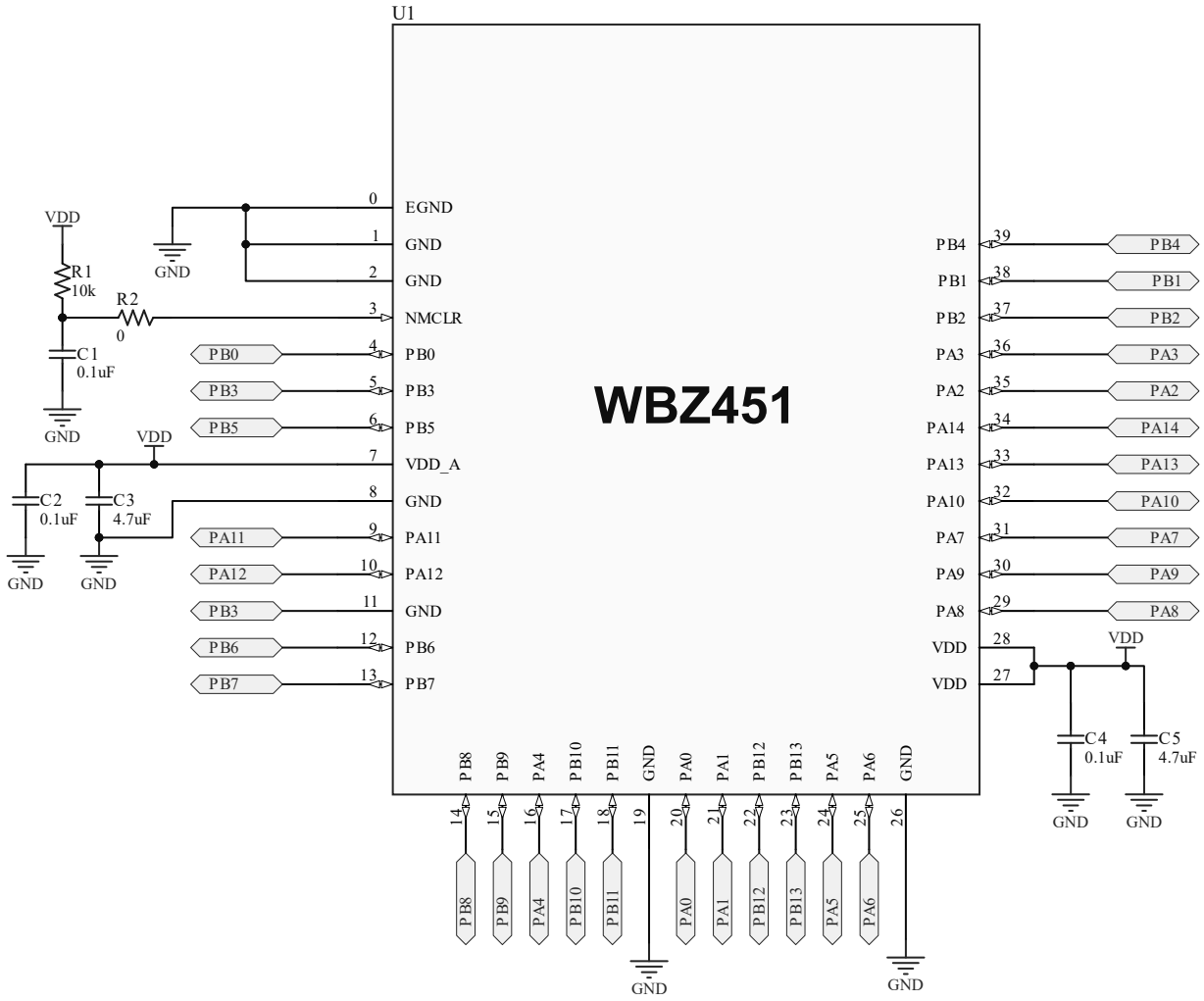
WBZ451

Note: It is required that the exposed paddle on the bottom of the module be connected to ground in the PCB.

4.2 Basic Connection Requirement

The WBZ45 module requires attention to a minimal set of device pin connections before proceeding with development.

Figure 4-3. Module Basic Connection and Interface Diagram



4.2.1 Power Pins

It is recommended that a bulk and a decoupling capacitor be added at the input supply pin (VDD, VDD_A and GND pins) of the WBZ45 module.

- It is recommended that 4.7 µF be on the VDD_A pin and 4.7 µF and a 0.1 µF be on the VDD pin.
- The value of the capacitors are based on typical application requirements and are the minimum recommended values. Depending on the application requirement (in other words, a noisy power line or other known noise sources), the values of capacitors can be adjusted to provide a clean supply to the module.
- All capacitors must be placed close to the Module Power supply pins.

4.2.2 Master Clear ($\overline{\text{MCLR}}$) Pin

The $\overline{\text{MCLR}}$ pin provides for two specific device functions:

- Device Reset
- Device programming and debugging

Pulling the $\overline{\text{MCLR}}$ pin low generates a device Reset. Module Basic Connection and Interface Diagram illustrates a typical $\overline{\text{MCLR}}$ circuit, see the *Module Basic Connection and Interface Diagram* in the *Basic Connection Requirement* from Related Links.

The module has sufficient filtering (0.1 µF) and pull-up (10k) on the Reset line. On a typical application, no extra filtering is required on this pin.

Related Links

4.2. Basic Connection Requirement

4.2.3 SWD Lines

The CM4_SWCLK, CM4_SWDIO and CM4_SWO pins are used for SWD Programming and debugging purposes. It is recommended that the CM4_SWCLK and CM4_SWDIO pin be used for the WBZ45 module for SWD as the default configuration (CM4_SWO can be optional).

Keep the trace length between the SWD pins of the WBZ45 module and the SWD header as short as possible. If the SWD connector is expected to experience an ESD event, a series resistor is recommended with the value in the range of a few tens of Ωs, not to exceed 100Ω.

Note: Provide an option for adding an external pull-up on SWDIO.

4.2.4 Unused I/O Pins

It is recommended that unused I/O pins not be allowed to float as inputs. They can be configured as inputs and pulled up. Alternatively, depending on the application, they can be pulled down as well.

Note: Pin PA3 has to be pulled down for optimal power consumption.

4.2.4.1 GPIO Pins/PPS Functions

Most of the WBZ45 module pins can be configured as GPIOs pins or for PPS functionality. To find the functionality supported by each of these GPIOs, see *I/O Ports and Peripheral Pin Select (PPS)* from Related Links. It is recommended that a series resistor be added on the host board for all critical, high frequency pins and clocks for EMI considerations. The value of the series resistor depends on the actual pin configuration. These resistors must be placed close to the module. Example of Host Board on Top Layer illustrates the placement of the series resistor; see the *Example of Host Board on Top Layer* figure in the *WBZ45 Module Routing Guidelines* from Related Links.

Related Links

6. I/O Ports and Peripheral Pin Select (PPS)

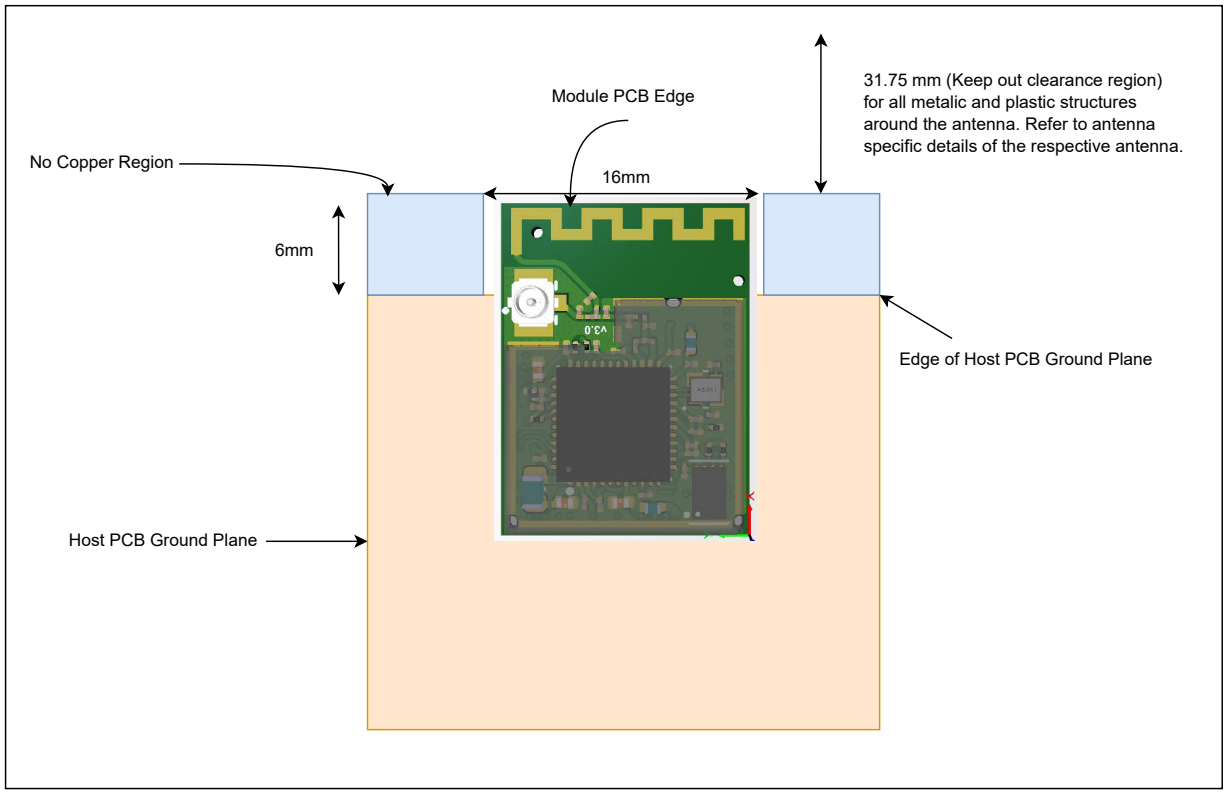
4.4. WBZ45 Module Routing Guidelines

4.3 WBZ45 Module Placement Guidelines

- For any Bluetooth Low Energy/Zigbee product, the antenna placement affects the performance of the whole system. The antenna requires free space to radiate RF signals and it must not be surrounded by the ground plane. Thus, for the best PCB antenna performance, it is recommended that the WBZ45 module be placed at the edge of the host board.
- It is recommended that the WBZ45 module ground outline edge be aligned with the edge of the host board ground plane as shown in the following figure.
- A low-impedance ground plane for the WBZ45 module ensures the best radio performance (best range and lowest noise). The ground plane can be extended beyond the minimum recommendation as required for the host board EMC and noise reduction.
- For the best performance, keep metal structures and components (such as mechanical spacers, bump-on and so on) at least 31.75 mm away from the PCB trace antenna as illustrated in the following figure.
- It is recommended that the antenna on the WBZ45 module not be placed in direct contact with or in close proximity to plastic casing or objects. Keep a minimum clearance of 10 mm in all directions around the PCB antenna as shown in the following figure. Keeping metallic and plastic objects close to the antenna can detune the antenna and reduce the performance of the device.
- Exposed GND pads on the bottom of the WBZ45 module must be soldered to the host board (see the *Example of Host Board on Top Layer* figure in the *WBZ45 Module Routing Guidelines* from Related Links).
- A PCB cutout or a copper keepout is required under the RF test point (see *WBZ451 Module Packaging Information* from Related Links).
- Copper keepout areas are required on the top layer under voltage test points (see *WBZ451 Module Packaging Information* from Related Links).
- Alternatively, the entire region, except the exposed ground paddle, can be solder-masked.

The following figure illustrates the examples of WBZ45 module placement on a host board with a ground plane. Refer to the following figure for placement-specific guidance.

Figure 4-4. Module Placement Guidelines



Related Links

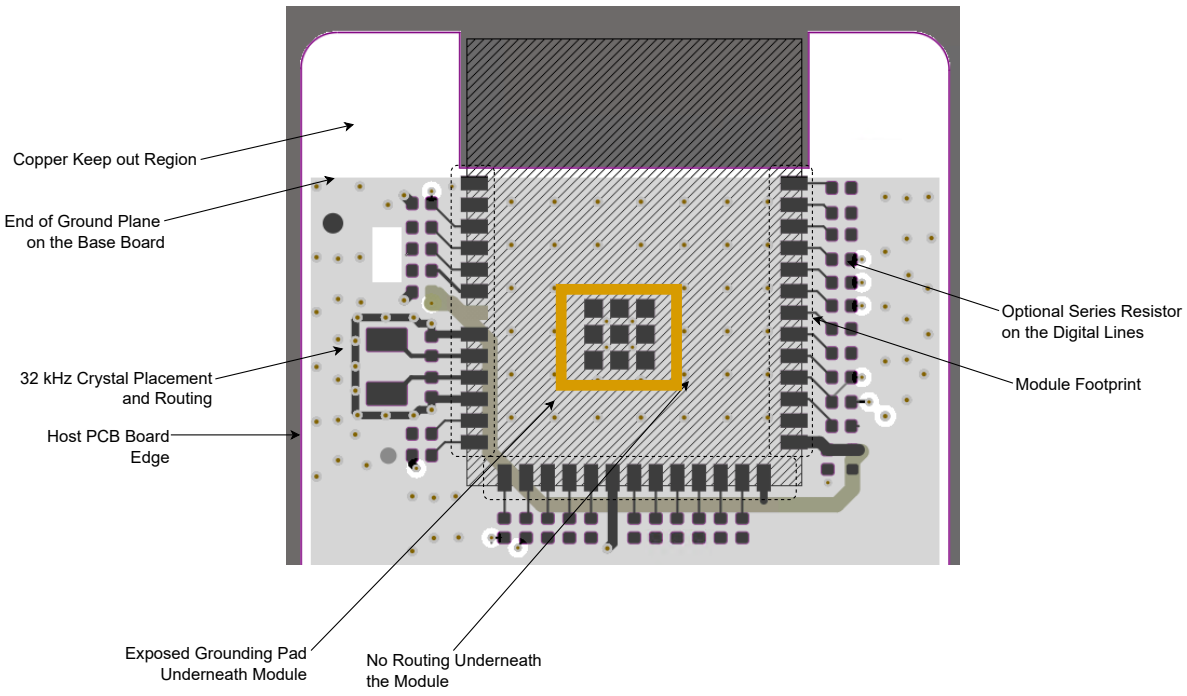
4.4. WBZ45 Module Routing Guidelines

44.2. WBZ451 Module Packaging Information

4.4 WBZ45 Module Routing Guidelines

- Use the multi-layer host board for routing signals on the inner layer and the bottom layer.
- The top layer (underneath the module) of the host board must be ground with as many GND vias as possible, shown in the following figure.
- Avoid fan-out of the signals under the module or antenna area. Use a via to fan-out signals to theedge of the WBZ45 module.
- For a better GND connection to the WBZ45 module, solder the exposed GND pads of the WBZ45 module on the host board.
- For the module GND pad, use a GND via of a minimum 10 mil (hole diameter) for good ground to all the layers and thermal conduction path.
- Having a series resistor on the host board for all GPIOs is recommended. These resistors must be placed close to the WBZ45 module. Refer to the following figure for the placement of the series resistor.
- The SOSC crystal (32.768 kHz) on the host board must be placed close to the WBZ45 module and follow the shortest trace routing length with no vias (see the following figure).

Figure 4-5. Example of Host Board on Top Layer



4.5 WBZ45 Module RF Considerations

The overall performance of the system is significantly affected by the product design, environment and application. The product designer must ensure system-level shielding (if required) and verify the performance of the product features and applications.

Consider the following guidelines for optimal RF performance:

- The WBZ45 module must be positioned in a noise-free RF environment and must be kept far away from high-frequency clock signals and any other sources of RF energy.
- The antenna must not be shielded by any metal objects.
- The power supply must be clean and noise-free.
- Make sure that the width of the traces routed to GND, VDD rails are sufficiently large for handling peak TX current consumption.
  Note: The WBZ45 module includes RF shielding on top of the board as a standard feature.

4.6 WBZ45 Module Antenna Considerations

4.6.1 PCB Antenna

For the WBZ45 module, the PCB antenna is fabricated on the top copper layer and covered in a solder mask. The layers below the antenna do not have copper trace. It is recommended that the module be mounted on the edge of the host board and to have no PCB material below the antenna structure of the module and no copper traces or planes on the host board in that area.

The following table lists the technical specification of the PCB antenna when tested with the WBZ45 module mounted on an Evaluation Board.

Table 4-1. PCB Antenna Specification for WBZ45

Parameter	Specification
Operating frequency	2400 to 2480 MHz
Peak gain	2.36 dBi at 2420 MHz
Efficiency	50%

4.6.2 External Antenna Placement Recommendations

The following recommendations must be applied for the placement of the antenna and its cable:

- The antenna cable must not be routed over circuits generating electrical noise on the host board or alongside or underneath the module. It is preferred that the cable is routed straight out of the module.
- The antenna must not be placed in direct contact or in close proximity of the plastic casing/objects. (Except when the selected antenna specifically recommends it).
- Do not enclose the antenna within a metal shield.
- Keep any components that may radiate noise, signals or harmonics within the 2.4-2.5 GHz frequency band away from the antenna and, if possible, shield those components. Any noise radiated from the host board in this frequency band degrades the sensitivity of the module.
- The antenna must be placed at a distance greater than 5 cm away from the module. The following figure illustrates the antenna keepout area where the antenna must not be placed.

These recommendations are based on an open-air measurement and do not take into account any metal shielding of the customer end product. When a metal enclosure is used, the antenna can be located closer to the WBZ45 Module.

Note: These are generic guidelines and it is recommended that customers check and fine-tune the antenna positioning in the final host product based on RF performance.

The following figure illustrates an indication on how to route the antenna cable depending on the location of the antenna with respect to the WBZ45 PCB; there are two possible options for the optimum routing of the cable.

Figure 4-6. WBZ45 Antenna Placement Guidelines

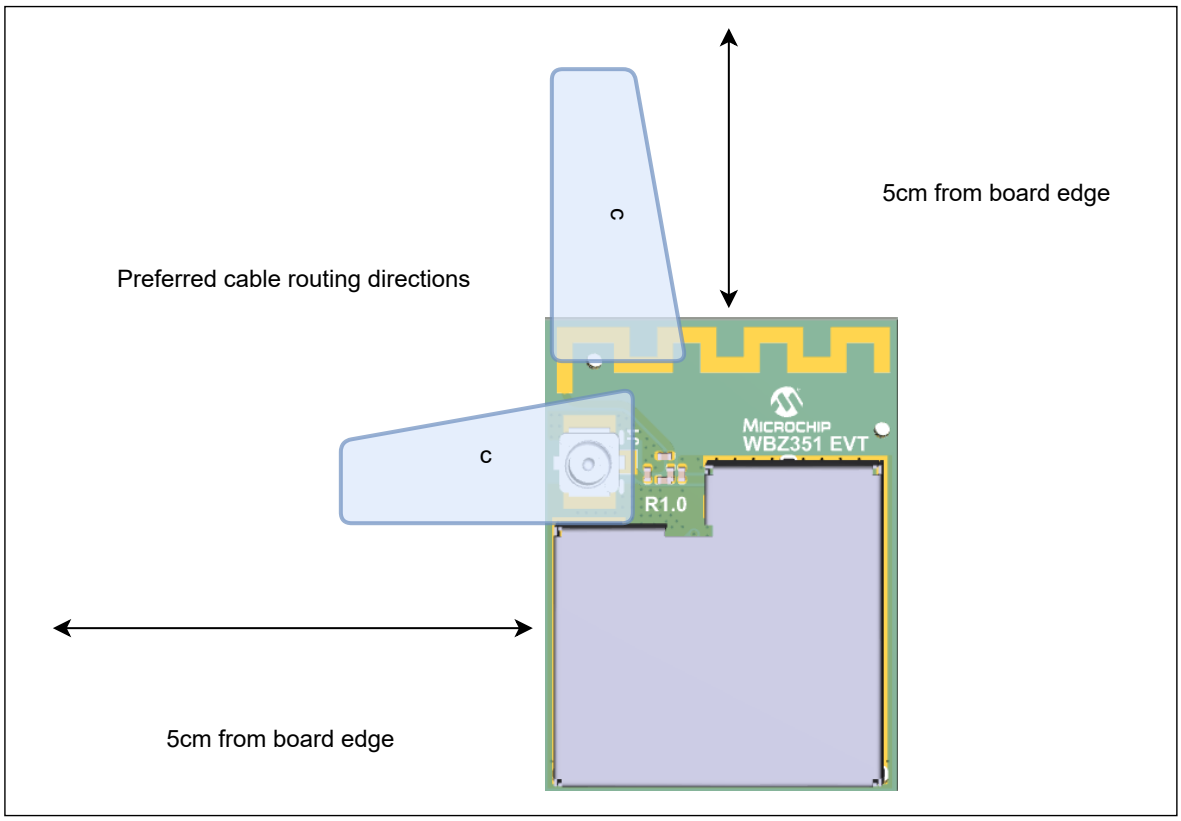


Table 4-2. List of Certified Antenna

Serial Number	Part Number	Vendor	Antenna Type	Gain	Comment
1	W3525B039	Pulse	PCB	2 dBi	Cable length 100 mm
2	001-0016	LSR	PIFA	2.5 dBi	Flex PIFA antenna
3	001-0001	LSR	Dipole	2 dBi	RPSMA connector*
4	1461530100	Molex	PCB	3 dBi	100 mm (Dual Band)
5	ANT-2.4-LPW-125	Linx Technologies	Dipole	2.8 dBi	125 mm
6	RFA-02-P05-D034	Alead	PCB	2 dBi	150 mm
7	RFA-02-P33-D034	Alead	PCB	2 dBi	150 mm
8	ABAR1504-S2450	ABRACON	PCB	2.28 dBi	250 mm
9	WBZ451 LGA	—	—	2.36 dBi	Only for WBZ451 module
10	WBZ450 LGA	—	—	4.14 dBi	Only for WBZ450 module

4.7 WBZ45 Module Reflow Profile Information

The WBZ45 module was assembled using the IPC/JEDEC J-STD-020 Standard lead free reflow profile. The WBZ45 module can be soldered to the host board using standard leaded or lead-free solder reflow profiles. To avoid damaging the module, adhere to the following recommendations:

- For Solder Reflow Recommendations, refer to the *Solder Reflow Recommendation Application Note* (AN233).

- Do not exceed a peak temperature (TP) of 250°C.
- Refer to the solder paste data sheet for specific reflow profile recommendations from the vendor.
- Use no-clean flux solder paste.
- Do not wash as moisture can be trapped under the shield.
- Use only one flow. If the PCB requires multiple flows, apply the module on the final flow.

4.7.1 Cleaning

The exposed GND pad helps to self-align the module, avoiding pad misalignment. The recommendation is to use the no clean solder pastes. Ensure full drying of no-clean paste fluxes as a result of the reflow process. As per the recommendation by the solder paste vendor, this requires longer reflow profiles and/or peak temperatures toward the high end of the process window. The uncured flux residues can lead to corrosion and/or shorting in accelerated testing and possibly the field.

4.8 WBZ45 Module Assembly Considerations

The WBZ45 module is assembled with an EMI shield to ensure compliance with EMI emission and immunity rules. The EMI shield is made of a tin-plated steel (SPTE) and is not hermetically sealed. Use the solutions such as IPA and similar solvents to clean this module. Cleaning solutions containing acid must never be used on the module.

4.8.1 Conformal Coating

The modules are not intended for use with a conformal coating, and the customer assumes all risks (such as the module reliability, performance degradation and so on) if a conformal coating is applied to the modules.

5. Pinout and Signal Descriptions List

The following table provides details on signal names classified by the peripherals along with the device pinout for each variant of the PIC32CX-BZ2 SoC and WBZ45 module.

Table 5-1. Pinout and Signal Descriptions List

PIC32CX-BZ2 SoC		WBZ45 Module		Pad Name	Peripherals									
24032	25048	450	451		AC	ADC	EIC(4)	GPIO (1, 2)	QSPI	RTCC	SERCOM	OSC	RF	DEBUG
		1, 5, 8, 18, 21, 27	1, 2, 8, 11, 19, 26	GND										
32	1			PMU_BK										
1	2			VPMU_VDD										
2	3			PMU_MLDO										
	4		20	PA0				RA0	QSPI_DATA2	RTC_IN3				
	5		21	PA1	AC_CMP1			RA1	QSPI_DATA3	RTC_IN2				
	6		35	PA2	AC_CMP0			RA2		RTC_IN1				
3	7	19	24	PA5				RA5			SERCOM0_PAD0			
4	8	16, 17	27, 28	VDD										
5	9	20	25	PA6	AC_CMP1_ALT			RA6			SERCOM0_PAD1			PGC2ENTRY
6	10	25	31	PA7				RA7			SERCOM1_PAD0			TRACECLK
7	11	24	29	PA8				RA8			SERCOM1_PAD1		FECTRL0	PGD2ENTRY
8	12	22	30	PA9				RA9		RTC_IN0_ALT	SERCOM1_PAD2		FECTRL1	
9	13	23	32	PA10				RA10		RTC_OUT_ALT	SERCOM1_PAD3		FECTRL2	
	14		22	PB12				RB12	QSPI_DATA0					
	15		23	PB13				RB13	QSPI_DATA1	RTC_EVENT				
	16		33	PA13				RA13			SERCOM2_PAD0		COEXCTRL0	
	17		34	PA14				RA14			SERCOM2_PAD1		COEXCTRL1	
10	18			CLDO_O										
11	19			BUCK_CLDO										
12	20			EXTR(8)										
13	21			BUCK_BB										
14	22			XO_N								XO−		
15	23			XO_P								XO+		
16	24			BUCK_PLL										
17	25			BUCK_LPA										
18	26			LPA_OUT									LPA	
	27			MPA_OUT									MPA	
	28			BUCK_MPA										
	29	2	3	NMCLR										
20	30	3	4	PB0	AC_AIN2	AN4		RB0					COEXCTRL2	
21	31	10	38	PB1	AC_AIN3	AN5		RB1						
	32		37	PB2	AC_AIN0	AN6		RB2						
	33		5	PB3	AC_AIN1	AN7		RB3						
22	34	11	39	PB4		AN0	INT0	RB4					FECTRL3	TRACEDATA3, PGC1

.....continued

PIC32CX-BZ2 SoC		WBZ45 Module		Pad Name	Peripherals										
24032	25048	450	451		AC	ADC	EIC(4)	GPIO (1, 2)	QSPI	RTCC	SERCOM	OSC	RF	DEBUG
23	35	9	6	PB5		AN1		RB5					FECTRL4	TRACEDATA0, PGC4
24	36	4	7	AVDD										
25	37	12	12	PB6		ANN0,AN2		RB6					FECTRL5	TRACEDATA1, PGD1
26	38	15	13	PB7	LVDIN	AN3		RB7						TRACEDATA2, CM4_SWO, PGD4
	39			VDD										
27	40	14	15	PB9				RB9						CM4_SWDIO
28	41	13	14	PB8				RB8						CM4_SWCLK
29(6)	42	6	16	PA4				RA4		RTC_OUT	SERCOM0_PAD3			
	43		17	PB10				RB10	QSPI_CS					
	44		18	PB11				RB11	QSPI_SCK					
29	45		9	PA11				RA11(5)				SOSCI		
30	46		10	PA12				RA12(5)				SOSCO		
30(6)	47	7	36	PA3				RA3		RTC_IN0	SERCOM0_PAD2	SCLKI		
31	48			VPMU_VDD										
		26, 28, 29, 30		NC										

Notes:

1. All GPIOs (RAn and RBn) can be used by remappable peripherals via PPS.
2. All GPIOs (RAn and RBn) can be used as I/O Change Notification (IOCAn and IOCBn).
3. The metal paddle at the bottom of the device must be connected to system ground.
4. These peripherals have signals that are only available via the PPS remappable pins.
5. This pin can be used as Input only pin if not using SOSC.
6. For 24032 only, pin 29 and pin 30 act as GPIO PA4/PA3 respectively ONLY if CFGCON0.GPSOSCE = 0. If CFGCON0.GPSOSCE = 1, these pins are SOSC 32 kHz crystal inputs OR as digital input only RA11/RA12 respectively.
7. These I/O pins are 5.5V tolerant: NMCLR, PA0, PA1, PA2, PA4, PA5, PA6, PA7, PA8, PA9, PA10, PA13, PA14, PB10, PB11, PB12, PB13. All other I/O pins are 3.3V tolerant.
8. External resistor used to set internal reference current of the SOC.

6. I/O Ports and Peripheral Pin Select (PPS)

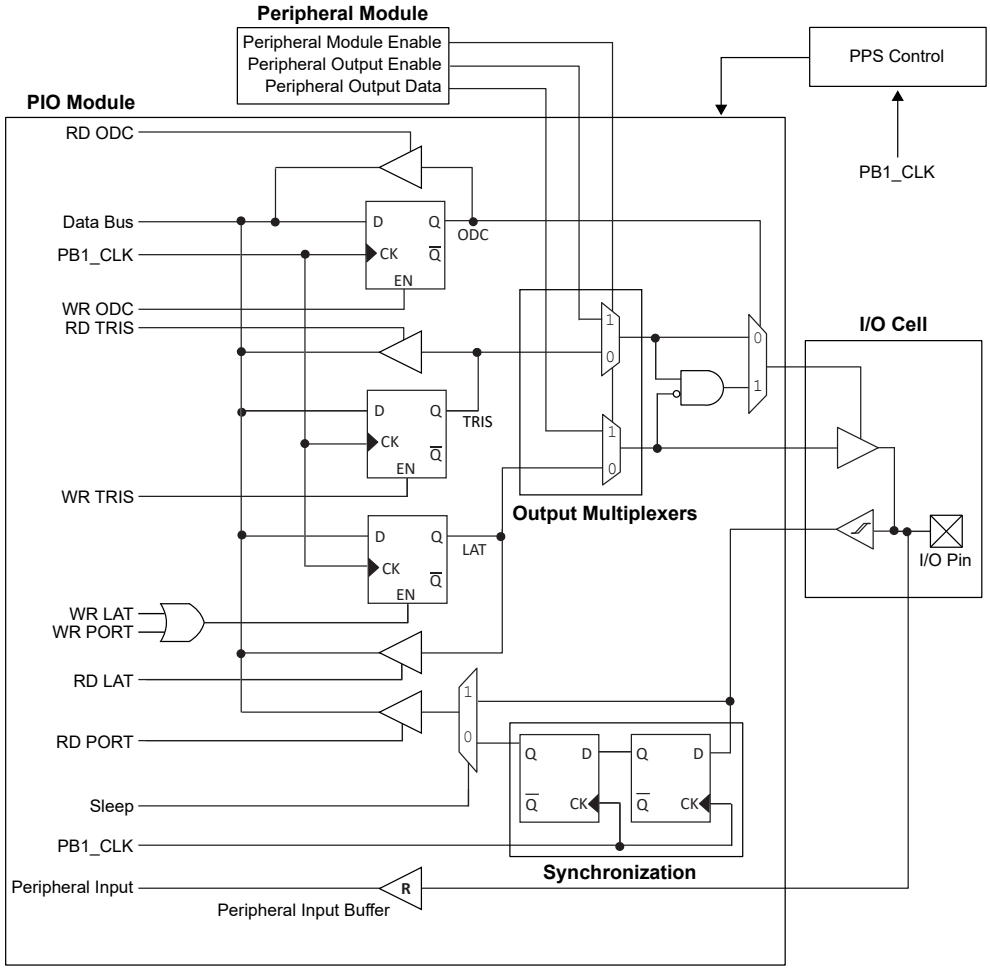
General purpose I/O (GPIO) pins allow the PIC32CX-BZ2 devices to monitor and control other devices. To add flexibility and functionality, some pins are multiplexed with alternate function(s). These functions depend on which peripheral features are on the device. In general, when a peripheral is functioning, that pin may not be used as a general purpose I/O pin. For more details on pin multiplexing, see *GPIO Pins/PPS Functions* from Related Links. There are default priorities for each GPIO pin as well. For details on priorities, see *Function Priority for Device Pins* from Related Links. It must be noted that 'fuse' values stored in NVR memory can be used to alter the power-on default function for certain GPIO pins. See *System Configuration Registers (CFG)* from Related Links.

Some of the key features of the I/O ports are:

- Individual output pin open-drain enable/disable
- Individual input pin weak pull-up and pull-down
- Monitor selective inputs and generate interrupt when change in pin state is detected
- Operation during Sleep and Idle modes
  - Fast bit manipulation using CLR, SET and INV registers
  - Slew rate control

The following figure illustrates a block diagram of a typical multiplexed I/O port.

Figure 6-1. Typical Multiplexed Port Structure Block Diagram



Related Links

4.2.4.1. GPIO Pins/PPS Functions
6.3. Function Priority for Device Pins
18. System Configuration and Register Locking (CFG)

6.1 Control Registers

Before reading and writing any I/O port, the desired pin(s) must be properly configured for the application. Each I/O port has nine registers directly associated with the operation of the port and one control register. Each I/O port pin has a corresponding bit in these registers. Throughout this section, the letter 'x', denotes any or all port module instances. For example, TRISx represents TRISA or TRISB. Any bit and its associated data and control registers that is not valid for a particular device will be disabled and will read as zeros.

6.1.1 Configuring Tri-State Functions (TRISx)

The TRISx registers configure the data direction flow through port I/O pins. The TRISx register bits determine whether a PORTx I/O pin is an input or an output:

- If a data direction bit is '1', the corresponding I/O port pin is an input.
- If a data direction bit is '0', the corresponding I/O port pin is an output.
- A read from a TRISx register reads the last value written to that register.
- All I/O port pins are defined as inputs after a Power-on Reset (POR).

6.1.2 Configuring Port Functions (PORTx)

The PORTx registers allow I/O pins to be accessed:

- A write to a PORTx register writes to the corresponding LATx register (PORTx data latch). Those I/O port pin(s) configured as outputs are updated.
- A write to a PORTx register is effectively the same as a write to a LATx register.
- A read from a PORTx register reads the synchronized signal applied to the port I/O pins.

6.1.3 Configuring Latch Functions (LATx)

The LATx registers (PORTx data latch) hold data written to port I/O pins:

- A write to a LATx register latches data to corresponding port I/O pins. Those I/O port pins configured as outputs are updated.
- A read from a LATx register reads the data held in the PORTx data latch, not from the port I/O pins.

6.1.4 Open-Drain Configuration (ODCx)

Each I/O pin can be individually configured for either normal digital output or open-drain output. This is controlled by the open-drain control register, ODCx, associated with each I/O pin. If the ODCx bit for an I/O pin is a '1', the pin acts as an open-drain output. If the ODCx bit for an I/O pin is a '0', the pin is configured for a normal digital output (the ODCx bit is valid only for output pins). After a Reset, the status of all the bits of the ODCx register is set to '0'.

The maximum open-drain voltage allowed is the same as the maximum V_{IH} specification. The ODCx register setting functions in all of the I/O modes, allowing the output to behave as an open-drain even if a peripheral is controlling the pin. Although, the user may achieve the same result by manipulating the corresponding LATx and TRISx bits, this procedure does not allow the peripheral to operate in the Open-Drain mode (except for the default operation of the I^2C pins). I^2C pins are already open-drain pins; therefore, the ODCx settings do not influence the I^2C pins.

6.1.5 Configuring Analog and Digital Port Pins (ANSELx)

The ANSELx register controls the operation of the analog port pins. The port pins that are to function as analog inputs must have their corresponding ANSEL and TRISx bits set. To use port pins for I/O functionality with digital modules, such as Timers, SERCOMs and so on, the corresponding ANSELx bit must be cleared. The ANSELx register has a default value of 0xFFFF; therefore, all pins that share analog functions are, by default, analog and not digital.

If the TRISx bit is cleared (output) while the ANSELx bit is set, the digital output level (V_{OH} or V_{OL}) is converted by an analog peripheral, such as the ADC module or the comparator module. When the PORTx register is read, all pins configured as analog input channels are read as cleared (a low-level). Pins configured as digital inputs do not convert an analog input. Analog levels on any pin defined as a digital input (including the ANx pins) can cause the input buffer to consume current that exceeds the device specifications.

6.1.6 Input Change Notification (CN)

The Input Change Notification (CN) function of the I/O ports allows PIC32CX-BZ2 devices to generate interrupt requests to the processor in response to a change-of-state on selected input pins. This feature can detect input change of states even in the Sleep mode, when the clocks are disabled.

The following control registers are associated with the CN functionality of each I/O port:

- Change Notice Pull-up Enable (CNPUEx)
- Change Notice Pull-down Enable (CNPDx)
- Change Notice Control (CNCONx)
- Change Notice Enable (CNENx/CNNEx)
- Change Notice Status (CNSTATx/CNFx) or the positive edge control

Each I/O pin also has a weak pull-up and a weak pull-down connected to it. The pull-ups act as a current source or sink source connected to the pin and eliminate the need for external resistors when push button or keypad devices are connected. The pull-ups and pull-downs are enabled separately using the CNPUEx and the CNPDx registers, which contain the control bits for each of the pins. Setting any of the control bits enables the weak pull-ups and/or pull-downs for the corresponding pins.

Note: Pull-ups and pull-downs on change notification pins must always be disabled when the port pin is configured as a digital output

The CNCONx registers provide change notice control.

The CNENx/CNNEx registers contain the CN interrupt enable control bits for each of the input pins. Setting any of these bits enables a CN interrupt for the corresponding pins. CNENx enables a mismatch CN interrupt condition when EDGEDETECT is not set. When EDGEDETECT is set, CNNEx controls the negative edge while CNENx controls the positive. On devices that do not have EDGEDETECT, this CN logic acts as if EDGEDETECT is not set.

The CNSTATx/CNFx registers indicate whether a change occurred on the corresponding pin since the last read of the PORTx bit. The CNFx registers indicate the type of change that occurred.

6.1.7 Registers for Peripheral Pin Select

The Peripheral Pin Select [pin name]R register (*[pin name]R*) and the Peripheral Pin Select Output (RPnR) register (*RPnR*) provide the control bits for the peripheral pin select input and output. See *[pin name]R* and *RPnR* from Related Links.

Related Links

6.2.7.4. [pin name]R
6.2.9.1. RPnR

6.1.8 Slew Rate Control

Some I/O pins can be configured for various types of slew rate control on its associated port. This is controlled by the slew rate control bits in the SRCON1x and SRCON0x registers that are associated with each I/O port. The slew rate control is configured using the corresponding bit in each register, as shown in the following table.

As an example, writing 0x0001, 0x0000 to SRCON1A and SRCON0A, respectively, can enable slew rate control on the RA0 pin and sets the slew rate to the slow edge rate.

Note: Slew rate control bits must not be enabled with pad configured as an input.

Table 6-1. Slew Rate Control Bit Settings(1)

SRCON1x	SRCON0x	Description
1	1	Slew rate control is enabled and is set to the slowest edge rate

.....continued		
SRCON1x	**SRCON0x**	**Description**
1	0	Slew rate control is enabled and is set to the slow edge rate
0	1	Slew rate control is enabled and is set to the medium edge rate
0	0	Slew rate control is disabled and is set to the fastest edge rate

1. By default, all the port pins are set to the fastest edge rate.

6.1.9 CLR, SET and INV Registers

Every I/O module register has corresponding SET, CLR and INV registers, which provide atomic bit manipulations. As the name of the registers imply, a value written to a SET, CLR or INV register effectively performs the implied operation, but only on the corresponding base register and only bits specified as '1' are modified. Bits specified as '0' are not modified. For example,

- Writing 0x0001 to the TRISASET register sets only bit 0 in the base register TRISA
- Writing 0x0020 to the PORTBCLR register clears only bit 5 in the base register PORTB
- Writing 0x9000 to the LATAINV register inverts only bits 15 and 12 in the base register LATA

Reading the SET, CLR and INV registers returns an undefined value. To see the influences of a write operation to a SET, CLR or INV register, the base register must be read instead.

A typical method to toggle an I/O pin requires a read-modify-write operation performed on a PORTx register in the software. For example, a read from a PORTx register, mask and modify the desired output bit or bits, and write the resulting value back to the PORTx register. This method is vulnerable to a read-modify-write issue where the port value may change after it is read and before the modified data can be written back, thus, changing the previous state. This method also requires more instructions.

A more efficient and atomic method uses the PORTxINV register. A write to the PORTxINV register effectively performs a read-modify-write operation on the target base register, equivalent to the software operation described previously; however, it is done in the hardware. To toggle an I/O pin using this method, a '1' is written to the corresponding bit in the PORTxINV register. This operation reads the PORTx register, inverts only those bits specified as '1', and writes the resulting value to the LATx register, thus, toggling the corresponding I/O pins all in a single atomic instruction cycle. PORTAINV = 0x0001.

6.2 Peripheral Pin Select (PPS)

A major challenge in general purpose devices is providing the largest possible set of peripheral features while minimizing the conflict of features on I/O pins. The challenge is even greater on low pin-count devices. In an application where more than one peripheral needs to be assigned to a single pin, inconvenient workarounds in application code or a complete redesign may be the only option.

The PPS configuration provides an alternative to these choices by enabling peripheral set selection and their placement on a wide range of I/O pins. By increasing the pinout options available on a particular device, the users can better modify the device to their entire application, rather than trimming the application to fit the device.

This feature operates over a fixed subset of digital I/O pins. The users may independently map the input and/or output of most digital peripherals to these I/O pins. The PPS configuration is performed in the software and generally does not require the device to be reprogrammed. The hardware safeguards that prevent accidental or spurious changes to the peripheral mapping are included once the PPS configuration is established.

In PPS mode, Maximum peripheral clock frequency = Direct mode clock frequency/2.

Note: Direct Mode is a mode in which peripherals are running based on Function Priority for Pins and not using PPS.

6.2.1 Re-Mappable Pin Groupings

The re-mappable pins, as well as the available input and output functions, are divided into four groups. The re-mappable pins of group k may be assigned pin functions only from group k (k = 1,2,3,4). The pins used by each peripheral are spread across all four groups when possible to maximize flexibility.

6.2.2 Enabling Remap Pins

Each remap pin (RPx) must be enabled by disabling all higher priority pin functions on that pin before it can be used. Typically, all functions other than GPIO are considered higher priority than remap pins.

6.2.3 RP Register Protection

The <INPUT>R and RPxxR registers are implemented with two levels of protection:

- I/O Lock Feature – All PPS registers may only be written while CFGCON0.IOLOCK = 0; once the IOLOCK is set, the registers cannot be written.
- IOLOCK Protection – The state of the IOLOCK bit can only be changed once it is unlocked using the CFGCON0.CFGLOCK[1:0] register.

These features prevent the RP registers from being inadvertently written during normal operation because changing the pinout functionality may have detrimental system-level outcome.

6.2.4 Available Pins

The number of available pins is dependent on the particular device and its pin count. Pins that support the PPS feature include the designation “RPn” in their full pin designation, where:

- RP – Designates a remappable peripheral
- n – Remappable port number

6.2.5 Available Peripherals

The peripherals managed by the PPS are all digital-only peripherals. These include general serial communications (SERCOM), general purpose timer clock inputs, timer-related peripherals (input capture and output compare), interrupt-on-change inputs and reference clocks (input and output).

In comparison, some digital-only peripheral modules are never included in the PPS feature. This is because the peripheral’s function requires special I/O circuitry on a specific port and cannot be easily connected to multiple pins. These modules include I^2C among others. A similar requirement excludes all modules with analog inputs, such as the ADC.

A key difference between remappable and non-remappable peripherals is that remappable peripherals are not associated with a default I/O pin. The peripheral must always be assigned to a specific I/O pin before it can be used. In contrast, non-remappable peripherals are always available on a default pin, assuming that the peripheral is active and not conflicting with another peripheral.

When a remappable peripheral is active on a given I/O pin, it takes priority over all other digital I/O and digital communication peripherals associated with the pin. Priority is given regardless of the type of peripheral that is mapped. Remappable peripherals never take priority over any analog functions associated with the pin.

6.2.6 Controlling PPS

The PPS features are controlled through two sets of SFRs: one to map peripheral inputs and another to map outputs. They are separately controlled; therefore, a particular peripheral’s input and output (if the peripheral has both) can be placed on any selectable function pin without constraint.

The association of a peripheral to a peripheral-selectable pin is handled in two different ways, depending on whether an input or output is mapped.

6.2.7 Remappable Inputs

When configuring a PAD for a new peripheral functionality, the existing PPS configuration must be cleared prior to using the same PAD for the different peripheral functionality.

6.2.7.1 Enabling Remappable Peripheral Inputs

With PPS, each remappable input pin function (EXTINT0, SERCOM0_PAD3 and so on) is assigned to be driven from a specific device pin by programming the corresponding <INPUT>R[3:0] register (meaning, EXTINT0R[3:0], SCOM0P3R[3:0], and so on) with a value defined in the *Input Pin Selection Group 1*, *Input Pin Selection Group 2*, *Input Pin Selection Group 3*, *Input Pin Selection Group 4* tables and *[pin name]R* register. See these tables in the *Remappable Input Example* and *[pin name]R* from Related Links.

Assigning a remappable input pin function does not automatically enable the digital input buffer on the pin. The buffer must be enabled for each remap pin (RPx) by disabling all higher priority pin functions on that pin. Typically, all functions other than GPIO are considered higher priority than remap pins. See *Function Priority for Device Pins* for the list and priority of pin functions on each pin from Related Links.

The mapping is dynamic; therefore, in order to avoid glitching outputs, the user is responsible for turning off the appropriate peripherals before remapping the pin functions associated with that peripheral. On Reset, all inputs are mapped to a default value and all outputs are disabled; therefore, the mapping may safely be performed after any device Reset.

Related Links

6.2.7.3. Remappable Input Example

6.2.7.4. [pin name]R

6.3. Function Priority for Device Pins

6.2.7.2 Remappable Input Priority

Only a single pin can be selected for any of the remappable peripheral inputs; therefore, priority encoding is not needed for RP inputs.

Note: A remappable input function does not have any control over the output of the RPx pin.

In this way, it is possible to drive a remappable output function on an RPx pin and a completely different remappable input function on the same pin. This can be useful, for instance, in driving an EVSYS output back into a Timer clock or gating input by assigning both functions to the same remap pin.

Note: To allow flexibility on 32 or 48-pin devices, the same input functions are repeated in multiple groups. Therefore, in order to differentiate between them, the “Off” code is provided in the *Input Pin Selection Group 1*, *Input Pin Selection Group 2*, *Input Pin Selection Group 3* and *Input Pin Selection Group 4* tables. See these tables in the *Remappable Input Example* from Related Links.

The software must ensure that the unused group register offset of a (repeated) input function is programmed to `4'h0` for proper operation. Failure to do so will lead to unknown behavior.

Related Links

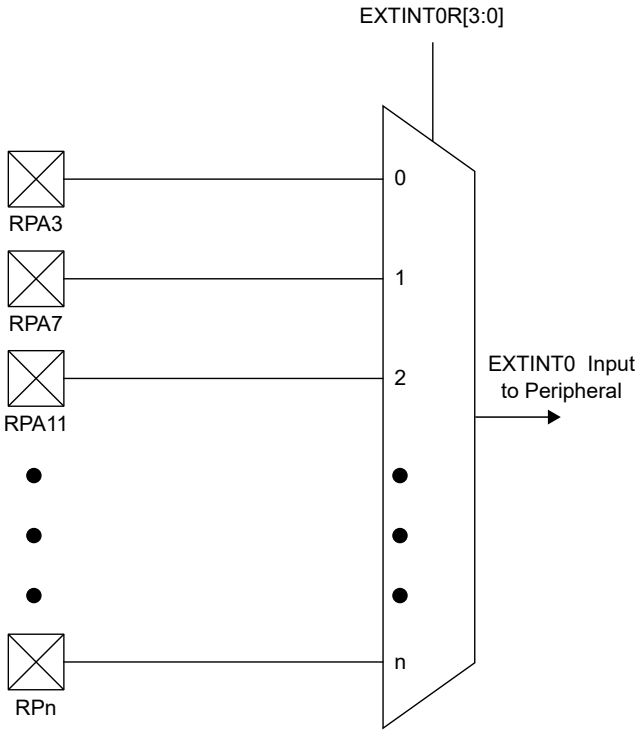
6.2.7.3. Remappable Input Example

6.2.7.3 Remappable Input Example

For example, the following figure illustrates the remappable pin selection for the EXTINT0 input. To remap the EXTINT0 input to a particular pin, the EXTINT0R remap register must be programmed. EXTINT0 is in group 1; therefore, it can be mapped to any pin that is in group 1 (RPA3, RPA7, RPA9, RPA11, RPB0, RPB4, RPB8 and so on).

To map it to RPB0, program the value 4 (4’b0100) into the EXINTR0R SFR register. See the following *Input Pin Selection Group 1* table.

Figure 6-2. EXTINT0 Remappable Pin Selection



Note: For input only, the PPS functionality does not have priority over the TRISx settings. Therefore, when configuring the RPn pin for input, the corresponding bit in the TRISx register must also be configured for input (set to '1').

Table 6-2. Input Pin Selection Group 1

Peripheral Pin	[*pin name*]R SFR	[*pin name*]R bits	[pin name]R Value to RPn Pin Selection
EXTINT0	EXTINT0R	EXTINT0[3:0]	0000 = OFF 0001 = RPA3 0010 = RPA7 0011 = RPA11 0100 = RPB0 0101 = RPB4 0110 = RPB8 0111 = RPB12 1000 = RPA2 1001 = RPA6 1010 = RPA10 1011 = RPA14 1100 = RPB3 1101 = RPB7 1110 = RPB11 1111 = RPA9
SERCOM0_PAD3	SCOM0P3R	SCOM0P3R[3:0]	
SERCOM1_PAD2	SCOM1P2R	SCOM1P2R[3:0]	
SERCOM2_PAD1	SCOM2P1R	SCOM2P1R[3:0]	
SERCOM3_PAD0	SCOM3P0R	SCOM3P0R[3:0]	
QSCK	QSCKR	QSCKR[3:0]	
QD1	QD1R	QD1R[3:0]	
REFI	REFIR	REFIR[3:0]	
CCLIN0	CCLIN0R	CCLIN0R[3:0]	
CCLIN3	CCLIN3R	CCLIN3R[3:0]	
TC0_WO0G1	TC0WO0G1R	TC0WO0G1R[3:0]	
TC1_WO0G1	TC1WO0G1R	TC1WO0G1R[3:0]	
TC2_WO0G1	TC2WO0G1R	TC2WO0G1R[3:0]	
TC3_WO0G1	TC3WO0G1R	TC3WO0G1R[3:0]	

Table 6-3. *Input Pin Selection Group 2*

Peripheral Pin	[*pin name*]R SFR	[*pin name*]R bits	[pin name]R Value to RPn Pin Selection
EXTINT1	EXTINT1R	EXTINT1R[3:0]	0000 = OFF 0001 = RPA4 0010 = RPA8 0011 = RPA12 0100 = RPB1 0101 = RPB5 0110 = RPB9 0111 = RPB13 1000 = RPA3 1001 = RPA7 1010 = RPA11 1011 = RPB0 1100 = RPB4 1101 = RPB8 1110 = RPB12 1111 = RPA0
SERCOM0_PAD0	SCOM0P0R	SCOM0P0R[3:0]	
SERCOM1_PAD3	SCOM1P3R	SCOM1P3R[3:0]	
SERCOM2_PAD2	SCOM2P2R	SCOM2P2R[3:0]	
SERCOM3_PAD1	SCOM3P1R	SCOM3P1R[3:0]	
QD2	QD2R	QD2R[3:0]	
CCLIN1	CCLIN1R	CCLIN1R[3:0]	
CCLIN4	CCLIN4R	CCLIN4R[3:0]	
TC0_WO0G2	TC0WO0G2R	TC0WO0G2R[3:0]	
TC1_WO1G2	TC1WO1G2R	TC1WO1G2R[3:0]	
TC2_WO1G2	TC2WO1G1R	TC2WO1G1R[3:0]	
TC3_WO1G2	TC3WO1G1R	TC3WO1G1R[3:0]	

Table 6-4. ***Input Pin Selection Group 3***

Peripheral Pin	[*pin name*]R SFR	[*pin name*]R bits	[pin name]R Value to RPn Pin Selection
EXTINT2	EXTINT2R	EXTINT2R[3:0]	0000 = OFF 0001 = RPA5 0010 = RPA9 0011 = RPA13 0100 = RPB2 0101 = RPB6 0110 = RPB10 0111 = RPA0 1000 = RPA4 1001 = RPA8 1010 = RPA12 1011 = RPB1 1100 = RPB5 1101 = RPB9 1110 = RPB13 1111 = RPA1
SERCOM0_PAD1	SCOM0P1R	SCOM0P1R[3:0]	
SERCOM1_PAD0	SCOM1P0R	SCOM1P0R[3:0]	
SERCOM2_PAD3	SCOM2P3R	SCOM2P3R[3:0]	
SERCOM3_PAD2	SCOM3P2R	SCOM3P2R[3:0]	
QD3	QD3R	QD3R[3:0]	
CCLIN2	CCLIN2R	CCLIN2R[3:0]	
CCLIN5	CCLIN5R	CCLIN5R[3:0]	
TC0_WO1G3	TC0WO1G3R	TC0WO1G3R[3:0]	
TC2_WO0G3	TC2WO0G3R	TC2WO0G3R[3:0]	
TC3_WO0G3	TC3WO0G3R	TC3WO0G3R[3:0]	

Table 6-5. ***Input Pin Selection Group 4***

Peripheral Pin	[*pin name*]R SFR	[*pin name*]R bits	[pin name]R Value to RPn Pin Selection
EXTINT3	EXTINT3R	EXTINT3R[3:0]	0000 = OFF 0001 = RPA6 0010 =RPA10 0011 = RPA14 0100 = RPB3 0101 =RPB7 0110 = RPB11 0111 = RPA1 1000 = RPA5 1001 = RPA9 1010 = RPA13 1011 = RPB2 1100 = RPB6 1101 = RPB10 1110 = RPA8 1111 = RPA2
NMI	NMIR	NMIR[3:0]	
SERCOM0_PAD2	SCOM0P2R	SCOM0P2R[3:0]	
SERCOM1_PAD1	SCOM1P1R	SCOM1P1R[3:0]	
SERCOM2_PAD0	SCOM2P0R	SCOM2P0R[3:0]	
SERCOM3_PAD3	SCOM3P3R	SCOM3P3R[3:0]	
QD0	QD0R	QD0R[3:0]	
TC0_WO1G4	TC0WO1G4R	TC0WO1G4R[3:0]	
TC2_WO1G4	TC2WO1G4R	TC2WO1G4R[3:0]	
TC3_WO1G4	TC3WO1G4R	TC3WO1G4R[3:0]	

6.2.7.4 Peripheral Pin Select Input Register

Name: *[pin name]*R
Offset: See the following Note
Reset: 0x00
Property: -

Notes:

1. For the Offset address, see *Peripheral Pin Select Input Registers* table in the *I/O Ports Control Registers* from Related Links.
2. Register values can only be changed if the IOLOCK Configuration bit (CFGCON0[13]) = 0.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					[pin name]R[3:0]			
Access					R/W-0	R/W-0	R/W-0	R/W-0
Reset					0	0	0	0

Bits 3:0 – *[pin name]*R[3:0] Peripheral Pin Select Input bits
Where [*pin name*] refers to the pins that are used to configure peripheral input mapping. See *Input Pin Selection Group 1*, *Input Pin Selection Group 2*, *Input Pin Selection Group 3* and *Input Pin Selection Group 4* tables in the *Remappable Input Example* for input pin selection values from Related Links.
Note: This field is only writable when CFGCON0.IOLOCK = 0.

Related Links

6.2.7.3. Remappable Input Example
6.4. I/O Ports Control Registers

6.2.8 Output Mapping

The remappable pin output assigns a peripheral output function to an output pin. Once the group for the output pin is identified, see the following table, which shows the peripheral output functions and its group.

Each remappable output can be programmed to an output function that is from its same output group number. As an example, if RPA0 is part of GROUP2, then it can be programmed to have any GROUP2 output function on its pin. Therefore, for a given output peripheral signal, the user must first choose which remappable pin to use, choose a Group number for that pin and, then, program the control registers for that pin. For example, RPA<0-10, 13, 14> G<1, 2, 3, 4> R or RPB<0-13> G<1, 2, 3, 4>R. See *Remappable Output Pin Configuration – Group1*, *Remappable Output Pin Configuration – Group2*, *Remappable Output Pin Configuration – Group3*, and *Remappable Output Pin Configuration – Group4* tables in the *Pin Output RP Registers* from Related Links.

The rules for which group belongs to which pin must be followed, such that multiple peripherals are not driving the same pin from different groups. For instance, pin RPA0 (PA0) as an output belongs to Group2 and Group3. If the peripheral driving the signal to RPA0 is coming from Group2, the software must ensure that all Group3 signals for RPA0 are disabled with an 'OFF' value in the corresponding RPA0G3R control register.

A null output is associated with the output register reset value of '0'. This is done to ensure that remappable outputs remain disconnected from all output pins, by default.

Table 6-6. PPS Output Groups

Group1	Group2	Group3	Group4
SERCOM0_PAD3	SERCOM0_PAD0	SERCOM0_PAD1	SERCOM0_PAD2
SERCOM0_PAD2	SERCOM0_PAD3	SERCOM0_PAD0	SERCOM0_PAD1
SERCOM0_PAD1	SERCOM0_PAD2	SERCOM0_PAD3	SERCOM0_PAD0
SERCOM1_PAD0	SERCOM1_PAD1	SERCOM1_PAD2	SERCOM1_PAD3
SERCOM1_PAD2	SERCOM1_PAD3	SERCOM1_PAD0	SERCOM1_PAD1
SERCOM1_PAD1	SERCOM1_PAD2	SERCOM1_PAD3	SERCOM1_PAD0
SERCOM2_PAD0	SERCOM2_PAD1	SERCOM2_PAD2	SERCOM2_PAD3
SERCOM2_PAD1	SERCOM2_PAD2	SERCOM2_PAD3	SERCOM2_PAD0
SERCOM3_PAD0	SERCOM3_PAD1	SERCOM3_PAD2	SERCOM3_PAD3
SERCOM3_PAD3	SERCOM3_PAD0	SERCOM3_PAD1	SERCOM3_PAD2
TCC0_WO0	TCC0_WO1	TCC0_WO2	TCC0_WO3
TCC0_WO4	TCC0_WO5	TCC0_WO0	TCC0_WO1
TCC0_WO2	TCC0_WO3	TCC0_WO4	TCC0_WO5
TCC1_WO0	TCC1_WO1	TCC1_WO2	TCC1_WO3
TCC1_WO4	TCC1_WO5	TCC1_WO0	TCC1_WO1
TCC1_WO2	TCC1_WO3	TCC1_WO4	TCC1_WO5
TCC2_WO0	TCC2_WO1	TCC2_WO0	TCC2_WO1
TC0_WO1	TC0_WO1	TC0_WO0	TC0_WO0
REFO1	REFO2	REFO3	REFO4
TC1_WO0	TC1_WO1	TC1_WO0	TC1_WO1
TC2_WO0	TC2_WO1	TC2_WO0	TC2_WO1
TC3_WO0	TC3_WO1	TC3_WO0	TC3_WO1
QSPI_SCK	QSPI_SCK	QSPI_SCK	QSPI_SCK
QSPI_CS	QSPI_CS	QSPI_CS	QSPI_CS
QSPI_DATA3	QSPI_DATA0	QSPI_DATA1	QSPI_DATA2
QSPI_DATA2	QSPI_DATA3	QSPI_DATA0	QSPI_DATA1
QSPI_DATA1	QSPI_DATA2	QSPI_DATA3	QSPI_DATA0
CCL_OUT0	CCL_OUT1	CCL_OUT0	CCL_OUT1

Related Links

6.2.9. Pin Output RP Registers

6.2.9 Pin Output RP Registers

Register *RPnR* shows the RP Remap Register format for output functions. See *RPnR* from Related Links. Each RP pin has a 4-bit field that can be assigned to the desired output function. See the following tables for a complete list of output function values and associated register names.

The mapping is dynamic; therefore, to avoid glitching outputs, the user must ensure that the appropriate peripherals are turned off before remapping the functions. On Reset, all inputs are mapped to a default value and all outputs are disabled; therefore, the mapping must be performed after any device Reset.

Figure 6-3. Example Muiltiplexing of Remappable Output Signal for RPA0 (Map Output Function to Pin)

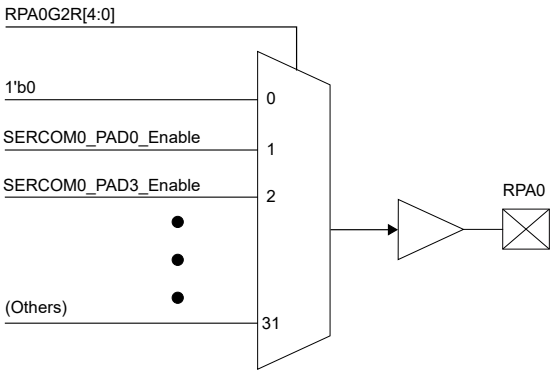


Table 6-7. Remappable Output Pin Configuration – Group1

Output Pin	[*pin name*]R SFR	[*pin name*]R bits	[pin name]R Value to RPn Pin Selection
RPA2	RPA2G1R	RPA2G1R[4:0]	00000 = OFF 00001 = SERCOM0_PAD3 00010 = SERCOM0_PAD2 00011 = SERCOM0_PAD1 00100 = SERCOM1_PAD0 00101 = SERCOM1_PAD2 00110 = SERCOM1_PAD1 00111 = SERCOM2_PAD0 01000 = SERCOM2_PAD1 01001 = SERCOM3_PAD0 01010 = SERCOM3_PAD3 01011 = TCC0_WO0 01100 = TCC0_WO4 01101 = TCC0_WO2 01110 = TCC1_WO0 01111 = TCC1_WO4 10000 = TCC1_WO2 10001 = TCC2_WO0 10010 = TC0_WO1 10011 = REFO1 10100 = TC1_WO0 10101 = TC2_WO0 10110 = TC3_WO0 10111 = QSCK 11000 = QCS 11001 = QD3 11010 = QD2 11011 = QD1 11100 = CCLOUT0 11101 = RESERVED 11110 = RESERVED 11111 = RESERVED
RPA3	RPA3G1R	RPA3G1R[4:0]	
RPA5	RPA5G1R	RPA5G1R[4:0]	
RPA6	RPA6G1R	RPA6G1R[4:0]	
RPA7	RPA7G1R	RPA7G1R[4:0]	
RPA9	RPA9G1R	RPA9G1R[4:0]	
RPA10	RPA10G1R	RPA10G1R[4:0]	
RPA14	RPA14G1R	RPA14G1R[4:0]	
RPB0	RPB0G1R	RPB0G1R[4:0]	
RPB3	RPB3G1R	RPB3G1R[4:0]	
RPB4	RPB4G1R	RPB4G1R[4:0]	
RPB7	RPB7G1R	RPB7G1R[4:0]	
RPB8	RPB8G1R	RPB8G1R[4:0]	
RPB11	RPB11G1R	RPB11G1R[4:0]	
RPB12	RPB12G1R	RPB12G1R[4:0]	

Table 6-8. Remappable Output Pin Configuration – Group2

Output Pin	[*pin name*]R SFR	[*pin name*]R bits	[pin name]R Value to RPn Pin Selection
RPA0	RPA0G2R	RPA0G2R[4:0]	00000 = OFF 00001 = SERCOM0_PAD0 00010 = SERCOM0_PAD3 00011 = SERCOM0_PAD2 00100 = SERCOM1_PAD1 00101 = SERCOM1_PAD3 00110 = SERCOM1_PAD2 00111 = SERCOM2_PAD1 01000 = SERCOM2_PAD2 01001 = SERCOM3_PAD1 01010 = SERCOM3_PAD0 01011 = TCC0_WO1 01100 = TCC0_WO5 01101 = TCC0_WO3 01110 = TCC1_WO1 01111 = TCC1_WO5 10000 = TCC1_WO3 10001 = TCC2_WO1 10010 = TC0_WO1 10011 = REFO2 10100 = TC1_WO1 10101 = TC2_WO1 10110 = TC3_WO1 10111 = QSCK 11000 = QCS 11001 = QD0 11010 = QD3 11011 = QD2 11100 = CCLOUT1 11101 = RESERVED 11110 = RESERVED 11111 = RESERVED
RPA3	RPA3G2R	RPA3G2R[4:0]	
RPA4	RPA4G2R	RPA4G2R[4:0]	
RPA6	RPA6G2R	RPA6G2R[4:0]	
RPA7	RPA7G2R	RPA7G2R[4:0]	
RPA8	RPA8G2R	RPA8G2R[4:0]	
RPB0	RPB0G2R	RPB0G2R[4:0]	
RPB1	RPB1G2R	RPB1G2R[4:0]	
RPB4	RPB4G2R	RPB4G2R[4:0]	
RPB5	RPB5G2R	RPB5G2R[4:0]	
RPB8	RPB8G2R	RPB8G2R[4:0]	
RPB9	RPB9G2R	RPB9G2R[4:0]	
RPB12	RPB12G2R	RPB12G2R[4:0]	
RPB13	RPB13G2R	RPB13G2R[4:0]	

Table 6-9. Remappable Output Pin Configuration - Group3

Output Pin	[*pin name*]R SFR	[*pin name*]R bits	[pin name]R Value to RPn Pin Selection
RPA0	RPA0G3R	RPA0G3R[4:0]	00000 = OFF 00001 = SERCOM0_PAD1 00010 = SERCOM0_PAD0 00011 = SERCOM0_PAD3 00100 = SERCOM1_PAD2 00101 = SERCOM1_PAD0 00110 = SERCOM1_PAD3 00111 = SERCOM2_PAD2 01000 = SERCOM2_PAD3 01001 = SERCOM3_PAD2 01010 = SERCOM3_PAD1 01011 = TCC0_WO2 01100 = TCC0_WO0 01101 = TCC0_WO4 01110 = TCC1_WO2 01111 = TCC1_WO0 10000 = TCC1_WO4 10001 = TCC2_WO0 10010 = TC0_WO0 10011 = REFO3 10100 = TC1_WO0 10101 = TC2_WO0 10110 = TC3_WO0 10111 = QSCK 11000 = QCS 11001 = QD1 11010 = QD0 11011 = QD3 11100 = CCLOUT0 11101 = RESERVED 11110 = RESERVED 11111 = RESERVED
RPA1	RPA1G3R	RPA1G3R[4:0]	
RPA3	RPA3G3R	RPA3G3R[4:0]	
RPA4	RPA4G3R	RPA4G3R[4:0]	
RPA5	RPA5G3R	RPA5G3R[4:0]	
RPA8	RPA8G3R	RPA8G3R[4:0]	
RPA9	RPA9G3R	RPA9G3R[4:0]	
RPA13	RPA13G3R	RPA13G3R[4:0]	
RPB1	RPB1G3R	RPB1GX3R[4:0]	
RPB2	RPB2G3R	RPB2G3R[4:0]	
RPB5	RPB5G3R	RPB5G3R[4:0]	
RPB6	RPB6G3R	RPB6G3R[4:0]	
RPB9	RPB9G3R	RPB9G3R[4:0]	
RPB10	RPB10G3R	RPB10G3R[4:0]	
RPB13	RPB13G3R	RPB13G3R[4:0]	

Table 6-10. Remappable Output Pin Configuration – Group4

Output Pin	[pin name]R SFR	[pin name]R Value to RPn Pin Selection
RPA1	RPA1G4R	00000 = OFF 00001 = SERCOM0_PAD2 00010 = SERCOM0_PAD1 00011 = SERCOM0_PAD0 00100 = SERCOM1_PAD3 00101 = SERCOM1_PAD1 00110 = SERCOM1_PAD0 00111 = SERCOM2_PAD3 01000 = SERCOM2_PAD0 01001 = SERCOM3_PAD3 01010 = SERCOM3_PAD2 01011 = TCC0_WO3 01100 = TCC0_WO1 01101 = TCC0_WO5 01110 = TCC1_WO3 01111 = TCC1_WO1 10000 = TCC1_WO5 10001 = TCC2_WO1 10010 = TC0_WO0 10011 = REFO4 10100 = TC1_WO1 10101 = TC2_WO1 10110 = TC3_WO1 10111 = QSCK 11000 = QCS 11001 = QD2 11010 = QD1 11011 = QD0 11100 = CCLOUT1 11101 = RESERVED 11110 = RESERVED 11111 = RESERVED
RPA2	RPA2G4R	
RPA4	RPA4G4R	
RPA5	RPA5G4R	
RPA6	RPA6G4R	
RPA8	RPA8G4R	
RPA9	RPA9G4R	
RPA10	RPA10G4R	
RPA13	RPA13G4R	
RPA14	RPA14G4R	
RPB2	RPB2G4R	
RPB3	RPB3G4R	
RPB6	RPB6G4R	
RPB7	RPB7G4R	
RPB10	RPB10G4R	
RPB11	RPB11G4R	

Related Links

6.2.9.1. RPnR

6.2.9.1 Peripheral Pin Select Output Register

Name: RPnR
Offset: See the following Note
Reset: 0x0
Property: -

Notes:

1. For the Offset address, see the *Peripheral Pin Select Output Registers* table in the *I/O Ports Control Registers* from Related Links.
2. Register values can only be changed if the IOLOCK Configuration bit (CFGCON0.IOLOCK) = 0.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

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

Bits 4:0 – RPnR[4:0] Peripheral Pin Select Output Register
Output bits. For output pin selection values, see *Remappable Output Pin Configuration – Group1*, *Remappable Output Pin Configuration – Group2*, *Remappable Output Pin Configuration – Group3*, and *Remappable Output Pin Configuration – Group4* tables in the *Pin Output RP Registers* from Related Links.
Note: This field is only writable, when CFGCON0.IOLOCK = 0.

Related Links

6.4. I/O Ports Control Registers
6.2.9. Pin Output RP Registers

6.3 Function Priority for Device Pins

The device pins have an associated priority order where functionality is exhibited on each pin. This priority order impacts the availability of PPS functionality. For example, if SERCOM0 is enabled with outputs chosen to be High Speed/Direct mode in the DEVCFG1 fuses (bit 17), pins PA3, PA4, PA5 and PA6 are given priority to be used as SERCOM0 pins instead of GPIO/PPS pins. See the following tables for the priority in which functions are brought out on each device pin.

Table 6-11. Priority for Device Pins PAn (n=0-14)

Pin Name (1)	Functions in Priority Order	Reference Peripheral	Pin Number (48-pin)	I/O	Type
pa0	QSPI_DATA2	QSPI	4	I/O	DIG/ST
	RTC_IN3	RTCC		I	ST/STMV
	RPA0	PPS		I/O	DIG/ST
	IOCA0	Change Notification		I	ST
	RA0	GPIO		I/O	DIG/ST
pa1	QSPI_DATA3	QSPI	5	I/O	DIG/ST
	AC_CMP1	Analog Comparator		O	DIG
	RTC_IN2	RTCC		I	ST/STMV
	RPA1	PPS		I/O	DIG/ST
	IOCA1	Change Notification		I	ST
	RA1	GPIO		I/O	DIG/ST
pa2	AC_CMP0	Analog Comparator	6	O	DIG
	RTC_IN1	RTCC		I	ST/STMV
	RPA2	PPS		I/O	DIG/ST
	IOCA2	Change Notification		I	ST
	RA2	GPIO		I/O	DIG/ST
pa3	SCLKI	Secondary Oscillator - Digital	47	I	ST/STMV
	SERCOM0_PAD2	SERCOM0		I/O	DIG/ST
	RTC_IN0	RTCC		I	ST/STMV
	RPA3	PPS		I/O	DIG/ST
	IOCA3	Change Notification		I	ST
	RA3	GPIO		I/O	DIG/ST
pa4	SERCOM0_PAD3	SERCOM0	42	I/O	DIG/ST
	RTC_OUT	RTCC		O	DIGMV
	RPA4	PPS		I/O	DIG/ST
	IOCA4	Change Notification		I	ST
	RA4	GPIO		I/O	DIG/ST
pa5	SERCOM0_PAD0	SERCOM0	7	I/O	DIG/ST/ I2C(1)/SMB
	RPA5	PPS		I/O	DIG/ST
	IOCA5	Change Notification		I	ST
	RA5	GPIO		I/O	DIG/ST

.....continued

Pin Name (1)	Functions in Priority Order	Reference Peripheral	Pin Number (48-pin)	I/O	Type
pa6	PGC2ENTRY	DEBUG	9	I	ST
	SERCOM0_PAD1	SERCOM0		I/O	DIG/ST/ I2C(1)/SMB
	AC_CMP1_ALT	Analog Comparator		O	DIG
	RPA6	PPS		I/O	DIG/ST
	IOCA6	Change Notification		I	ST
	RA6	GPIO		I/O	DIG/ST
pa7	TRACECLK	DEBUG	10	I	DIG
	SERCOM1_PAD0	SERCOM1		I/O	DIG/ST/ I2C(1)/SMB
	RPA7	PPS		I/O	DIG/ST
	IOCA7	Change Notification		I	ST
	RA7	GPIO		I/O	DIG/ST
pa8	PGD2ENTRY	DEBUG	11	I	ST
	FECTRL0	RF Radio		O	DIG
	SERCOM1_PAD1	SERCOM1		I/O	DIG/ST/ I2C(1)/SMB
	RPA8	PPS		I/O	DIG/ST
	IOCA8	Change Notification		I	ST
	RA8	GPIO		I/O	DIG/ST
pa9	FECTRL1	RF Radio	12	O	DIG
	SERCOM1_PAD2	SERCOM1		I/O	DIG/ST
	RTC_IN0_ALT	RTCC		I	ST/STMV
	RPA9	PPS		I/O	DIG/ST
	IOCA9	Change Notification		I	ST
	RA9	GPIO		I/O	DIG/ST
pa10	FECTRL2	RF Radio	13	O	DIG
	SERCOM1_PAD3	SERCOM1		I/O	DIG/ST
	RTC_OUT_ALT	RTCC		O	DIGMV
	RPA10	PPS		I/O	DIG/ST
	IOCA10	Change Notification		I	ST
	RA10	GPIO		I/O	DIG/ST
pa11	SOSCI	Secondary Oscillator	45	—	—
pa12	SOSCO	Secondary Oscillator	46	—	—

.....continued					
Pin Name (1)	**Functions in Priority Order**	**Reference Peripheral**	**Pin Number (48-pin)**	**I/O**	**Type**
pa13	COEXCTRL0	RF Radio	16	I/O	DIG/ST
	SERCOM2_PAD0	SERCOM2		I/O	DIG/ST/ I2C(1)/SMB
	RPA13	PPS		I/O	DIG/ST
	IOCA13	Change Notification		I	ST
	RA13	GPIO		I/O	DIG/ST
pa14	COEXCTRL1	RF Radio	17	I/O	DIG/ST
	SERCOM2_PAD1	SERCOM2		I/O	DIG/ST/ I2C(1)/SMB
	RPA14	PPS		I/O	DIG/ST
	IOCA14	Change Notification		I	ST
	RA14	GPIO		I/O	DIG/ST

Note:

1. SERCOMx_PADx supports SPI, UART, and I2C modes. SERCOM I2C mode is specifically mentioned in the "Type" column as it does not support Peripheral Pin Select (PPS) functionality.

Table 6-12. Priority for Device Pins PBn (n=0-13)

Pin Name (1)	Functions in Priority Order	Reference Peripheral	Pin Number (48-pin)	I/O	Type
pb0	COEXCTRL2	RF Radio	30	I/O	DIG/ST
	AN4	ADC		I	ANALOG
	AC_AIN2	Analog Comparator		I	ANALOG
	RPB0	PPS		I/O	DIG/ST
	IOCB0	Change Notification		I	ST
	RB0	GPIO		I/O	DIG/ST
pb1	AN5	ADC	31	I	ANALOG
	AC_AIN3	Analog Comparator		I	ANALOG
	RPB1	PPS		I/O	DIG/ST
	IOCB1	Change Notification		I	ST
	RB1	GPIO		I/O	DIG/ST
pb2	AN6	ADC	32	I	ANALOG
	AC_AIN0	Analog Comparator		I	ANALOG
	RPB2	PPS		I/O	DIG/ST
	IOCB2	Change Notification		I	ST
	RB2	GPIO		I/O	DIG/ST

.....continued					
Pin Name (1)	**Functions in Priority Order**	**Reference Peripheral**	**Pin Number (48-pin)**	**I/O**	**Type**
pb3	AN7	ADC	33	I	ANALOG
	AC_AIN1	Analog Comparator		I	ANALOG
	RPB3	PPS		I/O	DIG/ST
	IOCB3	Change Notification		I	ST
	RB3	GPIO		I/O	DIG/ST
pb4	PGC1ENTRY	Debug	34	I	ST
	FECTRL3	RF Radio		O	DIG
	AN0	ADC		I	ANALOG
	TP4	DO NOT USE		—	—
	RPB4	PPS		I/0	DIG/ST
	INT0	Edge Interrupt		I	ST/STMV
	IOCB4	Change Notification		I	ST
	RB4	GPIO		I/O	DIG/ST
pb5	PGC4ENTRY	Debug	35	I	ST
	TRACEDAT0	Debug		O	DIG
	FECTRL4	RF Radio		O	DIG
	AN1	ADC		I	ANALOG
	TP5	DO NOT USE		—	—
	RPB5	PPS		I/O	DIG/ST
	IOCB5	Change Notification		I	ST
	RB5	GPIO		I/O	DIG/ST
pb6	PGD1ENTRY	Debug	37	I	ST
	TRACEDAT1	Debug		O	DIG
	FECTRL5	RF Radio		O	DIG
	AN2	ADC		I	ANALOG
	ANN0	ADC (Differential)		I	ANALOG
	RPB6	PPS		I/O	DIG/ST
	IOCB6	Change Notification		I	ST
	RB6	GPIO		I/O	DIG/ST

.....continued					
Pin Name (1)	**Functions in Priority Order**	**Reference Peripheral**	**Pin Number (48-pin)**	**I/O**	**Type**
pb7	PGD4ENTRY	Debug	38	I	ST
	CM4_SWO	Debug		O	DIG
	TRACEDAT2	Debug		O	DIG
	AN3	ADC		I	ANALOG
	LVDIN	LVD Voltage Reference		I	ANALOG
	RPB7	PPS		I/O	DIG/ST
	IOCB7	Change Notification		I	ST
	RB7	GPIO		I/O	DIG/ST
pb8	CM4_SWCLK	DEBUG	41	I	ST
	RPB8	PPS		I/O	DIG/ST
	IOCB8	Change Notification		I	ST
	RB8	GPIO		I/O	DIG/ST
pb9	CM4_SWDIO	DEBUG	40	I/O	DIG/ST
	RPB9	PPS		I/O	DIG/ST
	IOCB9	Change Notification		I	ST
	RB9	GPIO		I/O	DIG/ST
pb10	QSPI_CS	QSPI	43	O	DIG
	RPB10	PPS		I/O	DIG/ST
	IOCB10	Change Notification		I	ST
	RB10	GPIO		I/O	DIG/ST
pb11	QSPI_SCK	QSPI	44	I/O	DIG/ST
	RPB11	PPS		I/O	DIG/ST
	IOCB11	Change Notification		I	ST
	RB11	GPIO		I/O	DIG/ST
pb12	QSPI_DATA0	QSPI	14	I/O	DIG/ST
	RPB12	PPS		I/O	DIG/ST
	IOCB12	Change Notification		I	ST
	RB12	GPIO		I/O	DIG/ST
pb13	QSPI_DATA1	QSPI	15	I/O	DIG/ST
	RTC_EVENT	RTCC		O	DIGMV
	RPB13	PPS		I/O	DIG/ST
	IOCB13	Change Notification		I	ST
	RB13	GPIO		I/O	DIG/ST

6.4 I/O Ports Control Registers

Notes: The following conventions are used in the following tables:

- x = Unknown value on Reset
- — = Unimplemented, read as '0'; Reset values are shown in hexadecimal

Table 6-13. PortA Register Map

Virtual Address (0x4400_2200)	Register	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
0010	TRISA	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	TRISA14	TRISA13	TRISA12*	TRISA11*	TRISA10	TRISA9	TRISA8	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	0000
0020	PORTA	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	RA14	RA13	RA12*	RA11*	RA10	RA9	RA8	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	0000
0030	LATA	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	LATA14	LATA13	LATA12*	LATA11*	LATA10	LATA9	LATA8	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0	0000
0040	ODCA	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	ODCA14	ODCA13	—	—	ODCA10	ODCA9	ODCA8	ODCA7	ODCA6	ODCA5	ODCA4	ODCA3	ODCA2	ODCA1	ODCA0	0000
0050	CNPUA	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	CNPUA14	CNPUA13	—	—	CNPUA10	CNPUA9	CNPUA8	CNPUA7	CNPUA6	CNPUA5	CNPUA4	CNPUA3	CNPUA2	CNPUA1	CNPUA0	0000
0060	CNPDA	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	CNPDA14	CNPDA13	—	—	CNPDA10	CNPDA9	CNPDA8	CNPDA7	CNPDA6	CNPDA5	CNPDA4	CNPDA3	CNPDA2	CNPDA1	CNPDA0	0000
0070	CNCONA	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	ON	FRZ	SIDL	—	EDGEDETECT	—	—	—	—	—	—	—	—	—	—	—	0000
0080	CNENA	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	CNENA14	CNENA13	—	—	CNENA10	CNENA9	CNENA8	CNENA7	CNENA6	CNENA5	CNENA4	CNENA3	CNENA2	CNENA1	CNENA0	0000
0090	CNSTATA	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	CN STATA14	CN STATA13	CN STATA12	CN STATA11	CN STATA10	CN STATA9	CN STATA8	CN STATA7	CN STATA6	CN STATA5	CN STATA4	CN STATA3	CN STATA2	CN STATA1	CN STATA0	0000
00A0	CNNEA	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	CNNEA14	CNNEA13	—	—	CNNEA10	CNNEA9	CNNEA8	CNNEA7	CNNEA6	CNNEA5	CNNEA4	CNNEA3	CNNEA2	CNNEA1	CNNEA0	0000
00B0	CNFA	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	CNFA14	CNFA13	—	—	CNFA10	CNFA9	CNFA9	CNFA7	CNFA76	CNFA5	CNFA4	CNFA3	CNFA2	CNFA71	CNFA0	0000

.....continued

Virtual Address (0x4400_2200)	Register	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
00C0	SRCON0A	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	SR014	SR013	—	—	SR010	SR09	SR08	SR07	SR06	SR05	SR04	SR03	—	SR01	SR00	0000
00D0	SRCON1A	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	SR114	SR113	—	—	SR110	SR19	SR18	SR17	SR16	SR15	SR14	SR13	—	SR11	SR10	0000

1. All registers in this table have corresponding CLR, SET and INV registers at their virtual addresses, plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET and INV Registers* from Related Links.

Note:
* - Not applicable for PIC32CX1012BZ25048

Table 6-14. PortB Register Map

Virtual Address (0x4400_2300)	Register	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
0100	ANSELB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	ANSB7	ANSB6	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0	FFFF
0110	TRISB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	TRISB13	TRISB12	TRISB11	TRISB10	TRISB9	TRISB8	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	0000
0120	PORTB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	RB13	RB12	RB11	RB10	RB9	RB8	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	0000
0130	LATB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	LATB13	LATB12	LATB11	LATB10	LATB9	LATB8	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	0000
0140	ODCB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	ODCB13	ODCB12	ODCB11	ODCB10	ODCB9	ODCB8	ODCB7	ODCB6	ODCB5	ODCB4	ODCB3	ODCB2	ODCB1	ODCB0	0000
0150	CNPUB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	CNPUB13	CNPUB12	CNPUB11	CNPUB10	CNPUB9	CNPUB8	CNPUB7	CNPUB6	CNPUB5	CNPUB4	CNPUB3	CNPUB2	CNPUB1	CNPUB0	0000
0160	CNPDB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	CNPDB13	CNPDB12	CNPDB11	CNPDB10	CNPDB9	CNPDB8	CNPDB7	CNPDB6	CNPDB5	CNPDB4	CNPDB3	CNPDB2	CNPDB1	CNPDB0	0000
0170	CNCONB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	ON	FRZ	SIDL	—	EDGE DETECT	—	—	—	—	—	—	—	—	—	—	—	0000
0180	CNENB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	CNENB13	CNENB12	CNENB11	CNENB10	CNENB9	CNENB8	CNENB7	CNENB6	CNENB5	CNENB4	CNENB3	CNENB2	CNENB1	CNENB0	0000
0190	CNSTATB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	CN STATB13	CN STATB12	CN STATB11	CN STATB10	CN STATB9	CN STATB8	CN STATB7	CN STATB6	CN STATB5	CN STATB4	CN STATB3	CN STATB2	CN STATB1	CN STATB0	0000
01A0	CNNEB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	CNNEB13	CNNEB12	CNNEB11	CNNEB10	CNNEB9	CNNEB8	CNNEB7	CNNEB6	CNNEB5	CNNEB4	CNNEB3	CNNEB2	CNNEB1	CNNEB0	0000

.....continued

Virtual Address (0x4400_2300)	Register	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
01B0	CNFB	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	CNFB13	CNFB12	CNFB11	CNFB10	CNFB9	CNFB8	CNFB7	CNFB6	CNFB5	CNFB4	CNFB3	CNFB2	CNFB1	CNFB0	0000
01C0	SRCON0B	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	SR013	SR012	SR011	SR010	—	—	—	—	—	—	—	—	—	—	0000
01D0	SRCON1B	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	SR113	SR112	SR111	SR110	—	—	—	—	—	—	—	—	—	—	0000

1. All registers in this table have corresponding CLR, SET and INV registers at their virtual address, plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET and INV Registers* from Related Links.

Notes: The following conventions are used in the following tables:

- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- -n = Value at POR
- '1' = Bit is set
- '0' = Bit is cleared
- x = Bit is unknown

Table 6-15. Peripheral Pin Select Input Registers

Virtual Address (4400_1000)	Register Name	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
0000h	EXTINT0R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	EXTINT0R[3:0]				0000
0004h	EXTINT1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	EXTINT1R[3:0]				0000
0008h	EXTINT2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	EXTINT2R[3:0]				0000
000Ch	EXTINT3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	EXTINT3R[3:0]				0000
003Ch	NMIR	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	NMIR[3:0]				0000
0040h	SCOM0P0R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM0P0R[3:0]				0000
0044h	SCOM0P1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM0P1R[3:0]				0000
0048h	SCOM0P2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM0P2R[3:0]				0000
004Ch	SCOM0P3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM0P3R[3:0]				0000
0050h	SCOM1P0R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM1P0R[3:0]				0000
0054h	SCOM1P1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM1P1R[3:0]				0000
0058h	SCOM1P2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM1P2R[3:0]				0000

.....continued

Virtual Address (4400_1000)	Register Name	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
005Ch	SCOM1P3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM1P3R[3:0]				0000
0060h	SCOM2P0R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM2P0R[3:0]				0000
0064h	SCOM2P1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM2P1R[3:0]				0000
0068h	SCOM2P2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM2P2R[3:0]				0000
006Ch	SCOM2P3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM2P3R[3:0]				0000
0070h	SCOM3P0R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM3P0R[3:0]				0000
0074h	SCOM3P1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM3P1R[3:0]				0000
0078h	SCOM3P2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM3P2R[3:0]				0000
007Ch	SCOM3P3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	SCOM3P3R[3:0]				0000
0080h	QSCKR	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	QSCKR[3:0]				0000
0084h	QD0R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	QD0R[3:0]				0000
0088h	QD1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	QD1R[3:0]				0000

.....continued

Virtual Address (4400_1000)	Register Name	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
008Ch	QD2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	QD2R[3:0]				0000
0090h	QD3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	QD3R[3:0]				0000
0094h	REFIR	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	REFIR[3:0]				0000
0098h	CCLIN0R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	CCLIN0R[3:0]				0000
009Ch	CCLIN1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	CCLIN1R[3:0]				0000
00A0h	CCLIN2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	CCLIN2R[3:0]				0000
00A4h	CCLIN3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	CCLIN3R[3:0]				0000
00A8h	CCLIN4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	CCLIN4R[3:0]				0000
00ACh	CCLIN5R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	CCLIN5R[3:0]				0000
00B0h	TC0WO0G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC0WO0G1R[3:0]				0000
00B4h	TC0WO0G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC0WO0G2R[3:0]				0000
00B8h	TC0WO1G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC0WO1G3R[3:0]				0000

.....continued

Virtual Address (4400_1000)	Register Name	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
00BCh	TC0WO1G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC0WO1G4R[3:0]				0000
00C0h	TC1WO0G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC1WO0G1R[3:0]				0000
00C4h	TC1WO1G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC1WO0G2R[3:0]				0000
00C8h	TC2WO0G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC2WO0G1R[3:0]				0000
00CCh	TC2WO0G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC2WO0G3R[3:0]				0000
00D0h	TC2WO1G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC2WO1G2R[3:0]				0000
00D4h	TC2WO1G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC2WO1G4R [3:0]				
00D8h	TC3WO0G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC3WO0G1R [3:0]				
00DCh	TC3WO0G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC3WO0G3R[3:0]				
00E0h	TC3WO1G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC3WO1G2R [3:0]				
00E4h	TC3WO1G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	TC3WO1G4R[3:0]				0000

Table 6-16. Peripheral Pin Select Output Registers

Virtual Address (4400_1000) (1)	Register Name	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
0200h	RPA0G2R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA0G2R[4:0]					0000
0204h	RPA0G3R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA0G3R[4:0]					0000
0208h	RPA1G3R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA1G3R[4:0]					0000
020Ch	RPA1G4R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA1G4R[4:0]					0000
0210h	RPA2G1R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA2G1R[4:0]					0000
0214h	RPA2G4R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA2G4R [4:0]					0000
0218h	RPA3G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA3G1R[4:0]					0000
021Ch	RPA3G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA3G2R[4:0]					0000
0220h	RPA3G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA3G3R[4:0]					0000
0224h	RPA4G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA4G2R[4:0]					0000
0228h	RPA4G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA4G3R[4:0]					0000
022Ch	RPA4G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA4G4R[4:0]					0000

.....continued

Virtual Address (4400_1000) (1)	Register Name	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
0230h	RPA5G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA5G1R[4:0]					0000
0234h	RPA5G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA5G3R[4:0]					0000
0238h	RPA5G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA5G4R[4:0]					0000
023Ch	RPA6G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA6G1R[4:0]					0000
0240h	RPA6G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA6G2R[4:0]					0000
0244h	RPA6G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA6G4R[4:0]					0000
0248h	RPA7G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA7G1R[4:0]					0000
024Ch	RPA7G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA7G2R[4:0]					0000
0250h	RPA8G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA8G2R[4:0]					0000
0254h	RPA8G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA8G3R[4:0]					0000
0258h	RPA8G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA8G4R[4:0]					0000
025Ch	RPA9G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA9G1R[4:0]					0000

.....continued

Virtual Address (4400_1000) (1)	Register Name	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
0260h	RPA9G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA9G3R[4:0]					0000
0264h	RPA9G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA9G4R[4:0]					0000
0268h	RPA10G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA10G1R[4:0]					0000
026Ch	RPA10G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA10G4R[4:0]					0000
0278h	RPA13G3R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA13G3R[4:0]					0000
027Ch	RPA13G4R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA13G4R[4:0]					0000
0280h	RPA14G1R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA14G1R[4:0]					0000
0284h	RPA14G4R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPA14G4R[4:0]					0000
028Ch	RPB0G1R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB0G1R[4:0]					0000
0290h	RPB0G2R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB0G2R[4:0]					0000
0294h	RPB1G2R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB1G2R[4:0]					0000
0298h	RPB1G3R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB1G3R[4:0]					0000

.....continued

Virtual Address (4400_1000) (1)	Register Name	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
029Ch	RPB2G3R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB2G3R[4:0]					0000
02A0h	RPB2G4R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB2G4R[4:0]					0000
02A4h	RPB3G1R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB3G1R[4:0]					0000
02A8h	RPB3G4R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB3G4R[4:0]					0000
02ACh	RPB4G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB4G1R[4:0]					0000
02B0h	RPB4G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB4G2R[4:0]					0000
02B4h	RPB5G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB5G2R[4:0]					0000
02B8h	RPB5G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB5G3R[4:0]					0000
02BCh	RPB6G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB6G3R[4:0]					0000
02C0h	RPB6G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB6G4R[4:0]					0000
02C4h	RPB7G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB7G1R[4:0]					0000
02C8h	RPB7G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB7G4R[4:0]					0000

.....continued

Virtual Address (4400_1000) (1)	Register Name	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
02CCh	RPB8G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB8G1R[4:0]					0000
02D0h	RPB8G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB8G2R[4:0]					0000
02D4h	RPB9G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB9G2R[4:0]					0000
02D8h	RPB9G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB9G3R[4:0]					0000
02DCh	RPB10G3R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB10G3R[4:0]					0000
02E0h	RPB10G4R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB10G4R[4:0]					0000
02E4h	RPB11G1R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB11G1R[4:0]					0000
02E8h	RPB11G4R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB11G4R[4:0]					0000
02ECh	RPB12G1R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB12G1R[4:0]					0000
02F0h	RPB12G2R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB12G2R[4:0]					0000
02F4h	RPB13G2R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB13G2R[4:0]					0000
02F8h	RPB13G3R*	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	RPB13G3R[4:0]					0000

Related Links

6.1.9. CLR, SET and INV Registers

6.4.1 Change Notice Control for PORTx Register

Name: CNCONx
Offset: See the following Note
Reset: 0x0
Property: -

Note:

1. For the Offset address, see the *PortA Register Map* and *PortB Register Map* tables in the *I/O Ports Control Registers* from Related Links.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON	FRZ	SIDL		EDGEDETECT			
Access	R/W	R/W	R/W		R/W			
Reset	0	0	0		0			

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – ON Change Notice (CN) Control ON bit
1 = Change Notice is enabled
0 = Change Notice is disabled

Bit 14 – FRZ Freeze in the Debug mode bit
1 = Freezes the module operation when the emulator is in the Debug mode
0 = Continues the module operation when the emulator is in the Debug mode

Bit 13 – SIDL Stop in the Idle mode bit
1 = Discontinues the module operation when device enters the Idle mode
0 = Continues the module operation even in the Idle mode

Bit 11 – EDGEDETECT Change Notification Style bit
1 = Edge Style. Detects edge transitions (CNFx used for CN Event).
0 = Mismatch Style. Detects change from last PORTx read (CNSTATx used for CN Event).

Related Links
6.4. I/O Ports Control Registers

6.5 Operation in Power Saving Modes

6.5.1 I/O Port Operation in Sleep Mode

As the device enters the Sleep mode, the system clock is disabled; however, the CN module continues to operate. If one of the enabled CN pins changes state, the corresponding interrupt flag is set in NVIC and device wakes from the Sleep (or Idle) mode and executes the CN Interrupt Service Routine.

6.5.2 I/O Port Operation in Idle Mode

As the device enters the Idle mode, the system clock sources remain functional. The SIDL bit (CNCONx[13]) selects whether the module will stop or continues to function in the Idle mode.

- If SIDL = 1, the module continues to sample Input CN I/O pins in the Idle mode; however, synchronization is disabled.
- If SIDL = 0, the module continues to synchronize and samples input CN I/O pins in the Idle mode.

6.6 Results of Various Resets

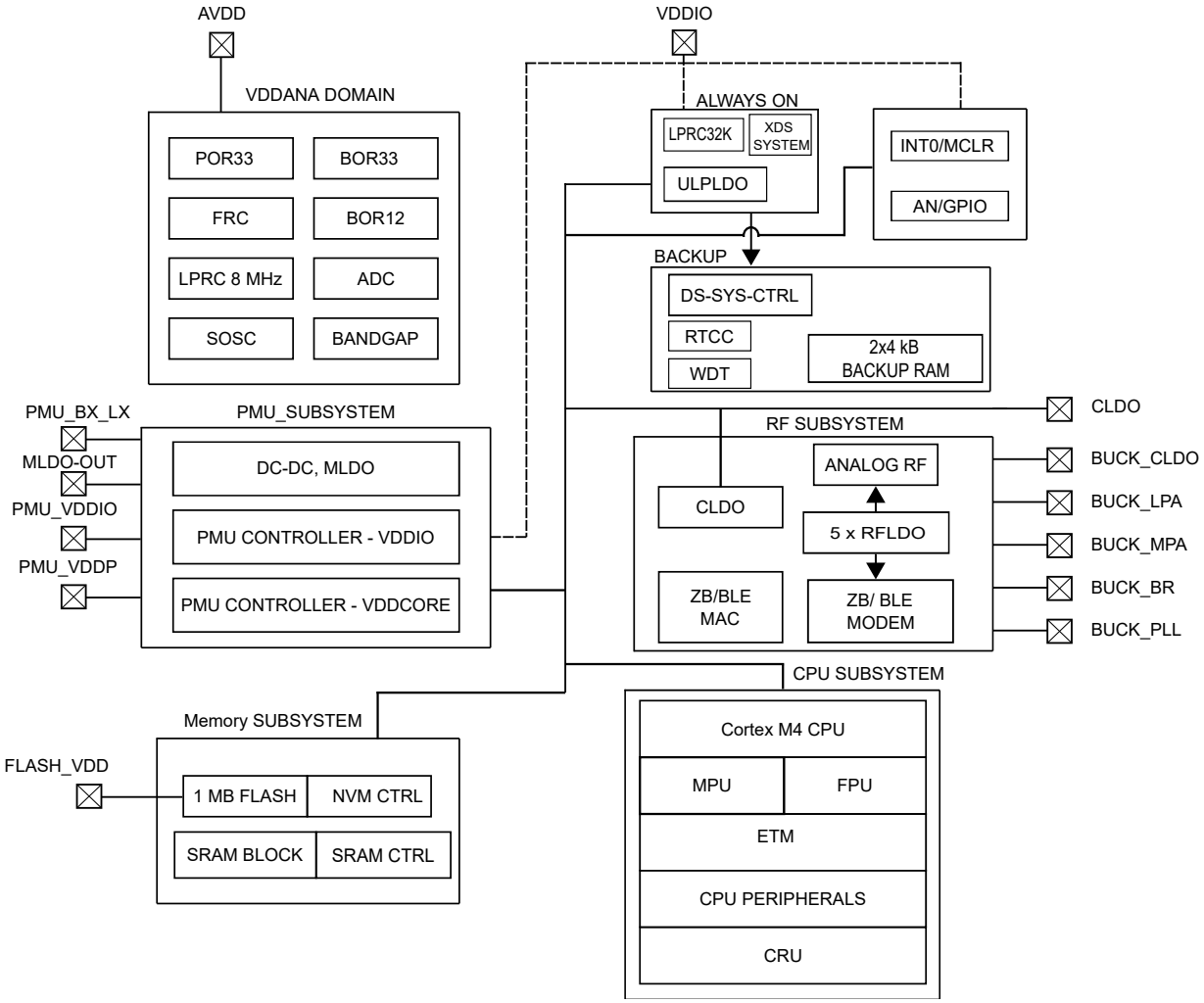
Table 6-17. Results of Resets Available

Reset Name	Description
Device Reset	All I/O registers are forced to their reset states upon a device Reset.
Power-on Reset (PoR)	All I/O registers are forced to their reset states upon a Power-on Reset (POR).
Watchdog Reset	All I/O registers are unchanged upon a Watchdog Reset.

7. Power Subsystem

7.1 Block Diagram

Figure 7-1. Power Subsystem Block Diagram



The power domains of PIC32CX-BZ2 and WBZ45 are as follows:

- VDD - 1.9V to 3.6V, Main Supply powering VDDIO, FLASH_VDD, AVDD, PMU_VDDIO, PMU_VDDP
  - VDDIO
    - 1.9V to 3.6, powering the AON, PMU-CTRL, AN/GPIO, INT0/MCLR, BKUP
  - FLASH_VDD
    - 1.9V to 3.6V, Filtered version of VDD (powering the Flash)
  - AVDD
    - 1.9V to 3.6V, Filtered version of VDD for system analog functionality
  - PMU_VDDIO
    - 1.9V to 3.6V, Filtered version of VDD (powering the PMU sub-system)
  - PMU_VDDP
    - 1.9V to 3.6V, Filtered version of VDD (powering the PMU sub-system)
- GND

  - Common GND for digital, analog and RF sub-systems

Other power supply pins are as follows:

- CLDO
  - Output pin for the internal voltage regulator for decoupling, this pin must not be used as an external power supply source
  - CLDO output is 1.2V
  - CLDO powers the VDDCORE
  - VDDCORE serves as the internal voltage regulator output. It powers the core, memories, peripherals
- PMU_BK_LX
  - Switching node for the Buck converter
- PMU_MLDO_OUT
  - 1.35V output pin from the internal LDO. This is the shared output pin for both MLDO and the DC-DC converter
  - PMU_MLDO_OUT powers the LDO's in the Wireless subsystem (BUCK_LPA, BUCK_MPA, BUCK_PLL, BUCK_BB and BUCK_CLDO)

For decoupling recommendations for the different power supplies, refer to the schematic checklist.

7.2 Voltage Regulators

The following voltage regulators are available in the power subsystem:

- MLDO – Linear voltage regulator generating 1.35V for powering CLDO and other RF LDOs operates from 1.9V to 3.6V
- DC-DC – Switching voltage regulator generating 1.35V operates from 2.4V to 3.6V
- ULP_VREG – Ultra low power voltage regulator for operation in back-up mode
- RF LDOs – Powering the different blocks of the RF subsystem
- CLDO – Powering the VDDCORE of PIC32CX-BZ2

7.3 Power Supply Modes

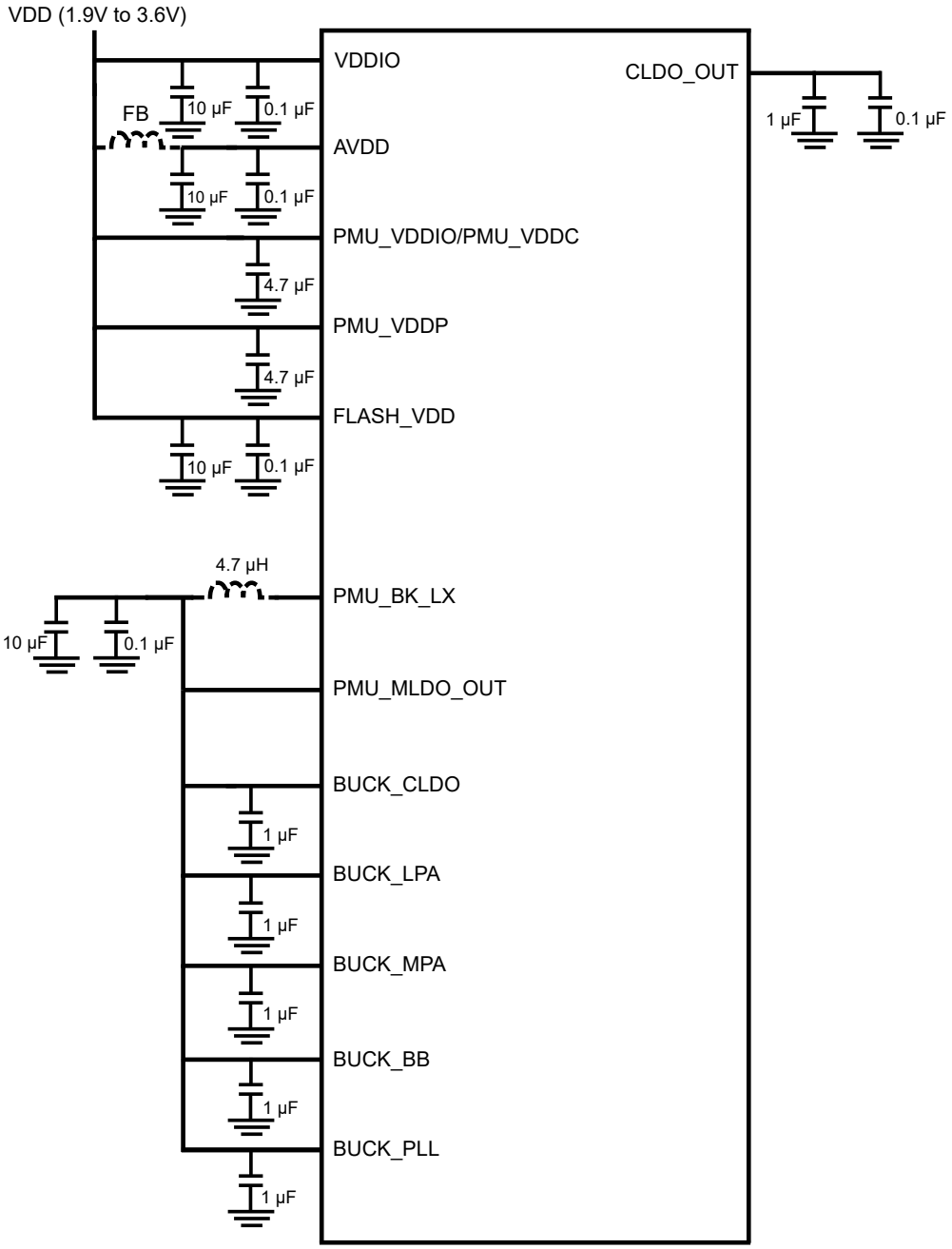
The PIC32CX-BZ2 supports a single power supply from 1.9V to 3.6V. The IO supply cannot be decoupled from the main supply. The same voltage must be applied to VDDx, PMU_VDDIO, VPMU_VDDC and AVDD with different levels of filtering. The internal voltage regulator has the following four different modes:

- Linear mode – This mode does not require any external inductor. This is the default mode when the CPU and peripherals are running. In this mode, the SoC is powered through the MLDO.
- Switching mode (Buck) – This is the most efficient mode when the CPU and peripherals are running. In this mode, the SoC is powered by the DC-DC converter.
- Low Power (PSK) mode – This is the mode used when the device is in Standby mode.
- Shutdown mode – When the device is in Backup mode, the internal regulator is turned off.

Selecting between the Switching mode and Linear mode can be done by software on-the-fly, but the power supply must be designed according to which mode is to be used.

7.4 Typical Power Supply Connection for SoC

Figure 7-2. Typical Power Supply Connection for SoC

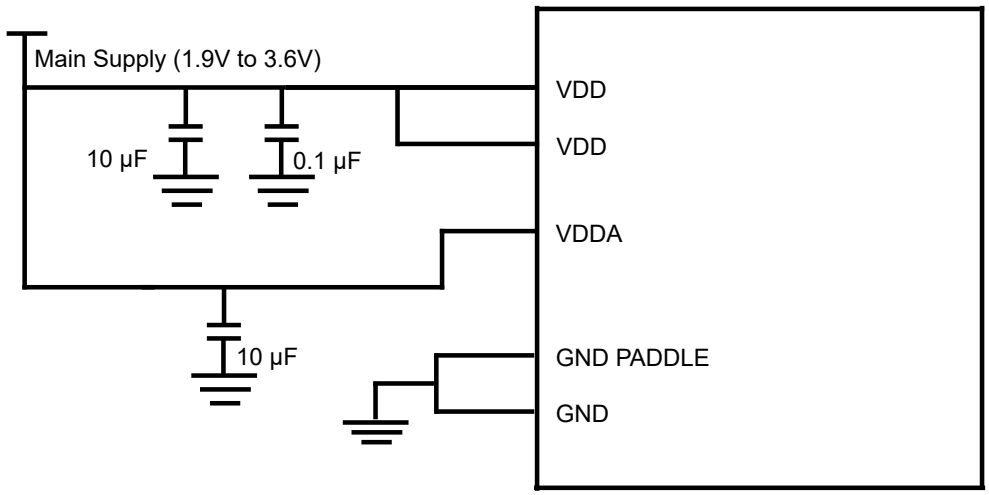


7.5 Typical Power Supply Connection for Module

The module requires only a single power supply on the VDD pins of the module.

- VDD can be from 1.9V to 3.6V

Figure 7-3. Module Schematics with VDD and Optional Bulk Capacitors



7.6 Power-Up Sequence

The characteristics of power-up sequence are as follows:

- The VDD/AVDD domains must rise at the same time
- On power-up, PIC32CX-BZ2 always start up on the MLDO mode
- After power-up in MLDO, the software can initiate the switch to Buck mode

7.6.1 Starting of Voltage Regulators

The characteristics of power-up voltage regulators are as follows:

- On power-up, the internal regulator starts in MLDO mode
- After MLDO boots up, the CLDO gets initialized
- Now the code execution can start
- The RF system is maintained in sleep-mode during the power-up time

7.6.2 Starting-up of Crystals

The characteristics of power-up crystals are as follows:

- The power-up of the SOC happens with the internal oscillators. After the power-up, the user software can request to switch on the SOSC and the XOSC crystals.

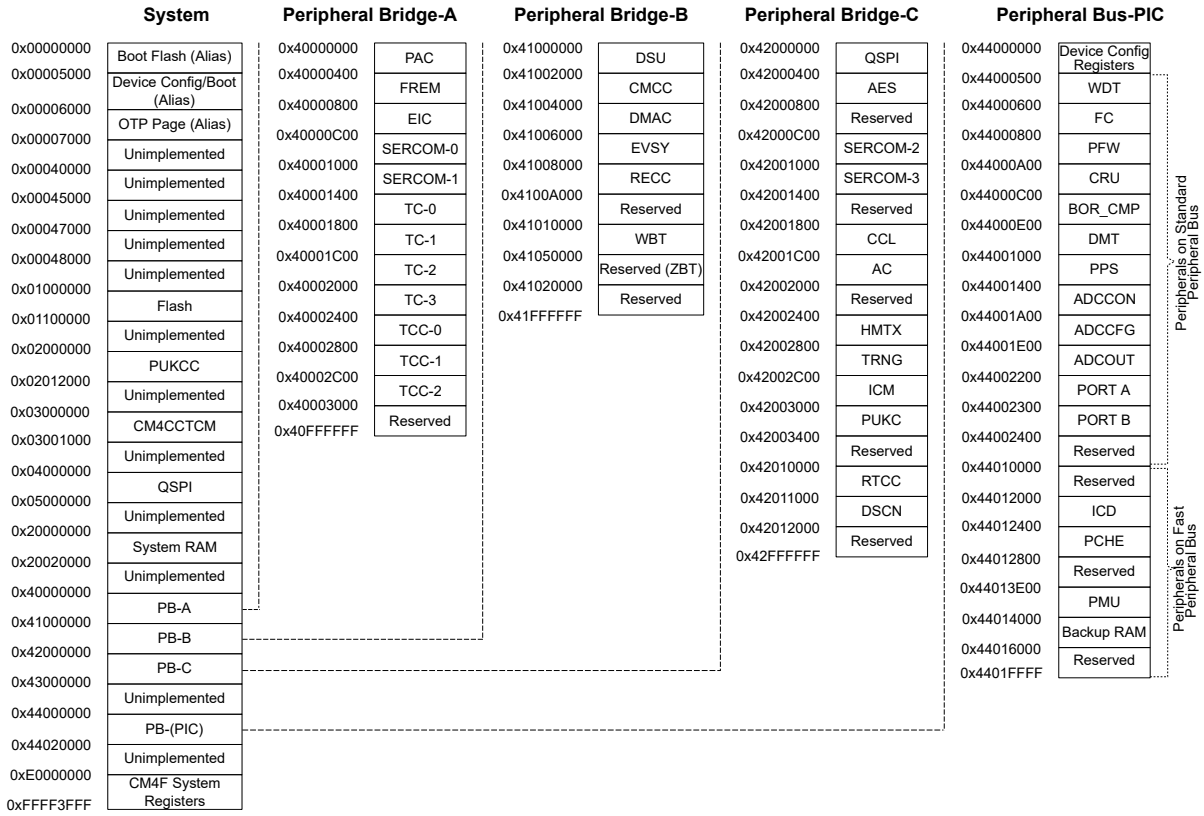
7.6.3 BOR and POR

The Brown-out Reset (BOR) monitors the VDD supply voltage. On detection of a brown-out condition, the BOR re-arms the POR. In this device, the min BOR trip point is the voltage below which the IO is deemed to be un-trusted; thus, it generates a reset. There are three BOR in PIC32CX-BZ2 and WBZ45:

- BOR3.3 to monitor VDD
- BOR1.2 to monitor VDDCORE
- ZPBOR monitors VDD during the deep sleep and extreme deep sleep mode if enabled

8. Product Memory Mapping Overview

Figure 8-1. Product Mapping



Notes:

1. Access attempts to any unimplemented memory location generates a bus error.
2. The PUKCC space is fixed as non-cacheable and bufferable.
3. The QSPI space “cacheable and bufferable” attribute is controlled using CFGCON1.QSCHE_EN.

Table 8-1. Device Configuration Map

Register Name		Reset value from NVR memory?	Writable in User Mode?	Corresponding Write Lock Bit(s)	Purpose
System Configuration					
CFGCON0(L)	0x4400_0000	X	X	CFGLOCK[1:0]	Misc. System Configuration
CFGCON1(L)	0x4400_0010	X	X	CFGLOCK[1:0]	Misc. System Configuration
CFGCON2(L)	0x4400_0020	X	X	CFGLOCK[1:0]	Misc. System Configuration
CFGCON3	0x4400_0030	—	X	CFGLOCK[1:0]	Misc. System Configuration
CFGCON4(L)	0x4400_0040	X	X	CFGLOCK[1:0]	Misc. System Configuration
CFGPGQOS	0x4400_0050	—	X	CFGCON0.PGLOCK	Bus Matrix Permission Groups
CFGPCLKGEN1	0x4400_0060	—	X	CFGLOCK[1:0]	Peripheral Clock Gen Reg-1
CFGPCLKGEN2	0x4400_0070	—	X	CFGLOCK[1:0]	Peripheral Clock Gen Reg-2
CFGPCLKGEN3	0x4400_0080	—	X	CFGLOCK[1:0]	Peripheral Clock Gen Reg-3
ID[2]	0x4400_0090	X	—	N/A- Read Only	32-bit Device ID
USER_ID(L)	0x4400_00A0	X	X	CFGLOCK[1:0]	16-bit User ID
SYSKEY	0x4400_00B0	—	X	Sequence	System Lock feature
PMD1	0x4400_00C0	—	X	CFGCON0.PMDLOCK	Peripheral Module Disable
PMD2	0x4400_00D0	—	X	CFGCON0.PMDLOCK	Peripheral Module Disable
PMD3	0x4400_00E0	—	X	CFGCON0.PMDLOCK	Peripheral Module Disable
Boot Configuration					
BCFG0	0x4400_0200	X	—	None	Pre-boot user configuration

Notes:

1. Registers marked with (L) are loadable from Flash, and they can be controlled by software after the boot with the correct unlock sequence.
2. ID Register is a JTAB ID register, maintained for legacy reasons. The actual external-facing ID register is DSU.DID register. See *DID* register in the *Device Service Unit (DSU)* from Related Links.

Table 8-2. CM4 System Components Register Offset Map

Peripheral		TA Clock	Size (Bytes)	Virtual Address		Physical Address	
ABRV.	Description			Start	End	Start	End
CM4 System Components accessible via CM4 PPB Bus (Only DAP and CM4 can access the registers)				Base Address 0xE000_0000		Base Address 0xE000_0000	

.....continued

Peripheral		TA Clock	Size (Bytes)	Virtual Address		Physical Address	
ABRV.	Description			Start	End	Start	End
ITM	Inst. TM	SYS_CLK	4 KB	0000_0000	0000_0FFF	0000_0000	0000_0FFF
DWT	Data WT	SYS_CLK	4 KB	0000_1000	0000_1FFF	0000_1000	0000_1FFF
FPB	Flash PB	SYS_CLK	4 KB	0000_2000	0000_2FFF	0000_2000	0000_2FFF
RSVD	Reserved	—	4 KB*n	0000_3000	0000_DFFF	0000_3000	0000_DFFF
SCS	Sys. CS	SYS_CLK	4 KB	0000_E000	0000_EFFF	0000_E000	0000_EFFF
RSVD	Reserved	—	4 KB*n	0000_F000	0003_FFFF	0000_F000	0003_FFFF
TPIU	Trace PIU	SYS_CLK	4 KB	0004_0000	0004_0FFF	0004_0000	0004_0FFF
ETM	ETM	SYS_CLK	4 KB	0004_1000	0004_1FFF	0004_1000	0004_1FFF
ETB	ETB	SYS_CLK	4 KB	0004_2000	0004_2FFF	0004_2000	0004_2FFF
RSVD	Reserved	—	4 KB*n	0004_3000	000F_EFFF	0004_3000	000F_EFFF
CROM	CSight ROM	SYS_CLK	4 KB	000F_F000	000F_FFFF	000F_F000	000F_FFFF
RSVD	Reserved	—	4 KB*n	0010_0000	FFFF_FFFF	0010_0000	FFFF_FFFF

Notes:

1. All system and debug components carry a unique ID accessible via its own register space.
2. The DAP derives the base address of the components from CROM entry values.
3. Component Base address = CROM Base address + CROM Entry value.
4. Core sight ROM entries are not provided in this document.
5. Refer to *CM4F* documentation for details on each component register space (developer.arm.com/documentation/ddi0439/b/System-Control/Register-summary).

Related Links

12.13.8. DID

9. Prefetch Cache (PCHE)

9.1 Introduction

The Prefetch Cache is a performance-enhancing module included in the PIC32CX-BZ2 devices, along with the L1 cache (Cortex M Cache Controller CMCC) to the Cortex M4F CPU.

9.2 Features

The Prefetch module increases the system performance for most of the applications.

The Prefetch module includes the following features:

- Fully associative lines for:
  - Four lines for CPU instructions cache
  - Two lines for CPU data cache
  - Two lines for peripheral data cache
- 16-byte cache lines and 128-bits parallel memory fetch
- Configurable predictive prefetch for CPU instructions cache
- Error detection and correction (ECC)

9.3 Overview

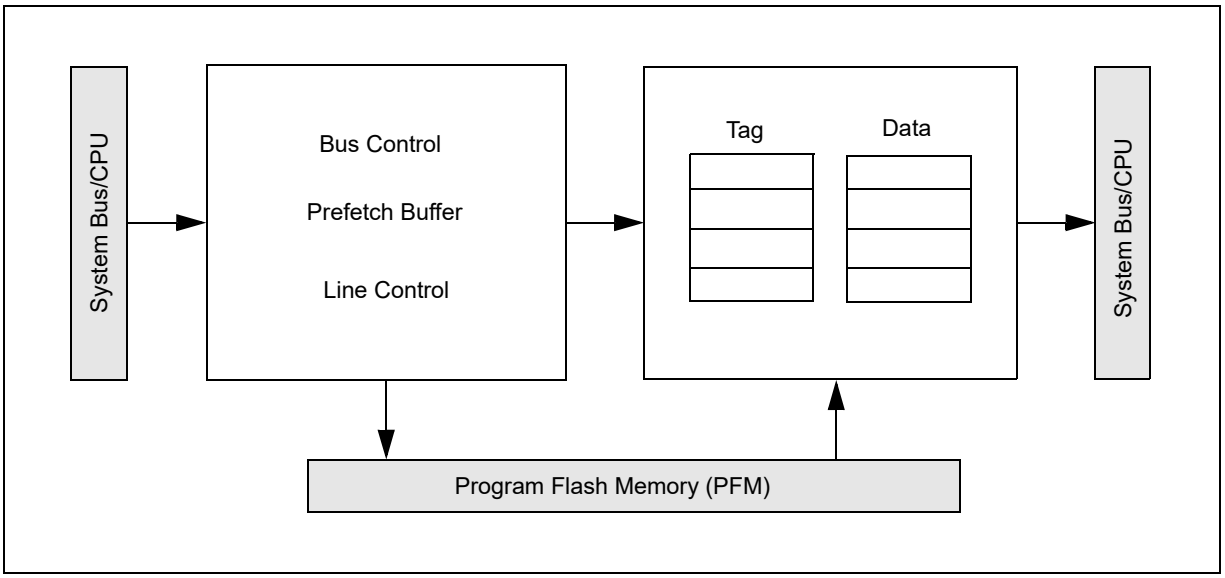
When running the Prefetch module at high-clock rates, insert the Wait states into Program Flash Memory (PFM) read transactions to meet the access time of the PFM. The user can hide the Wait states to the core by prefetching and storing the instructions in a temporary holding area that the CPU can access quickly. Although, the data path to the CPU is 32 bits wide, the data path to the PFM is 128 bits wide. This wide data path provides the same bandwidth to the CPU as a 32-bit path running at four times the frequency.

The Prefetch module holds a subset of PFM in temporary holding spaces known as lines. Each line contains a tag and data field. Normally, the lines hold a copy of what is currently in memory to make instructions or data available to the CPU without the Wait states.

The CPU or a peripheral may request the data located in the PFM. If the requested data is not currently stored in the Prefetch module line, a read is performed to the PFM at the correct address, and the data is supplied to the Prefetch module and to the CPU or peripheral. If the requested data is stored in the Prefetch module and is valid, the data is supplied to the CPU or peripheral without Wait states.

The following figure shows a block diagram of the Prefetch module. Logically, the Prefetch module fits between the System Bus module and the PFM.

Figure 9-1. Prefetch Cache Block Diagram



9.3.1 Line Organization

The Prefetch module consists of two arrays, data and tag, each of which hold four lines. A data array consists of program instructions, program data or peripheral data. Address matches are based on the physical address, not the virtual address.

Each line in the tag array contains the following information:

- Tag – Physical address of the data held in the data line
- Valid bit

Each line in the data array, contains 16 bytes of data. Depending on the line, the data can be CPU instructions, CPU data or peripheral data.

The following figures illustrate the organization of a line.

Figure 9-2. Tag Line

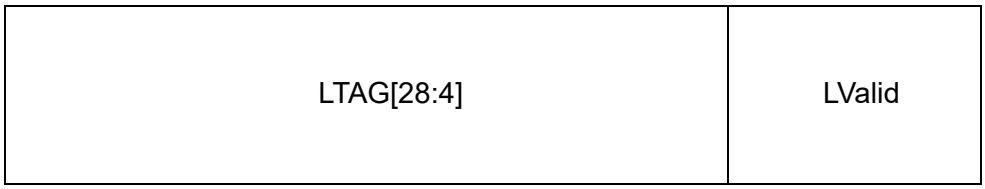
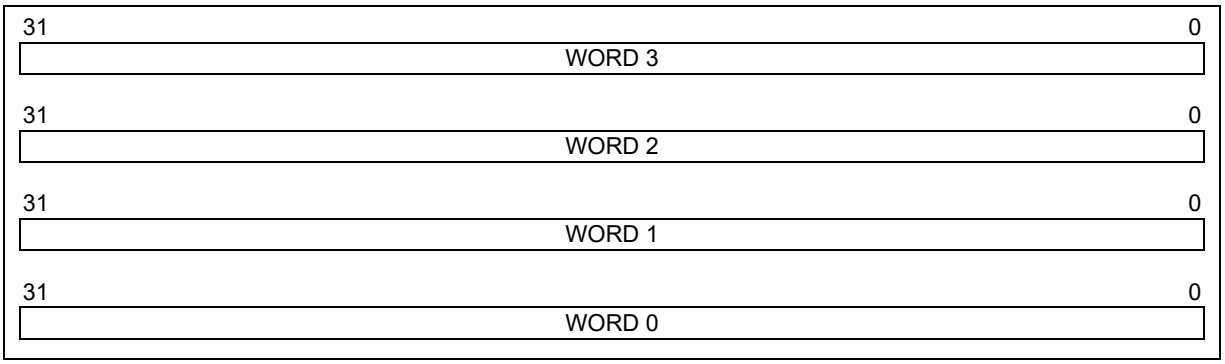


Figure 9-3. Data Line



9.4 Prefetch Behavior

The Prefetch module complements an L1 CPU (CMCC) cache rather than replacing it. A four 128-bit (16-byte) lines hold instructions, two 128-bit (16-byte) lines hold CPU data and two 128-bit (16-byte) lines hold peripheral data from the PFM. The Prefetch module uses the Wait state's value from the PFMWS[3:0] bits (CHECON[3:0]) and Address Wait state ADRWS bit (CHECON[8]) to determine how long it must wait for Flash access when it reads instructions or data from the PFM.

If the instructions or data already reside in the Prefetch module line, the Prefetch module returns the instruction or data in zero Wait states. For CPU instructions, if predictive prefetch is enabled and the code is 100% linear, the Prefetch module will provide instructions back to the CPU with the Wait states only on the first instruction of the Prefetch module line.

If the CPU accesses uncacheable addresses, it bypasses the cache. During the bypass, the prefetch module accesses the PFM for every instruction, incurring an address setup time defined by ADRWS and a Flash access time as defined by PFMWS bits. Therefore, the total Flash wait states is a sum of ADRWS and PFMWS. The Bypass mode is also forced for a cache if its associated I/D/A CHEEN bit (CHECON) is zero.

To allow caching for I and/or D caches, set the I and/or D *CHEEN bit to one. To enable a cache, set the ACHEEN bit to one.

9.5 Configurations

The CHECON register controls the general configurations available for accelerating the instruction and data accesses to the Flash memory system.

The Prefetch module implements the following general options:

- The PFMWS[3:0] bits (CHECON[3:0]) control the number of system clock cycles required to access the PFM. The total Flash Wait states is a sum of ADRWS and PFMWS.
- The PREFEN[1:0] bits (CHECON[5:4]) control the predictive and prefetched instruction, which allows the cache controller to fetch the next 16-byte aligned set of instructions.
- The PFMSECEN bit (CHECON[7]) controls the Prefetch module that generates an interrupt event on a specific count of single bit errors corrected by the Flash Error Correction Code (ECC).
- The ADRWS bit (CHECON[8]) controls the number of system clock cycles required for address setup to PFM.
- The CHEPERF bit (CHECON[12]) controls the gathering statistics of the CPU instruction cache.
- The ICHECOH bit (CHECON[16]) controls the auto invalidate for the CPU instruction cache.
- The DCHECOH bit (CHECON[17]) controls the auto invalidate for the CPU data cache.
- The ACHECOH bit (CHECON[18]) controls the auto invalidate for the peripheral data cache.
- The ICHEINV bit (CHECON[20]) controls the manual invalidate for the CPU instruction cache.
- The DCHEINV bit (CHECON[21]) controls the manual invalidate for the CPU data cache.
- The ACHEINV bit (CHECON[22]) controls the manual invalidate for the peripheral data cache.

- The ICHEEN bit (CHECON[24]) controls the CPU instruction cache enable.
- The DCHEEN bit (CHECON[25]) controls the CPU data cache enable.
- The ACHEEN bit (CHECON[26]) controls the peripheral data cache enable.

9.6 Predictive Prefetch Behavior

When the user configures the module for predictive prefetch, the module predicts the next line address, fetches the instruction and, then, stores it in the prefetch buffer. If the requested instruction is not in a Prefetch module line and the read address matches the predicted address, the content of the prefetch buffer is loaded in the Prefetch module line while simultaneously returning the critical word to the read initiator.

On enabling the predictive prefetch, the prefetch function starts predicting based on the first address read to the PFM. When the user places the first line in the Prefetch module, the module increments the address to the next 16-byte aligned address and starts a PFM access.

The predictive prefetches, like all PFM read accesses, are never aborted. If a new address request does not match the predicted address, a new PFM access occurs after the current access finishes. The PREFEN [1:0] bits (CHECON[5:4]) can start a predictive prefetch. This allows the cache controller to speculatively fetch the next 16-byte aligned set of instructions. The predictive prefetch feature is available only for CPU instruction but not for CPU data and peripheral.

If the selected system clock speed is sufficiently low enough to access the Flash at zero Wait states, the predictive prefetch is detrimental and may be disabled.

9.7 Coherency Support

When a PFM programming event occurs flash programming initiated by the Flash controller, the Prefetch module invalidates all lines and the contents of the prefetch buffer. If a transaction is in progress, the invalidation occurs after completion. When programming or erasing a Flash page, a read of that Flash page will cause the transaction to stall until the erase or program event completes.

The Prefetch module provides two methods for coherency control:

- Auto Invalidate via I/D/A CHECOH
- Manual Invalidate via I/D/A CHEINV

The user can choose to auto invalidate the each/any cache on a PFM programming event by setting *CHECOH = `1`. This is the safest option. However, the user has the option to never auto invalidate each/any cache by setting *CHECOH = `0`.

In addition to using *CHECOH, *CHEINV can be used as an alternate invalidate method to invalidate each/any cache manually. If using *CHEINV to manually invalidate each/any cache due to a PFM programming event, stop all instruction/data fetches from the desired Flash, set *CHEINV, wait for it to clear and, then, start the programming sequence. When using *CHEINV to invalidate each/any cache for reasons other than programming, it can be set at any time but only takes effect after any pending transactions complete.

9.8 Effects of Reset

9.8.1 On Reset

Upon a device Reset, the following occurs:

- All lines are invalidated
- All tag bits are cleared

9.8.2 After Reset

The module operates as per the values in the CHECON register. See the *CHECON* register from Related Links.

Related Links

9.12.1. CHECON

9.9 Error Conditions

The Prefetch module handles and reports information about two error types: ECC Double-bit Error Detected (DED) and ECC Single-bit Error Corrected (SEC). The ECC Error detection logic is enabled and disabled using the configuration bits, ECCCTL[1:0] (CFGCON0/DEVCFG0[29:28]).

The ECC logic increases the read access delay from the PFM. Depending on the frequency of the system clock, the Wait states may be different between ECC-enabled and ECC-disabled.

Note: ECC errors are captured for predictive prefetch reads of the PFM. However, those errors are not reported until, and unless, that data is used by the system.

9.9.1 ECC Double-bit Error Detected (DED)

A read from the Flash memory that results in a PFM ECC DED causes the Prefetch module to return a bus exception error to the initiator. If that initiator is the CPU, it recognizes the bus exception error, prevents the instruction from executing, or read data from loading, and generates an exception using the bus exception error vector.

When an ECC DED error occurs, the PFMDED bit (CHESTAT[27]) is set. The exception handling code can, then, check this bit to determine whether the exception was caused by a PFM ECC DED event. This bit must be cleared in software by the exception handler.

Note: CPU instructions or data prefetched from the PFM will always be loaded into the Prefetch module, even if a DED error is generated. The Prefetch module line containing the DED data will be tagged as valid until the line is replaced.

9.9.2 ECC Single Error Corrected (SEC)

A PFM ECC SEC event is not a critical error and, as such, is reported through an interrupt. The user has the option to enable or disable this interrupt through the PFMSECEN bit (CHECON[7]). The data in the Prefetch module is correct, and no further ECC events are generated for addresses that hit the data line as long as that data is in the Prefetch module.

Each read that returns from the PFM with an ECC SEC status causes the PFMSECCNT[7:0] bits (CHESTAT[7:0]) to decrement by one. If PFMSECCNT[7:0] is zero and a PFM ECC SEC event occurs, the PFMSEC bit (CHESTAT[26]) is set and an interrupt is generated. Therefore, the PFMSECCNT[7:0] bits must be set to the number of PFM ECC SEC events desired for an interrupt minus 1. For example, to generate an interrupt after five PFM ECC SEC events, PFMSECCNT[7:0] must be set to four (‘00000100’). The Prefetch module does not reload the PFMSECCNT[7:0] bits when it reaches zero. Software must write the desired count each time it services the PFMSEC interrupt.

Software can generate an ECC SEC interrupt by setting the PFMSECEN bit, then setting the PFMSEC bit. If the PFMSEC bit is already set when PFMSECEN is set, the Prefetch module will also generate an ECC SEC interrupt. The ECC SEC interrupt persists as long as the PFMSECEN and PFMSEC bits remain set. See *PCACHE Interrupt* in the *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

9.10 Operation in Power-saving Modes

9.10.1 Sleep Mode

When the device enters the Sleep mode, the Prefetch module is disabled and placed into a low-power state where no clocking occurs in the module.

9.10.2 Idle Mode

When the device enters the Idle mode, iCache, Prefetch and dCache clocks are internally gated-off, the aCache (peripheral data) clock remains functional for peripheral accesses and the CPU stops executing code. Any outstanding prefetch completes before the Prefetch module stops its clock through automatic clock gating.

9.10.3 Debug Mode

The behavior of the Prefetch module is unaltered in the Debug mode.

9.11 Register Summary (PCHE)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CHECON	7:0	PFMSECEN		PREFEN[1:0]		PFMWS[3:0]			
		15:8				CHEPERF				ADRWS
		23:16		ACHEINV	DCHEINV	ICHEINV		ACHECOH	DCHECOH	ICHECOH
		31:24						ACHEEN	DCHEEN	ICHEEN
0x04 ... 0x0F	Reserved									
0x10	CHESTAT	7:0	PFMSECCNT[7:0]							
		15:8								
		23:16								
		31:24					PFMDED	PFMSEC		
0x14 ... 0x1F	Reserved									
0x20	CHEHIT	7:0	CHEHIT[7:0]							
		15:8	CHEHIT[15:8]							
		23:16	CHEHIT[23:16]							
		31:24	CHEHIT[31:24]							
0x24 ... 0x2F	Reserved									
0x30	CHEMIS	7:0	CHEMIS[7:0]							
		15:8	CHEMIS[15:8]							
		23:16	CHEMIS[23:16]							
		31:24	CHEMIS[31:24]							

9.12 Register Description

The CHECON and CHESTAT registers in the Register Summary table have corresponding CLR, SET and INV registers at its virtual address, plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET and INV Registers* from Related Links.

The following are the description of the legends:

- R = Readable bit
- W = Writable bit
- S = Settable bit
- C = Clearable bit
- HC = Hardware Cleared
- HS = Hardware Set

Related Links

6.1.9. CLR, SET and INV Registers

9.12.1 CHECON - Prefetch Module Control Register

Name: CHECON
Offset: 0x00
Reset: 0x0700010F
Property: -

Bit	31	30	29	28	27	26	25	24
						ACHEEN	DCHEEN	ICHEEN
Access						R/W	R/W	R/W
Reset						1	1	1

Bit	23	22	21	20	19	18	17	16
		ACHEINV	DCHEINV	ICHEINV		ACHECOH	DCHECOH	ICHECOH
Access		R/S/HC	R/S/HC	R/S/HC		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit	15	14	13	12	11	10	9	8
				CHEPERF				ADRWS
Access				R/W				R/W
Reset				0				1

Bit	7	6	5	4	3	2	1	0
	PFMSECEN		PREFEN[1:0]		PFMWS[3:0]			
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	1	1	1	1

Bit 26 – ACHEEN Peripheral Data Cache Enable bit

Value	Description
1	Caching is enabled
0	Caching is disabled (and all lines invalidated)

Bit 25 – DCHEEN Data Cache Enable bit

Value	Description
1	Caching is enabled
0	Caching is disabled (and all lines invalidated)

Bit 24 – ICHEEN Instruction Data Cache Enable bit

Value	Description
1	Caching is enabled
0	Caching is disabled (and all lines invalidated)

Bit 22 – ACHEINV Manual Invalidate Control for Peripheral Data Cache

Note: The hardware auto clears this bit when cache invalidate completes. Bits may clear at different times.

Value	Description
1	Force invalidate cache/invalidate busy
0	Cache invalidation follows ACHECOH/invalid complete

Bit 21 – DCHEINV Manual Invalidate Control for Data Cache

Note: The hardware auto clears this bit when cache invalidate completes. Bits may clear at different times.

Value	Description
1	Force invalidate cache/invalidate busy
0	Cache invalidation follows DCHECOH/invalid complete

Bit 20 – ICHEINV Manual Invalidate Control for Instruction Cache

Notes:

1. The Predictive Prefetch Buffer (PFB) is included with iCache invalidate.
2. The hardware auto clears this bit when cache invalidate completes. Bits may clear at different times.

Value	Description
1	Force invalidate cache/invalidate busy
0	Cache invalidation follows ICHECOH/invalid complete

Bit 18 – ACHECOH Auto Cache Coherency Control for Peripheral Data Cache

Note: ACHECOH must be stable before initiation of programming to ensure correct invalidation of data.

Value	Description
1	Auto invalidate cache on a programming event
0	No auto invalidated cache on a programming event

Bit 17 – DCHECOH Auto Cache Coherency Control for Data Cache

Note: DCHECOH must be stable before initiation of programming to ensure correct invalidation of data.

Value	Description
1	Auto invalidate cache on a programming event
0	No auto invalidated cache on a programming event

Bit 16 – ICHECOH Auto Cache Coherency Control for Instruction Cache

Note: ICHECOH must be stable before initiation of programming to ensure correct invalidation of data.

Value	Description
1	Auto invalidate cache on a programming event
0	No auto invalidated cache on a programming event

Bit 12 – CHEPERF Cache Performance Counters Enable

Note: Performance counters are reset on 0 to 1 transition of this bit.

Value	Description
1	Performance counters is enabled
0	Performance counters is disabled

Bit 8 – ADRWS Address Wait State Enable

Total Flash Wait states are ADRWS + PFMWS.

Value	Description
1	Add 1 address Wait state - allowing for higher clock frequencies
0	Add 0 address Wait states - allowing for higher performance at lower clock frequencies

Bit 7 – PFMSECEN Flash Single-bit Error Corrected (SEC) Interrupt Enable bit

Value	Description
1	Generate an interrupt when PFMSEC is set
0	Do not generate an interrupt when PFMSEC is set

Bits 5:4 – PREFEN[1:0] Instruction Predictive Prefetch Enable

Value	Description
01	Instruction predictive prefetch enabled for cacheable regions only
00	Instruction predictive prefetch disabled

Note: Other values are unavailable.

Bits 3:0 – PFMWS[3:0] PFM Access Time Defined in Terms of SYSCLK Wait States bits

Total Flash Wait states are ADRWS + PFMWS.

Value	Description
1111	Fifteen Wait states

Value	Description
1110	Fourteen Wait states
...	
0001	One Wait state
0000	Zero Wait state

Notes:

1. This is not the Wait state seen by the CPU.
2. For the Wait states to SYSCLK relationship, see *Electrical Characteristics* from Related Links.

Related Links

43. Electrical Characteristics

9.12.2 CHESTAT - Prefetch Module Status Register

Name: CHESTAT
Offset: 0x10
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
					PFMDED	PFMSEC		
Access					HS, R/C	HS, R/W		
Reset					0	0		

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	PFMSECCNT[7:0]							
Access	HS, HC, R/W	HS, HC, R/W	HS, HC, R/W	HS, HC, R/W	HS, HC, R/W	HS, HC, R/W	HS, HC, R/W	HS, HC, R/W
Reset	0	0	0	0	0	0	0	0

Bit 27 – PFMDED Flash Double-bit Error Detected (DED) Status bit
This bit is set in hardware and can only be cleared (i.e., set to '0') in software.

Value	Description
1	A DED error has occurred
0	A DED error has not occurred

Bit 26 – PFMSEC Flash Single-bit Error Corrected (SEC) Status bit
Note: The error event is reported to the CPU via using the PCACHE interrupt event (See *Nested Vector Interrupt Controller (NVIC)* from Related Links).

Value	Description
1	A SEC error occurred when PFMSECCNT[7:0] was equal to zero
0	A SEC error has not occurred

Bits 7:0 – PFMSECCNT[7:0] Flash SEC Count bits
Decrements by 1 its count value each time an SEC error occurs. Holds at zero. When an SEC error occurs when PFMSECCNT[7:0] is zero, the PFMSEC status bit is set. If PFMSECEN is also set, a Prefetch module interrupt event is generated.

Related Links
10.2. Nested Vector Interrupt Controller (NVIC)

9.12.3 CHEHIT – Prefetch Module Hit Statistics Register

Name: CHEHIT
Offset: 0x20
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	CHEHIT[31:24]							
Access	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CHEHIT[23:16]							
Access	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CHEHIT[15:8]							
Access	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CHEHIT[7:0]							
Access	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CHEHIT[31:0] Instruction Cache Hit Count bits
When CHECON.CHEPERF = 1, CHEHIT increments once per iCache or Predictive Prefetch Buffer (PFB) hit.
Note: CHEHIT is Reset on 0 to 1 transition of CHECON.CHEPERF.

9.12.4 CHEMIS – Prefetch Module Miss Statistics Register

Name: CHEMIS
Offset: 0x30
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	CHEMIS[31:24]							
Access	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CHEMIS[23:16]							
Access	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CHEMIS[15:8]							
Access	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CHEMIS[7:0]							
Access	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC	R/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CHEMIS[31:0] Instruction Cache Miss Count bits
When CHECON.CHEPERF = 1, CHEMIS increments once per iCache or Predictive Prefetch Buffer (PFB) miss.
Note: CHEMIS is Reset on 0 to 1 transition of CHECON.CHEPERF.

10. Processor and Architecture

10.1 Cortex M4 Processor

The ARM®Cortex™-M4 processor is a high performance 32-bit processor designed for the microcontroller market. It offers the following significant benefits to developers:

- Outstanding processing performance combined with fast interrupt handling
- Enhanced system debug with extensive breakpoint and trace capabilities
- Efficient processor core, system, and memories
- Ultra low-power consumption with integrated sleep modes
- Platform security robustness, with integrated memory protection unit (MPU).

The implemented ARM Cortex-M4 is revision r0p1

For additional information, refer to http://www.arm.com

The Cortex-M4 processor is built on a high-performance processor core with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware including IEEE 754-compliant single-precision floating-point computation, a range of single-cycle and SIMD multiplication and multiply-with-accumulate capabilities, saturating arithmetic, and dedicated hardware division.

To facilitate the design of cost-sensitive devices, the Cortex-M4 processor implements tightly-coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The Cortex-M4 processor implements a version of the Thumb instruction set based on Thumb®-2 technology, ensuring high code density and reduced program memory requirements. The Cortex-M4 instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.

The Cortex-M4 processor closely integrates a configurable *Nested Vector Interrupt Controller* (NVIC), to deliver industry-leading interrupt performance. The NVIC includes a *Non-Maskable interrupt* (NMI), and provides up to 8 interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of *Interrupt Service Routines* (ISRs), dramatically reducing interrupt latency. This is achieved through the hardware stacking of registers, and the ability to suspend load-multiple and store-multiple operations. Interrupt handlers do not require wrapping in assembler code, removing any code overhead from the ISRs. A tail-chain optimization also significantly reduces the overhead when switching from one ISR to another.

To optimize low-power designs, the NVIC integrates with the sleep modes, that include a deep sleep function that enables the entire device to be rapidly powered down while still retaining program state.

10.1.1 System Level Interface

The Cortex-M4 processor provides multiple interfaces using AMBA technology to provide high-speed, low-latency memory accesses. It supports unaligned data accesses and implements atomic bit manipulation that enables faster peripheral controls, system spinlocks, and thread-safe Boolean data handling.

The Cortex-M4 processor has a Memory Protection Unit (MPU) that provides fine grain memory control, enabling applications to utilize multiple privilege levels, separating and protecting code, data, and stack on a task-by-task basis. Such requirements are becoming critical in many embedded applications such as automotive.

10.1.2 Integrated Configurable Debug

The Cortex-M4 processor implements a complete hardware debug solution. This provides high system visibility of the processor and memory through a 2-pin *Serial Wire Debug* (SWD) port that is ideal for microcontrollers and other small package devices.

For system trace the processor integrates an *Instrumentation Trace Macrocell* (ITM) alongside data watchpoints and a profiling unit. The *Embedded Trace Macrocell* (ETM) delivers unrivaled instruction trace capture in an area far smaller than traditional trace units, enabling many low cost MCUs to implement full instruction trace for the first time.

To enable simple and cost-effective profiling of the system events these generate, a stream of software-generated messages, data trace, and profiling information is exported over three different ways:

- Output off chip using the TPIU, through a single pin, called *Serial Wire Viewer* (SWV). Limited to ITM system trace
- Output off chip using the TPIU, through a 4-bit pin interface. Bandwidth is limited
- Internally stored in RAM, using the CoreSight ETB. Bandwidth is then optimal but capacity is limited

The *Flash Patch and Breakpoint Unit* (FPB) provides up to 8 hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions of up to 8 words in the program code in the Code memory region. This enables applications stored on a non-erasable, ROM-based microcontroller to be patched if a small programmable memory, for example Flash, is available in the device. During initialization, the application in ROM detects, from the programmable memory, whether a patch is required. If a patch is required, the application programs the FPB to remap a number of addresses. When those addresses are accessed, the accesses are redirected to a remap table specified in the FPB configuration, which means the program in the non-modifiable ROM can be patched.

10.1.3 Cortex-M4 Processor Features and Configuration

- Thumb® instruction set combines high code density with 32-bit performance
- IEEE 754-compliant single-precision Floating Point Unit (FPU)
- Integrated sleep modes for low power consumption
- Fast code execution permits slower processor clock or increases Sleep mode time
- Hardware division and fast digital-signal-processing orientated multiply accumulate
- Saturating arithmetic for signal processing
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory Protection Unit (MPU) for safety-critical applications
- Extensive debug and trace capabilities: Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging, tracing and code profiling.

Features	Cortex-M4 Options	PIC32CX-BZ2 Configuration
Interrupts	1 to 240	40
Number of priority bits	3 to 8	3 = eight levels of priority
Data endianness	Little-endian or big-endian	Little-endian
SysTick Timer calibration value	—	0x80000000
MPU	Present or Not present	Present
Debug support level	0 = No debug. No DAP, breakpoints, watchpoints, Flash patch or halting debug 1 = Minimum debug. Two breakpoints, one watchpoint, no Flash patch 2 = Full debug minus DWT data matching 3 = Full debug plus DWT data matching	3 = Full debug plus DWT data matching
Trace support level	0 = No trace. No ETM, ITM or DWT triggers and counters 1 = Standard trace. ITM and DWT triggers and counters, but no ETM 2 = Full trace. Standard trace plus ETM 3 = Full trace plus HTM port	2 = Full trace. Standard trace plus ETM
JTAG	Present or Not present	Not present

.....continued		
Features	**Cortex-M4 Options**	**PIC32CX-BZ2 Configuration**
Bit Banding	Present or Not present	Not present
FPU	Present or Not present	Present

10.1.4 Cortex-M4 Core Peripherals

Nested Vectored Interrupt Controller The Nested Vector Interrupt Controller (NVIC) is an embedded interrupt controller that supports low latency interrupt processing.

System Control Block The System Control Block (SCB) is the programmers model interface to the processor. It provides system implementation information and system control, including configuration, control and reporting of system exceptions. Refer to the *Cortex-M4 Technical Reference Manual* for more details (http://www.arm.com).

System Timer The system timer, SysTick, is a 24-bit countdown timer. Use this as a Real-Time Operating System (RTOS) tick timer or as a simple counter. The SysTick timer runs on the processor clock and it does not decrement when the processor is halted for debugging. Refer to the *Cortex-M4 Technical Reference Manual* for more details (http://www.arm.com).

Memory Protection Unit The Memory Protection Unit (MPU) improves system reliability by defining the memory attributes for different memory regions. It provides up to eight different regions and an optional predefined background region. Refer to the *Cortex-M4 Technical Reference Manual* for more details (http://www.arm.com).

Floating-Point Unit The Floating Point Unit (FPU) provides IEEE 754-compliant operations on single-precision, 32-bit, floating-point values. Refer to the *Cortex-M4 Technical Reference Manual* for more details (http://www.arm.com).

10.1.5 Cortex-M4 Address Map

Address	Core Peripheral
0xE000E008-0xE000E00F	System control block
0xE000E010-0xE000E01F	System timer
0xE000E100-0xE000E4EF	Nested Vectored Interrupt Controller
0xE000ED00-0xE000ED3F	System control block
0xE000ED90-0xE000ED93	MPU Type Register
0xE000ED94-0xE000EDB8	Memory Protection Unit
0xE000EF00-0xE000EF03	Nested Vectored Interrupt Controller
0xE000EF30-0xE000EF44	Floating Point Unit

10.2 Nested Vector Interrupt Controller (NVIC)

10.2.1 Overview

The Nested Vectored Interrupt Controller (NVIC) in the PIC32CX-BZ2 family devices supports 40 interrupts with eight different priority levels. For more details, refer to the *Cortex-M4 Technical Reference Manual* (www.arm.com).

10.2.2 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, located in the peripheral's Interrupt Flag Status and Clear (INTFLAG) register.

An interrupt flag is set when the interrupt condition occurs. Each interrupt in the peripheral can be individually enabled by writing a '1' to the corresponding bit in the peripheral's Interrupt Enable Set (INTENSET) register, and disabled by writing '1' to the corresponding bit in the peripheral's Interrupt Enable Clear (INTENCLR) register.

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

Depending on their criticality, the interrupt requests for one peripheral are either ORed together on system level, generating one interrupt or directly connected to an NVIC interrupt lines. This is described in the table below.

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.

Module	Source	Line
EIC NMI – External Interrupt Control	NMI	NMI
RTCC – Real-Time Counter and Calendar	CMP A 0	0
	CMP A 1	
	CMP A 2	
	CMP A 3	
	OVF A	
	PER A 0	
	PER A 1	
	PER A 2	
	PER A 3	
	PER A 4	
	PER A 5	
	PER A 6	
	PER A 7	
	TAMPER A	
EIC – External Interrupt Controller	EXTINT 0	1
	EXTINT 1	
	EXTINT 2	
	EXTINT 3	
FREQM – Frequency Meter	DONE	2
Flash Subsystem	Flash Controller	3
	PFW	
	PCACHE	
PORT-A	PortA Input Change Interrupt	4
PORT-B	PortB Input Change Interrupt	5

.....continued

Module	Source	Line
DMAC – Direct Memory Access Controller	SUSP 0..3	6
	TCMPL 0..3	
	TERR 0..3	
	SUSP 4..15	7
	TCMPL 4..15	
	TERR 4..15	
EVSYS – Event System Interface	EVD 0..3	8
	OVR 0..3	
	EVD 4..11	9
	OVR 4..11	
PAC – Peripheral Access Controller	ERR	10
RAM ECC	SINGLEE-0	11
	DualE-1	
SERCOM0 – Serial Communication Interface 0(1) Order: USART, I2CM, I2CS, SPI	0	12
	1	
	2	
	3	
	4	
	5	
	7	
SERCOM1 – Serial Communication Interface 1(1) Order: USART, I2CM, I2CS, SPI	0	13
	1	
	2	
	3	
	4	
	5	
	7	
SERCOM2 – Serial Communication Interface 2(1) Order: USART, I2CM, I2CS, SPI	0	14
	1	
	2	
	3	
	4	
	5	
	7	

.....continued

Module	Source	Line
SERCOM3 – Serial Communication Interface 3(1) Order: USART, I2CM, I2CS, SPI	0	15
	1	
	2	
	3	
	4	
	5	
	7	
TCC0 – Timer Counter Control 0	CNT A	16
	DFS A	
	ERR A	
	FAULTA A	
	FAULTB A	
	FAULT0 A	
	FAULT1 A	
	OVF	
	TRG	
	UFS A	
	MC 0	
	MC 1	
	MC 2	
	MC 3	
	MC 4	
	MC 5	

.....continued

Module	Source	Line
TCC1 – Timer Counter Control 1	CNT A	17
	DFS A	
	ERR A	
	FAULTA A	
	FAULTB A	
	FAULT0 A	
	FAULT1 A	
	OVF	
	TRG	
	UFS A	
	MC 0	
	MC 1	
	MC 2	
	MC 3	
	MC 4	
	MC 5	
TCC2 – Timer Counter Control 2	CNT A	18
	DFS A	
	ERR A	
	FAULTA A	
	FAULTB A	
	FAULT0 A	
	FAULT1 A	
	OVF	
	TRG	
	UFS A	
	MC 0	
	MC 1	
TC0 – Basic Timer Counter 0	ERR A	19
	MC 0	
	MC 1	
	OVF	

.....continued

Module	Source	Line
TC1 – Basic Timer Counter 1	ERR A	20
	MC 0	
	MC 1	
	OVF	
TC2 – Basic Timer Counter 2	ERR A	21
	MC 0	
	MC 1	
	OVF	
TC3 – Basic Timer Counter 3	ERR A	22
	MC 0	
	MC 1	
	OVF	
ADCTRL	ADC_GIRQ	23
	ADC_DIRQ0, ADC_DIRQ1	
	ADC_AIRQ0, ADC_AIRQ1	
	ADC_FLT	34
	ADC_EOS	35
	ADC_BGVR_RDY	36
AC – Analog Comparators	COMP 0	24
	COMP 1	
	WIN 0	
AES – Advanced Encryption Standard	ENCCMP	25
	GFMCMP	
TRNG – True Random Generator	IS0	26
ICM – Integrity Check Monitor	ICM	27
QSPI – Quad SPI interface	QSPI	29
Wireless Radio	ZB_INT0	30
	BT_INT0	31
	BT_INT1	32
	ARBITER	33
	CLKI_WAKEUP_NMI	37
	BT_LC	38
	BT_RC	39

Note:

1. The integer number specified in the source refers to the respective bit position in the INTFLAG register of the respective peripheral.

10.3 High-Speed Bus System

10.3.1 Features

High-Speed Bus Matrix 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 one-to-one clock frequency with the bus Hosts

10.3.2 Configuration

Figure 10-1. Host-Client Relations High-Speed Bus Matrix

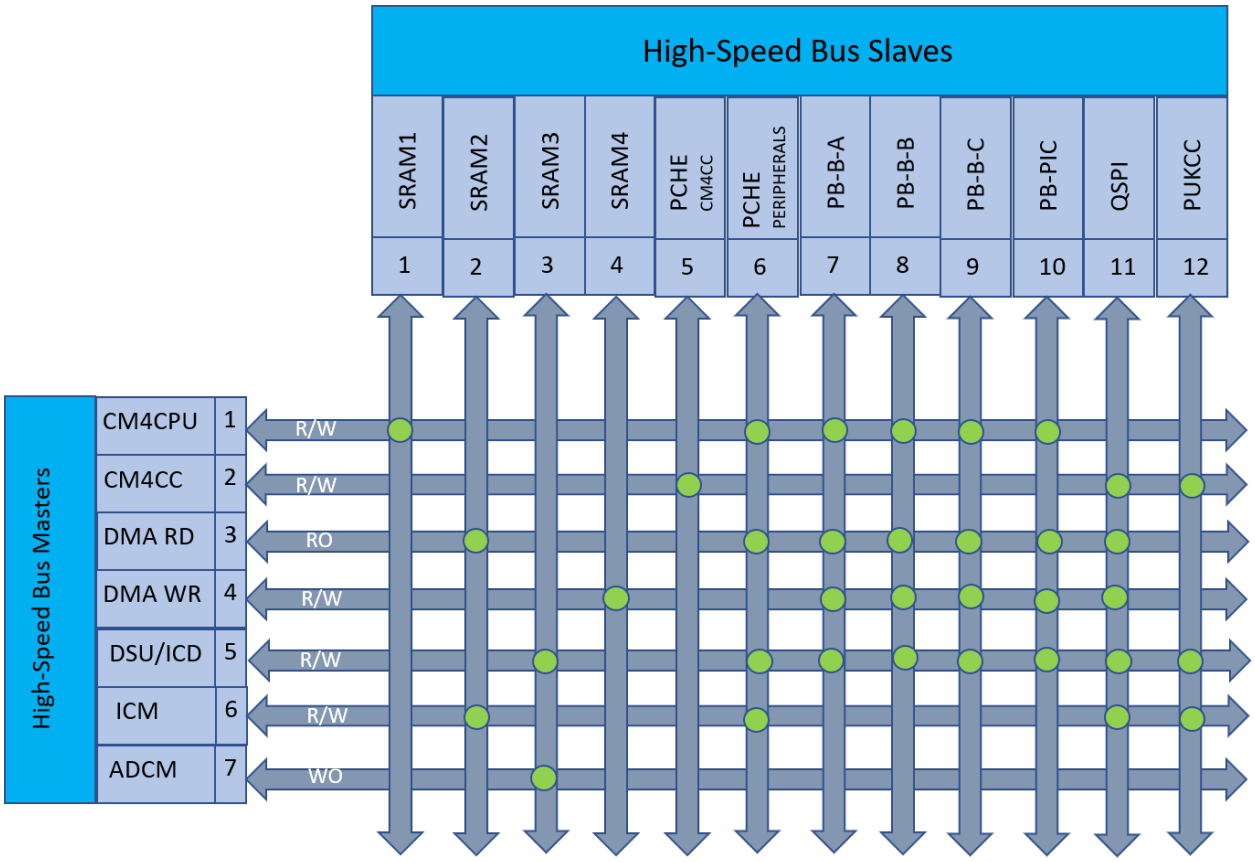


Table 10-1. High Speed Bus Matrix Host

High-Speed Bus Matrix Host	Host ID
CM4CPU - Cortex M4 Processor	1
CM4CC - Cortex-M Cache Controller	2
DMA RD - DMA-Read	3
DMA-WR - DMA-Write	4

.....continued	
High-Speed Bus Matrix Host	**Host ID**
DSU/ICD (private test mode only) - Device Service Unit/In-Chip Debugger	5
ICM - Integrity Check Monitor	6
ADCM - ADC Controller Module	7

Table 10-2. High-Speed Bus Matrix Client

High-Speed Bus Matrix Client	Client ID
SRAM1 - SRAM Port 1	1
SRAM2 - SRAM Port 2	2
SRAM3 - SRAM Port 3	3
SRAM4 - SRAM Port 4	4
PCHE - Pre-fetch Cache of CM4CC	5
PCHE - Pre-fetch Cache of Peripherals	6
PB-B-A - Peripheral Bridge A	7
PB-B-B - Peripheral Bridge B	8
PB-B-C - Peripheral Bridge C	9
PB-PIC - Peripheral Bus PIC	10
QSPI - Quad SPI Interface	11
PUKCC - Public Key Cryptography Controller	12

11. Cortex M Cache Controller (CMCC)

11.1 Overview

The Cortex M Cache Controller provides an L1 cache to the Cortex M CPU. The CMCC sits transparently between the CPU and the cache leading to improved performance.

The CMCC interfaces with the CPU through the AHB and is connected to the APB bus interface for its configuration.

11.2 Features

- Physically addressed and physically tagged
- L1 data and instruction cache set to 4 KB
- L1 cache line size set to 16 Bytes
- L1 cache integrates 32-bit bus host interface
- Unified 4-Way set associative cache architecture
- Lock-Down feature, which allows cached to be locked per way
- Write through cache operations, read allocate
- Configurable as data and instruction Tightly Coupled Memory (TCM)
- Round Robin victim selection policy
- Event Monitoring, with one programmable 32-bit counter
- Cache Interface includes cache maintenance operations registers

11.3 Block Diagram

Figure 11-1. CMCC Block Diagram

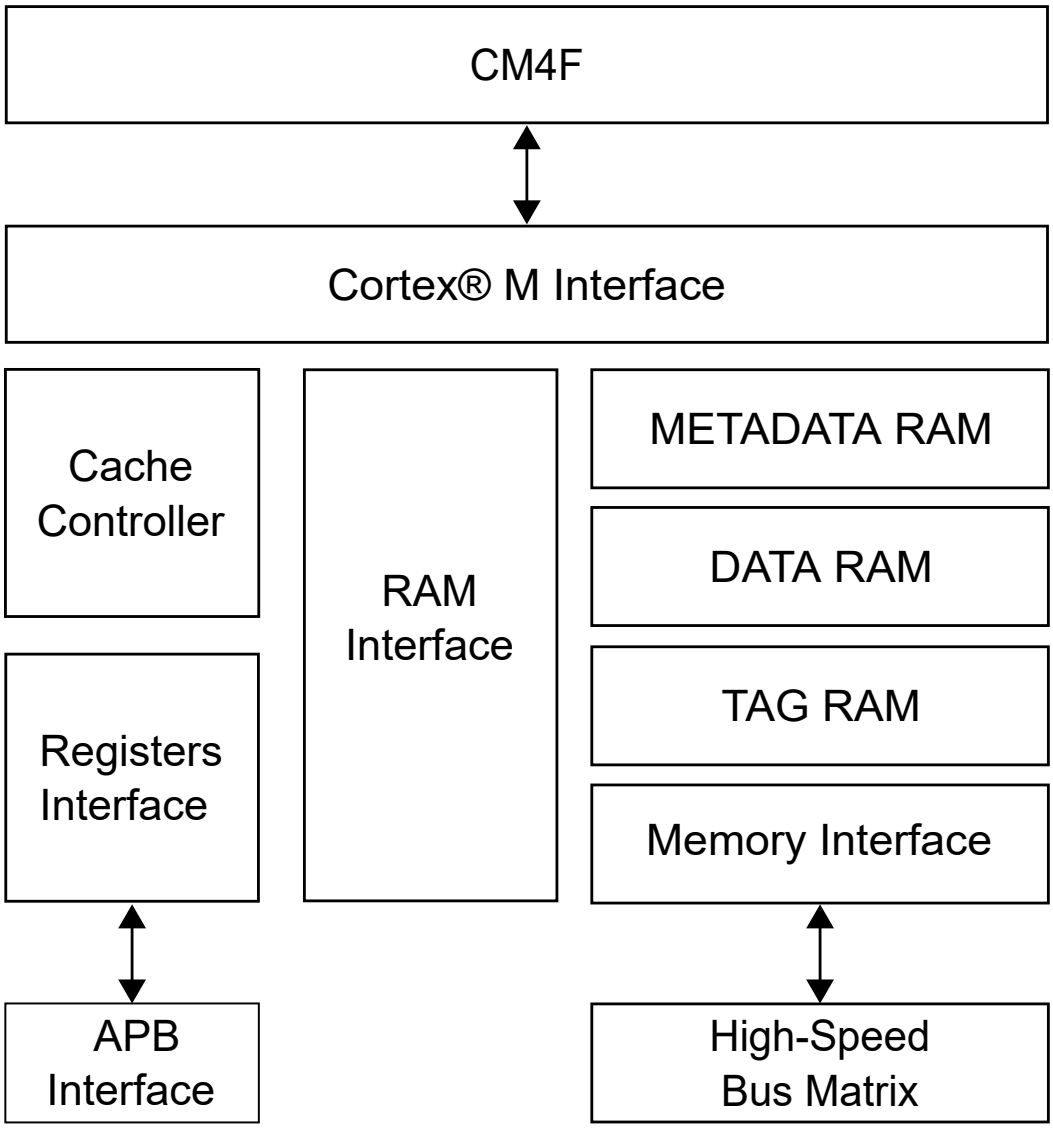
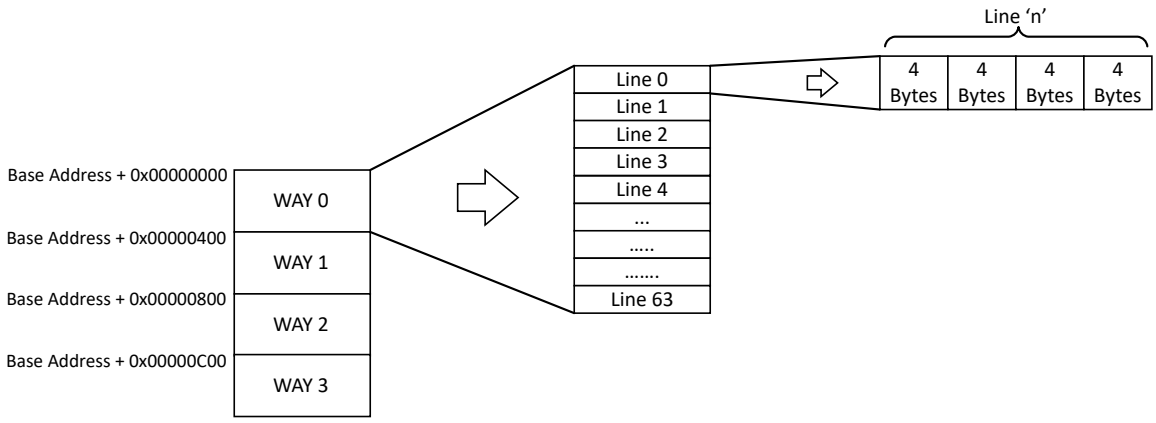


Figure 11-2. CMCC Organization



11.4 Signal Description

Not applicable.

11.5 Product Dependencies

Not applicable.

11.5.1 I/O Lines

Not applicable.

11.5.2 Power Management

The CMCC will continue to function as long as the CPU is not sleeping and the CMCC is enabled.

11.5.3 Clocks

Not applicable.

11.5.4 DMA

Not applicable.

11.5.5 Interrupts

Not applicable.

11.5.6 Events

Not applicable.

11.5.7 Debug Operation

When the CPU is halted in debug mode, the CMCC is halted. Any read access by the debugger in cached zones are not cached.

11.5.8 Register Access Protection

Not applicable.

11.5.9 Analog Connections

Not applicable.

11.6 Functional Description

11.6.1 Principle of Operation

11.6.2 Initialization and Normal Operation

On reset, the cache controller data entries are all invalidated, and the cache is disabled. The cache is transparent to processor operations. The cache controller is activated through the use of its configuration registers. The configuration interface is memory mapped in the APB bus.

Use the following sequence to enable the cache controller:

- Verify that the CMCC is disabled, reading the value of the SR.CSTS.
- Enable the CMCC by writing '1' in CTRL.CEN. The MODULE is disabled by writing a '0' in CTRL.CEN.

11.6.3 Change Cache Size

It is possible to change the cache size by writing to the Cache Size Configured By Software bits in the Cache Configuration register (CFG.CSIZESW).

Use the following sequence to change the cache size:

- Disable the CMCC controller by writing a zero to the Cache Controller Enable bit in the Cache Control register (CTRL.CEN=0).
- Check the Cache Controller Status bit in the Cache Status register to verify that the CMCC is successfully disabled (SR.CSTS=0).
- Change CFG.CSIZESW to its new value.
- Enable the CMCC by writing CTRL.CEN=1.

11.6.4 Data Cache Disable

The Instructions alone can be cached by disabling the Data cache, as described in the following steps:

1. Disable the cache controller by writing a ‘0’ to CTRL.CEN.
2. Check SR.CSTS to verify whether the CMCC is successfully disabled.
3. Write CFG.DCDIS = 1.
4. Enable the CMCC by writing CTRL.CEN = 1.

11.6.5 Instruction Cache Disable

The Data alone can be cached by disabling the Instruction cache, as described in the following steps:

1. Disable the cache controller by writing CTRL.CEN = 0.
2. Check SR.CSTS to verify that the CMCC is successfully disabled.
3. Write CFG.ICDIS = 1.
4. Enable the CMCC by writing CTRL.CEN = 1.

11.6.6 Cache Load and Lock

It is possible to lock a specific way for code optimization by writing the Lock Way register (LCKWAY.LCKWAY). The locked way will not be updated by the CMCC as part of cache operations.

The load and lock mechanism can be implemented to use cache memory in a deterministic way. Follow these steps to load and lock a way:

1. Disable cache controller by clearing the CTRL.CEN bit.
2. Invalidate the desired WAY line by line. This will reset the round robin algorithm of the invalidated line, that will become eligible for the next load operation.
3. Disable the Instruction cache, but keep the Data cache enabled.

4. Enable the cache by setting the CTRL.CEN bit.
5. Place the respective piece of code and/or data to the corresponding WAY due to simple LOAD operations. Loading the piece of code and/or data will force the cache to refill the previous invalidated line in the right way. No need to load all the bytes of the line, only the first byte. The cache will automatically refill the complete line.
6. Lock the specific WAY by setting LCKWAY.LCKWAY[3:0].
7. Re-enable the instruction cache. The locked WAY is now loaded and ready to operate. The remaining WAYS can be used as I-cache or D-cache as required.

11.6.7 Tightly Coupled Memory

Users can use a part of the cache as Tightly Coupled Memory (TCM). The cache size is determined by the Cache Size Configuration by Software bits in the Cache Configuration register (CFG.CSIZESW). The relation between cache and TCM is as given below:

TCM size = maximum Cache size – configured Cache size.

The TCM start address can be obtained from the product memory mapping. The cache memory starts first from the address followed by the TCM memory. Size of the Way is fixed and the number of ways varies according to the available size for the cache memory. See 8. Product Memory Mapping Overview.

Table 11-1. TCM Sizes

Max. Cache	Configured Cache	TCM Size
4 KB	4 KB	0 KB
4 KB	1 KB	3 KB
4 KB	2 KB	2 KB
4 KB	0 KB	4 KB

The TCM is also accessible in its maximum size when the CMCC is disabled. The TCM does not need to be locked in order to operate.

Note: Writing into the cache DATA RAM region through the CPU can overwrite the valid cache lines. This can result in data corruption when the cache controller is accessing the data for cache transactions. Access the DATA RAM region only after configuring it as TCM.

Related Links

8. Product Memory Mapping Overview

11.6.8 Cache Maintenance

11.6.8.1 Cache Invalidate by Line Operation

When an invalidate by line command is issued, the CMCC resets the valid bit information of the decoded cache line. As the line is no longer valid, the replacement counter points to that line.

- Disable the cache controller by writing a zero to the Cache Controller Enable bit in the Cache Control register (CTRL.CEN).
- Check SR.CSTS to verify that the CMCC is successfully disabled.
- Perform an invalidate by line by writing the set {index,way} in the Cache Maintenance 1 register (MAINT1.INDEX, MAINT1.WAY).
- Enable the CMCC by writing a '1' to CTRL.CEN.

11.6.8.2 Cache Invalidate All Operation

Use the following sequence to invalidate all cache entries.

- Disable the cache controller by writing a zero to the Cache Enable bit in the Cache Control register (CTRL.CEN).
- Check SR.CSTS to verify that the CMCC is successfully disabled.
- Perform a full invalidate operation by writing a '1' to the Cache Controller Invalidate All bit in the Cache Maintenance 0 register (MAINT0.INVALL).

- Enable the CMCC by writing a '1' to CTRL.CEN.

11.6.9 Cache Performance Monitoring

The Cortex M cache controller includes a programmable monitor/32-bit counter. The monitor can be configured to count the number of clock cycles, the number of data hit or the number of instruction hit.

It is important to know that the Cortex-M4 processor prefetches instructions ahead of execution. It performs only 32-bit read access on the Instruction Bus, which means:

- One arm instruction is fetched per bus access
- Two thumb instructions are fetched per bus access

As a consequence, two thumb instructions (e.g., `NOP`) need one bus access, which results in the HIT counter incrementing by 1.

Use the following sequence to activate the counter:

- Configure the monitor counter by writing the MCFG.MODE.
  - CYCLE_COUNT is used to increment the counter along with the program counter, to count the number of cycles.
  - IHIT_COUNT is the instruction Hit counter, which increments the counter when there is a hit for the instruction in the cache.
  - DHIT_COUNT is the data Hit counter which increments the counter when there is a hit for the data in the cache.
- Enable the counter by writing a '1' to the Cache Controller Monitor Enable bit in the Cache Monitor Enable register (MEN.MENABLE).
- If required, reset the counter by writing a '1' to the Cache Controller Software Reset bit in the Cache Monitor Control register (MCTRL.SWRST).
- Check the value of the monitor counter by reading the MSR.EVENT_CNT bit field.

11.7 DEBUG Mode

In Debug mode, TAG and METADATA RAM blocks content is read/written through the AHB bus interface if the CMCC is disabled. When the CMCC is enabled, the TAG and METADATA RAM blocks are non readable.

Debug access has the same R/W properties as the CPU access for the DATA RAM block.

The TAG, METADATA and DATA RAM blocks' R/W properties are summarized in RAM Properties. See *RAM Properties* from the Related Links.

Use the following sequence to perform read access with the Debugger to the three RAM blocks:

- Disable the cache controller by writing a zero to the Cache Controller Enable bit in the Cache Control register (CTRL.CEN).
- Check the Cache Controller Status bit in the Cache Status register (SR.CSTS) to verify that the CMCC is successfully disabled.
- Perform a read or write access through Debugger:
  - @ CMCC_AHB_ADDR for DATA RAM,
  - @ CMCC_AHB_ADDR_TAG for TAG RAM,
  - @ CMCC_AHB_ADDR_MTDATA for METADATA RAM.
- If a write access has been performed in the TAG, METADATA, or DATA RAM in the cache section, an invalid operation must be performed before re-enabling the CMCC.

Related Links

11.8. RAM Properties

11.8 RAM Properties

The following table shows the different access properties of the three RAM blocks, according the different modes described in the previous chapters.

Table 11-2. Access to RAM

Access Condition	DATA RAM	TAG RAM	METADATARAM
CPU access when CMCC DISABLED	R/W	no R/W - hardfault	no R/W - hardfault
CPU access when CMCC ENABLED	CACHE section configured: R/W(1) TCM section configured: R/W	no R/W - hardfault	no R/W - hardfault
Debugger access when CMCC DISABLED	R/W	R/W	R/W
Debugger access when CMCC ENABLED	CACHE section configured: R/W(1) TCM section configured: R/W	no R/W	no R/W

Note:

1. A write operation in this zone can corrupt the coherency of the cache. An invalidate operation may be needed.

11.9 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	TYPE	7:0	LCKDOWN	WAYNUM[1:0]		RRP	LRUP	RANDP	GCLK	AP
		15:8			CLSIZE[2:0]			CSIZE[2:0]		
		23:16								
		31:24								
0x04	CFG	7:0		CSIZESW[2:0]				DCDIS	ICDIS	
		15:8								
		23:16								
		31:24								
0x08	CTRL	7:0								CEN
		15:8								
		23:16								
		31:24								
0x0C	SR	7:0								CSTS
		15:8								
		23:16								
		31:24								
0x10	LCKWAY	7:0					LCKWAY[3:0]			
		15:8								
		23:16								
		31:24								
0x14 ... 0x1F	Reserved									
0x20	MAINT0	7:0								INVALL
		15:8								
		23:16								
		31:24								
0x24	MAINT1	7:0	INDEX[3:0]							
		15:8					INDEX[7:4]			
		23:16								
		31:24	WAY[3:0]							
0x28	MCFG	7:0							MODE[1:0]	
		15:8								
		23:16								
		31:24								
0x2C	MEN	7:0								MENABLE
		15:8								
		23:16								
		31:24								
0x30	MCTRL	7:0								SWRST
		15:8								
		23:16								
		31:24								
0x34	MSR	7:0	EVENT_CNT[7:0]							
		15:8	EVENT_CNT[15:8]							
		23:16	EVENT_CNT[23:16]							
		31:24	EVENT_CNT[31:24]							

11.10 Register Description

11.10.1 Cache Type

Name: TYPE
Offset: 0x00
Reset: 0x000012D2
Property: R

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			CLSIZE[2:0]			CSIZE[2:0]		
Access			R	R	R	R	R	R
Reset			0	1	0	0	1	0

Bit	7	6	5	4	3	2	1	0
	LCKDOWN	WAYNUM[1:0]		RRP	LRUP	RANDP	GCLK	AP
Access	R	R	R	R	R	R	R	R
Reset	1	1	0	1	0	0	1	0

Bits 13:11 – CLSIZE[2:0] Cache Line Size
This field configures the Cache Line Size.

Value	Name	Description
0x2	CLSIZE_16B	Cache Line Size is 16 bytes
0x3-0x7	—	Reserved

Bits 10:8 – CSIZE[2:0] Cache Size
This bit field configures the cache size.

Value	Name	Description
0x0	CSIZE_1KB	Cache Size is 1 KB
0x1	CSIZE_2KB	Cache Size is 2 KB
0x2	CSIZE_4KB	Cache Size is 4 KB
0x3-0x7	—	Reserved

Bit 7 – LCKDOWN Lock Down Supported
Writing a '0' to this bit disables the Lock Down feature.
Writing a '1' to this bit enables the Lock Down feature.

Bits 6:5 – WAYNUM[1:0] Number of Way
This bit field configures the mapping of the cache.

Value	Name	Description
0x0	DMAPPED	Direct Mapped Cache
0x1	ARCH2WAY	2-WAY set associative
0x2	ARCH4WAY	4-WAY set associative
0x3	ARCH8WAY	8-WAY set associative

Bit 4 – RRP Round Robin Policy Supported
Writing a '0' to this bit disables Round Robin Policy.

Writing a '1' to this bit enables Round Robin Policy.

Bit 3 – LRUP Least Recently Used Policy Supported
Writing a '0' to this bit disables the Least Recently Used Policy Supported.
Writing a '1' to this bit enables the Least Recently Used Policy Supported.

Bit 2 – RANDP Random Selection Policy Supported
Writing a '0' to this bit disables the Random Selection Policy Supported.
Writing a '1' to this bit enables the Random Selection Policy Supported.

Bit 1 – GCLK Dynamic Clock Gating
Writing a '0' to this bit disables the Dynamic Clock Gating feature.
Writing a '1' to this bit enables the Dynamic Clock Gating feature.

Bit 0 – AP Access Port Access Allowed
Writing a '0' to this bit disables the Access Port Access Allowed.
Writing a '1' to this bit enables the Access Port Access Allowed.

11.10.2 Cache Configuration

Name: CFG
Offset: 0x04
Reset: 0x00000020
Property: R/W

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		CSIZESW[2:0]				DCDIS	ICDIS	
Access		R/W	R/W	R/W		R/W	R/W	
Reset		0	1	0		0	0	

Bits 6:4 – CSIZESW[2:0] Cache Size Configured by Software
This field configures the cache size.

Value	Name	Description
0x0	CONF_CSIZE_1KB	The Cache Size is configured to 1KB
0x1	CONF_CSIZE_2KB	The Cache Size is configured to 2KB
0x2	CONF_CSIZE_4KB	The Cache Size is configured to 4KB
0x3	CONF_CSIZE_8KB	The Cache Size is configured to 8KB
0x4	CONF_CSIZE_16KB	The Cache Size is configured to 16KB
0x5	CONF_CSIZE_32KB	The Cache Size is configured to 32KB
0x6	CONF_CSIZE_64KB	The Cache Size is configured to 64KB
0x7		Reserved

Bit 2 – DCDIS Data Cache Disable
Writing a '0' to this bit enables data caching.
Writing a '1' to this bit disables data caching.

Bit 1 – ICDIS Instruction Cache Disable
Writing a '0' to this bit enables instruction caching.
Writing a '1' to this bit disables instruction caching.

11.10.3 Cache Control

Name: CTRL
Offset: 0x08
Reset: 0x00000000
Property: Write-only

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								CEN
Access								W
Reset								0

Bit 0 – CEN Cache Controller Enable
Writing a '0' to this bit disables the CMCC.
Writing a '1' to this bit enables the CMCC.

11.10.4 Cache Status

Name: SR
Offset: 0x0C
Reset: 0x00000000
Property: Read-only

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								CSTS
Access								R
Reset								0

Bit 0 – CSTS Cache Controller Status
Writing a '0' to this bit disables the CMCC.
Writing a '1' to this bit enables the CMCC.

11.10.5 Cache Lock per Way

Name: LCKWAY
Offset: 0x10
Reset: 0x00000000
Property: Read/Write

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

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

Bits 3:0 – LCKWAY[3:0] Lockdown Way Register
This field selects which way is locked.

11.10.6 Cache Maintenance 0

Name: MAINT0
Offset: 0x20
Reset: 0x00000000
Property: Write-only

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

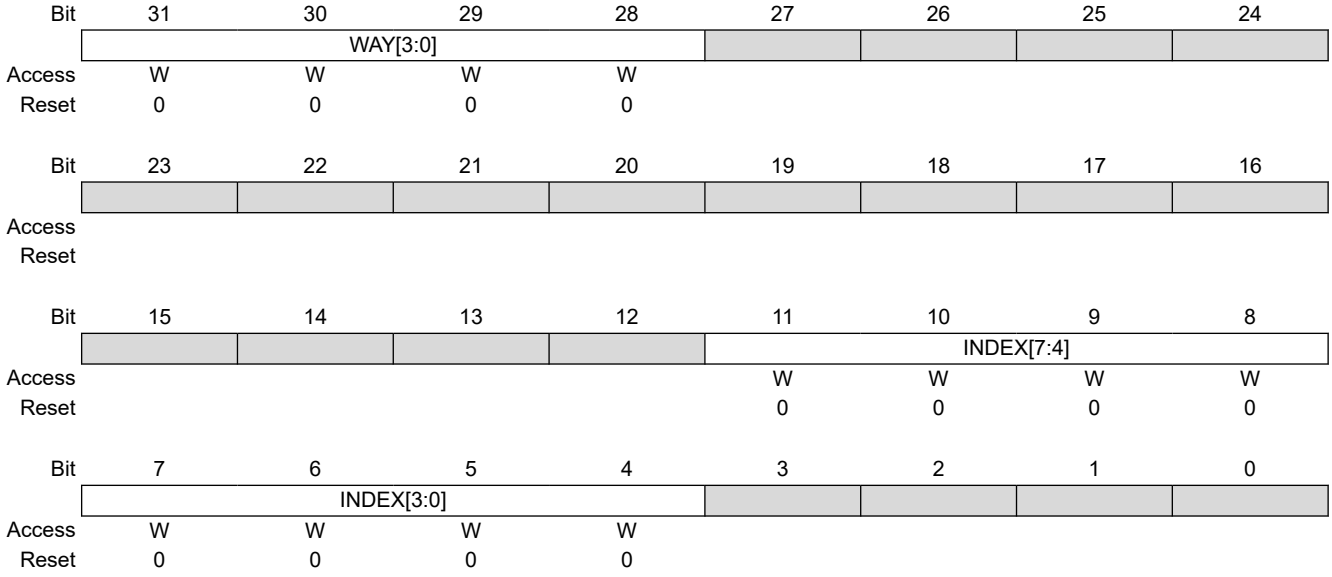
Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								INVALL
Access								W
Reset								0

Bit 0 – INVALL Cache Controller Invalidate All
Writing a '0' to this bit has no effect.
Writing a '1' to this bit invalidates all cache entries.

11.10.7 Cache Maintenance 1

Name: MAINT1
Offset: 0x24
Reset: 0x00000000
Property: Write-only



Bits 31:28 – WAY[3:0] Invalidate Way

Value	Name	Description
0x0	WAY0	Way 0 is selection for index invalidation
0x1	WAY1	Way 1 is selection for index invalidation
0x2	WAY2	Way 2 is selection for index invalidation
0x3	WAY3	Way 3 is selection for index invalidation
0x4-0xF		Reserved

Bits 11:4 – INDEX[7:0] Invalidate Index
This field selects the index value for invalidation

11.10.8 Cache Monitor Configuration

Name: MCFG
Offset: 0x28
Reset: 0x00000000
Property: Read/Write

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

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

Bits 1:0 – MODE[1:0] Cache Controller Monitor Counter Mode
This field selects the type of data monitored.

Value	Name	Description
0x0	CYCLE_COUNT	Cycle counter
0x1	IHIT_COUNT	Instruction hit counter
0x2	DHIT_COUNT	Data hit counter
0x3		Reserved

11.10.9 Cache Monitor Enable

Name: MEN
Offset: 0x2C
Reset: 0x00000000
Property: Read/Write

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								MENABLE
Access								R/W
Reset								0

Bit 0 – MENABLE Cache Controller Monitor Enable
Writing a '0' to this bit disables the monitor counter.
Writing a '1' to this bit enables the monitor counter.

11.10.10 Cache Monitor Control

Name: MCTRL
Offset: 0x30
Reset: 0x00000000
Property: Write-only

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								SWRST
Access								W
Reset								0

Bit 0 – SWRST Cache Controller Software Reset
Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets the event counter register.

11.10.11 Cache Monitor Status

Name: MSR
Offset: 0x34
Reset: 0x00000000
Property: Read-only

Bit	31	30	29	28	27	26	25	24
	EVENT_CNT[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EVENT_CNT[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EVENT_CNT[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – EVENT_CNT[31:0] Monitor Event Counter
This field indicates the Monitor Event Counter value.

12. Device Service Unit (DSU)

12.1 Overview

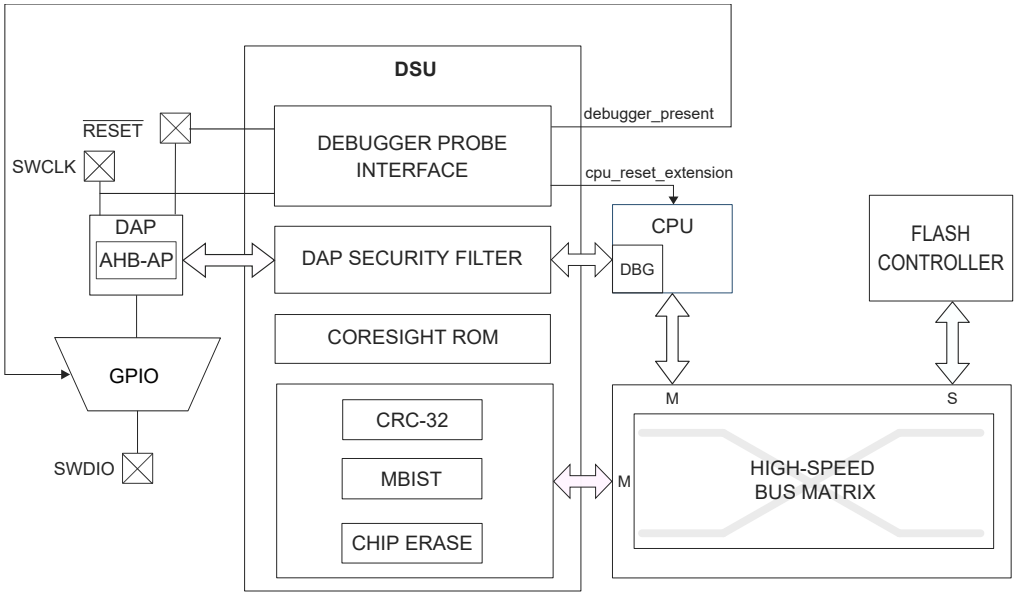
The Device Service Unit (DSU) provides a means of detecting debugger probes. It enables the ARM Debug Access Port (DAP) to have control over multiplexed debug pads and CPU Reset. The DSU also provides system-level services to debug adapters in an ARM debug system. It implements a CoreSight Debug ROM that provides device identification as well as identification of other debug components within the system. Hence, it complies with the ARM Peripheral Identification specification. The DSU also provides system services to applications that need memory testing, as required for IEC60730 Class B compliance, for example. The DSU can be accessed simultaneously by a debugger and the CPU, as it is connected on the High-Speed Bus Matrix. For security reasons, some of the DSU features will be limited or unavailable when the device is protected by the Code Protect bit.

12.2 Features

- CPU Reset Extension
- Debugger Probe Detection (Cold- and Hot-Plugging)
- Chip-Erase Command and Status
- 32-Bit Cyclic Redundancy Check (CRC32) of any Memory Accessible Through the Bus Matrix
- ARM® CoreSight™ Compliant Device Identification
- Two Debug Communications Channels
- Debug Access Port Security Filter

12.3 Block Diagram

Figure 12-1. DSU Block Diagram



12.4 Signal Description

The DSU uses three signals to function.

Signal Name	Type	Description
$\overline{\text{RESET}}$	Digital Input	External Reset pin
SWCLK	Digital Input	SW clock pin
SWDIO	Digital I/O	SW bidirectional data pin

12.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

12.5.1 I/O Lines

The SWCLK pin is by default assigned to the DSU module to allow debugger probe detection and to stretch the CPU Reset phase (see *Debugger Probe Detection* from Related Links). The Hot-Plugging feature depends on the GPIO configuration. If the SWCLK pin function is changed in the port, the Hot-Plugging feature is not disabled. Hot-Plugging is disabled with the CFGCON0.HPLUGDIS bit, which is enabled by default. Therefore to use the SWCLK pin for GPIO functions, it must be disabled by setting CFGCON0.HPLUGDIS = `1`.

Related Links

12.6.3. Debugger Probe Detection

12.5.2 Power Management

The DSU will continue to operate in any sleep mode where the selected source clock is running.

12.5.3 DMA

The DMA request lines are connected to the DMA Controller (DMAC). To use DMA requests with this peripheral, the DMAC must be configured first (see *Direct Memory Access Controller* from Related Links).

Related Links

22. Direct Memory Access Controller (DMAC)

12.5.4 Interrupts

Not applicable.

12.5.5 Events

Not applicable.

12.5.6 Register Access Protection

Registers with write access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

- Debug Communication Channel 0 register (DCC0)
- Debug Communication Channel 1 register (DCC1)

Note: Optional write protection is indicated by the "PAC Write Protection" property in the register description.

Write protection does not apply for accesses through an external debugger.

12.5.7 Analog Connections

Not applicable.

12.6 Debug Operation

12.6.1 Principle of Operation

The DSU provides basic services to allow on-chip debug using the ARM Debug Access Port and the ARM processor debug resources:

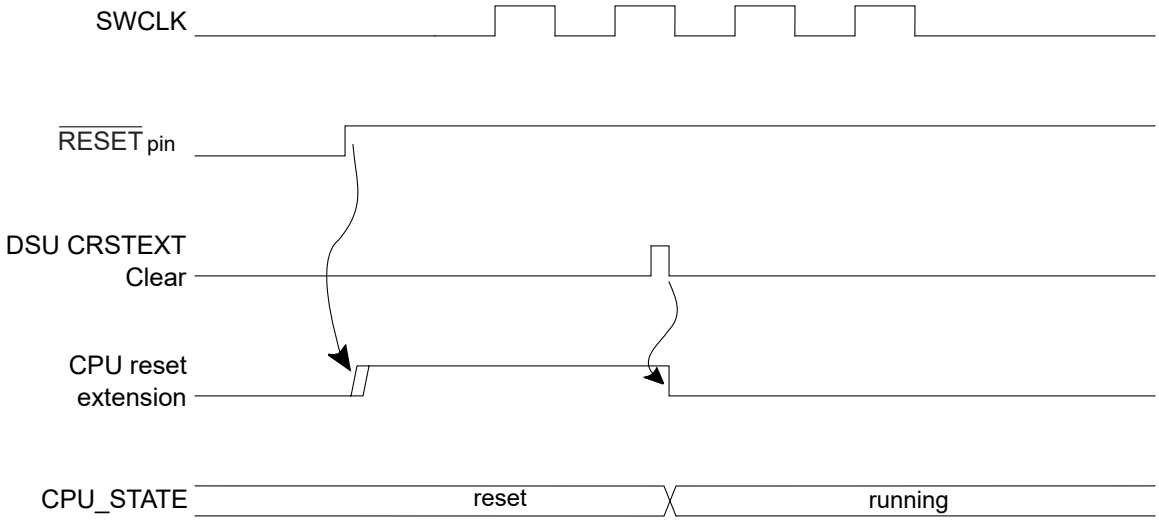
- CPU Reset extension
- Debugger probe detection

For more details on the ARM debug components, refer to the *ARM Debug Interface v5 Architecture Specification*.

12.6.2 CPU Reset Extension

“CPU Reset extension” refers to the extension of the Reset phase of the CPU core after the external Reset is released. This ensures that the CPU is not executing code at start-up while a debugger connects to the system. The debugger is detected on a $\overline{\text{RESET}}$ release event when SWCLK is low. At start-up, SWCLK is internally pulled up to avoid false detection of a debugger if the SWCLK pin is left unconnected. When the CPU is held in the Reset extension phase, the CPU Reset extension bit of the Status A register (STATUSA.CRSTEXT) is set. To release the CPU, write a '1' to STATUSA.CRSTEXT. STATUSA.CRSTEXT will, then, be set to '0'. Writing a '0' to STATUSA.CRSTEXT has no effect. For security reasons, it is not possible to release the CPU Reset extension when the device is protected by the Code Protect bit. Trying to do so sets the Protection Error bit (PERR) of the Status A register (STATUSA.PERR).

Figure 12-2. Typical CPU Reset Extension Set and Clear Timing Diagram



12.6.3 Debugger Probe Detection

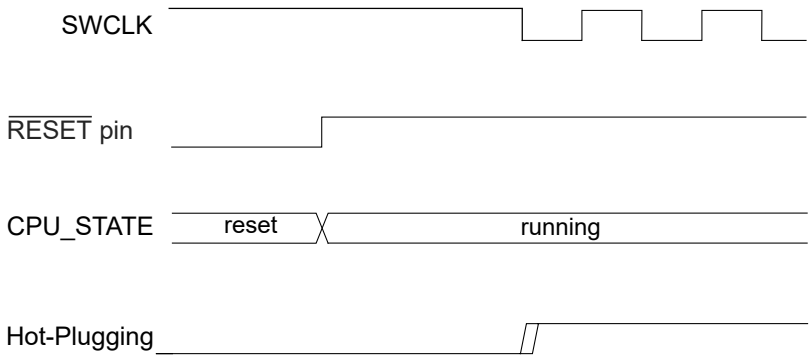
12.6.3.1 Cold Plugging

Cold-Plugging is the detection of a debugger when the system is in Reset. Cold-Plugging is detected when the CPU Reset extension is requested, as described above.

12.6.3.2 Hot Plugging

Hot-Plugging is the detection of a debugger probe when the system is not in Reset. Hot-Plugging is not possible under Reset because the detector is reset when POR or $\overline{\text{RESET}}$ are asserted. Hot-Plugging is active when a SWCLK falling edge is detected. The SWCLK pad is multiplexed with other functions and the user must ensure that its default function is assigned to the debug system. If the SWCLK pin function is changed in the port, the Hot-Plugging feature is not disabled. Hot-Plugging is disabled with the CFGCON0.HPLUGDIS bit, which is enabled by default. Therefore, to use the SWCLK pin for GPIO functions, it must be Disabled by setting CFGCON0.HPLUGDIS=1. Availability of the Hot-Plugging feature can be read from the Hot-Plugging Enable bit of the Status B register (STATUSB.HPE).

Figure 12-3. Hot-Plugging Detection Timing Diagram



The presence of a debugger probe is detected when either Hot-Plugging or Cold-Plugging is detected. Once detected, the Debugger Present bit of the Status B register (STATUSB.DBGPRES) is set. For security reasons, Hot-Plugging is not available when the device is protected by the Code Protect bit.

This detection requires that pads are correctly powered. Thus, at cold start-up, this detection cannot be done until POR is released. If the device is protected, Cold-Plugging is the only way to detect a debugger probe, and so the external Reset timing must be longer than the POR timing. If external Reset is deasserted before POR release, the user must retry the procedure above until it gets connected to the device.

12.7 Chip Erase

Chip erase consists of removing all sensitive information stored in the chip and clearing the Code Protect bit. Therefore, all volatile memories and the Flash memory (including the EEPROM emulation area) will be erased. The Flash auxiliary rows, including the user row, will not be erased.

When the device is protected, the debugger must first reset the device in order to be detected. This ensures that internal registers are reset after the Protected state is removed. The chip erase operation is triggered by writing a '1' to the chip erase bit in the Control register (CTRL.CE). This command will be discarded if the DSU is protected by the Peripheral Access Controller (PAC). Once issued, the module clears volatile memories prior to erasing the Flash array. To ensure that the chip erase operation is completed, check the Done bit of the Status A register (STATUSA.DONE).

The chip erase operation depends on clocks and power management features that can be altered by the CPU. For that reason, it is recommended to issue a chip erase after a Cold-Plugging procedure to ensure that the device is in a known and Safe state.

The recommended sequence is as follows:

1. Issue the Cold-Plugging procedure (see *Cold Plugging* from Related Links). The device then:
   a. Detects the debugger probe.
   b. Holds the CPU in Reset.
2. Issue the chip erase command by writing a '1' to CTRL.CE. The device then:
   a. Clears the system volatile memories.
   b. Erases the whole Flash array (excluding OTP).
   c. Erases the Code Protect bit protection.
3. Check for completion by polling STATUSA.DONE (read as '1' when completed).
4. Reset the device to let the Flash Controller update the fuses.

Related Links

12.6.3.1. Cold Plugging

12.8 Programming

Programming the Flash or RAM memories is only possible when the device is not protected by the Code Protect bit. The programming procedure is as follows:

1. At power-up, $\overline{\text{RESET}}$ is driven low by a debugger. The on-chip regulator holds the system in a POR state until the input supply is above the POR threshold (see *Power-on Reset (POR)* from Related Links for the characteristics). The system continues to be held in this Static state until the internally regulated supplies have reached a safe Operating state.
2. The PM starts, clocks are switched to the slow clock (Core Clock, System Clock, Flash Clock and any Bus Clocks that do not have clock gate control). Internal Resets are maintained due to the external Reset.
3. The debugger maintains a low level on SWCLK. $\overline{\text{RESET}}$ is released, resulting in a debugger Cold-Plugging procedure.
4. The debugger generates a clock signal on the SWCLK pin, the Debug Access Port (DAP) receives a clock.
5. The CPU remains in Reset due to the Cold-Plugging procedure; meanwhile, the rest of the system is released.
6. A chip erase is issued to ensure that the Flash is fully erased prior to programming.
7. Programming is available through the AHB-AP. Refer to the *PIC32CX-BZ2 Programming Specification* for more details.
8. After the operation is completed, the chip can be restarted either by asserting $\overline{\text{RESET}}$ or toggling power. Make sure that the SWCLK pin is high when releasing $\overline{\text{RESET}}$ to prevent extending the CPU Reset.

Related Links

13.18.4.2. Power-on Reset (POR)

12.9 Intellectual Property Protection

Intellectual property protection consists of restricting access to internal memories from external tools when the device is protected, and this is accomplished by setting the Code Protect bit. This Protected state can be removed by issuing a chip erase (see *Chip Erase* from Related Links). When the device is protected, read/write accesses using the AHB-AP are limited to the DSU address range and DSU commands are restricted. When issuing a chip erase, sensitive information is erased from volatile memory and Flash.

The DSU implements a security filter that monitors the AHB transactions inside the DAP. If the device is protected, then AHB-AP read/write accesses outside the DSU external address range are discarded, causing an error response that sets the ARM AHB-AP sticky error bits (For more details, refer to the *ARM Debug Interface v5 Architecture Specification* on www.arm.com).

The DSU is intended to be accessed either:

- Internally from the CPU, without any limitation, even when the device is protected
- Externally from a debug adapter, with some restrictions when the device is protected

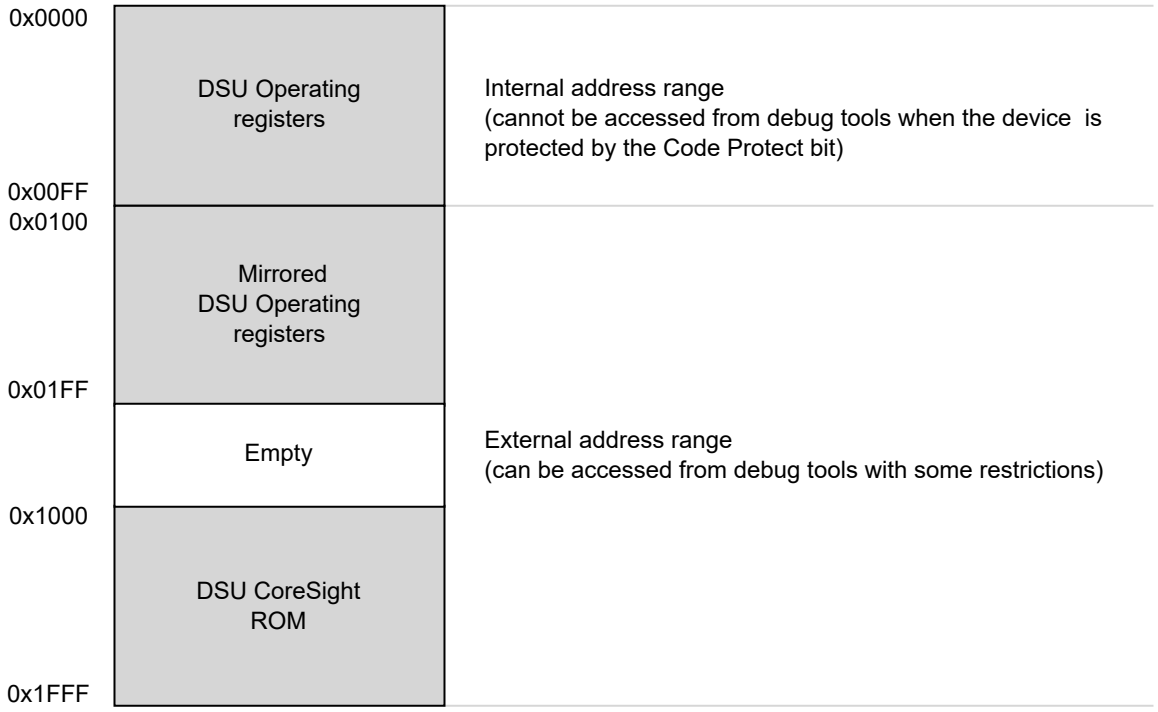
For security reasons, DSU features have limitations when used from a debug adapter. To differentiate external accesses from internal ones, the first 0x100 bytes of the DSU register map has been mirrored at offset 0x100:

- The first 0x100 bytes form the internal address range
- The next 0x100 bytes form the external address range

When the device is protected, the DAP can only issue MEM-AP accesses in the DSU range 0x0100-0x2000.

The DSU Operating registers are located in the 0x0000-0x00FF area and remapped in 0x0100-0x01FF to differentiate accesses coming from a debugger and the CPU. If the device is protected and an access is issued in the region 0x0100-0x01FF, it is subject to security restrictions. For more information, refer to the following table.

Figure 12-4. APB Memory Mapping



Some features not activated by APB transactions are not available when the device is protected:

Table 12-1. Feature Availability Under Protection

Features	Availability When the Device is Protected
CPU Reset Extension	Yes
Clear CPU Reset Extension	No
Debugger Cold-Plugging	Yes
Debugger Hot-Plugging	No

Related Links

12.7. Chip Erase

12.10 Device Identification

Device identification relies on the ARM CoreSight component identification scheme, which allows the chip to be identified as a SAM device implementing a DSU. The DSU contains identification registers to differentiate the device.

12.10.1 CoreSight Identification

A system-level ARM® CoreSight™ ROM table is present in the device to identify the vendor and the chip identification method. Its address is provided in the MEM-AP BASE register inside the ARM Debug Access Port. The CoreSight ROM implements a 64-bit conceptual ID composed as follows from the PID0 to PID7 CoreSight ROM Table registers:

Figure 12-5. Conceptual 64-bit Peripheral ID

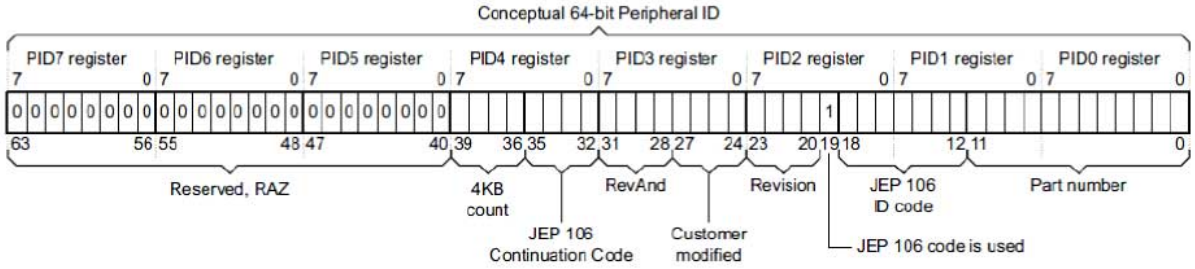


Table 12-2. Conceptual 64-Bit Peripheral ID Bit Descriptions

Field	Size	Description	Location
JEP-106 CC code	4	Microchip continuation code: 0x0	PID4
JEP-106 ID code	7	Microchip device ID: 0x1F	PID1+PID2
4KB count	4	Indicates that the CoreSight component is a ROM: 0x0	PID4
RevAnd	4	Not used; read as 0	PID3
CUSMOD	4	Not used; read as 0	PID3
PARTNUM	12	Contains 0xCD0 to indicate that DSU is present	PID0+PID1
REVISION	4	DSU revision (starts at 0x0 and increments by 1 at both major and minor revisions). Identifies DSU identification method variants. If 0x0, this indicates that device identification can be completed by reading the Device Identification register (DID)	PID2

For more details, refer to the *ARM Debug Interface Version 5 Architecture Specification*.

12.10.2 Chip Identification Method

The DSU DID register identifies the device as shown in the following table:

Table 12-3. DSU DID Encoding

Field	Size	Value	Comments
Revision	4 bits	0x0	Immutable Field (0x0=Rev-A0)
Family	5 bits	0b00000	Family[4:0]
Series	6 bits	0b00000	Series[5:0]
Die	8 bits	0x9B	Immutable Field = Mask ID [7:0]
DEVSEL	8 bits	Flash Fuses	Determines variants of product {VSEL[7:0]} [0x8F|0x0F|0x0B]

12.11 Functional Description

12.11.1 Principle of Operation

The DSU provides memory services, such as CRC32 that require almost the same interface. Hence, the Address, Length and Data registers (ADDR, LENGTH, DATA) are shared. These shared registers must be configured first; then a command can be issued by writing the Control register. When a command is ongoing, other commands are

discarded until the current operation is completed. Hence, the user must wait for the STATUSA.DONE bit to be set prior to issuing another one.

12.11.2 Basic Operation

12.11.2.1 Initialization

The module is enabled by enabling its clocks, see *Clock and Reset Unit (CRU)* from Related Links. The DSU registers can be PAC write-protected, see *Peripheral Access Controller (PAC)* from Related Links.

Related Links

13. Clock and Reset Unit (CRU)
26. Peripheral Access Controller (PAC)

12.11.2.2 Operation From a Debug Adapter

Debug adapters must access the DSU registers in the external address range 0x100 – 0x2000. If the device is protected by the Code Protect bit, accessing the first 0x100 bytes causes the system to return an error. See *Intellectual Property Protection* from Related Links.

Related Links

12.9. Intellectual Property Protection

12.11.2.3 Operation From the CPU

There are no restrictions when accessing DSU registers from the CPU. However, the user must access DSU registers in the internal address range (0x0–0x100) to avoid external security restrictions. See *Intellectual Property Protection* from Related Links.

Related Links

12.9. Intellectual Property Protection

12.11.3 32-bit Cyclic Redundancy Check CRC32

The DSU unit provides support for calculating a cyclic redundancy check (CRC32) value for a memory area (including Flash and AHB RAM).

When the CRC32 command is issued from:

- The internal range, the CRC32 can be operated at any memory location
- The external range, the CRC32 operation is restricted; DATA, ADDR, and LENGTH values are forced (see below)

Table 12-4. AMOD Bit Descriptions when Operating CRC32

AMOD[1:0]	Short name	External range restrictions
0	ARRAY	CRC32 is restricted to the full Flash array area (EEPROM Emulation area not included) DATA forced to 0xFFFFFFFF before calculation (no seed)
1	EEPROM	CRC32 of the whole EEPROM Emulation area DATA forced to 0xFFFFFFFF before calculation (no seed)
2-3	Reserved	-

The algorithm employed is the industry standard CRC32 algorithm using the generator polynomial 0xEDB88320 (reversed representation).

12.11.3.1 Starting CRC32 Calculation

CRC32 calculation for a memory range is started after writing the start address into the Address register (ADDR) and the size of the memory range into the Length register (LENGTH). Both must be word-aligned.

The initial value used for the CRC32 calculation must be written to the Data register (DATA). This value will usually be 0xFFFFFFFF, but can be, for example, the result of a previous CRC32 calculation if generating a common CRC32 of separate memory blocks.

Once completed, the calculated CRC32 value can be read out of the Data register. The read value must be complemented to match standard CRC32 implementations or kept noninverted if used as starting point for subsequent CRC32 calculations.

The actual test is started by writing a '1' in the 32-bit Cyclic Redundancy Check bit of the Control register (CTRL.CRC). A running CRC32 operation can be canceled by resetting the module (writing '1' to CTRL.SWRST).

12.11.3.2 Interpreting the Results

The user must monitor the Status A register. When the operation is completed, STATUSA.DONE is set. Then the Bus Error bit of the Status A register (STATUSA.BERR) must be read to ensure that no bus error occurred.

12.11.4 Debug Communication Channels

The Debug Communication Channels (DCCO and DCC1) consist of a pair of registers with associated handshake logic, accessible by both CPU and debugger even if the device is protected by the Code Protect bit. The registers can be used to exchange data between the CPU and the debugger, during run time as well as in Debug mode. This enables the user to build a custom debug protocol using only these registers.

The DCC0 and DCC1 registers are accessible when the Protected state is active. When the device is protected, however, it is not possible to connect a debugger while the CPU is running (STATUSA.CRSTEXT is not writable and the CPU is held under Reset).

Two Debug Communication Channel status bits in the Status B registers (STATUS.DCCDx) indicate whether a new value has been written in DCC0 or DCC1. These bits, DCC0D and DCC1D, are located in the STATUSB registers. They are automatically set on write and cleared on read.

Note: The DCC0 and DCC1 registers are shared with the on-board memory testing logic (MBIST). Accordingly, DCC0 and DCC1 must not be used while performing MBIST operations.

12.11.5 System Services Availability when Accessed Externally and Device is Protected

External access: Access performed in the DSU address offset 0x200-0x1FFF range.

Internal access: Access performed in the DSU address offset 0x000-0x100 range.

Table 12-5. Available Features when Operated From The External Address Range and Device is Protected

Features	Availability From The External Address Range and Device is Protected
Chip erase command and status	Yes
CRC32	Yes, only full array or full EEPROM
CoreSight Compliant Device identification	Yes
Debug communication channels	Yes
STATUSA.CRSTEXT clearing	No (STATUSA.PERR is set when attempting to do so)

12.12 DSU Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRL	7:0				CE				SWRST
0x01	STATUSA	7:0				PERR	FAIL	BERR	CRSTEXT	DONE
0x02	STATUSB	7:0			CELCK	HPE	DCCD1	DCCD0	DBGPRES	PROT
0x03	Reserved									
0x04	ADDR	7:0	ADDR[5:0]						AMOD[1:0]	
		15:8	ADDR[13:6]							
		23:16	ADDR[21:14]							
		31:24	ADDR[29:22]							
0x08	LENGTH	7:0	LENGTH[5:0]							
		15:8	LENGTH[13:6]							
		23:16	LENGTH[21:14]							
		31:24	LENGTH[29:22]							
0x0C	DATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							
0x10	DCC0	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							
0x14	DCC1	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							
0x18	DID	7:0	DEVSEL[7:0]							
		15:8	DIE[7:0]							
		23:16	FAMILY[0]		SERIES[5:0]					
		31:24	REVISION[3:0]				FAMILY[4:1]			
0x1C ... 0x0FFF	Reserved									
0x1000	ENTRY0	7:0							FMT	EPRES
		15:8	ADDOFF[3:0]							
		23:16	ADDOFF[11:4]							
		31:24	ADDOFF[19:12]							
0x1004	ENTRY1	7:0							FMT	EPRES
		15:8	ADDOFF[3:0]							
		23:16	ADDOFF[11:4]							
		31:24	ADDOFF[19:12]							
0x1008	END	7:0	END[7:0]							
		15:8	END[15:8]							
		23:16	END[23:16]							
		31:24	END[31:24]							
0x100C ... 0x1FCB	Reserved									
0x1FCC	MEMTYPE	7:0								SMEMP
		15:8								
		23:16								
		31:24								
0x1FD0	PID4	7:0	FKBC[3:0]				JEPCC[3:0]			
		15:8								
		23:16								
		31:24								

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1FD4 ... 0x1FDF	Reserved									
0x1FE0	PID0	7:0	PARTNBL[7:0]							
		15:8								
		23:16								
		31:24								
0x1FE4	PID1	7:0	JEPIDCL[3:0]				PARTNBH[3:0]			
		15:8								
		23:16								
		31:24								
0x1FE8	PID2	7:0	REVISION[3:0]				JEPU	JEPIDCH[2:0]		
		15:8								
		23:16								
		31:24								
0x1FEC	PID3	7:0	REVAND[3:0]				CUSMOD[3:0]			
		15:8								
		23:16								
		31:24								
0x1FF0	CID0	7:0	PREAMBLEB0[7:0]							
		15:8								
		23:16								
		31:24								
0x1FF4	CID1	7:0	CCLASS[3:0]				PREAMBLE[3:0]			
		15:8								
		23:16								
		31:24								
0x1FF8	CID2	7:0	PREAMBLEB2[7:0]							
		15:8								
		23:16								
		31:24								
0x1FFC	CID3	7:0	PREAMBLEB3[7:0]							
		15:8								
		23:16								
		31:24								

12.13 Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the "PAC Write-Protection" property in each individual register description. See *Register Access Protection* from Related Links.

Related Links

12.5.6. Register Access Protection

12.13.1 Control

Name: CTRL
Offset: 0x0000
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
				CE				SWRST
Access				W				W
Reset				0				0

Bit 4 – CE Chip-Erase
Writing a '0' to this bit has no effect.
Writing a '1' to this bit starts the Chip-Erase operation.

Bit 0 – SWRST Software Reset
Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets the module.

12.13.2 Status A

Name: STATUSA
Offset: 0x0001
Reset: 0x00
Property: PAC Write Protection

Bit	7	6	5	4	3	2	1	0
				PERR	FAIL	BERR	CRSTEXT	DONE
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit 4 – PERR Protection Error
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Protection Error bit.
This bit is set when a command that is not allowed in Protected state is issued.

Bit 3 – FAIL Failure
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Failure bit.
This bit is set when a DSU operation failure is detected.

Bit 2 – BERR Bus Error
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Bus Error bit.
This bit is set when a bus error is detected.

Bit 1 – CRSTEXT CPU Reset Phase Extension
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the CPU Reset Phase Extension bit.
This bit is set when a debug adapter Cold-Plugging is detected, which extends the CPU Reset phase.

Bit 0 – DONE Done
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Done bit.
This bit is set when a DSU operation is completed.

12.13.3 Status B

Name: STATUSB
Offset: 0x0002
Reset: 0x0x
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
			CELCK	HPE	DCCD1	DCCD0	DBGPRES	PROT
Access			R	R	R	R	R	R
Reset			0	0	0	0	x	x

Bit 5 – CELCK Chip Erase Locked
Writing a '0' to this bit has no effect.
Writing a '1' to this bit has no effect.
This bit is set when Chip Erase is locked.
This bit is cleared when Chip Erase is unlocked.

Bit 4 – HPE Hot-Plugging Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit has no effect.
This bit is set when Hot-Plugging is enabled.
This bit is cleared when Hot-Plugging is disabled. This is the case when the SWCLK function is changed. Only a power-reset or a external reset can set it again.

Bits 2, 3 – DCCD Debug Communication Channel x Dirty
Writing a '0' to this bit has no effect.
Writing a '1' to this bit has no effect.
This bit is set when DCC is written.
This bit is cleared when DCC is read.

Bit 1 – DBGPRES Debugger Present
Writing a '0' to this bit has no effect.
Writing a '1' to this bit has no effect.
This bit is set when a debugger probe is detected.
This bit is never cleared.

Bit 0 – PROT Protected
Writing a '0' to this bit has no effect.
Writing a '1' to this bit has no effect.
This bit is set at power-up when the device is protected.
This bit is never cleared.

12.13.4 Address

Name: ADDR
Offset: 0x04
Reset: 0x00000000
Property: PAC Write Protection

Bit	31	30	29	28	27	26	25	24
	ADDR[29:22]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ADDR[21:14]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[13:6]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[5:0]						AMOD[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:2 – ADDR[29:0] Address
Initial word start address needed for memory operations.

Bits 1:0 – AMOD[1:0] Access Mode
The functionality of these bits is dependent on the operation mode.
Bit description when operating CRC32 (see *32-bit Cyclic Redundancy Check (CRC32)* from Related Links).
Related Links
12.11.3. 32-bit Cyclic Redundancy Check CRC32

12.13.5 Length

Name: LENGTH
Offset: 0x0008
Reset: 0x00000000
Property: PAC Write Protection

Bit	31	30	29	28	27	26	25	24
	LENGTH[29:22]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LENGTH[21:14]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LENGTH[13:6]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:2 – LENGTH[29:0] Length
Length in words needed for memory operations.

12.13.6 Data

Name: DATA
Offset: 0x000C
Reset: 0x00000000
Property: PAC Write Protection

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – DATA[31:0] Data
Memory operation initial value or result value.

12.13.7 Debug Communication Channel x

Name: DCC
Offset: 0x10 + n*0x04 [n=0..1]
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – DATA[31:0] Data
Data register.

12.13.8 Device Identification

Name: DID
Offset: 0x0018
Property: PAC Write Protection

Bit	31	30	29	28	27	26	25	24
	REVISION[3:0]				FAMILY[4:1]			
Access	R	R	R	R	R	R	R	R
Reset	p	p	p	p	f	f	f	f

Bit	23	22	21	20	19	18	17	16
	FAMILY[0]		SERIES[5:0]					
Access	R		R	R	R	R	R	R
Reset	f		s	s	s	s	s	s

Bit	15	14	13	12	11	10	9	8
	DIE[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	d	d	d	d	d	d	d	d

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

Bits 31:28 – REVISION[3:0] Processor
The value of this field identifies the die revision number. 0x0=rev.A0.

Bits 27:23 – FAMILY[4:0] Product Family
The value of this field corresponds to the product family part of the ordering code. For this device, the value of this field is 0x0.

Bits 21:16 – SERIES[5:0] Product Series
The value of this field corresponds to the product series part of the ordering code. For this device, the value of this field is 0x01, corresponding to a product with the Cortex-M0+ processor with DMA and USB features.

Bits 15:8 – DIE[7:0] Die Number
Identifies the die family.

Bits 7:0 – DEVSEL[7:0] Device Selection
This bit field identifies a device within a product family and product series. The value corresponds to the Flash memory density, pin count and device variant parts of the ordering code.

12.13.9 CoreSight ROM Table Entry x

Name: ENTRY
Offset: 0x1000 + n*0x04 [n=0..1]
Reset: 0xxxxxx00x
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	ADDOFF[19:12]							
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
	ADDOFF[11:4]							
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
	ADDOFF[3:0]							
Access	R	R	R	R				
Reset	x	x	x	x				

Bit	7	6	5	4	3	2	1	0
							FMT	EPRES
Access							R	R
Reset							1	x

Bits 31:12 – ADDOFF[19:0] Address Offset
The base address of the component, relative to the base address of this ROM table.

Bit 1 – FMT Format
Always reads as '1', indicating a 32-bit ROM table.

Bit 0 – EPRES Entry Present
This bit indicates whether an entry is present at this location in the ROM table.
This bit is set at power-up if the device is not protected indicating that the entry is not present.
This bit is cleared at power-up if the device is not protected indicating that the entry is present.

12.13.10 CoreSight ROM Table End

Name: END
Offset: 0x1008
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	END[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	END[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	END[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – END[31:0] End Marker
Indicates the end of the CoreSight ROM table entries.

12.13.11 CoreSight ROM Table Memory Type

Name: MEMTYPE
Offset: 0x1FCC
Reset: 0x0000000X
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								SMEMP
Access								R
Reset								x

Bit 0 – SMEMP System Memory Present

This bit indicates whether system memory is present on the bus that connects to the ROM table.
This bit is set at power-up if the device is not protected, indicating that the system memory is accessible from a debug adapter.
This bit is cleared at power-up if the device is protected, indicating that the system memory is not accessible from a debug adapter.

12.13.12 Peripheral Identification 4

Name: PID4
Offset: 0x1FD0
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	FKBC[3:0]				JEPCC[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:4 – FKBC[3:0] 4KB Count
These bits will always return zero when read, indicating that this debug component occupies one 4KB block.

Bits 3:0 – JEPCC[3:0] JEP-106 Continuation Code
These bits will always return zero when read.

12.13.13 Peripheral Identification 0

Name: PID0
Offset: 0x1FE0
Reset: 0x000000D0
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

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

Bits 7:0 – PARTNBL[7:0] Part Number Low
These bits will always return 0xD0 when read, indicating that this device implements a DSU module instance.

12.13.14 Peripheral Identification 1

Name: PID1
Offset: 0x1FE4
Reset: 0x000000FC
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	JEPIDCL[3:0]				PARTNBH[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	0	0

Bits 7:4 – JEPIDCL[3:0] Low Part of the JEP-106 Identity Code
These bits will always return 0xF when read (JEP-106 identity code is 0x1F).

Bits 3:0 – PARTNBH[3:0] Part Number High
These bits will always return 0xC when read, indicating that this device implements a DSU module instance.

12.13.15 Peripheral Identification 2

Name: PID2
Offset: 0x1FE8
Reset: 0x00000009
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	REVISION[3:0]				JEPU	JEPIDCH[2:0]		
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	1	0	0	1

Bits 7:4 – REVISION[3:0] Revision Number
Revision of the peripheral. Starts at 0x0 and increments by one at both major and minor revisions.

Bit 3 – JEPU JEP-106 Identity Code is Used
This bit will always return one when read, indicating that JEP-106 code is used.

Bits 2:0 – JEPIDCH[2:0] JEP-106 Identity Code High
These bits will always return 0x1 when read, (JEP-106 identity code is 0x1F).

12.13.16 Peripheral Identification 3

Name: PID3
Offset: 0x1FEC
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	REVAND[3:0]				CUSMOD[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:4 – REVAND[3:0] Revision Number
These bits will always return 0x0 when read.

Bits 3:0 – CUSMOD[3:0] ARM CUSMOD
These bits will always return 0x0 when read.

12.13.17 Component Identification 0

Name: CID0
Offset: 0x1FF0
Reset: 0x0000000D
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	PREAMBLEB0[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	1	1	0	1

Bits 7:0 – PREAMBLEB0[7:0] Preamble Byte 0
These bits will always return 0x0000000D when read.

12.13.18 Component Identification 1

Name: CID1
Offset: 0x1FF4
Reset: 0x00000010
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	CCLASS[3:0]				PREAMBLE[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	1	0	0	0	0

Bits 7:4 – CCLASS[3:0] Component Class
These bits will always return 0x1 when read indicating that this ARM CoreSight component is ROM table (For more details, refer to the *ARM Debug Interface v5 Architecture Specification* at http://www.arm.com).

Bits 3:0 – PREAMBLE[3:0] Preamble
These bits will always return 0x0 when read.

12.13.19 Component Identification 2

Name: CID2
Offset: 0x1FF8
Reset: 0x00000005
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	PREAMBLEB2[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	1	0	1

Bits 7:0 – PREAMBLEB2[7:0] Preamble Byte 2
These bits will always return 0x00000005 when read.

12.13.20 Component Identification 3

Name: CID3
Offset: 0x1FFC
Reset: 0x000000B1
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	PREAMBLEB3[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	1	0	1	1	0	0	0	1

Bits 7:0 – PREAMBLEB3[7:0] Preamble Byte 3
These bits will always return 0x000000B1 when read.

13. Clock and Reset Unit (CRU)

13.1 Overview

The PIC32CX-BZ2 Clock System provides both clocking and reset functions. This chapter describes the clocking functionality and summarizes the clock distribution and terminology in the PIC32CX-BZ2 device. For more details on configuration, see the respective peripherals' descriptions. Clock control is handled by the CRU to provide system clocks and interface peripheral clocks. The CRU controls switching and synchronization of clock sources.

For details on the Reset functionality, see *Resets* from Related Links.

Related Links

13.18. Resets

13.2 Features

The Clock and Reset Unit has the following features:

- Supports the following as system clock sources:
  - 16 MHz Primary Crystal Oscillator (POSC)
  - 8 MHz Fast RC Oscillator (FRC)
  - 32 kHz Low Power RC Oscillator (LPRC)
  - 32.768 kHz Secondary Crystal Oscillator (SOSC)
  - 96 MHz System PLL (RFPLL)
- Provides control registers for all PLLs
- Provides for glitch-free clock switching between various clock sources
- Post dividers on processor clock generator to slow down system clock for power save
- A fail safe clock monitor that detects clock failure and provides automatic switching to the FRC
- Provides control registers for user interface of clocks and resets
- Provides configuration bits for oscillator selection and calibration of on-chip oscillators
- Provides control registers to generate a reference clock output
- Provide resets for the system
- Provides NMI interrupts for the system
- Multiple PB clock dividers
- One system clock, SYS_CLK, from which almost all clocks used throughout the system are derived
- Three peripheral clocks, created by independent integer dividers of the SYS_CLK:
  - PB1_CLK: PB-PIC and PB-Bridge-A bus
  - PB2_CLK: PB-Bridge-B and PB-Bridge-C
  - PB3_CLK: DS/XDS bus clock
- Six reference output clocks (REFO1 – REFO6) with the following clock sources:
  - System clock (SYS_CLK)
  - PB1 bus clock (PB1_CLK)
  - 16 MHz Primary Crystal Oscillator (POSC)
  - 8 MHz Fast RC Oscillator (FRC)
  - 32 kHz Low Power RC Oscillator (LPRC)
  - 32.768 kHz Secondary Crystal Oscillator (SOSC)
  - 96 MHz System PLL (RFPLL)
  - 64 MHz System PLL (RFPLL PGM MHz)
  - REFI pin
  - Sleep control, supporting Req/Ack signaling with the bus matrix to determine that no transactions are in flight when initiating sleep

- JTAG TCK clock control

13.3 Block Diagram

The CRU, along with the PMD, provides gated clock output for all peripheral buses. The following figure illustrates the CRU block diagram.

Figure 13-1. Clock and Reset Unit Block Diagram

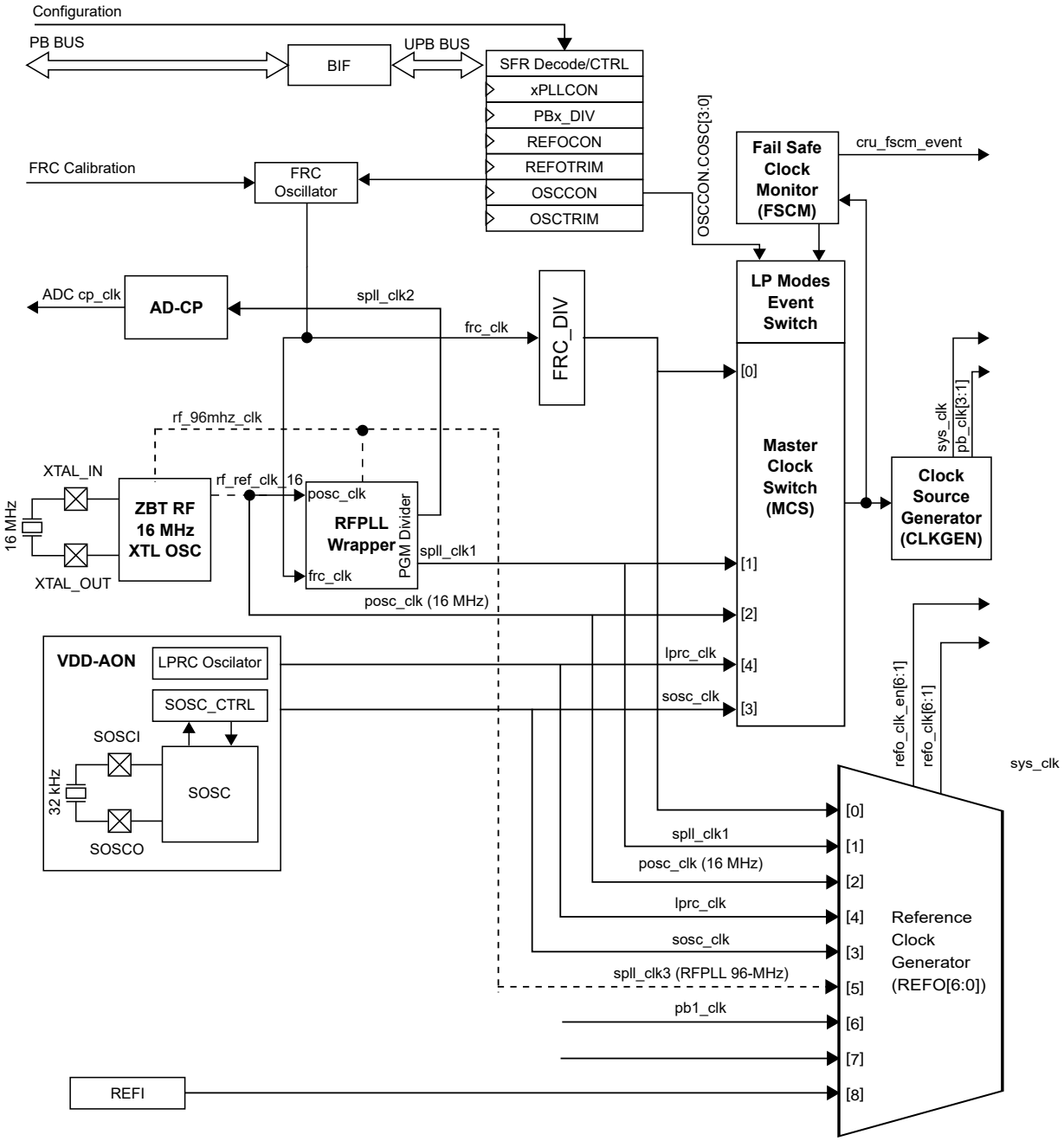


Figure 13-2. RFPLL Wrapper

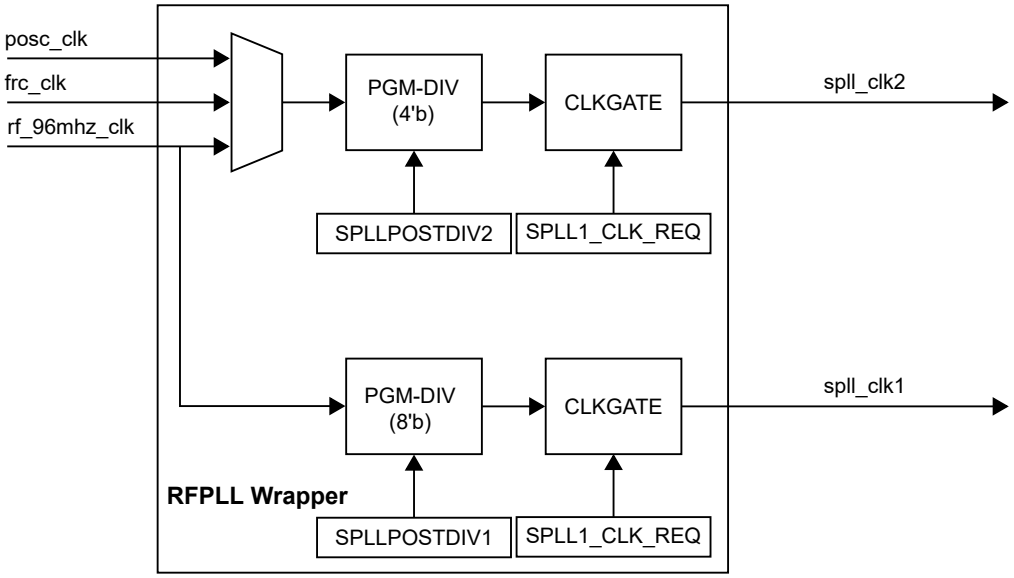
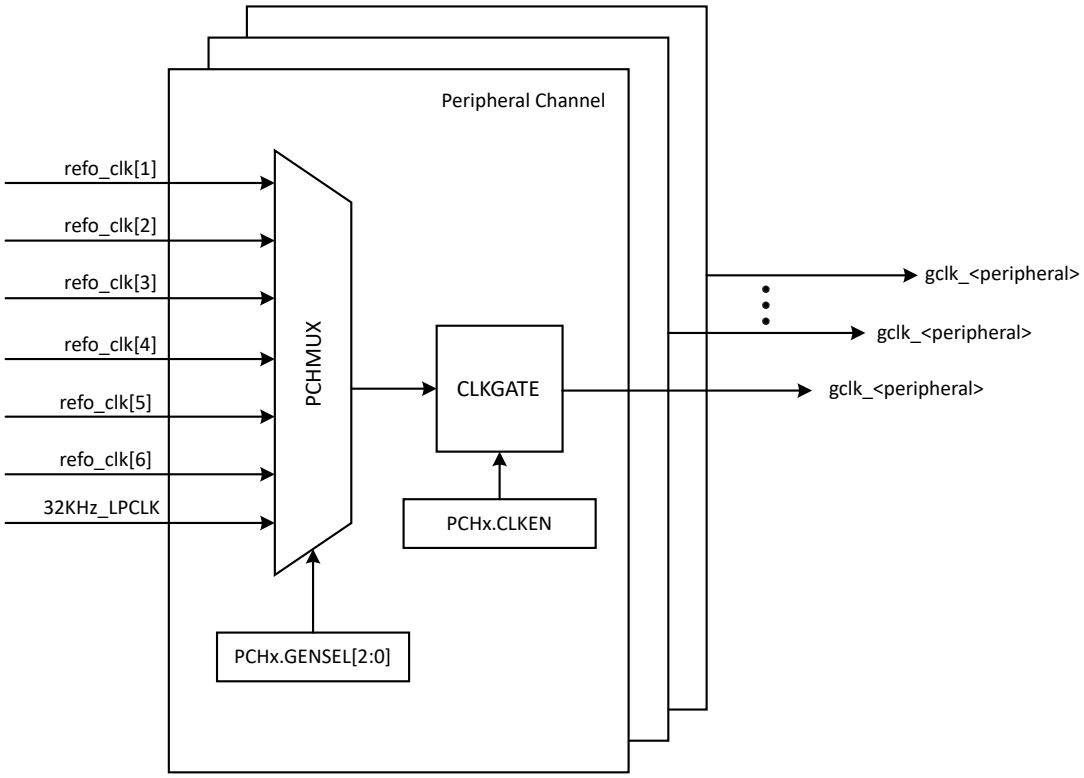


Figure 13-3. Peripheral Clock Generation



The CRU Master Clock Switch selects the input clock, which needs to be fed to the CLKGEN Synchronous Clock Generator. The CLKGEN generates and controls the synchronous clocks on the system. This includes the CPU, bus clocks (APB and AHB) and the synchronous (to the CPU) user interfaces of the peripherals. It contains prescalers for the CPU and bus clocks.

13.4 System and Peripheral Clock Generation (CLKGEN)

This sub-module generates the system clocks needed for the device from a single source clock. In addition, this module also shuts down these clocks during the Sleep mode.

There are two types of clocks generated by this block called core clocks and peripheral clocks. The system clock (SYS_CLK) is typically used by the CPU. It supports components, such as memory subsystems, and fast peripherals. The peripheral bus clocks (pb_clk) are used to clock slow peripheral devices attached to the pb_bus. The pb_clk[n] outputs are based on the SYS_CLK frequency with a fixed divisor. The divisor is determined by the value of the PBxDIV registers.

The system and peripheral clocks are stopped when in the Sleep mode. The clocks are restarted by disabling the sleep enable.

13.4.1 Sleep Mode

The Sleep mode is entered when DSCON[DSEN] is clear and the OSCCON[SLPEN] bit is set and the CPU executes a WFI instruction. This causes the device clocks to be held low ('0').

13.4.2 Sleep Mode Entry

Entry into the Sleep mode from any other mode does not require a clock switch. This is due to the fact that once in the Sleep mode, no clocks are needed.

Note: If software writes to the CRU SFRs before going into the Sleep mode, it is recommended a read be done of the SFRs to Flash the write before executing the WAIT instruction that initiates the Sleep mode.

13.4.3 Sleep Mode Exit

Unless the Two-Speed Start-up is enabled, there are no clock-switching events when exiting the Sleep mode. With Two-Speed Start-up disabled, the Sleep mode is effectively extended until the selected clock is ready. Once the clock is ready, the device enters the RUN mode.

If Two-Speed Start-up is enabled, the device will come out of Sleep running with the FRC as the clock source and perform an automatic clock switch to the selected clock source.

If Two-Speed Start-up is enabled, the RUN mode is entered immediately using the Two-Speed Start-up clock source. A clock switch to the selected clock source occurs once the source is ready.
Note: Two-Speed Start-up is not permanently enabled and is software programmable.

13.4.4 Sleep Mode with Delayed Exit

In some of the low power Sleep modes, some of the on-chip voltage regulators may be turned off in the system. If this is the case, the Sleep mode cannot be exited immediately on a wake-up event. The wake from Sleep will be delayed until the system is ready to have clocks turned on again.

13.5 Idle Mode

The Idle mode is entered when the OSCCON[SLPEN] bit is low and the CPU executes a WFI instruction. Only the CPU's internal clock is stopped in this mode.

Note: When exiting from the Sleep mode, the CRU will transition the system to the Idle mode before transitioning to the Run mode. This transition to the Idle mode is used to support the Dream mode and always occurs regardless of the CRU transitioning to the Dream mode or Run mode. Therefore, when exiting the Sleep mode, both the RCON bits [SLEEP] and [IDLE] will be set.

13.6 Dream Mode

When the DRMEN bit in the OSCCON register is set, it allows the DMA controller to switch between the Idle mode and the Sleep mode. When the OSCCON.SLPEN bit is set and the OSCCON.DRMEN bit is set, the CRU monitors the DMAC to make sure all transfers are complete before going into the Sleep mode.

If OSCCON.SLPEN = 1, OSCCON.DRMEN = 1, and peripheral clock requests are active, the CRU goes into the Idle mode until all peripheral clock requests are non-active; at which time the CRU goes into the Sleep mode.

If OSCCON.SLPEN is not set, the DRMEN bit has no affect as the DMA clocks are still running in the Idle mode.

If the CRU recognizes a wake/interrupt event whose priority will wake the DMA but not the CPU, the CRU transitions to the Idle mode. Therefore, the DMA can perform the needed operations and, when the DMA is finished, the CRU will go back to the Sleep mode. During this time, the CPU is still asleep.

If the wake event is such that the CPU must handle the event, the whole system will exit the Sleep mode and transition back to the Run mode.

13.7 FRCDIV

The FRC can be divided and used as a system clock. The user controls the divider setting using the OSCCON.FRCDIV[2:0] register bits. The divisor is configured for eight divider selections: /1, /2, /4, /8, /16, /32, /64, /256.

13.8 RFPLL Wrapper

The RFPLL wrapper generates two clocks:

- spll_clk1 (PGM MHz)
  - Clock frequencies = 96 MHz/(1-255) and 64 MHz frequency choices and optional clock disable option
  - Clock ready indication
- spll_clk2 (PGM MHz)
  - Clock frequencies = 96 MHz/(1-15) and optional clock disable option
  - Clock ready indication

Clocks are produced only when there is a request generated by the user; for clk1, it is CRU and for clk2, it is ADC charge-pump. Along with the clocks, individual clock ready is also generated, which indicates that clocks are ready for consumption.

13.9 Start-up Considerations

The presence of hardware NVR fuses on the PIC32CX-BZ2 device allows the system configuration fuses to be ready upon exiting Reset. The following start-up conditions exist:

- On any device Reset, no start-up time is required to transfer configuration values from the NVR memory into the configuration holding registers.
- Once the device is active, the user may change the primary system clock source from FRC to SPLL by using the OSCCON register.

13.10 Fail-Safe Clock Monitor

The Clock System includes a Fail-safe Clock Monitor (FSCM). The FSCM monitors the SYS_CLK for continuous operation. If it detects that the SYS_CLK failed, it switches the SYS_CLK over to the FRC oscillator and triggers an NMI. The FRC is an untuned 8 MHz oscillator that drives the SYS_CLK during an FSCM event. When the NMI is executed, software can restart the main oscillator or shut down the system.

In the Sleep mode, both the SYS_CLK and the FSCM halt, preventing FSCM detection.

13.11 Fast RC Oscillator

The on-chip 8 MHz Fast RC Oscillator (FRC) is intended to be a fast, with precise frequency, internal RC oscillator. The FRC supports calibration to ± 0.25% accuracy pre-package. However, package-induced stress lowers the accuracy.

The FRC oscillator is accurate to provide the clock frequency necessary to maintain baud rate tolerance for serial data transmissions. FRC is enabled with conditions in 13.11.1. Enabling the FRC; otherwise, it is not enabled. Power-on Reset sets NOSC[3:0] = 0000; therefore, it is always ON when powered-up.

The oscillator module provides a 6-bit wide user tuning adjustment capability using the OSCTRM.TUN[5:0] bits.

13.11.1 Enabling the FRC

The FRC oscillator is powered when OSCCON.NOSC[3:0] = 4'b0000 or a Fail-safe clock monitor is enabled and a clock fail is detected, forcing a switch to FRC. It is also enabled whenever it gets requested by: ADC requesting for FRC, Configuring an SPLL2 source as FRC, PMU Controller requesting for FRC, Flash Controller requesting for FRC, Configuring LPCLK (32 KHz) source to FRC.

13.11.2 Frequency Tuning in User Mode

In addition to the factory calibration, the base frequency can be tuned in the user's application. This frequency tuning capability allows the user to deviate from the factory calibrated frequency. The user can tune the frequency by writing to the OSCTRM register. The tuning range of the FRC oscillator is ±1.5% of nominal in 3.75 kHz steps.

13.12 Secondary Oscillator

The Secondary Oscillator (SOSC) is a low-power 32.768 kHz crystal oscillator that provides accurate time keeping.

The Secondary Oscillator has the following features:

- 32.768 kHz operation
- Provides system clock output
- Provided to CRU or LPCLKGEN on request
- Can be disabled to reduce power
- Ultra-low power driver
- No calibration is required

13.13 Low Power RC Oscillator (LPRC)

The Low Power Internal RC Oscillator (LPRC) operates at a nominal frequency of 32.768 kHz.
Note: The LPRC is not a 50% duty cycle clock; however, it maintains an average frequency over a number of base clocks.

The LPRC can be used as both a source for the system clock and a reference for Backup core modules. These modules include the Deep Sleep Watchdog (DSWDT), clock monitor circuits and other modules that require a 32 kHz reference clock.

13.14 Reference Clock Generator

The Reference Clock Generator provides the Generic Clocks (GCLK_<Periperhal>) for system peripherals via Peripheral Channels. There are a total of 24 Peripheral Channels with the mapping as shown in following table.

Table 13-1. Peripheral Clock Generation

Peripheral Clock	Pchannel Index
GCLK_EIC, GCLK_CCL	0
GCLK_FREQM_MSR	1
GCLK_FREQM_REF	2
GCLK_SERCOM0_CORE, GCLK_SERCOM1_CORE	3
GCLK_SERCOM2_CORE, GCLK_SERCOM3_CORE	4

.....continued

Peripheral Clock	Pchannel Index
GCLK_TC0	5
GCLK_TC1	6
GCLK_TC2, GCLK_TC3	7
GCLK_EVSYS_CH_0	8
GCLK_EVSYS_CH_1	9
GCLK_EVSYS_CH_2	10
GCLK_EVSYS_CH_3	11
GCLK_EVSYS_CH_4	12
GCLK_EVSYS_CH_5	13
GCLK_EVSYS_CH_6	14
GCLK_EVSYS_CH_7	15
GCLK_EVSYS_CH_8	16
GCLK_EVSYS_CH_9	17
GCLK_EVSYS_CH_10	18
GCLK_EVSYS_CH_11	19
GCLK_TCC0	20
GCLK_TCC1, GCLK_TCC2	21
GCLK_AC	22
GCLK_CM4_TRACE	23

The mapping for the source of the clocks for both the CLKGEN generator and Reference clock generator are shown in following table.

Table 13-2. CRU Clock Mapping

clock_in[x]	MCS/COSC Mapping	REFO/ROSEL Mapping	FSCM Clock Source	Clock to Switch to on a FSCM Fail
0 - FRC	0000	0000	—	X
1 - SPLL_CLK1	0001	0001	—	—
2 - POSC (16 MHz)	0010	0010	—	—
3- SOSC	0011	0011	—	—
4 - LPRC	0100	0100	X	—
5 - SPLL_CLK3 (RFPLL, 96 MHz)	—	0101	—	—
6 - PB1_CLK	—	0110	—	—
7 - SYS_CLK	—	0111	—	—
8 - REFI Pin	—	1000	—	—

13.15 CRU Configuration Registers

The BASE address of the CRU registers is 0x4400_0A00. The Register Summary table shows the mapping of the registers in memory as well as the details of the bit fields in each register. Each register has an associated SET/CLR/INV function register with the suffix appended to the register name, for example: <reg>SET, <reg>CLR, <reg>INV.

13.16 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	OSCCON	7:0	CLKLOCK			SLPEN	CF		SOSCEN	OSWEN
		15:8	COSC[3:0]				NOSC[3:0]			
		23:16	DRMEN		2SPDSLP					
		31:24						FRCDIV[2:0]		
0x04	OSCTRM	7:0			TUN[5:0]					
		15:8								
		23:16								
		31:24								
0x08	SPLLCON	7:0			SPLLRST	SPLLFLOCK	SPLLPWDN			
		15:8	SPLL1POSTDIV1[7:0]							
		23:16					SPLL2POSTDIV2[3:0]			
		31:24	SPLL_BYP[1:0]							
0x0C	RCON	7:0	EXTR	SWR	DMTO	WDTO	SLEEP	IDLE	BOR	POR
		15:8						DPSLP	CMR	
		23:16								VBAT
		31:24	POR_IO	POR_CORE			BCFGERR	BCFGFAIL	NVMLTA	NVMEOL
0x10	RSWRST	7:0								SWRST
		15:8								
		23:16								
		31:24								
0x14	RNMICON	7:0	NMICNT[7:0]							
		15:8	NMICNT[15:8]							
		23:16	SWNMI				EXT	PLVD	CF	WDTS
		31:24							DMTO	WDTR
0x18 ... 0x1B	Reserved									
0x1C	REFO1CON	7:0					ROSEL3	ROSEL2	ROSEL1	ROSEL0
		15:8	ON	FRZ	SIDL	OE	RSLP		DIVSW_EN	ACTIVE
		23:16	RODIV7	RODIV6	RODIV5	RODIV4	RODIV3	RODIV2	RODIV1	RODIV0
		31:24		RODIV14	RODIV13	RODIV12	RODIV11	RODIV10	RODIV9	RODIV8
0x20	REFO1TRIM	7:0								
		15:8								
		23:16	ROTRIM0							
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1
0x24	REFO2CON	7:0					ROSEL3	ROSEL2	ROSEL1	ROSEL0
		15:8	ON	FRZ	SIDL	OE	RSLP		DIVSW_EN	ACTIVE
		23:16	RODIV7	RODIV6	RODIV5	RODIV4	RODIV3	RODIV2	RODIV1	RODIV0
		31:24		RODIV14	RODIV13	RODIV12	RODIV11	RODIV10	RODIV9	RODIV8
0x28	REFO2TRIM	7:0								
		15:8								
		23:16	ROTRIM0							
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1
0x2C	REFO3CON	7:0					ROSEL3	ROSEL2	ROSEL1	ROSEL0
		15:8	ON	FRZ	SIDL	OE	RSLP		DIVSW_EN	ACTIVE
		23:16	RODIV7	RODIV6	RODIV5	RODIV4	RODIV3	RODIV2	RODIV1	RODIV0
		31:24		RODIV14	RODIV13	RODIV12	RODIV11	RODIV10	RODIV9	RODIV8
0x30	REFO3TRIM	7:0								
		15:8								
		23:16	ROTRIM0							
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1
0x34	REFO4CON	7:0					ROSEL3	ROSEL2	ROSEL1	ROSEL0
		15:8	ON	FRZ	SIDL	OE	RSLP		DIVSW_EN	ACTIVE
		23:16	RODIV7	RODIV6	RODIV5	RODIV4	RODIV3	RODIV2	RODIV1	RODIV0
		31:24		RODIV14	RODIV13	RODIV12	RODIV11	RODIV10	RODIV9	RODIV8

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x38	REFO4TRIM	7:0								
		15:8								
		23:16	ROTRIM0							
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1
0x3C	REFO5CON	7:0					ROSEL3	ROSEL2	ROSEL1	ROSEL0
		15:8	ON	FRZ	SIDL	OE	RSLP		DIVSW_EN	ACTIVE
		23:16	RODIV7	RODIV6	RODIV5	RODIV4	RODIV3	RODIV2	RODIV1	RODIV0
		31:24		RODIV14	RODIV13	RODIV12	RODIV11	RODIV10	RODIV9	RODIV8
0x40	REFO5TRIM	7:0								
		15:8								
		23:16	ROTRIM0							
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1
0x44	REFO6CON	7:0					ROSEL3	ROSEL2	ROSEL1	ROSEL0
		15:8	ON	FRZ	SIDL	OE	RSLP		DIVSW_EN	ACTIVE
		23:16	RODIV7	RODIV6	RODIV5	RODIV4	RODIV3	RODIV2	RODIV1	RODIV0
		31:24		RODIV14	RODIV13	RODIV12	RODIV11	RODIV10	RODIV9	RODIV8
0x48	REFO6TRIM	7:0								
		15:8								
		23:16	ROTRIM0							
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1
0x4C	PB1DIV	7:0		PB1DIV[6:0]						
		15:8	PB1DIVON				PB1DIVRDY			
		23:16								
		31:24								
0x50	PB2DIV	7:0		PB1DIV[6:0]						
		15:8	PB1DIVON				PB1DIVRDY			
		23:16								
		31:24								
0x54	PB3DIV	7:0		PB1DIV[6:0]						
		15:8	PB1DIVON				PB1DIVRDY			
		23:16								
		31:24								
0x58	SLEWCON	7:0						SLW_UP	SLW_DN	SLW_BUSY
		15:8						SLW_DIV[2:0]		
		23:16					SYS_DIV[3:0]			
		31:24					SLW_DELAY[3:0]			
0x5C	CLKSTAT	7:0		SPLL3RDY		LPRCRDY	SOSCRDY	POSCRDY	SPLL1RDY	FRCRDY
		15:8								
		23:16								
		31:24								
0x60 ... 0x63	Reserved									
0x64	CLK_DIAG	7:0		SPLL3_STOP	SPLL2_STOP	SPLL1_STOP	LPRC_STOP	FRC_STOP	SOSC_STOP	POSC_STOP
		15:8								
		23:16	NMICTR7	NMICTR6	NMICTR5	NMICTR4	NMICTR3	NMICTR2	NMICTR1	NMICTR0
		31:24	NMICTR15	NMICTR14	NMICTR13	NMICTR12	NMICTR11	NMICTR10	NMICTR9	NMICTR8

13.17 Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the “PAC Write-Protection” property in each individual register description.

Some registers are synchronized when read and/or written. Synchronization is denoted by the “Write-Synchronized” or the “Read-Synchronized” property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable protection is denoted by the “Enable-Protected” property in each individual register description.

Note: All registers in this table have corresponding CLR, SET and INV registers at its virtual address, plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET, and INV Registers* from Related Links.

Related Links

6.1.9. CLR, SET and INV Registers

13.17.1 CRU Oscillator Control

Name: OSCCON
Offset: 0x00
Reset: 0x00000000

Note: The system unlock sequence must be done before this register can be written.

Bit	31	30	29	28	27	26	25	24
						FRCDIV[2:0]		
Access						R/W/L	R/W/L	R/W/L
Reset						0	0	0

Bit	23	22	21	20	19	18	17	16
	DRMEN		2SPDSLP					
Access	R/W/L		R/W/L					
Reset	0		1					

Bit	15	14	13	12	11	10	9	8
	COSC[3:0]				NOSC[3:0]			
Access	R	R	R	R	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLKLOCK			SLPEN	CF		SOSCEN	OSWEN
Access	R/W/L			R/W/L	R/W/HS/L		R/W/L	R/W/HC/L
Reset	0			0	0		1	1

Bits 26:24 – FRCDIV[2:0] Fast RC Clock Divider bits

Value	Description
000	FRC Divide by 1 (default value)
001	FRC Divide by 2
010	FRC Divide by 4
011	FRC Divide by 8
100	FRC Divide by 16
101	FRC Divide by 32
110	FRC Divide by 64
111	FRC Divide by 256

Bit 23 – DRMEN Enable the Dream Mode bit

Value	Description
1	When the cpu has executed WFI instruction and SLPEN = 1, peripheral clock requests are NOT active causes to enter the Sleep mode
0	DMA transfer has no effect

Bit 21 – 2SPDSLP 2-Speed Start-up enabled in the Sleep mode bit

Note: Default Reset Value is specified by cfg_two_speed_startup_en input.

Value	Description
1	When the device exits the Sleep Mode, the SYS_CLK will be from FRC until the selected clock is ready
0	When the device exits the Sleep Mode, the SYS_CLK will be from the selected clock

Bits 15:12 – COSC[3:0] Current Oscillator Selection bits (Read-only)

Notes:

- The default value on Reset is 4'b0000, which ensures that a virgin die has frc_clk running for ICDJTAG or EJTAG to program the NVR.
- Loaded with NOSC[3:0] at the completion of a successful clock switch.
- Set to FRC value (0000) when FSCM detects a failure and switches clock to FRC.

Value	Description
0000	Fast RC Oscillator (FRC) divided by OSCCON.FRCDIV
0001	System PLL Clock-1 (SPLL1 Module) (input clock and divider set by SPLLCON)
0010	Primary Oscillator (POSC)
0011	Secondary Oscillator (SOSC)
0100	Low Power RC Oscillator (LPRC)
0101-1111	Reserved for future use

Bits 11:8 – NOSC[3:0] New Oscillator Selection bits

Note: Default value on Reset is 4'b0000, which ensures that a virgin die has frc_clk running for ICDJTAG or EJTAG to program the NVR.

Value	Description
0000	Fast RC Oscillator (FRC) divided by OSCCON.FRCDIV
0001	System PLL Clock-1 (SPLL1 Module) (input clock and divider set by SPLLCON)
0010	Primary Oscillator (POSC)
0011	Secondary Oscillator (SOSC)
0100	Low Power RC Oscillator (LPRC)
0101-1111	Reserved for future use

Bit 7 – CLKLOCK Clock Lock Enabled bit

Notes:

- Once set, this bit can only be cleared via a Device Reset.
- When active, this bit prevents writes to the following registers: NOSC[3:0] and OSWEN.

Value	Description
1	All clock and PLL configuration registers are locked. These include OSCCON, OSCTRIM, SPLLCON, UPLLCON, PBxDIV
0	Clock and PLL selection registers are not locked, configurations may be modified.

Bit 4 – SLPEN Enable the Sleep Mode bit

Value	Description
1	When a WAIT Instruction is executed device will enter SLEEP Mode
0	When a WAIT instruction is executed device will enter IDLE Mode

Bit 3 – CF Clock Fail Detect bit (Read/writable/Clearable by application)

Notes:

- Writing a '1' to this bit will cause a clock switching sequence to be initiated by the clock switch state machine
- Resets when a valid clock switching sequence is initiated by the clock switch state machine
- This bit is set when clock fail event is detected

Value	Description
1	FSCM has detected clock failure
0	FSCM has not detected clock failure

Bit 1 – SOSCEN 32 kHz Secondary (LP) Oscillator Enable bit

Value	Description
1	Enable Secondary Oscillator

Value	Description
0	Disable Secondary Oscillator

Bit 0 – OSWEN Oscillator Switch Enable bit

Notes:

- A Write of value '0' has no effect.
- Cleared by hardware after a successful clock switch
- Cleared by hardware after a redundant clock switch (NOSC = COSC)
- Cleared by hardware after FSCM switches the oscillator to Fail-Safe Clock Source (FRC)

Value	Description
1	Request oscillator switch to selection specified by NOSC[3:0] bits
0	Oscillator switch is complete

13.17.2 CRU Oscillator Trimming

Name: OSCTRM
Offset: 0x04
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			TUN[5:0]					
Access			R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset			0	0	0	0	0	0

Bits 5:0 – TUN[5:0] Internal Fast RC (FRC) Oscillator Tuning bits
This bit field specifies the user-tuning capability for the internal fast RC oscillator.
Note: The system unlock sequence must be done before this register can be written.

Value	Description
011111	Maximum Frequency
011110	
...	
000001	
000000	Center Frequency, oscillator is running at calibrated frequency
111111	
111110	
...	
100001	
100000	Minimum Frequency

13.17.3 SPLL (RFPLL/Wrapper) Control

Name: SPLLCON
Offset: 0x08
Reset: 0x00000000

Note: The system unlock sequence must be done before these registers can be written.

Bit	31	30	29	28	27	26	25	24
	SPLL_BYP[1:0]							
Access	R/W/L	R/W/L						
Reset	1	1						

Bit	23	22	21	20	19	18	17	16
					SPLL2POSTDIV2[3:0]			
Access					R/W/L	R/W/L	R/W/L	R/W/L
Reset					0	0	0	1

Bit	15	14	13	12	11	10	9	8
	SPLL1POSTDIV1[7:0]							
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			SPLLRST	SPLLFLOCK	SPLLPWDN			
Access			R/W/L	R/W/L	R/W/L			
Reset			1	0	1			

Bits 31:30 – SPLL_BYP[1:0] SPLL Bypass; when this bit is set, the input clock REF bypasses PLL to PLLOUTx.

Notes:

- Dictates clock source for ADC CP (Analog-to-Digital Converter Charge Pump) (SPLL2) Clock generation only
- Clock source must be preselected and kept ready before the need of ADC CP arrives. Failure to do so will result in the loss of clock for one or two cycles when ADC CP is enabled.

Value	Description
00	RFPLL Clock is the clock source for ADC CP clock generation.
x1	FRC is used as clock source for ADC CP clock generation.
10	POSC is used as clock source for ADC CP clock generation.

Bits 19:16 – SPLL2POSTDIV2[3:0] ADC-CP Post Divide Value

Value	Description
1 ≤ SPLLPOSTDIV2 ≤ 15	Divide-by SPLLPOSTDIV2
0	No Clock; Clock disabled

Bits 15:8 – SPLL1POSTDIV1[7:0] First Post Divide Value

Value	Description
2 ≤ SPLLPOSTDIV ≤ 255	Divide-by SPLLPOSTDIV
0	Divide-by 1
1	Divide-by 1.5

Bit 5 – SPLLRST System PLL Reset

Value	Description
1	Assert the Reset to the SPLL
0	De-assert the Reset to the SPLL

Bit 4 – SPLLFLOCK System PLL Force Lock

Value	Description
1	Force the SPLL lock signal to be asserted
0	Do not force the SPLL lock signal to be asserted

Bit 3 – SPLLPWDN PLL Power Down Register bit

Value	Description
1	PLL is powered down
0	PLL is active

13.17.4 Reference Oscillator x Control

Name: REFOxCON
Offset: 0x1C + (x-1)*0x08 [x=1..6]
Reset: 0x00000000

Notes:

- REFOCON.ROSEL must not be written while the REFOCON.ACTIVE bit is '1' – This will result in undefined behavior.
- REFOCON must not be written when REFOCON[ON] != REFOCON[ACTIVE] – This will result in undefined behavior.
- This register can always be accessed regardless of the cfg_sys_unlock value.

Bit	31	30	29	28	27	26	25	24
		RODIV14	RODIV13	RODIV12	RODIV11	RODIV10	RODIV9	RODIV8
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RODIV7	RODIV6	RODIV5	RODIV4	RODIV3	RODIV2	RODIV1	RODIV0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON	FRZ	SIDL	OE	RSLP		DIVSW_EN	ACTIVE
Access	R/W	R/W	R/W	R/W	R/W		HC/ R/W	HS/HC/ R
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
					ROSEL3	ROSEL2	ROSEL1	ROSEL0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30 – RODIV Reference Clock Divider bits
Specifies 1/2 period of reference clock in the source clocks.
Example: Period of refo_clk = [Reference source * 2] * RODIV

Value	Description
111111111111111	REFOx clock is Base clock frequency divided by 65,534 (32,767 *2)
111111111111110	REFOx clock is Base clock frequency divided by 65,532 (32,766 * 2)
...	
...	
...	
000000000000011	REFOx clock is Base clock frequency divided by 6 (3*2)
000000000000010	REFOx clock is Base clock frequency divided by 4 (2*2)
000000000000001	REFOx clock is Base clock frequency divided by 2 (1*2)
000000000000000	REFOx clock is same frequency as Base clock (no divider)

Bit 15 – ON Output Enable bit

Value	Description
1	Reference Oscillator Module is enabled
0	Reference Oscillator Module is disabled

Bit 14 – FRZ Freeze in Debug mode bit

Value	Description
1	When the emulator is in the Debug mode, module freezes operation

Value	Description
0	When the emulator is in the Debug mode, module continues operation

Bit 13 – SIDL Peripheral Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when device enters the Idle mode
0	Continues module operation in the Idle mode

Bit 12 – OE Reference Clock Output Enable bit

Value	Description
1	Reference clock is driven out on REFOx pin
0	Reference clock is not driven out on REFOx pin

Bit 11 – RSLP Reference Oscillator Run in Sleep bit

Note: This bit is ignored when ROSEL[3:0] = (0000 or 0001).

Value	Description
1	Reference Oscillator output continues to run in the Sleep mode
0	Reference Oscillator output is disabled in the Sleep mode

Bit 9 – DIVSW_EN Clock RODIV/ROTRIM switch enabled

Value	Description
1	Clock Divider Switching is currently in progress
0	Clock Divider Switch has completed

Bit 8 – ACTIVE Reference Clock Request Status bit

Value	Description
1	Reference clock request is active (User must not update this REFOCON register)
0	Reference clock request is not active (User can update this REFOCON register)

Bits 0, 1, 2, 3 – ROSEL Reference Clock Source Select bits

Select one of various clock sources to be used as the reference clock.

Value	Description
1001-1111	Reserved
1000	REFI pin
0111	System clock (reference clock reflects any device clock switching)
0110	Peripheral clock (reference clock reflects any peripheral clock switching)
0101	System PLL (Clock-3)
0100	LPRC
0011	SOSC
0010	POSC
0001	System PLL (Clock-1)
0000	FRC

13.17.5 Reference Oscillator x Trim

Name: REFOxTRIM
Offset: 0x20 + (x-1)*0x08 [x=1..6]
Reset: 0x00000000

Notes:

- REFOxTRIM must not be written when REFOxCON[ON] != REFOxCON[ACTIVE] – This will result in undefined behavior.
- This register can always be accessed regardless of the cfg_sys_unlock value.

Bit	31	30	29	28	27	26	25	24
	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ROTRIM0							
Access	R/W							
Reset	0							

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 23, 24, 25, 26, 27, 28, 29, 30, 31 – ROTRIM Trim bits – Provides fractional additive to RODIV value for 1/2 period of REFOx clock

Note: ROTRIM values greater than zero are only valid when RODIV values are greater than 0.

Value	Description
0000_0000_0	0/512 (0.0) divisor added to RODIV value
0000_0000_1	1/512 (0.001953125) divisor added to RODIV value
0000_0001_0	2/512 (0.00390625) divisor added to RODIV value
...	...
...	...
100000000	256/512 (0.5000) divisor added to RODIV value
...	...
...	...
1111_1111_0	510/512 (0.99609375) divisor added to RODIV value
1111_1111_1	511/512 (0.998046875) divisor added to RODIV value

13.17.6 PBx Clock Divisor Control

Name: PBxDIV
Offset: 0x4C + (x-1)*0x04 [x=1..3]
Reset: 0x00000000

Note: The system unlock sequence must be done before this register can be written. PBx registers include PB1DIV, PB2DIV and PB3DIV. Ensure the PB3DIV[6:0] value is equal to 0x09 or greater if the user is not using Microchip-provided boot code.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PB1DIVON				PB1DIVRDY			
Access	R				R			
Reset	1				1			

Bit	7	6	5	4	3	2	1	0
		PB1DIV[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	1

Bit 15 – PB1DIVON Output Enable bit

Value	Description
1	PB1 Output clock is enabled
0	PB1 Output clock is disabled **Note:** PB1DIV[PB1DIVON] bit cannot be written to a '0', as this clock is used by the system CLK_RST macro.

Bit 11 – PB1DIVRDY PB1 Peripheral Clock Divisor Ready

Value	Description
1	Indicates the PB clock divisor logic is not switching divisors and the PB1DIV may be written.
0	Indicates the PB clock divisor logic is currently switching values and the PB1DIV cannot be written.

Bits 6:0 – PB1DIV[6:0] PB1 Peripheral Clock Divisor Control value

Value	Description
000_0000	Divide by 1 PB1 Clock same frequency as SYS_CLK
000_0001	Divide by 2 PB1 Clock is 1/2 of SYS_CLK
000_0010	Divide by 3 PB1 Clock is 1/3 of SYS_CLK
000_0011	Divide by 4 PB1 Clock is 1/4 of SYS_CLK
...	...
...	...
000_1111	Divide by 16 PB1 Clock is 1/16 of SYS_CLK
001_0000	Divide by 17 PB1 Clock is 1/17 of SYS_CLK
...	...
...	...
111_1110	Divide by 127 PB1 Clock is 1/127 of SYS_CLK
111_1111	Divide by 128 PB1 Clock is 1/128 of SYS_CLK

13.17.7 Slew Rate Control for Clock Switching

Name: SLEWCON
Offset: 0x58
Reset: 0x00000000

Notes:
- The system unlock sequence must be done before this register can be written.
- Updates to this register do not take effect until OSCCON[OSWEN] is set.

Bit	31	30	29	28	27	26	25	24
					SLW_DELAY[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					c	c	c	c

Bit	23	22	21	20	19	18	17	16
					SYS_DIV[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					c	c	c	c

Bit	15	14	13	12	11	10	9	8
						SLW_DIV[2:0]		
Access						R/W	R/W	R/W
Reset						c	c	c

Bit	7	6	5	4	3	2	1	0
						SLW_UP	SLW_DN	SLW_BUSY
Access						R/W	R/W	R/W
Reset						c	c	c

Bits 27:24 – SLW_DELAY[3:0] Number of clocks generated at each slew step for a clock switch
Note: The reset value of this register field is defined by the input cfg_slewcon_sel[].

Value	Description
0000	1 clock will be generated at each slew step
0001	2 clocks will be generated at each slew step
...	...
1111	16 clocks will be generated at each slew step

Bits 19:16 – SYS_DIV[3:0] PBx Peripheral Clock Divisor Control value

Value	Description
0000	Divide by 1 – SYS_CLK_OUT same frequency as SYS_CLK source - Default
0001	Divide by 2 – SYS_CLK_OUT is 1/2 of SYS_CLK source
0010	Divide by 3 – SYS_CLK_OUT is 1/3 of SYS_CLK source
...	...
1111	Divide by 16 – SYS_CLK_OUT is 1/16 of SYS_CLK source

Bits 10:8 – SLW_DIV[2:0] Divisor steps used when doing slewed clock switches
Note: Each Divisor step lasts four clocks

Value	Description
000	No divisor is used
001	Divide by 2 (2^1), then no divisor
010	Divide by 4 (2^2), then by 2, then no divisor
011	Divide by 8 (2^3), then by 4, then by 2, then no divisor
100	Divide by 16 (2^4), then by 8, then by 4, then by 2, then no divisor
...	...

Value	Description
111	Divide by 128 (2^7), then by 64, then by 32, then by 16, then by 8, then by 4, then by 2, then no divisor

Bit 2 – SLW_UP Clock slew enable for switching up to faster clocks

Value	Description
0	Clock Slewing is disabled
1	Clock Slewing is enabled on a clock switch OR exit from Sleep

Bit 1 – SLW_DN Clock slew enable for switching down to slower clocks

Value	Description
0	Clock Slewing is disabled
1	Clock Slewing is enabled on a clock switch

Bit 0 – SLW_BUSY Clock Switch Slewing Active Status Bit – Read-Only

Value	Description
0	Clock Switch has reached its final value
1	Clock frequency is being actively Slewed

13.17.8 Clock Status

Name: CLKSTAT
Offset: 0x5C
Reset: 0x00000000

Note: The corresponding RDY bits are updated only after the clock switch request is initiated via OSCCON.NOSC[3:0].

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		SPLL3RDY		LPRCRDY	SOSCRDY	POSCRDY	SPLL1RDY	FRCRDY
Access		R/HS/HC		R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset		0		0	0	0	0	0

Bit 6 – SPLL3RDY System PLL (Clock-3) Ready Status value

Value	Description
1	SPLL3 is stable and ready
0	SPLL3 is not stable and not ready

Bit 4 – LPRCRDY LPRC Ready Status value

Value	Description
1	LPRC is stable and ready
0	LPRC is not stable and not ready

Bit 3 – SOSCRDY SOSC Ready Status value

Value	Description
1	SOSC is stable and ready
0	SOSC is not stable and not ready

Bit 2 – POSCRDY Primary Oscillator Ready Status value

Value	Description
1	POSC is stable and ready
0	POSC is not stable and not ready

Bit 1 – SPLL1RDY System PLL (Clock-1) Ready Status value

Value	Description
1	SPLL1 is stable and ready
0	SPLL1 is not stable and not ready

Bit 0 – FRCRDY FRC Ready Status value

Value	Description
1	FRC is stable and ready

Value	Description
0	FRC is not stable and not ready

13.17.9 User Clock Diagnostics Control

Name: CLK_DIAG
Offset: 0x64
Reset: 0x00000000

Note: The system unlock sequence must be done before this register can be written.

Bit	31	30	29	28	27	26	25	24
	NMICTR15	NMICTR14	NMICTR13	NMICTR12	NMICTR11	NMICTR10	NMICTR9	NMICTR8
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	NMICTR7	NMICTR6	NMICTR5	NMICTR4	NMICTR3	NMICTR2	NMICTR1	NMICTR0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		SPLL3_STOP	SPLL2_STOP	SPLL1_STOP	LPRC_STOP	FRC_STOP	SOSC_STOP	POSC_STOP
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 – NMICTR Internal value of internal NMI Counter
This field reflects the actual value of the internal NMI counter

Bit 6 – SPLL3_STOP SPLL Clock Stop Control value
Note: Gating logic is outside of this macro

Value	Description
0	SPLL3 clock source runs as normal
1	SPLL3 clock source is stopped

Bit 5 – SPLL2_STOP SPLL Clock Stop Control value
Note: Gating logic is outside of this macro

Value	Description
0	SPLL2 clock source runs as normal
1	SPLL2 clock source is stopped

Bit 4 – SPLL1_STOP SPLL Clock Stop Control value
Note: Gating logic is outside of this macro

Value	Description
0	SPLL1 clock source runs as normal
1	SPLL1 clock source is stopped

Bit 3 – LPRC_STOP LPRC Clock Stop Control value
Note: Gating logic is outside of this macro

Value	Description
0	LPRC clock source runs as normal
1	LPRC clock source is stopped

Bit 2 – FRC_STOP FRC Clock Stop Control value
Note: Gating logic is outside of this macro

Value	Description
0	FRC clock source runs as normal
1	FRC clock source is stopped

Bit 1 – SOSC_STOP SOSC Clock Stop Control value
Note: Gating logic is outside of this macro

Value	Description
0	SOSC clock source runs as normal
1	SOSC clock source is stopped

Bit 0 – POSC_STOP POSC Clock Stop Control value
Note: Gating logic is outside of this macro

Value	Description
0	POSC clock source runs as normal
1	POSC clock source is stopped

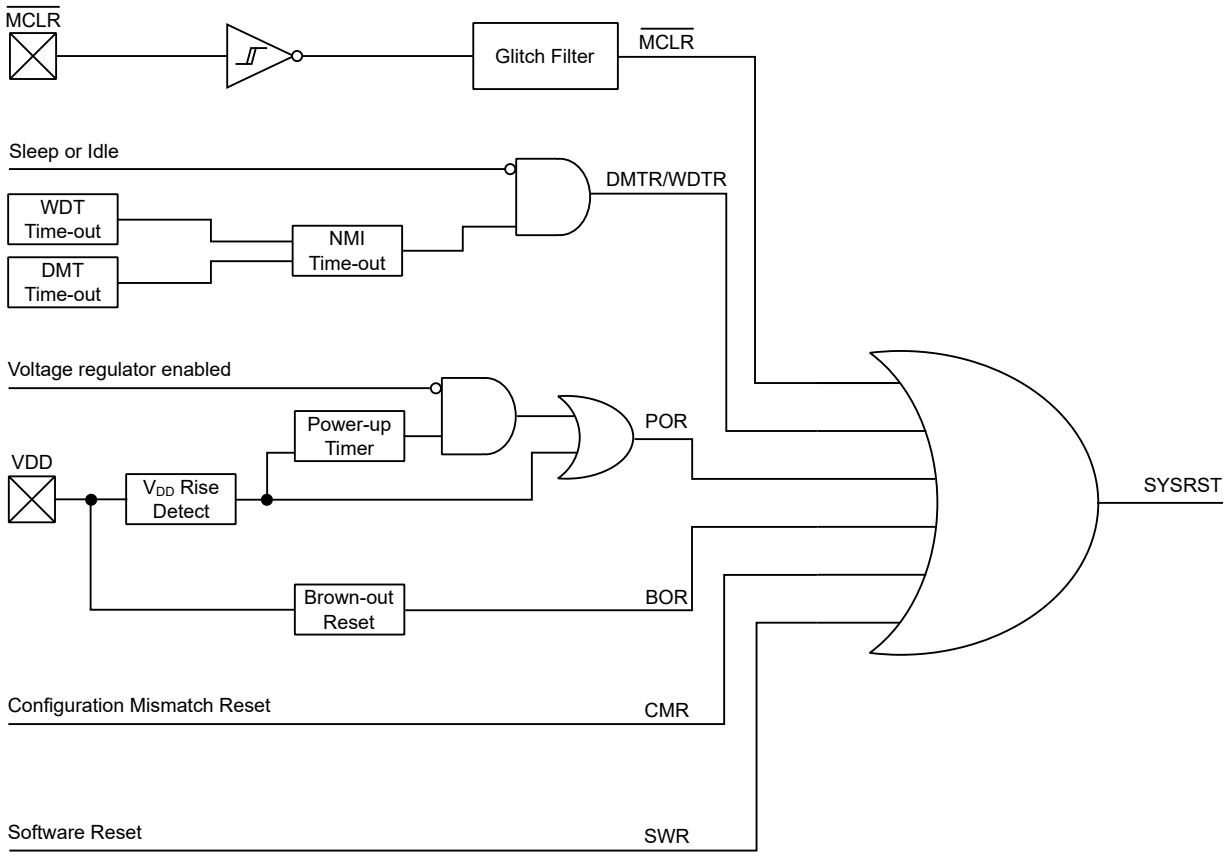
13.18 Resets

The Reset module combines all Reset sources and controls the device Master Reset signal, SYSRST. The device Reset sources are as follows:

- Power-on Reset (V_{dd}, IO, or POR)
- Brown-out Reset (BOR/ZPBOR)
- Master Clear Reset ($\overline{MCLR}$)
- Watchdog Timer Reset (NMI Counter)
- Dead Man Timer Reset (NMI Counter)
- Software Reset (SWR)
- Test mode Entry and Exit
- JTAG Reset
- Configuration Mismatch Reset (CMR)

A simplified block diagram of the Reset module is shown in the following figure. Any active source of reset will make the system Reset signal active. Many registers associated with the CPU and peripherals are forced to a known Reset state.

Figure 13-4. System Reset Block Diagram



13.18.1 Control Registers

Most types of device resets will set corresponding Status bits in the RCON register to indicate the type of Reset (see *RCON* register from Related Links). The one exception is the Non-maskable Interrupt (NMI) time-out Reset. A Power-on Reset (POR) will clear all bits, except for the BOR and POR bits (RCON[1:0]), which are set. The user software may set or clear any of the bits at any time during code execution. The RCON bits serve only as Status bits. Setting a Reset status bit in software will not cause a system Reset to occur.

The RCON register also has other bits associated with the Watchdog Timer (WDT) and device power-saving states. For more information on the function of these bits, see *Using the RCON Status Bits* from Related Links.

The RSWRST control register has only one bit, SWRST. This bit is used to force a software Reset condition.

A delay equal to the duration of the number of NMICNT system clocks begins as it is decremented to zero. During this interval, the program can clear the WDT or DMT flag bits, if desired, to avoid a Reset. If the active flag is not cleared, the device will be reset at the end of the interval. The NMICNT value can be set to zero for no delay and up to 255 SYSCLK cycles.

The NMI interrupt can also be triggered by setting the SWNMI bit in software or if the CF bit is set by the FSCM, but these do not begin the countdown and do not automatically lead to a reset.

The Resets module consists of the following Special Function Registers (SFRs):

- RCON: Reset Control Register
- RSWRST: Software Reset Register
- RNMICON: Non-Maskable Interrupt (NMI) Control Register

The base address of these registers is 0x4400_0A00. The offset for each register is shown in Reset Regsiter Map (see *Reset Register Map* in the *Reset Control Registers* from Related Links). Multiply the address offset in the table by 4. The other three addresses represent the CLR/SET/INV bitwise registers.

Related Links

13.18.2.1. RCON
13.18.3. Reset Control Registers
13.18.5.3. Using the RCON Status Bits

13.18.2 Reset Register Summary

The following registers provide a brief summary of the Reset Control registers.

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00 ... 0x0B	Reserved									
0x0C	RCON	7:0	EXTR	SWR	DMTO	WDTO	SLEEP	IDLE	BOR	POR
		15:8						DPSLP	CMR	
		23:16								VBAT
		31:24	POR_IO	POR_CORE			BCFGERR	BCFGFAIL	NVMLTA	NVMEOL
0x10	RSWRST	7:0								SWRST
		15:8								
		23:16								
		31:24								
0x14	RNMICON	7:0	NMICNT[7:0]							
		15:8	NMICNT[15:8]							
		23:16	SWNMI				EXT	PLVD	CF	WDTS
		31:24							DMTO	WDTR

13.18.2.1 Reset Control Register

Name: RCON
Offset: 0xC
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	POR_IO	POR_CORE			BCFGERR	BCFGFAIL	NVMLTA	NVMEOL
Access	R/W/HS	R/W/HS			R/W/HS	R/W/HS	R/W/HS	R/W/HS
Reset	0	0			0	0	0	0

Bit	23	22	21	20	19	18	17	16
								VBAT
Access								R/W/HS
Reset								0

Bit	15	14	13	12	11	10	9	8
						DPSLP	CMR	
Access						R/W/HS	R/W/HS	
Reset						0	0	

Bit	7	6	5	4	3	2	1	0
	EXTR	SWR	DMTO	WDTO	SLEEP	IDLE	BOR	POR
Access	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS
Reset	0	0	0	0	0	0	0	0

Bit 31 – POR_IO I/O Voltage POR Flag bit
Set by hardware at detection of an I/O POR event. User software must clear this bit to view next detection.
Note: User may write this bit to '1'. Does not cause a POR_IO.

Value	Description
1	A Power-on Reset has occurred due to I/O voltage
0	A Power-on Reset has not occurred due to I/O voltage

Bit 30 – POR_CORE Core Voltage POR Flag bit
Set by hardware at detection of a core POR event. User software must clear this bit to view the next detection.
Note: User may write this bit to '1'. Does not cause a POR_CORE.

Value	Description
1	A Power-on Reset has occurred due to I/O voltage
0	A Power-on Reset has not occurred due to I/O voltage

Bit 27 – BCFGERR BCFG Error Flag bit
A primary BCFG value had an error, but the secondary BCFG value was valid and used.

Value	Description
1	A BCFG error has occurred
0	A BCFG error has not occurred

Bit 26 – BCFGFAIL BCFG Failure Flag bit
Both the Primary and Secondary BCFG values had an unrecoverable error. Default values are in effect.

Value	Description
1	A BCFG error has occurred
0	A BCFG error has not occurred

Bit 25 – NVMLTA NVM Life Time Alert Flag bit
NVM Life Time Alert – Due to charge leakage, the NVM is nearing EOL.

Value	Description
1	A NVM LTA error has occurred
0	A NVM LTA error has not occurred

Bit 24 – NVMEOL NVM End of Life Flag bit

NVM End of Life – may not be visible to user, because the part will not come out of Reset if the bit is asserted.

Value	Description
1	A NVM EOL failure has occurred
0	A NVM EOL failure has not occurred

Bit 16 – VBAT VBAT Mode Flag bit

Value	Description
1	A POR exit from VBAT has occurred. A true POR must be established with the valid VBAT voltage level on the VBAT pin.
0	A POR exit from VBAT has not occurred.

Bit 10 – DPSLP Deep Sleep Mode Flag bit

Set by hardware at time of entry into Deep Sleep mode. User software must clear this bit to view next detection.

Value	Description
1	Deep Sleep mode has occurred
0	Deep Sleep mode has not occurred

Bit 9 – CMR Configuration Mismatch Reset Flag bit

Note: User may write this bit to '1'. Does not cause a Mismatch Reset.

Value	Description
1	A CMR event has occurred
0	A CMR event has not occurred

Bit 7 – EXTR External Reset ($\overline{MCLR}$) Status bit

Note: User may write this bit to '1'. Does not cause a ($\overline{MCLR}$).

Value	Description
1	A Master Clear (pin) Reset has occurred
0	A Master Clear (pin) Reset not occurred

Bit 6 – SWR Software Reset Flag bit

Note: User may write this bit to '1'. Does not cause SWR.

Value	Description
1	A SWR has occurred
0	A SWR not occurred

Bit 5 – DMTO Deadman Timer Time-out Flag bit

Note: User may write this bit to '1'. Does not cause DMT Reset.

Value	Description
1	DMT Time-out has occurred and caused a Reset
0	DMT Time-out has not occurred

Bit 4 – WDTO Watchdog Timer Time-out Flag bit

Note: User may write this bit to '1'. Does not cause WDT Reset.

Value	Description
1	WDT Time-out has occurred and caused a Reset
0	WDT Time-out has not occurred

Bit 3 – SLEEP Wake from Sleep Flag bit

Note: User may write this bit to '1'. Does not invoke Sleep mode.

Value	Description
1	Device has been in Sleep mode
0	Device has not been in Sleep mode

Bit 2 – IDLE Wake from Idle Flag bit

Note: User may write this bit to '1'. Does not invoke Idle mode.

Value	Description
1	Device was in the Idle mode
0	Device was not in the Idle mode

Bit 1 – BOR BOR Flag bit

Set by hardware at detection of a BOR event. User software must clear this bit to view next detection.

Note: User may write this bit to '1'. Does not cause a BOR.

Value	Description
1	A BOR has occurred
0	A BOR has not occurred

Bit 0 – POR POR Flag bit

Set by hardware at detection of a POR event. User software must clear this bit to view next detection.

Note: User may write this bit to '1'. Does not cause a POR.

Value	Description
1	A Power-on Reset has occurred
0	A Power-on Reset has not occurred

13.18.2.2 Software Reset Register

Name: RSWRST
Offset: 0x10
Reset: 0x00000000
Property: -

Note: The system unlock sequence must be done before this register can be written.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								SWRST
Access								W/HC
Reset								0

Bit 0 – SWRST Software Reset Trigger bit
'1' = Enable SWR event. A subsequent read of this register triggers the system Reset sequence. The system unlock sequence must be done before the bit can be written. This bit always reads a value of logic '0'.

13.18.2.3 NMI Control Register

Name: RNMICON
Offset: 0x14
Reset: 0x00000000
Property: -

Note: The system unlock sequence must be done before this register can be written.

Bit	31	30	29	28	27	26	25	24
							DMTO	WDTR
Access							R/W	R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
	SWNMI				EXT	PLVD	CF	WDTS
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NMICNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bit 25 – DMTO Deadman Timer Time-out Flag bit (This will cause a Reset when NMICNT expires.)
Note: User may write this bit to '1'. Causes a user-initiated DMT NMI event and NMICNT start.

Value	Description
1	DMT Time-out has occurred and caused a NMI
0	DMT Time-out has not occurred

Bit 24 – WDTR Watchdog Timer Time-out in Run Flag bit
Note: User may write this bit to '1'. Causes a user-initiated WDT NMI event and NMICNT start.

Value	Description
1	WDT Time-out has occurred and caused an NMI (This may cause a Reset if NMICNT expires.)
0	WDT Time-out has not occurred

Bit 23 – SWNMI Software NMI Trigger bit

Value	Description
1	Writing a '1' to this bit will cause an NMI to be generated
0	Writing a '0' to this bit will have no effect

Bit 19 – EXT External / Generic NMI Event bit
Note: User may write this bit to '1'. Causes a user-initiated EXT NMI event.

Value	Description
1	A general NMI event was detected and caused an NMI. Writing '0' to this bit will clear the NMI event
0	A general NMI event was not detected

Bit 18 – PLVD Programmable Low Voltage Detect Event bit
Note: User may write this bit to '1'. Causes a user-initiated PLVD NMI event.

Value	Description
1	PLVD detected a low voltage condition and caused an NMI
0	PLVD did not detect a low voltage condition

Bit 17 – CF Clock Fail Detect bit (Read/Clear-able by application)

Note: Writing a '1' to the CF bit will cause a user-initiated clock failure NMI event, but will not actually cause a clock switch.

Value	Description
1	FSCM detected clock failure and caused an NMI
0	FSCM did not detect a clock failure

Bit 16 – WDTS Watch-Dog Timer Time-out in Sleep Flag bit

Note: User may write this bit to '1'. Causes a user-initiated WDT NMI event.

Value	Description
1	WDT Time-out has occurred during Sleep mode and caused a wake-up from sleep
0	WDT Time-out has not occurred during Sleep mode

Bits 15:0 – NMICNT[15:0] NMI Reset counter value bit

This bit field specifies the reload value used by the NMI Reset counter.

```
0000_0000_0000_0000 = No delay between NMI assertion and device Reset event
0000_0000_0000_0001
0000_0000_0000_0010
.................
.................
.................
1111_1111_1111_1110
1111_1111_1111_1111 = Number of SYSCLK cycles that Software has to clear the NMI event before
a device Reset is performed. If the NMI event is cleared before the counter reached zero,
then NO device Reset is asserted.
```

Note: When a WDT NMI event occurs (when not in the Sleep mode), the NMICNT starts incrementing from the zero NMICNT value. When a DMT NMI event is triggered, the NMICNT starts decrementing. When NMICNT reaches zero, the device is Reset. This NMI reset counter is only applicable to these two specific NMI events.

13.18.3 Reset Control Registers

Table 13-3. Reset Register Map

	Register	Bit Range	Bits																All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
0XC	RCON	31:16	POR_IO	POR_CORE	—	—	BCFGERR	BCFGFAIL	NVMLTA	NVMEOL	—	—	—	—	—	—	—	VBAT	0000
		15:0	—	—	—	—	—	DPSLP	CMR	—	EXTR	SWR	DMTO	WDTO	SLEEP	IDLE	BOR	POR	0000
0X10	RSWRST	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	SWRST	0000
0X14	RNMICON	31:16	—	—	—	—	—	—	DMTO	WDTR	SWNMI	—	—	—	EXT	PLVD	CF	WDTS	0000
		15:0	NMICNT[15:0]																0000

13.18.4 Modes of Operation

13.18.4.1 System Reset (SYSRST)

The internal System Reset (SYSRST) can be generated from multiple Reset sources, such as:

- Power-on Reset (POR)
- Brown-out Reset (BOR/ZPBOR)
- Master Clear Reset ($\overline{\text{MCLR}}$)
- Watchdog Time-out Reset (WDTO)
- Deadman Timer Reset (DMTR)
- Software Reset (SWR)
- Configuration Mismatch Reset (CMR)
- Test Mode Entry and Exit Reset
- JTAG Reset

A system Reset is active at the first POR and asserted until device configuration settings are loaded and the oscillator clock sources become stable. The system Reset is then de-asserted, allowing the CPU to start fetching code after eight system clock cycles (SYSCLK). On any device Reset, no start-up time is required to transfer configuration values from the NVR memory into the configuration-holding registers. Once the device is active, the user may change the primary system clock source from FRC to SPLL by using the OSCCON register.

13.18.4.2 Power-on Reset (POR)

A power-on event generates an internal POR pulse when a VDD rise is detected above VPOR. The device supply voltage characteristics must meet the specified starting voltage and rise rate requirements to generate the POR pulse. In particular, VDD must fall below VPOR before a new POR is initiated. For more information on the VPOR and VDD rise-rate specifications, see *Electrical Characteristics* from Related Links.

This device has an on-chip internal voltage regulator and its power-on delay is designated as TPU. For more information on the TPU specification, see *Electrical Characteristics* from Related Links.

At this point, the POR event has expired but the device Reset is still asserted while the device configuration settings are loaded and the clock oscillator sources are configured. The clock monitoring circuitry waits for the oscillator source to become stable. The clock source of this device when exiting from Reset is always FRC (NOSC[3:0] bits (OSCCON[11:8]).

After these delays expire, the system Reset, SYSRST, is de-asserted. Before allowing the CPU to start code execution, eight system clock cycles are required before the synchronized Reset to the CPU core is de-asserted.

The power-on event sets the BOR and POR status bits (RCON[1:0]).

For more information on the values of the delay parameters, see *Electrical Characteristics* from Related Links.

Note: When the device exits the Reset condition (begins normal operation), the device operating parameters (voltage, frequency, temperature and so on) must be within their operating ranges; otherwise, the device will not function correctly. The user software must ensure that the delay between the time power is first applied and the time the system Reset is released is adequate to get all the operating parameters within the specification.

Related Links

43. Electrical Characteristics

13.18.4.3 Master Clear Reset

Whenever the master clear pin ($\overline{\text{MCLR}}$) is driven low, the Reset event is synchronized with the system clock, SYSCLK, before asserting the system Reset, SYSRST, provided the input pulse on $\overline{\text{MCLR}}$ is longer than a certain minimum width, as specified in the *Electrical Specifications*.

The $\overline{\text{MCLR}}$ pin provides a filter to minimize the effects of noise and to avoid unwanted Reset events. The status bit, EXTR (RCON[7]), is set to indicate the $\overline{\text{MCLR}}$ Reset.

The $\overline{\text{MCLR}}$ pin can be configured to generate a POR event, rather than a normal SYSRST event. This is configured through the SMCLR bit (CFGCON1[14]).

13.18.4.4 Software Reset (SWR)

This device does not provide a specific `RESET` instruction; however, a hardware Reset can be performed in software (software Reset) by executing a software Reset command sequence. The software Reset acts like a $\overline{\text{MCLR}}$ Reset. The software Reset sequence requires the system unlock sequence to be executed before the SWRST bit (RSWRST[0]) can be written.

A software Reset is performed as follows:

1. Write the system unlock sequence.
2. Set the SWRST bit (RSWRST[0]) = `1`.
3. Read the RSWRST register.

Setting the SWRST bit (RSWRST[0]) will arm the software Reset. The subsequent read of the RSWRST register triggers the software Reset, which must occur on the next clock cycle following the read operation. To ensure no other user code is executed before the Reset event occurs, it is recommended that four `NOP` instructions or a `while(1)` statement be placed after the `READ` instruction.

The SWR Status bit (RCON[6]) is set to indicate the software Reset.

13.18.4.5 Watchdog Timer Reset (WDTR)

A Watchdog Timer (WDT) Reset event is synchronized with the system clock (SYSCLK) before asserting the system Reset.

Note: A WDT time-out during the Sleep or Idle mode will wake-up the processor and branch to the reset vector, but it does not Reset the processor.

The only bits affected are WDTO and SLEEP or IDLE in the RCON register. See *Clock and Reset Unit (CRU)* from Related Links for more information on the WDT Reset.

Related Links

13. Clock and Reset Unit (CRU)

13.18.4.6 Brown-out Reset (BOR)

This device has a simple Brown-out Reset (BOR) capability. If the voltage supplied to the regulator is inadequate to maintain a regulated level, the regulator Reset circuitry will generate a BOR event, which is synchronized with the system clock, SYSCLK, before asserting the system Reset. This event is captured by the BOR flag bit (RCON[1]), see *Electrical Characteristics* from Related Links.

Related Links

43. Electrical Characteristics

13.18.4.7 Configuration Mismatch Reset (CMR)

To maintain the integrity of the stored configuration values, all device Configuration bits are loaded and implemented as a complementary set of bits. As the Configuration Words are being loaded, for each bit loaded as '`1`', a complementary value of '`0`' is stored into its corresponding background word location and vice versa. The bit pairs are compared every time the Configuration Words are loaded, including in Standby Sleep mode. During this comparison, if the Configuration bit values are not found opposite to each other, a configuration mismatch event is generated, which causes a device Reset.

If a device Reset occurs as a result of a configuration mismatch, the CMR Status bit (RCON[9]) is set.

13.18.4.8 Deadman Timer Reset (DMTR)

A Deadman Timer (DMT) Reset is generated when the DMT count has expired.

The primary function of the DMT is to Reset the processor in the event of a software malfunction. The DMT is a free-running instruction fetch timer, which is clocked whenever an instruction fetch occurs until a count match occurs. Instructions are not fetched when the processor is in the Sleep mode.

The DMT consists of a 32-bit counter with a time-out count match value as specified by the DMTCNT[4:0] bits in the CFGCON2 Configuration register.

A DMT is typically used in mission critical and safety critical applications, where any single failure of the software functionality and sequencing must be detected. For more information on the DMT reset, see *Deadman Timer (DMT)* from Related Links.

Related Links

17. Deadman Timer (DMT)

13.18.4.9 Non-maskable Interrupt (NMI) Timer

The NMI timer provides a delay between DMT or WDT events and a device Reset. Set the delay in System Clock counts from 0 to 255 in the NMICNT[15:0] bits (RNMICON[15:0]). If these bits are set to zero, there will be no delay between the DMTO or WDTO flag and a device Reset. If set to a non-zero value, the NMI interrupt has that number of system clocks to clear flags or save data for debugging purposes.

If the corresponding NMI flag is not cleared in RNMICON before the counter reaches zero, then a device Reset will be issued. If the corresponding NMI flag in RNMICON is cleared before the counter reaches zero, then the counter is stopped, then reloaded with the NMICNT value again and waits for another NMI event to occur. A device Reset will not be asserted in this case and software can to return from this interrupt.

The DMTO flag will be set if there is a DMT event. The device will be Reset after the NMI counter expires.

The WDTO flag will be set if there is a WDT event. The device will be Reset after the NMI counter expires.

The WDTS flag will be set if there is a WDT event during the Sleep mode. The WDTS flag will trigger the NMI interrupt but will not start the NMI counter nor cause a Reset.

The CF bit (RNMICON[17]) may be set by the Fail-Safe Clock Monitor (FSCM) if a clock failure is detected. The CF flag will trigger the NMI interrupt but will not start the timer nor cause a Reset.

The SWNMI bit (RNMICON[23]) can be set in software to cause an NMI interrupt but will not start the NMI counter nor cause a Reset.

13.18.4.10 Determining the Source of Device Reset

After a device Reset, the RCON register can be examined to confirm the source of the Reset. All reset status bits in the RCON register must be cleared after reading them to ensure the RCON value will provide meaningful results after the next device Reset.

13.18.5 Effects of Various Resets

The Reset value for the Reset Control register, RCON, will depend on the type of device Reset, as indicated in the following table.

Table 13-4. Status Bits, Their Significance and the Initialization Condition for RCON Register(1)

Condition	Program Counter	EXTR	SWR	WDTO	DMTO	SLEEP(2)	IDLE(2)	CMR	BOR	POR
Power-on Reset or $\overline{\text{MCLR}}$ set as POR	—	0	0	0	0	0	0	0	1	1
Brown-out Reset		0	0	0	0	0	0	0	1	u
$\overline{\text{MCLR}}$ Reset during the Run mode		1	u	u	u	u	u	u	u	u
$\overline{\text{MCLR}}$ Reset during the Idle mode		1	u	u	u	u	1	u	u	u
$\overline{\text{MCLR}}$ Reset during the Sleep mode		1	u	u	u	1	u	u	u	u
Software Reset command		u	1	u	u	u	u	u	u	u
Configuration Word Mismatch Reset		u	u	u	u	u	u	1	u	u
WDT Time-out Reset during the Run mode		u	u	1	u	u	u	u	u	u
WDT Time-out Reset during the Idle mode		u	u	1	u	u	1	u	u	u
WDT Time-out Reset during the Sleep mode		u	u	1	u	1	u	u	u	u
DMT Time-out Reset		u	u	u	1	u	u	u	u	u

.....continued										
Condition	Program Counter	EXTR	SWR	WDTO	DMTO	SLEEP(2)	IDLE (2)	CMR	BOR	POR
Interrupt Exit from the Idle mode	Vector	u	u	u	u	u	1	u	u	u
Interrupt Exit from the Sleep mode		u	u	u	u	1	u	u	u	u

1. Legends:
   – u = unchanged
2. The SLEEP or IDLE states are entered when the correct sequence plus WAIT instruction is executed.

13.18.5.1 Special Function Register (SFR) Reset States

Most of the SFRs associated with the CPU and peripherals are reset to a particular value at a device Reset. This also applies to a WDT/DMT NMI condition, which is treated as a full device Reset by the CPU and peripherals.

The Reset value for the Reset Control register, RCON, is depending on the type of device Reset, see *Status Bits, Their Significance and Initialization Condition for RCON Register* table in *Effects of Various Resets* from Related Links.

Related Links

13.18.5. Effects of Various Resets

13.18.5.2 Configuration Word Register Reset States

All Reset conditions force the configuration settings to be reloaded. The POR and BOR reset all the Configuration Word registers before loading the configuration settings. For all other Reset conditions, the Configuration Word registers are not Reset prior to being reloaded.

13.18.5.3 Using the RCON Status Bits

The user software can read the RCON register after any system Reset to determine the cause of the reset. The following table provides a summary of the Reset flag bit operation.

Note: The status bits in the RCON register must be cleared after they are read, so that the next RCON register value after a device Reset will be meaningful.

Table 13-5. Reset Flag Bit Operation

Flag Bit	Set by	Cleared by
POR (RCON[0])	POR	User Software
BOR (RCON[1])	POR, BOR	User Software
IDLE (RCON[2])	WAIT Instruction	User Software, POR, BOR
STANDBY SLEEP (RCON[3])	WAIT Instruction	User Software, POR, BOR
WDTO (RCON[4])	WDT timeout and NMI counter expires	User Software, POR, BOR
DMTO (RCON[5])	DMT Timeout and NMI counter expires	User Software, POR, BOR
SWR (RCON[6])	Software Reset Command	User Software, POR, BOR
EXTR (RCON[7])	$\overline{\text{MCLR}}$ Reset	User Software, POR, BOR
CMR (RCON[9])	Configuration Mismatch Reset	User Software, POR, BOR
BCFGFAIL (RCON[26])	Non-recoverable error in Primary and Alternate configuration words	User Software, POR, BOR

.....continued		
Flag Bit	**Set by**	**Cleared by**
BCFGERR (RCON[27])	Recoverable error in primary configuration words	User Software, POR, BOR

14. RAM Error Correction Code (RAMECC)

14.1 Overview

For safety applications, the SAM D5x/E5x family can embed error correction codes (ECC) to detect and correct single bit errors or to enable dual error detection in SRAM. As discussed in Memories chapter, when the RAMECC is enabled, the top half of SRAM memory will be reserved to store error correction codes and will not be available for the application.

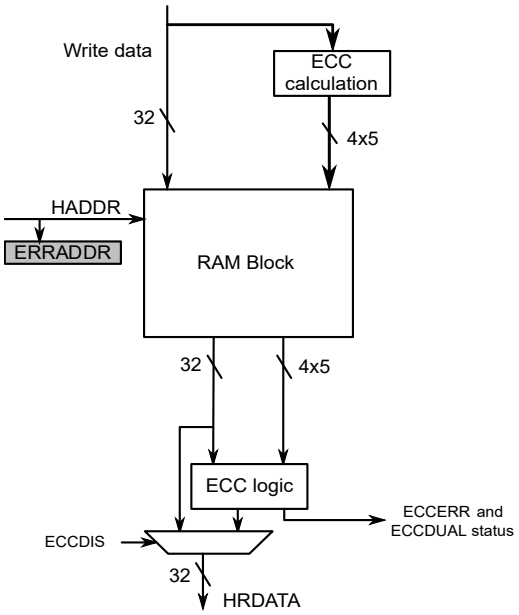
ECC calculation is software selectable through the RAM ECCDIS bit in the NVM User Row. For additional information, refer to Table 9-2.

14.2 Features

- Single bit correction and dual bit detection.
- Error Interrupt.
- Operates in any Sleep mode.
- Interrupts generated by RAMECC can be used to wake up the device from sleep modes.

14.3 Block Diagram

Figure 14-1. RAMECC Block Diagram



14.4 Signal Description

Not applicable.

14.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

14.5.1 I/O Lines

Not applicable.

14.5.2 Power Management

The RAMECC will continue to operate in any sleep mode where the selected source clock is running. The RAMECC's interrupts can be used to wake up the device from sleep modes. See *Power Management Unit (PMU)* from Related Links for details on the different sleep modes.

Related Links

15. Power Management Unit (PMU)

14.5.3 DMA

Not applicable.

14.5.4 Interrupts

The interrupt request line is connected to the interrupt controller. Using the RAMECC interrupt(s) requires the interrupt controller to be configured first.

14.5.5 Events

Not applicable.

14.5.6 Debug Operation

When the CPU is halted in debug mode the RAMECC will correct and log ECC errors based on the table below.

Table 14-1. ECC Debug Operation

DBGCTRL.ECCELOG	DBGCTRL.ECCDIS	Description
0	0	ECC errors from debugger reads are corrected but not logged in INTFLAG.
1	0	ECC errors from debugger reads are corrected and logged in INTFLAG.
X	1	ECC errors from debugger reads are not corrected or logged in INTFLAG.

If the RAMECC is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

14.5.7 Register Access Protection

All registers with write-access are optionally write-protected by the Peripheral Access Controller (PAC), except the following registers:

- Interrupt Flag Status and Clear (INTFLAG) register
- Status (STATUS) register.

Write-protection is denoted by the Write-Protected property in the register description.

Write-protection does not apply to accesses through an external debugger (see *Peripheral Access Controller (PAC)* from Related Links).

Related Links

26. Peripheral Access Controller (PAC)

14.5.8 Analog Connections

Not applicable.

14.6 Functional Description

14.6.1 Principle of Operation

Error Correcting Code (ECC) is implemented to detect and correct errors that may arise in the RAM arrays. The ECC logic is capable of double error detection and single error correction per 8-bit byte.

Upon single bit error detection, the Single Bit Error interrupt flag is raised (INTFLAG.SINGLEE). If a dual error is detected, the Dual Error interrupt flag (INTFLAG.DUALE) is raised. When the first error is detected, the ERRADDR register is frozen with the failing address and remains frozen until INTFLAG.DUALE and INTFLAG.SINGLEE are cleared. If a dual bit error occurs while INTFLAG.SINGLEE is set, the ERRADDR register is updated with the dual bit error information and INTFLAG.DUALE is also set.

The INTFLAG.SINGLEE and INTFLAG.DUALE bits are both cleared on ERRADDR read.

The block diagram shows the ECC interface. When ECC is disabled (CTRLA.ECCDIS=1), the ECC field in RAM is left unchanged on writes. On reads, ECC errors are not corrected or flagged.

Related Links

14.3. Block Diagram

14.6.2 Interrupts

The RAMECC has the following interrupt sources:

- Dual Bit Error (DUALE): Indicates that a dual bit error has been detected.
- Single Bit Error (SINGLEE): Indicates that a single bit error has been detected.

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear (INTFLAG) register is set when the interrupt condition occurs.

Each interrupt can be individually enabled by writing a '1' to the corresponding bit in the Interrupt Enable Set (INTENSET) register, and disabled by writing a '1' to the corresponding bit in the Interrupt Enable Clear (INTENCLR) register.

An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the ERRADDR register is read, the interrupt is disabled, or the RAMECC is reset.

All interrupt requests from the peripheral are ORed together on system level to generate one combined interrupt request to the NVIC. The user must read the INTFLAG register to determine which interrupt condition is present.

Note: Interrupts must be globally enabled for interrupt requests to be generated.

Related Links

14.8.3. INTFLAG

14.7 Register Summary - RAMECC

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	INTENCLR	7:0							DUALE	SINGLEE
0x01	INTENSET	7:0							DUALE	SINGLEE
0x02	INTFLAG	7:0							DUALE	SINGLEE
0x03	STATUS	7:0								ECCDIS
0x04	ERRADDR	7:0	ERRADDR[7:0]							
		15:8	ERRADDR[15:8]							
		23:16								ERRADDR[16]
		31:24								
0x08 ... 0x0E	Reserved									
0x0F	DBGCTRL	7:0							ECCELOG	ECCDIS

14.8 Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the “PAC Write-Protection” property in each individual register description (see *Register Access Protection* from Related Links).

Related Links

14.5.7. Register Access Protection

14.8.1 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x00
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set (INTENSET) register.

Bit	7	6	5	4	3	2	1	0
							DUALE	SINGLEE
Access							R/W	R/W
Reset							0	0

Bit 1 – DUALE Dual Bit Error Interrupt Enable Clear
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Dual Bit Error Interrupt Enable bit, which disables the Dual Bit Error interrupt.

Value	Description
0	The Dual Bit Error interrupt is disabled.
1	The Dual Bit Error interrupt is enabled.

Bit 0 – SINGLEE Single Bit Error Interrupt Enable Clear
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Single Bit Error Interrupt Enable bit, which disables the Single Bit Error interrupt.

Value	Description
0	The Single Bit Error interrupt is disabled.
1	The Single Bit Error interrupt is enabled.

14.8.2 Interrupt Enable Set

Name: INTENSET
Offset: 0x01
Reset: 0x00
Property: Write-Protected

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

Bit	7	6	5	4	3	2	1	0
							DUALE	SINGLEE
Access							R/W	R/W
Reset							0	0

Bit 1 – DUALE Dual Bit Error Interrupt Enable Set
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Dual Bit Error Interrupt Enable bit, which enables the Dual Bit Error interrupt.

Value	Description
0	The Dual Bit Error interrupt is disabled.
1	The Dual Bit Error interrupt is enabled.

Bit 0 – SINGLEE Single Bit Error Interrupt Enable Set
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Single Bit Error Interrupt Enable bit, which disables the Single Bit Error interrupt.

Value	Description
0	The Single Bit Error interrupt is disabled.
1	The Single Bit Error interrupt is enabled.

14.8.3 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x02
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
							DUALE	SINGLEE
Access							R/W	R/W
Reset							0	0

Bit 1 – DUALE Dual Bit ECC Error Interrupt

This flag is set on the occurrence of a dual bit ECC error.
Writing a '0' to this bit has no effect.
Reading the ECCADDR register will clear the Dual Bit Error interrupt flag.

Value	Description
0	No dual bit errors have been received since the last clear.
1	At least one dual bit error has occurred since the last clear.

Bit 0 – SINGLEE Single Bit ECC Error Interrupt

This flag is set on the occurrence of a single bit ECC error.
Writing a '0' to this bit has no effect.
Reading the ECCADDR register will clear the Single Bit Error interrupt flag.

Value	Description
0	No errors have been received since the last clear.
1	At least one single bit error has occurred since the last clear.

14.8.4 Status

Name: STATUS
Offset: 0x03
Reset: 0x00
Property: Read-only

Bit	7	6	5	4	3	2	1	0
								ECCDIS
Access								R
Reset								0

Bit 0 – ECCDIS ECC Disable
This bit is fuse updated at startup. When enabled, the calculated ECC is written to RAM along with data. ECC correction and detection is enabled for reads.

Value	Description
0	ECC detection and correction is enabled.
1	ECC detection and correction is disabled.

14.8.5 Error Address

Name: ERRADDR
Offset: 0x04
Reset: 0x00000000
Property: Read-only

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
								ERRADDR[16]
Access								R
Reset								0

Bit	15	14	13	12	11	10	9	8
	ERRADDR[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

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

Bits 16:0 – ERRADDR[16:0] ECC Error Address
The RAM address offset from RAM start that caused an ECC error. If a single bit error is followed by a dual bit error, this register will be updated with the address of the dual bit error, otherwise it stalls on the first error occurrence. This register will read as zero unless INTFLAG.SINGLEE and/or INTFLAG.DUALE are 1.

14.8.6 Debug Control

Name: DBGCTRL
Offset: 0x0F
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
							ECCELOG	ECCDIS
Access							R/W	R/W
Reset							0	0

Bit 1 – ECCELOG ECC Error Log

When DBGCTRL.ECCDIS=0, This bit controls whether ECC errors are logged in the INTFLAG register. When DBGCTRL.ECCDIS=1, this bit has no meaning.

Value	Description
0	ECC errors for debugger reads are not logged.
1	ECC errors for debugger reads are logged if DBGCTRL.ECCDIS=0.

Bit 0 – ECCDIS ECC Disable

By default, ECC errors during debugger reads are corrected and logged based on DBGCTRL.ECCELOG. Setting this bit will disable ECC correction and logging.

Value	Description
0	ECC errors are are corrected for debugger reads and logged based on DBGCTRL.ECCELOG.
1	ECC errors are masked for debugger reads.

15. Power Management Unit (PMU)

15.1 Overview

This chapter describes the Power Management Unit (PMU) in the PIC32CX-BZ2 wireless microcontroller with the various modes it provides. It is for information purposes only.

Note: The PMU is a complex power controller that requires specific configuration and handling by the software for correct, safe operation of the SOC. The software SDK and operational stacks provided by Microchip handles the start-up and operation of the PMU. Therefore, Microchip highly recommends using this software framework for all application development on the device.

15.2 Power Modes

To minimize power consumption, the PIC32CX-BZ2 incorporates a PMU that provides both DC-DC (Buck) and LDO power conversion. The input voltage range to the PMU is 1.9V to 3.6V – VDD (main). On power-up, the PIC32CX-BZ2 operates in LDO mode; the software must enable the DC-DC Buck converter.

See *Electrical Characteristics* from Related Links for detailed electrical information.

The power modes and power management features provided by the PMU are shown in the following table.

Table 15-1. Power Modes and Power Management Features

Functions	Device Power Modes					
	Run	Idle	Dream[1]	Sleep(Standby)	Deep Sleep(Backup)	eXtreme Deep Sleep(Off)
CPU	On	Clock Gated	Clock Gated	Clock Gated	Off	Off
Peripherals	On	On	On Demand	Clock Gated	Off	Off
Flash	On	On	On Demand	Clock Gated	Off	Off
Core System Memory	On	On	On Demand	Retention Mode	Off	Off
Radio	On	On	On Demand	Clock Gated	Off	Off
DSWDT	On	On	On	On	On	Off
RTCC	On	On	On	On	On	Off
Backup RAM	On	On	On Demand	Retention Mode	Retention Mode	Off
XTAL(16 MHz)	On	On	On	On	Off	Off
SPLL	On	On	On Demand	Off	Off	Off
ADC	On	On	On Demand	On Demand	Off	Off
Analog Comp	On	On	On Demand	Off	Off	Off
FRC	On	On	On Demand	Off	Off	Off
LPRC	On	On	On	On	On	On

Note:

1. Dream (Sleep Walking) is not a mode by itself but is a companion mode to the Sleep mode. This mode cannot be entered directly through a software command.

Current consumption can be reduced on peripherals/modules in "On" state by clock gating and/or disabling the module, see *Module Enable/Disable Controls* in the *Register Description* from Related Links. All transitions from

the Run state to any of the low power states is simply initiated by WFI command from the CPU. However, prior to initiating the WFI command.

The software performs the following actions:

- Disabling all interrupts
- Setting up the DSCON[DSEN], OSCCON[SLPEN], OSCCON[DRMEN] and Wireless Subsystem Sleep mode Controls
- Set the Wireless Subsystem into the Low Power mode
- Enable the appropriate wake-up events
- Checking for any pending interrupts and, if present, abort deep sleep and service the interrupt
- If no pending interrupts, then issue a SLEEP/WAIT command from the CPU to get into the appropriate Low Power mode

In the Run mode, the CPU is actively executing code. Run mode provides normal operation of the processor and all peripherals that are currently enabled.

In the Idle mode, all active peripherals can be clocked, but the CPU core is clock gated off and no code is executed.

In the Dream mode (or Sleep Walking mode), the CPU is clock gated but peripherals can be turned on on-demand by events related to those peripherals. No code is executed.

In the Sleep and Deep Sleep modes, the backup RAM is in retention mode, while the CPU and most peripherals are clock gated off and no code is executed.

In the eXtreme Deep Sleep mode, only the Low Power RC oscillator is enabled. Exiting Deep sleep/XDS is equivalent to Power-on Reset. The RCON register provides the status whether it is a normal power up or exiting from deep sleep/XDS.
Note: All desired register/configuration inputs (for example, DSZPBOREN, DSWDTPS) to be preserved must be set to expected values before setting DSCON.DSEN = `1`; failure to do so would result in faulty configuration fault indications.

The low power mode entry and exit commands and wake up resources are shown in the following table.

Table 15-2. Low Power Mode Entry and Exit commands and Wake-up Resources

Device Power Modes	Entry Commands	Wake-up Resources
Run	—	—
Idle	WFI Instruction + ~OSCCON[SLPEN]	IRQ, Reset, Others(1)
Dream	OSCCON[DREAM] + Event + Sleep mode	IRQ, Reset, Others
Sleep	~DSCON[DSEN]+OSCCON[SLPEN]+Wireless Sleep followed by WFI	IRQ, Reset, Others
Deep Sleep	DSCON[DSEN]+ {RTCC (Enabled) or DSWDT (Enabled)} + Wireless Sleep followed by WFI	INT0, RTCC, DSWDT(2), Reset
eXtreme Deep Sleep	DSCON[DSEN]+ {RTCC (Disabled) and DSWDT (Disabled)} + Wireless Sleep followed by WFI	INT0, Reset

Notes:

1. Others = System Wake-up (Dream), WDT (Timeout Event), DMT (Timeout Event), PLVD Event, PMU Event, External NMI/INT, DSU/ICD Debug Event, CPU Debug Event.
2. To enable the Deep Sleep Mode Watchdog Timer(DSWDT) in deep sleep mode, Configuration Control Register 4(CFGCON4). DSWDT is a separate timer from the device's watchdog timer that is used in run mode. The device Watchdog Timer (WDT) does not have to be enabled for the DSWDT to function.

Related Links

20.4.2. Register Description
43. Electrical Characteristics

16. Watchdog Timer (WDT)

16.1 Overview

The Watchdog Timer (WDT) can be used to detect system software malfunctions by resetting the device if the WDT is not cleared periodically in software. The user can configure the WDT in Windowed mode or non-Windowed mode. Various WDT time-out periods can be selected using the WDT postscaler. The WDT can also be used to wake the device from Sleep or Idle mode.

16.2 Features

- Configuration using Fuses (DEVCFG) or Software controlled
- Up to 32 Configurable Time-Out Periods
- Can wake the Device from Standby Sleep or Idle mode
- Independent Run and Standby Sleep mode Counters
- WDT may use alternate Clock source and Postscaler for Run mode Counter
- Independent 5-bit Postscalers for Run and Standby Sleep mode Counters
- Hardware and Software enabled
- Two Clock sources
- Windowed WDT

Note: When the CPU is running on the same clock or clock frequency as the WDT, the lowest pre-scale values may not allow the CPU to have enough time to reset the WDT before it expires.

16.3 Applications

The WDT is a free-running timer with a configurable postscaler. The counter is clocked with an external reference clock until the counter value exceeds the selected WDT period. If enabled, the WDT will continue to operate even if the main processor clock (for example, the crystal oscillator) fails.

The WDT uses separate internal counters for use in Run mode and Sleep/Idle modes. One counter operates only in Run mode; the count value of this counter is frozen when the device is in Sleep or Idle modes.

The second counter operates only in Sleep and Idle modes; it is reset when entering Sleep or Idle. This provides the following benefits:

- A different WDT clock source can be used in Run mode(PB1_CLK or LPRC) vs. Sleep/Idle modes (LPRC only).
- A different post-scale value may be used in Run mode vs. Sleep/Idle modes.
- The Run mode WDT count is preserved while in Sleep or Idle modes, which makes it easier to manage the WDT while in windowed mode.

The WDT can have two modes of operation, Windowed and non-Windowed. In Windowed mode, software can clear the WDT only when the counter is in its final window before a period match occurs. There are four window size options (25%, 37.5%, 50% and 75% of the total WDT period). This window is active when the timer counter is greater than a predetermined value for each option. The window size is determined by the WDTWINSZ configuration fuses (see *System Configuration Registers (CFG)* from Related Links). Any attempts to clear the WDT when the window is not active would cause a device Reset. In non-Windowed configuration, software can clear the WDT any time before the period match occurs.

Note: Windowed WDT operation is not applicable in Sleep or Idle modes.

Related Links

18. System Configuration and Register Locking (CFG)

16.3.1 Enabling the WDT

The WDT can be enabled through hardware or software control. The WDT is enabled if WDTEN=1 in the DEVCFG2 fuse register. If WDTEN=0, the WDT is disabled, and can be controlled using the ON bit (WDTCON[15]).

16.3.2 Selecting WDT Clock Source

The Sleep Mode Counter always uses the 32 kHz LPRC clock source. The Run Mode Counter will use the clock source selected by the WDTRMCS bit in fuses Configuration Bits register DEVCFG2 to be either the 32 kHz LPRC clock or the PB1_CLK bus clock (sysclk). See *WDTRMCS[1:0]* bits in the *CFGCON2(L)* register from Related Links.

Related Links

18.4.3. CFGCON2(L)

16.3.3 Clearing the Watch Dog Timer

To clear the WDT, software must write to the WDTKEY register with the value 16'h5743 before the timer expires. The upper two bytes of WDTCON must be written simultaneously, as a 16-bit write. Any other write with a different value or byte size will NOT clear the timer.

16.3.4 Selecting WDT Period

The WDT can be configured for various time-out periods by controlling the WDT postscaler value. The settings are chosen using the WDTPSR[4:0] inputs for the Run Mode Counter and the WDTPSS[4:0] inputs for the Sleep Mode Counter. These values range from 1 to 2^20 and allow for time-out periods of from 1 ms to over 17 minutes when using the 21 kHz LPRC clock.

16.3.5 Events that Reset both WDT Counters

The following events will reset both internal WDT counters:

- A generic device reset
- Disabling the WDT via the ON bit

16.3.6 Events that Reset only the Run Mode Counter

The following events reset the Run Mode Counter:

- Any counter value greater than the selected WDT period
- Detecting a correct write value to WDTCON.WDTKEY within the correct time window

16.3.7 Events that Reset only the Sleep Mode Counter

The following events reset the Sleep Mode Counter:

- Entry into Sleep or Idle modes
- Any counter value greater than the selected WDT period

16.3.8 Run Mode WDT Event

Once the WDT period is exceeded, or if a correct write value to the WDTCON.WDTKEY is executed in a Windowed mode when the window is not yet active, the WDT event is activated. The event remains active for one clock period.

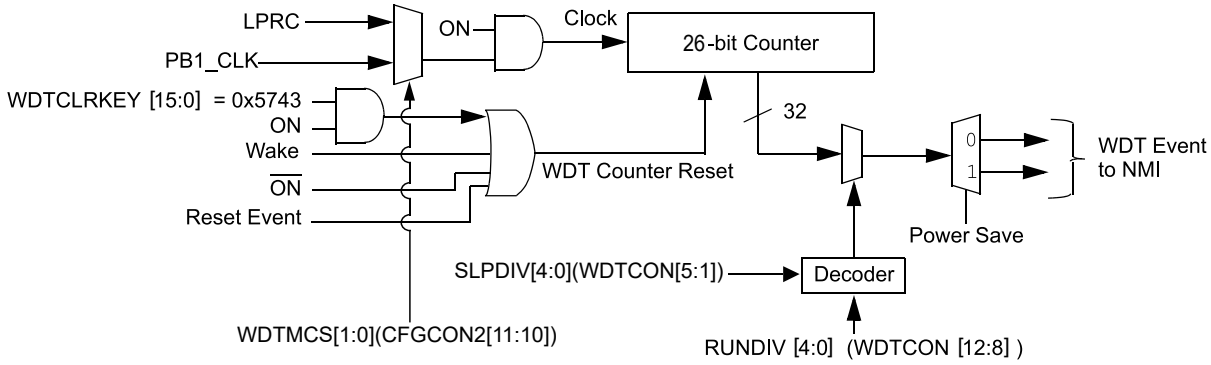
The WDT is reset after the count is exceeded and continues to run.

16.3.9 Sleep Mode WDT Event

Once the WDT period is exceeded, the WDT event is activated. The event remains active for one clock period. The WDT is reset after the count is exceeded.

16.4 Block Diagram

Figure 16-1. Watchdog Timer Block Diagram



16.5 Configuration

The WDT is configured using the following config register bits/fields:

- Window size (CFGCON2.FWINSZ[1:0])
- Windowing disable (CFGCON2.WINDIS)
- Post-scaler selection (CFGCON2.WDTPS[4:0])
- Run Mode Counter clock source (CFGCON2.WDTRMCS[1:0])

16.6 Register Summary - WDT

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	WDTCON	7:0			SLPDIV[4:0]					WDTWINEN
		15:8	ON			RUNDIV[4:0]				
		23:16	WDTCLRKEY[7:0]							
		31:24	WDTCLRKEY[15:8]							

16.7 Register Description

Note: All registers in this table have corresponding CLR, SET and INV registers at its virtual address, plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET, and INV Registers* from Related Links.

Following conventions are used in the register description:

- R = Readable bit
- W = Writable bit
- — = Unimplemented bit, read as '0'
- -n = Value at POR
- 1 = Bit is set
- 0 = Bit is cleared
- x = Bit is unknown
- y = Values set from Configuration bits on POR
- Reset values are shown in hexadecimal.

Related Links

6.1.9. CLR, SET and INV Registers

16.7.1 WDTCON - Watchdog Timer Control Register

Name: WDTCON
Offset: 0x00
Reset: 0x00
Property: -

Bit	31	30	29	28	27	26	25	24
	WDTCLRKEY[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	WDTCLRKEY[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON			RUNDIV[4:0]				
Access	R/W			R	R	R	R	R
Reset	y			y	y	y	y	y

Bit	7	6	5	4	3	2	1	0
			SLPDIV[4:0]					WDTWINEN
Access			R	R	R	R	R	R/W
Reset			y	y	y	y	y	0

Bits 31:16 – WDTCLRKEY[15:0] Watchdog Timer Clear Key bits
To clear the WDT to prevent a time-out, software must write the value 0x5743 to this location using a single 16-bit write. Anything other than a 16-bit write will not reset the WDT. You must use a 16-bit write for the WDTCLRKEY[15:0] bits.

Bit 15 – ON Watchdog Timer Enable bit
Note:
1. This bit only has control when the WDTEN bit (DEVCFG2/CFGCON2[23]) = 0.

Value	Description
1	WDT is enabled
0	WDT is disabled

Bits 12:8 – RUNDIV[4:0] Watchdog Timer Postscaler Run Counter Value bits
On Reset, these bits are set to the values of the WDTPSR[4:0] Configuration bits in CFGCON2.

Bits 5:1 – SLPDIV[4:0] Watchdog Timer Postscaler Sleep Counter Value bits
On Reset, these bits are set to the values of the WDTPSS[4:0] Configuration bits in CFGCON1.

Bit 0 – WDTWINEN Watchdog Timer Window Enable bit

Value	Description
1	Enable windowed WDT
0	Disable windowed WDT

17. Deadman Timer (DMT)

17.1 Overview

The Deadman Timer (DMT) module is designed to enable users to be able to monitor the health of their application software by requiring periodic timer interrupts within a user-specified timing window. The DMT module is a synchronous counter and when enabled, counts instruction fetches and causes a system reset if the DMT counter is not cleared within a set number of instructions. The DMT is typically connected to the system clock that drives the processor. The user specifies the timer time-out value and a mask value that specifies the range of the window, which is the range of counts that is not considered for the comparison event.

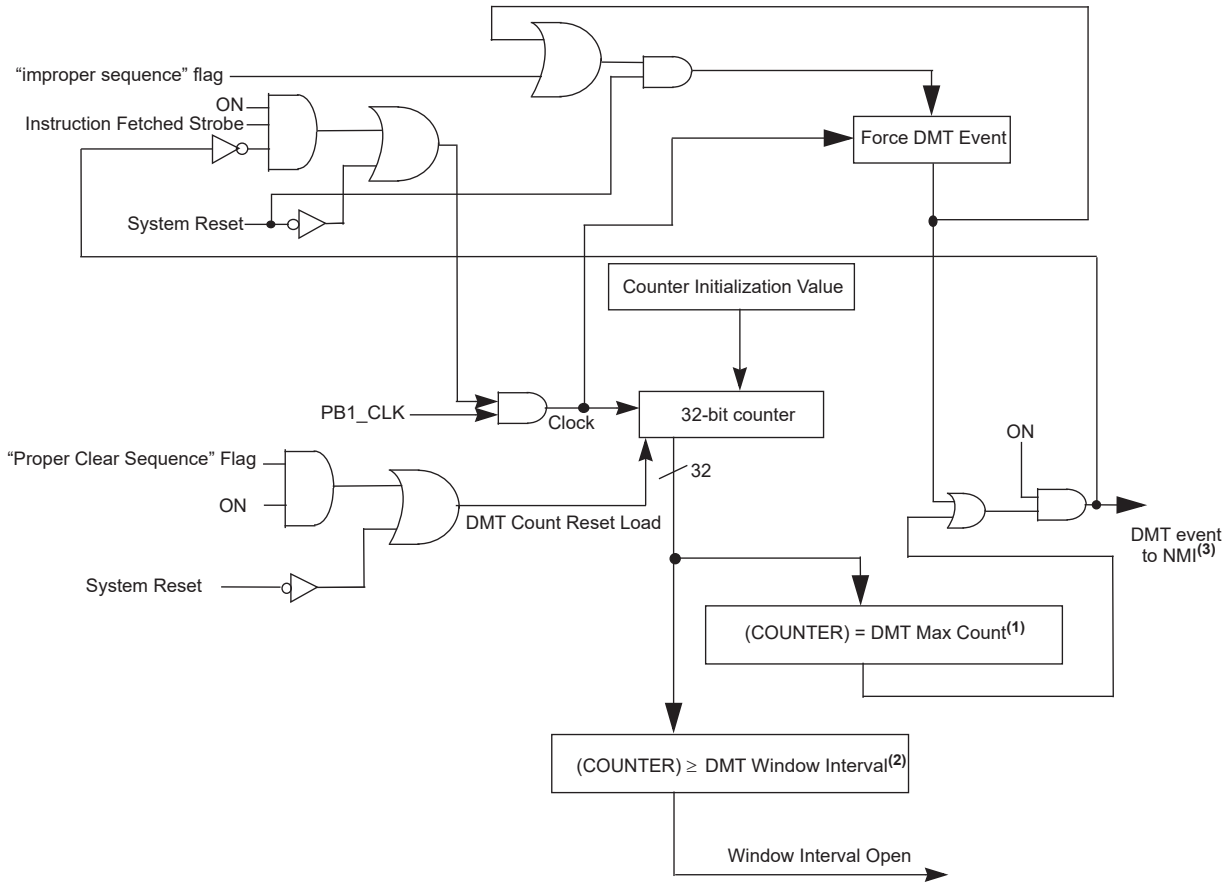
17.2 Features

- 32-bit configurable count-limit based on counting instructions fetched
- Hardware- or software-enabled control
- User-configurable time-out period or instruction count
- Two instruction sequence to clear timer
- 32-bit configurable window to clear timer

17.3 Block Diagram

The following figure shows the block diagram of the Deadman Timer.

Figure 17-1. Deadman Timer Block Diagram



Notes:

1. The DMT Max Count is controlled by the DMTCNT[4:0] bits in the CFGCON2 register "Maximum = 2^31".
2. The DMT Window Interval is controlled by the DMTINTV[2:0] bits in the CFGCON2 register.
3. For more details, see *Resets* from Related Links.

Related Links

13.18. Resets

17.4 DMT Operation

17.4.1 Mode of Operation

The primary function of the Deadman Timer (DMT) module is to reset the processor in the event of a software malfunction. The DMT module, which works on the system clock, is a free running instruction fetch timer, which is clocked whenever an instruction fetch occurs until a count match occurs. The instructions are not fetched when the processor is in the Standby Sleep mode.

The DMT module consists of a 32-bit counter, the read-only DMTCNT register with a time-out count match value as specified by the 32-bit DMT count configuration fuse bits CFGCON2.DMTCNT[4:0]. Whenever the count match occurs, a DMT reset event will occur and the DMTEVENT bit in DMTSTAT register will be set.

A DMT module is typically used in mission critical and safety critical applications, where any failure of the software functionality and sequencing must be detected.

17.4.2 Enabling and Disabling the DMT Module

Because of the nature of the Deadman Timer, the PMD register bit is not provided to enable/disable the module. The DMT module can be enabled or disabled by the DMT enable (DMTEN) bit in the Configuration Control Register 2 (CFGCON2) fuse register or it can be enabled through software by writing to the Deadman Timer Control (DMTCON) register. Once the DMT is enabled, it may not be disabled without a device reset.

If the DMTEN configuration bit in the CFGCON2 fuse register is set, the DMT is always enabled. The ON control bit (DMTCON[15]) in DMTCON register will reflect this by reading a '1'. In this mode, the ON bit (DMTCON[15]) cannot be cleared in software. To disable the DMT, the DMTEN configuration bit must be cleared in the CFGCON2 fuse register. When DMTEN is cleared to '0' in the CFGCON2, the DMT is disabled in hardware.

Software can enable the DMT by setting the ON bit in the DMTCON register. However, for software control, the DMTEN configuration bit in the CFGCON2 fuse register must be set to '0'.

17.4.3 DMT Count Windowed Interval

The DMT module has the Windowed Operation mode. The DMT interval (DMTINV[2:0]) bits in the Configuration Control Register 2 (CFGCON2) fuse register sets the window interval value. The PSINTV[31:0] bits in DMT interval post status register (DMTPSINTV) allows the software to read the DMT window interval value. That means this register reads the value that is written to the DMT interval (DMTINV[2:0]) bits in the Configuration Control Register 2 (CFGCON2) fuse register.

In the Windowed mode, software can clear the DMT only when the counter is in its final window before a count match occurs. That is, if the DMT counter value is greater than or equal to the value written to the window interval value, only then the DMT clear sequence can be executed in the DMT module. If the DMT is cleared before the allowed window, a DMT reset event is immediately generated.

17.4.4 DMT Count Selection

The Deadman Timer count is set by the DMT count configuration (DMTCNT[4:0]) bits in the Configuration Control Register 2 (CFGCON2) fuse register. The current DMT count value can be obtained by reading the DMT count register DMTCNT.

The PSCNT[31:0] bits in the DMT count post status register (DMTPSCNT) allows the software to read the maximum count selected for the DMT. The PSCNTx bit values are the values that are initially written to the DMTCNTx bits in the CFGCON2 fuse register. Whenever the DMT event occurs, the user can always compare to see whether the current counter value in the DMTCNT register is equal to the value of the DMTPSCNT register, which holds the maximum count value.

Whenever the DMT current counter value in DMTCNT reaches the value of the DMTPSINTV register, the window interval opens permitting the user to execute the DMT clear sequence. The open window interval is indicated by the WINOPN bit in DMTSTAT register.

The UPRCNT[15:0] bits in the DMT hold register (DMTHOLDREG) holds the value of the last read DMT upper count values whenever DMTCNT is last read.

17.4.5 DMT Operation in Power-Saving Modes

As the DMT module is only incremented by instruction fetches, the count value will not change when the core is inactive. The DMT module remains inactive in the Standby Sleep and Idle modes. As soon as the device wakes-up from Standby Sleep or Idle, the DMT counter starts incrementing again for every instruction fetch.

17.4.6 Resetting the DMT

The DMT can be reset in two ways: one way is after a system reset and another way is by writing an ordered sequence to the DMT pre-clear register (DMTPRECLR) and DMT clear register (DMTCLR) in a specific two-step sequence.

Clearing the DMT counter value requires the following sequence of operations:

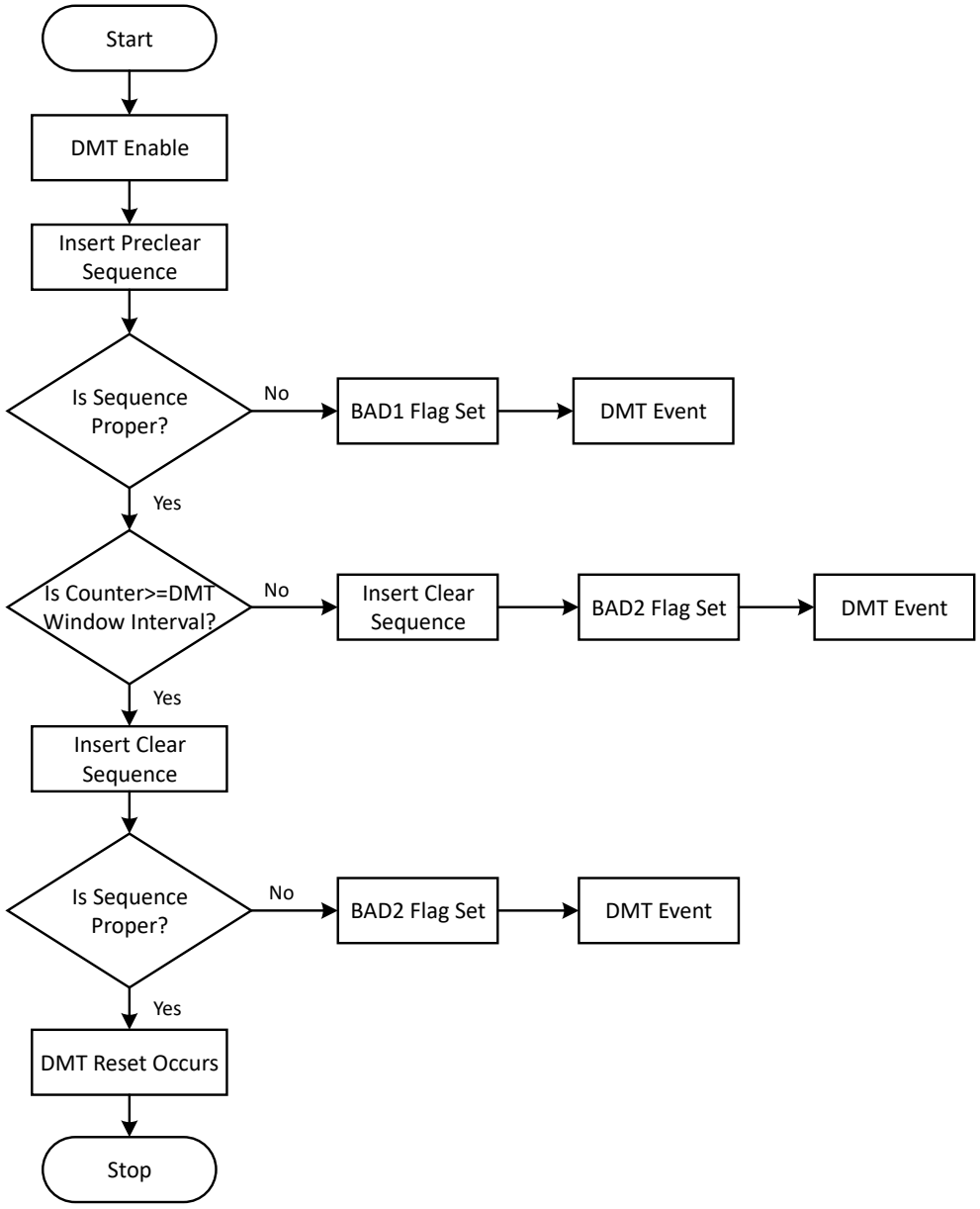
1. The STEP1[7:0] bits in the DMTPRECLR register must be written as '01000000' (0x40). This action sets the "enable for clearing" state, which enables the DMT to be cleared by step 2.
2. The STEP2[7:0] bits in the DMTCLR register must be written as '00001000' (0x08). This can only be done if preceded by step 1 and if the DMT is in the open window interval.

Once these values are written, following are cleared to zero:

- DMTCNT counter
- DMTPRECLR register
- DMTCLR register
- DMTSTAT register

If any value other than 0x40 is written to the STEP1x bits, the BAD1 bit in the DMTSTAT register will be set and it causes a DMT event to occur. Any value other than 0x08, written to the STEP2x bits, will cause the BAD2 bit to be set in the DMTSTAT register. Also, if step 2 is not preceded by step 1, or step 2 is not carried out in the open window interval, it causes the BAD2 flag to be set. Immediately, a DMT event will occur. In both these cases, the DMTEVENT bit in the DMTSTAT register will be set. Refer to the flowchart shown in the following figure.

Figure 17-2. Flowchart for Clearing the DMT



17.5 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	DMTCON	7:0								
		15:8	ON							
		23:16								
		31:24								
0x04 ... 0x0F	Reserved									
0x10	DMTPRECLR	7:0								
		15:8	STEP1[7:0]							
		23:16								
		31:24								
0x14 ... 0x1F	Reserved									
0x20	DMTCLR	7:0	STEP2[7:0]							
		15:8								
		23:16								
		31:24								
0x24 ... 0x2F	Reserved									
0x30	DMTSTAT	7:0	BAD1	BAD2	DMT_EVENT					WINOPN
		15:8								
		23:16								
		31:24								
0x34 ... 0x3F	Reserved									
0x40	DMTCNT	7:0	COUNTER[7:0]							
		15:8	COUNTER[15:8]							
		23:16	COUNTER[23:16]							
		31:24	COUNTER[31:24]							
0x44 ... 0x4F	Reserved									
0x50	DMTHOLDREG	7:0	UPRCNT[7:0]							
		15:8	UPRCNT[15:8]							
		23:16								
		31:24								
0x54 ... 0x5F	Reserved									
0x60	DMTPSCNT	7:0	PSCNT[7:0]							
		15:8	PSCNT[15:8]							
		23:16	PSCNT[23:16]							
		31:24	PSCNT[31:24]							
0x64 ... 0x6F	Reserved									
0x70	DMTPSINTV	7:0	PSINTV[7:0]							
		15:8	PSINTV[15:8]							
		23:16	PSINTV[23:16]							
		31:24	PSINTV[31:24]							

17.6 Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the “PAC Write-Protection” property in each individual register description.

Some registers are synchronized when read and/or written. Synchronization is denoted by the “Write-Synchronized” or the “Read-Synchronized” property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable protection is denoted by the “Enable-Protected” property in each individual register description.

Note: All registers in this table have corresponding CLR, SET and INV registers at their virtual addresses plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET and INV Registers* from Related Links

Related Links

6.1.9. CLR, SET and INV Registers

17.6.1 Deadman Timer Control

Name: DMTCON
Offset: 0x00
Reset: 0x00
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON							
Access	HC							
Reset	0							

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – ON On bit

The ON bit reflects the status of the configuration fuse CFGCON2.DMTEN, if the fuse is set.

Value	Description
1	Enables the Deadman Timer if the event configuration fuse is not enabled.
0	The DMT disabled.

17.6.2 Deadman Timer Preclear

Name: DMTPRECLR
Offset: 0x10
Reset: 0x00
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	STEP1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 15:8 – STEP1[7:0] Pre-Clear Enable bit when write pattern is:

Value	Description
01000000	Enables the Deadman Timer Pre-Clear (STEP 1).
all other write patterns	Sets DMTSTAT.BAD1 flag to '1'. **Note:** Bits 15:8 are cleared when a DMT reset event occurs. STEP1 is also cleared if DMTCLR.STEP2 is loaded with the correct value in the correct sequence.

17.6.3 Deadman Timer Clear

Name: DMTCLR
Offset: 0x20
Reset: 0x00
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	STEP2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – STEP2[7:0] Clear Timer bit when write pattern is:

Value	Description
00001000	Clears DMTPRECLR.STEP1, DMTCLR.STEP2 and the Dead Man Timer if and only if preceded by the correct loading of Pre-Clear Enable (STEP1) in the correct sequence. The write to the DMTCLR.STEP2 field may be verified by reading DMTCNT and observing the counter being reset.
all other write patterns	The DMTSTAT.BAD2 flag is set to '1', the value in the DMTPRECLR.STEP1 will remain unchanged, and the new value being written DMTCLR.STEP2 will be captured. **Note:** These bits 7:0 are also cleared when a DMT reset event occurs.

17.6.4 Deadman Timer Status

Name: DMTSTAT
Offset: 0x30
Reset: 0x00
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	BAD1	BAD2	DMT_EVENT					WINOPN
Access	R	R	R					R
Reset	0	0	0					0

Bit 7 – BAD1 When an incorrect DMTPRECLR.STEP1 value is detected, this bit is set. It is cleared by a Reset.

Bit 6 – BAD2 When an incorrect value of DMTCLR.STEP2 is detected, this bit is set. It is cleared by a Reset.

Bit 5 – DMT_EVENT This bit is set when the Deadman timer event is detected (counter expired or bad STEP1[7:0] or STEP2[7:0] value is entered prior to the counter increment). This bit remains set and is cleared only by a Reset.

Bit 0 – WINOPN Deadman Timer Clear Window bit.
A value of '1' indicates that a STEP2 "clear" action can take place, and if this "clearing" action occurs as part of a correct sequence of actions, the DMT counter will be cleared.

17.6.5 Deadman Timer Count

Name: DMTCNT
Offset: 0x40
Reset: 0x00
Property: -

Bit	31	30	29	28	27	26	25	24
	COUNTER[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	COUNTER[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COUNTER[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – COUNTER[31:0] Read current contents of DMT Counter.

17.6.6 Deadman Timer Count Holding Register

Name: DMTHOLDREG
Offset: 0x50
Reset: 0x00
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	UPRCNT[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

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

Bits 15:0 – UPRCNT[15:0]

It is the content of DMTCNT.COUNTER[31:16] when the counter was last read to ensure a synchronous snapshot of the counter. This register is initialized to '0' on reset and is only loaded when the DMTCNT register is read.

17.6.7 Post Status Configure DMT Count Status Register

Name: DMTPSCNT
Offset: 0x60
Reset: 0x00
Property: -

Bit	31	30	29	28	27	26	25	24
	PSCNT[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PSCNT[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSCNT[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – PSCNT[31:0]
DMT Instruction Count Value Configuration Fuse Status bits. This bit always reflects the value of CFGCON2.DMTCNT.

17.6.8 Post Status Configure DMT Interval Status Register

Name: DMTPSINTV
Offset: 0x70
Reset: 0x00
Property: -

Bit	31	30	29	28	27	26	25	24
	PSINTV[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PSINTV[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSINTV[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – PSINTV[31:0] DMT Window Interval Configuration Status bits.
This bit reflects the value of CFGCON2.DMTINTV.

18. System Configuration and Register Locking (CFG)

18.1 Overview

This device provides several user writable configuration registers related to the configuration and operation of the system. The registers marked with (L) are loadable from Flash via their corresponding FBCFG* registers in the following table. The user must program these FBCFG* registers, which, then, loads the appropriate register after Reset.

This device provides a single user writable configuration register related to boot configuration of the device. The BCFG0 register provides control, selection and locking for various features of the device, including Flash BCFG7-0 valid, Flash Signature Bit (read only), Code Protect Status (read only). BCFG0 is a read-only register loaded from Flash register FBCFG0 in the following table.

Table 18-1. Writable Configuration Registers

Register	Address	Destination
FBCFG0	0x0004_5F9C	BCFG0 (0x4400_0200)
FBCFG1/DEVCFG0	0x0004_5F98	CFGCON0(L) (0x4400_0000)
FBCFG2/DEVCFG1	0x0004_5F94	CFGCON1(L) (0x4400_0010)
FBCFG3/DEVCFG2	0x0004_5F90	CFGCON2(L) (0x4400_0020)
FBCFG4/DEVCFG4	0x0004_5F8C	CFGCON4(L) (0x4400_0040)
FBCFG5/FUSERID	0x0004_5F88	USERID(L) (0x4400_00A0)

CFGCON0(L) – Provides control, selection and locking for various features of the device.

- PPS register locking
- PMD register locking
- CFGPG register locking
- Config register locking
- JTAG port enable and configuration
- Trace port enable
- Flash ECC control

CFGCON1(L) – Provides control, selection and locking for various features of the device.

- Debug port and feature configuration CFGCON0 locking control
- Class B functionality enable

CFGCON2(L) – Provides control, selection and locking for various features of the device.

- Deadman timer enable and configuration
- Watchdog timer enable and configuration
- Clock monitoring and control
- Oscillator enable and configuration
- 2-Speed start-up enabled in the Sleep mode bit

CFGCON4(L) – Provides control, selection and locking for various features of the device.

- Deep sleep modules control
- SOSC configuration control

CFGPGQOS – The CFGPGQOS register defines the permission group settings for various bus hosts on the device bus matrix.

USER_ID(L) – The USER_ID register is used to provide the end user with a 16-bit ID field that may be read out directly through the JTAG interface via the USER_ID JTAG instruction.

BCFG0 – Used to set Code Protect, Sign Bit and control PCHE cache mode.

18.2 Applications

18.2.1 Loading User/Device Configuration Registers

The non-BCFG system configuration registers (CFGCON* and CFGCON(L)*) are available in a memory mapped area under software-based locking control.

The following registers that do not have permanent storage:

- Wi-Fi (RF, BBP, PHY) System Configurations
- Calibration values, System PLL
- CRU clock switching
- Analog calibration values

For these registers, the firmware (boot code and/or application code) must allocate nonvolatile storage of system configuration values and load them into the memory mapped system configuration registers during a device boot. The recommended non-volatile storage space in NVR boot memory is shown in the Flash Memory Organization table. See *Flash Memory Organization* from Related Links.

In general, if the Reset value or Flash-loaded value of a system configuration register is the value that is required, then it is not necessary to load the system configuration register during a device boot.

Related Links

24.5.2. Flash Memory Organization

18.2.2 Locking and Unlocking the System Configuration Registers

Write access to the system configuration registers is controlled via the CFGCON0.CFGLOCK[1:0] register bits.

18.2.3 NMI Events

The only system configuration that gets Reset on an NMI event are CPUPG bits in the CFGPGQOS register. This allows application firmware to pass control back to the bootloader and re-enable reads of all configuration words from Boot Flash NVR pages if reads of the Boot Flash pages were disabled using group permissions.

18.2.4 Alternate System Configuration

To provide better data retention for the configuration data (compared to the data retention of the rest of the device Flash), each set of DEVCFGx, FBCFG0 registers are duplicated in an alternate set. This improves the chances of correct system configuration out of Reset. In the event of an uncorrectable error in a word (ECC DED), the Flash controller uses the alternate set to obtain the correct data.

Table 18-2. Alternate Register

Register	Alternate Register
FBCFG0	ALTFBCFG0(0004_5E9C)
DEVCFG0	ALTDEVCFG0(0004_5E98)
DEVCFG1	ALTDEVCFG1(0004_5E94)
DEVCFG2	ALTDEVCFG2(0004_5E90)
DEVCFG4	ALTDEVCFG4(0004_5E8C)
FUSERID	ALTFUSERID(0004_5E88)

18.3 CFG Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CFGCON0(L)	7:0	CPENFILT	ACCMP1_ALTEN	GPSOSCE*	ADCOPVR	JTAGEN	TROEN	SWOEN	TDOEN
		15:8	CFGLOCK[1:0]		IOLOCK	PMDLOCK	PGLOCK	PMULOCK	RTCOUT_ALTEN	RTCIN0_ALTEN
		23:16	SLRTEN2	SLRTEN1	SLRTEN0	HPLUGDIS	SMBUSEN2	SMBUSEN1	SMBUSEN0	VBCMODE
		31:24		FRECCDIS	FECCCON/ECCCTL[1:0]		ADCFCEN	INT0P	INT0E	PCM
0x04 ... 0x0F	Reserved									
0x10	CFGCON1(L)	7:0	ZBTWKSYS		TRCEN	ICESEL[1:0]			DEBUG[1:0]	
		15:8	QSCHE_EN	SMCLR	SLRCTRL2	SLRCTRL1	SLRCTRL0	CLASSBDIS	CMP1_OE	CMP0_OE
		23:16	I2CDSEL2	I2CDSEL1	I2CDSEL0	CCL_OE	SCOM_HSEN[2:0]			QSPI_HSEN
		31:24		CLKZBREF	QSPIDDRM	WDTPSS[4:0]				
0x14 ... 0x1F	Reserved									
0x20	CFGCON2(L)	7:0	PMUTEST_VDD_EN		DMTINTV[2:0]			ACMP_CYCLE[2:0]		
		15:8	FSCMEN	CKSWEN	WAKE2SPD	SOSCSEL	WDTRMCS[1:0]		POSCMD[1:0]	
		23:16	WDTEN	WINDIS	WDTSPGM	WDTPSR[4:0]				
		31:24	DMTEN	DMTCNT[4:0]					WINSZ[1:0]	
0x24 ... 0x3F	Reserved									
0x40	CFGCON4(L)	7:0	SOSC_CFG[7:0]							
		15:8	MLPCLK_MOD	VBKP_DIVSEL	VBKP_32KCSEL[1:0]		VBKP_1KCSEL	RTCEVENT_EN	RTCEVENTSEL[1:0]	
		23:16	DSWDTPS[2:0]			DSZPBOREN	CPEN_DLY[2:0]			RTCEVTYPE
		31:24	RTCNTM_CSEL	LPOSCEN	UVREGROVR	DSBITEN	DSWDTEN	DSWDTLPRC	DSWDTPS[4:3]	
0x44 ... 0x4F	Reserved									
0x50	CFGPGQOS	7:0	ICDJQOS[1:0]		ICDJPG[1:0]		CPUQOS[1:0]		CPUPG[1:0]	
		15:8							DMAPG[1:0]	
		23:16	ICMQOS[1:0]		ICMPG[1:0]		ADCQOS[1:0]		ADCPG[1:0]	
		31:24	WISIBQOS[1:0]		FCQOS[1:0]				DSUPG[1:0]	
0x54 ... 0x5F	Reserved									
0x60	CFGPCLKGEN1	7:0	RCD	FREQMRCSEL[2:0]			EICCD	EICCSEL[2:0]		
		15:8	S01D	SERCOM01CSEL[2:0]			MCD	FREQMMCSEL[2:0]		
		23:16	TCC12CD	TCC12CSEL[2:0]			S23D	SERCOM23CSEL[2:0]		
		31:24	CM4TD	CM4TCSEL[2:0]			TC23CD	TC23CSEL[2:0]		
0x64 ... 0x6F	Reserved									
0x70	CFGPCLKGEN2	7:0	C2D	EVSYSC2SEL[2:0]			C1D	EVSYSC1SEL[2:0]		
		15:8	C4D	EVSYSC4SEL[2:0]			C3D	EVSYSC3SEL[2:0]		
		23:16	C6D	EVSYSC6SEL[2:0]			C5D	EVSYSC5SEL[2:0]		
		31:24	C8D	EVSYSC8SEL[2:0]			C7D	EVSYSC7SEL[2:0]		
0x74 ... 0x7F	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x80	CFGPCLKGEN3	7:0	C10D	EVSYSC10SEL[2:0]			C9D	EVSYSC9SEL[2:0]		
		15:8	C12D	EVSYSC12SEL[2:0]			C11D	EVSYSC11SEL[2:0]		
		23:16	TCC0CD	TCC0CSEL[2:0]			ACCD	ACCLKSEL[2:0]		
		31:24	TC1CD	TC1CSEL[2:0]			TC0CD	TC0CSEL[2:0]		
0x84 ... 0x9F	Reserved									
0xA0	USER_ID	7:0	USER_ID[7:0]							
		15:8	USER_ID[15:8]							
		23:16								
		31:24								
0xA4 ... 0x01FF	Reserved									
0x0200	BCFG0	7:0							PCSCMODE	
		15:8								
		23:16								
		31:24	BINFOVALID0		SIGN	CP				

18.4 Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the “PAC Write-Protection” property in each individual register description.

Some registers are synchronized when read and/or written. Synchronization is denoted by the “Write-Synchronized” or the “Read-Synchronized” property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable protection is denoted by the “Enable-Protected” property in each individual register description.

Note: All registers in this table have corresponding CLR, SET and INV registers at its virtual address, plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET, and INV Registers* from Related Links.

Related Links

6.1.9. CLR, SET and INV Registers

18.4.1 Configuration Control Register 0

Name: CFGCON0(L)
Offset: 0x00
Reset: 0x7100000b
Property: -

The CFGLOCK[1:0] register bits are writable only when CFGLOCK[0] = 1'b0.
The IOLOCK, PMDLOCK and PGLOCK register bits can only be cleared on a system reset. Thereafter, it is controlled as described above.

This register is loaded with trusted data from FBCFG1 during the pre-boot period. Trusted data from Flash means when there is no BCFG* fail status and BINFOVALID = 0 during Flash configuration word reads. If accompanied by a fail status or blank/erase indication, then reset values (described in the register description below) are retained and new values from FBCFG1 are not loaded.

Under all conditions, Flash loading is omitted for the following bits in the CFGCON0 and HPLUGDIS register:

- IOLOCK
- CFGLOCK[1:0]
- PMDLOC
- PGLOCK
- PMULOCK
- JTAGEN

Bit	31	30	29	28	27	26	25	24
		FRECCDIS	FECCCON/ECCCTL[1:0]		ADCFCEN	INT0P	INT0E	PCM
Access		R/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset		1	1	1	0	0	0	1

Bit	23	22	21	20	19	18	17	16
	SLRTEN2	SLRTEN1	SLRTEN0	HPLUGDIS	SMBUSEN2	SMBUSEN1	SMBUSEN0	VBCMODE
Access	R/W/L	R/W/L	R/W/L	R/W	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CFGLOCK[1:0]		IOLOCK	PMDLOCK	PGLOCK	PMULOCK	RTCOUT_ALTEN	RTCIN0_ALTEN
Access	R/W/L	R/W/L	R/S/L	R/S/L	R/S/L	R/S/L	R/W/L	R/W/L
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CPENFILT	ACCMP1_ALTEN	GPSOSCE*	ADCOPVR	JTAGEN	TROEN	SWOEN	TDOEN
Access	R/W/L	R/W/L	R/W/L	R/W	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	0	0	0	1	0	1	1

Bit 30 – FRECCDIS Flex RAM ECC Control

Notes:

- This bit is only writable when CFGLOCK[1:0] is '00'.
- Only a read-only fuse bit, sets the initialization value of RAMECC Control. "True" RAMECC override is available in RAMECC module.

Value	Description
1	ECC is disabled
0	ECC is enabled

Bits 29:28 – FECCCON/ECCCTL[1:0] Flash ECC Control

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
11	ECC and dynamically ECC are disabled
10	ECC and dynamically ECC are disabled
01	Dynamically ECC is enabled
00	ECC is enabled (NVMOP = Word Programming disabled)

Bit 27 – ADCFCEN ADC FC Channel Enable

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Exclusive ADC FC Channel Enable (Disables all second/third class channels)
0	ADC FC Channel Disable (Only second/third class channels are enabled)

Bit 26 – INT0P INT0P Polarity

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	INT0 Polarity (Positive)
0	INT0 Polarity (Negative)

Bit 25 – INT0E INT0 Enable

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	INT0 is enabled
0	INT0 is disabled

Bit 24 – PCM PCHE I/D Cacheable Mode

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Always enabled from outside. Can be further enabled/disabled by PCHE SFR registers.
0	The cache-ability is controlled by the CPU via HPROT[3]. This feature is not available on all the ARM cores.

Bit 23 – SLRTEN2 SLRT Enable for SERCOM2

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Slew rate is enabled
0	Slew rate is disabled

Bit 22 – SLRTEN1 SLRT Enable for SERCOM1

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Slew rate is enabled
0	Slew rate is disabled

Bit 21 – SLRTEN0 SLRT Enable for SERCOM0

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Slew rate is enabled
0	Slew rate is disabled

Bit 20 – HPLUGDIS Hot Plugging Disable (outside fuse loading)

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Hot Plugging is disabled
0	Hot Plugging is enabled

Bit 19 – SMBUSEN2 SMBus Enable for SERCOM2

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	SMBus is enabled
0	SMBus is disabled

Bit 18 – SMBUSEN1 SMBus Enable for SERCOM1

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	SMBus is enabled
0	SMBus is disabled

Bit 17 – SMBUSEN0 SMBus Enable for SERCOM0

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	SMBus is enabled
0	SMBus is disabled

Bit 16 – VBCMODE VBC Operating Mode

Notes:

- This bit is only writable when CFGLOCK[1:0] is '00'.
- Do not change this field if there are pending accesses to VDDBKUPCORE memory map. Failing to do so may result in unexpected data.

Value	Description
1	Indirect addressing. The VDDBKUPCORE IO mapped using PMU Controller.
0	Direct addressing. The VDDBKUPCORE memory mapped on PB-Bridge-B.

Bits 15:14 – CFGLOCK[1:0] Configuration Register Lock

Note: This bit is only writable when CFGLOCK[1:0] is '00' or '10'.

Value	Description
11	All NVR memory self-writes, Boot Configuration (BCFG0) and System Configuration registers (CFG* and USER_ID) are locked and cannot be written – CFGLOCK value cannot be changed.
10	All NVR memory self-writes, Boot Configuration (BCFG0) and System Configuration registers (CFG* and USER_ID) are locked and cannot be written – CFGLOCK value can be changed.
01	Reserved for future use
00	All NVR memory self-writes, Boot Configuration (BCFG0) and System Configuration registers (CFG* and USER_ID) are not locked and can be written – CFGLOCK value can be changed.

Bit 13 – IOLOCK I/O Lock

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	I/O Remap SFR bits are locked and cannot be modified
0	I/O Remap SFR are not locked and can be modified

Bit 12 – PMDLOCK Peripheral Module Disable (PMD) Lock

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	PMDx SFR bits are locked and cannot be modified
0	PMDx SFR bits are not locked and can be modified

Bit 11 – PGLOCK Permission Group Lock

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	CFGPG SFR bits are locked and cannot be modified
0	CFGPG SFR bits are not locked and can be modified

Bit 10 – PMULOCK PMU Controller Register Lock

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	PMU* SFR bits are locked and cannot be modified
0	PMU* SFR bits are not locked and can be modified

Bit 9 – RTCOUT_ALTEN RTCOUT Alternate Enable

Notes:

- This bit is only writable when CFGLOCK[1:0] is '00'.
- RTC alternate output is unavailable on PA10 in sleep modes (Deep Sleep and Extreme Deep Sleep).

Value	Description
1	RTC/OUT is available on PA10
0	RTC/OUT is available on PA4

Bit 8 – RTCIN0_ALTEN RTCIN0 Alternate Enable

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	RTC/IN0 is available on PA9
0	RTC/IN0 is available on PA3

Bit 7 – CPENFILT ADC CP Filter Enable

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	ADC CP filter is enabled
0	ADC CP filter is disabled

Bit 6 – ACCMP1_ALTEN AC CMP1 Alternate Enable

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	AC/CMP1 Out is available on PA6
0	AC/CMP1 Out is available on PA1

Bit 5 – GPSOSCE* GPIO/SOSC Enable* This bit is not applicable to 48-pin devices PIC32CX1012BZ25048.

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	SOSC is selected
0	GPIO is selected

Bit 4 – ADCOPVR ADC Charge Pump Override

Notes:

- This bit is only writable when CFGLOCK[1:0] is '00'.
- This bit is not fuse loadable.

Value	Description
1	Overriden (software controlled)
0	Hardware controlled

Bit 3 – JTAGEN JTAG Enable

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	JTAG port is enabled
0	JTAG port is disabled

Bit 2 – TROEN Trace Output Enable

Notes:

- When CFGCON1.TRCEN = 0, the value of this bit is ignored but has the effect of being '0'.
- This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Starts Trace Clock and enables Trace Outputs (Trace probe must be present)
0	Stops Trace Clock and disables Trace Outputs

Bit 1 – SWOEN SWO enable on 2-wire debug interface

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	SWO is enabled
0	SWO is disabled

Bit 0 – TDOEN TDO enable for 2-wire JTAG

Implementing the JTAG protocol over the 2-wire interface requires four 2-wire clocks for each TCK if TDO is required. However, if the values shifted out TDO are predetermined, then TDO can be disabled, saving two 2-wire clocks.

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	2-wire JTAG protocol uses TDO (Four phase (Full Duplex) protocol)
0	2-wire JTAG protocol does not use TDO (Two phase (Half Duplex) protocol)

18.4.2 Configuration Control Register 1

Name: CFGCON1(L)
Offset: 0x10
Reset: 0x1f00443b
Property: -

This register is loaded with trusted data from FBCFG2 during the pre-boot period. Thereafter, it is controlled as described above.

Trusted data from Flash means when there is no BCFG* fail status during Flash configuration word reads. If accompanied by fail status or blank/erase indication, then reset values (described in the register description below) are retained, and new values from FBCFG2 are not loaded.

Under all conditions, Flash loading is omitted for the following bits in the CFGCON1 register:

- DEBUG[1:0]

Bit	31	30	29	28	27	26	25	24
		CLKZBREF	QSPIDDRM	WDTPSS[4:0]				
Access		R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset		0	0	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	I2CDSEL2	I2CDSEL1	I2CDSEL0	CCL_OE	SCOM_HSEN[2:0]			QSPI_HSEN
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	QSCHE_EN	SMCLR	SLRCTRL2	SLRCTRL1	SLRCTRL0	CLASSBDIS	CMP1_OE	CMP0_OE
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	1	0	0	0	1	0	0

Bit	7	6	5	4	3	2	1	0
	ZBTWKSYS		TRCEN	ICESEL[1:0]			DEBUG[1:0]	
Access	R/W/L		R/W/L	R/W/L	R/W/L		R/W/L	R/W/L
Reset	0		1	1	1		1	1

Bit 30 – CLKZBREF External Reference Clock Zigbee Enable
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Enable clk_zb_to_ext on PPS.REFO1
0	No clk_zb_to_ext on PPS.REFO1, PPS.REFO1 is unchanged

Bit 29 – QSPIDDRM QSPI DDR Mode Clock Enable
Notes:

- When using the QSPI DDR Mode, System Clock (SYS_CLK) must be <= 48 MHz.
- This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	QSPI DDR Mode Clock Enable
0	Disabled

Bits 28:24 – WDTPSS[4:0] Watchdog Timer Post-scale Select Sleep bits
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
10100	1:1048576
10011	1:524288

Value	Description
10010	1:262144
10001	1:131072
10000	1:65536
01111	1:32768
01110	1:16384
01101	1:8192
01100	1:4096
01011	1:2048
01010	1:1024
01001	1:512
01000	1:256
00111	1:128
00110	1:64
00101	1:32
00100	1:16
00011	1:8
00010	1:4
00001	1:2
00000	1:1

Bit 23 – I2CDSEL2 I2C Delay Select for SERCOM2
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	I^2C Delay Enabled
0	I^2C Delay Disabled

Bit 22 – I2CDSEL1 I2C Delay Select for SERCOM1
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	I^2C delay is enabled
0	I^2C delay is disabled

Bit 21 – I2CDSEL0 I2C Delay Select for SERCOM0
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	I^2C delay is enabled
0	I^2C delay is disabled

Bit 20 – CCL_OE CCL Pads (via PPS) Output Enable
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	CCL pads (via PPS) output is enabled
0	CCL pads (via PPS) output is disabled

Bits 19:17 – SCOM_HSEN[2:0] SCOM (Direct) Enable, 17 = SCOM0, 18 = SCOM1, 19 = SCOM2
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Direct mode (High-Speed)
0	via PPS

Bit 16 – QSPI_HSEN QSPI (Direct) Enable

Notes:

- This bit is only writable when CFGLOCK[1:0] is '00'.
- In Direct mode.

Value	Description
1	Direct mode (High-Speed)
0	via PPS

Bit 15 – QSCHE_EN QSPI Address Space Cache Attribute

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Cache attribute is enabled
0	Caching is disabled

Bit 14 – SMCLR Selects CRU handling of MCLR Control

Note: This bit is only writable when CFGLOCK[1:0] is '00' or '10'.

Value	Description
1	Legacy mode (system clear does not reset all state of device)
0	MCLR causes a faux POR

Bit 13 – SLRCTRL2 I2C Delay Select for SERCOM2

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Slew rate control is configured via SERCOM configuration
0	Slew rate control is configured via GPIO configuration

Bit 12 – SLRCTRL1 I2C Delay Select for SERCOM1

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Slew rate control is configured via SERCOM configuration
0	Slew rate control is configured via GPIO configuration

Bit 11 – SLRCTRL0 I2C Delay Select for SERCOM0

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Slew rate control is configured via SERCOM configuration
0	Slew rate control is configured via GPIO configuration

Bit 10 – CLASSBDIS Disable CLASSB Device Functionality

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	CLASSB functions are disabled
0	CLASSB functions are enabled

Bit 9 – CMP1_OE Analog Comparator-1 Output Enable

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	AC1 Output is enabled
0	AC1 Output is disabled

Bit 8 – CMP0_OE Analog Comparator-0 Output Enable

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	AC0 Output is enabled
0	AC0 Output is disabled

Bit 7 – ZBTWKSYS Zigbee Bluetooth Subsystem External Wake-up source

Notes:

- Write-only bit, with read-as-zero; when written to '1', creates one clk_lp_cycle wide pulse on Zigbee Bluetooth Subsystem.external_NMI0 pin. This enables external system wake-up to Bluetooth subsystem. This allows CPU and Bluetooth subsystem wake-up/sleep to be independent of each other.
- Flash fuse loading is excluded for this bit.

Bit 5 – TRCEN Trace Enable

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Trace features in the CPU are enabled
0	Trace features in the CPU are disabled

Bits 4:3 – ICESEL[1:0] EMUC/EMUD Communication Channel Select

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
11	ICE EMUC1/EMUD1 pins are shared with PGC1/PGD1
10	ICE EMUC2/EMUD2 pins are shared with PGC2/PGD2
01	ICE EMUC3/EMUD3 pins are shared with PGC3/PGD3 (Not used on this device)
00	ICE EMUC4/EMUD4 pins are shared with PGC4/PGD4

Bits 1:0 – DEBUG[1:0] Background Debugger Access Selection

Notes:

1. JTAGEN = 0 prevents 4-wire JTAG Debugging but not EMUC/EMUD debugging.
2. If CPN = 0, then the JTAG TAP controller denies access to the EJTAG TAP Controller (i.e., the SWTAP command is ignored) and, therefore, external access to the debugging features is denied.
3. This bit is only writable when CFGLOCK[1:0] = '00'.

Value	Description
11	4-wire JTAG I/F is enabled; EMUC/EMUD is disabled; ICD module is disabled
10	4-wire JTAG I/F is enabled; EMUC/EMUD is disabled; ICD module is enabled
01	EMUC/EMUD is enabled; 4-wire JTAG I/F is disabled; ICD module is disabled
00	EMUC/EMUD is enabled; 4-wire JTAG I/F is disabled; ICD module is enabled

18.4.3 Configuration Control Register 2

Name: CFGCON2(L)
Offset: 0x20
Reset: 0x00
Property: -

This register is loaded with trusted data from FBCFG3 during pre-boot period. Thereafter, it is controlled as described above.

Trusted data from Flash means when there is no BCFG* fail status during Flash configuration word reads. If accompanied by fail status or blank/erase indication then reset values (described in the register description below) are retained and new values from FBCFG3 are not loaded.

Bit	31	30	29	28	27	26	25	24
	DMTEN	DMTCNT[4:0]					WINSZ[1:0]	
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	WDTEN	WINDIS	WDTSPGM	WDTPSR[4:0]				
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	FSCMEN	CKSWEN	WAKE2SPD	SOSCSEL	WDTRMCS[1:0]		POSCMD[1:0]	
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	PMUTEST_VDD_EN		DMTINTV[2:0]			ACMP_CYCLE[2:0]		
Access	R/W/L		R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0		1	1	1	0	0	0

Bit 31 – DMTEN Dead Man Timer Enable bit

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	DMT is enabled
0	DMT is disabled (control is placed on the DMTCON.ON bit)

Bits 30:26 – DMTCNT[4:0] Dead Man Timer Count Select bits

Notes:

- This bit is only writable when CFGLOCK[1:0] is '00'.
- On devices where the DMTCNT[4:0] configuration field is loaded.

Value	Description
00000	Counter value is 2^8 for cfg_dmt_cnt[31:0]
00001	Counter value is 2^9 for cfg_dmt_cnt[31:0]
...	...
10100	Counter value is 2^28 for cfg_dmt_cnt[31:0]
10101	Counter value is 2^29 for cfg_dmt_cnt[31:0]
10110	Counter value is 2^30 for cfg_dmt_cnt[31:0]
10111	Counter value is 2^31 for cfg_dmt_cnt[31:0]
11000 - 11111	Reserved

Bits 25:24 – WINSZ[1:0] Watchdog Timer Window Size bits

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
00	Window size is 75%
01	Window size is 50%
10	Window size is 37.5%
11	Window size is 25%

Bit 23 – WDTEN Watchdog Timer Enable bit

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	WDT is enabled
0	WDT is disabled (control is placed on the SWDTEN bit)

Bit 22 – WINDIS Windowed Watchdog Timer Disable bit

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Standard WDT is selected; windowed WDT disabled
0	Windowed WDT is enabled

Bit 21 – WDTSPGM Watchdog Timer Stop during Flash Programming bit

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	The WDT stops during NVR programming (legacy)
0	The WDT runs during NVR programming (for read/execute while programming Flash systems)

Bits 20:16 – WDTPSR[4:0] Watchdog Timer Post-scale Select Run bits

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
10100	1:1048576
10011	1:524288
10010	1:262144
10001	1:131072
10000	1:65536
01111	1:32768
01110	1:16384
01101	1:8192
01100	1:4096
01011	1:2048
01010	1:1024
01001	1:512
01000	1:256
00111	1:128
00110	1:64
00101	1:32
00100	1:16
00011	1:8
00010	1:4
00001	1:2
00000	1:1

Bit 15 – FSCMEN Fail-Safe Clock Monitor Enable

Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	FSCM enabled
0	FSCM disabled

Bit 14 – CKSWEN Software Clock Switching Enable
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Software clock switching is enabled
0	Software clock switching is disabled

Bit 13 – WAKE2SPD 2-Speed startup enabled in Sleep mode bit
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	When the device exits the Sleep mode, the SYS_CLK will be from FRC until the selected clock is ready.
0	Not applicable.

Bit 12 – SOSCSEL SOSC Selection Configuration bit
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Crystal (SOSCI/SOSCO) mode is selected
0	Digital (SCLKI) mode is selected

Bits 11:10 – WDTRMCS[1:0] WDT RUN Mode Clock Select
Note: This bit is only writable when CFGLOCK[1:0] is '00' or '10'.

Value	Description
11	LPRC
10	Do not use
01	Do not use
00	Module PB Clock

Bits 9:8 – POSCMD[1:0] Primary Oscillator Configuration bits
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
11	Primary oscillator is disabled
10	HS oscillator mode is selected
01	HS oscillator mode is selected
00	HS oscillator mode is selected

Bit 7 – PMUTEST_VDD_EN PMU Test Output or VDD/2 Enable via ADC IE12
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	PMU test output monitor is enabled
0	VDD/2 monitor is enabled

Bits 5:3 – DMTINTV[2:0] Dead Man Timer Count Window Interval bits
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
000	Window/Interval value is zero for cfg_dmt_intv[31:0] - windowed mode is disabled
001	Window/Interval value is 1/2 Counter value for cfg_dmt_intv[31:0]
010	Window/Interval value is 3/4 Counter value for cfg_dmt_intv[31:0]
011	Window/Interval value is 7/8 Counter value for cfg_dmt_intv[31:0]
100	Window/Interval value is 15/16 Counter value for cfg_dmt_intv[31:0]

Value	Description
101	Window/Interval value is 31/32 Counter value for cfg_dmt_intv[31:0]
110	Window/Interval value is 63/64 Counter value for cfg_dmt_intv[31:0]
111	Window/Interval value is 127/128 Counter value for cfg_dmt_intv[31:0]

Bits 2:0 – ACMP_CYCLE[2:0] AC SIB Comparator Result Wait Cycles
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
n	Wait for 32µs* ACMP_CYCLE+1 cycles to generate comparator done indication

18.4.4 Configuration Control Register 4

Name: CFGCON4(L)
Offset: 0x40
Reset: 0x840e4000
Property: -

This register is loaded with trusted data from FBCFG4/DEVCFG4 during the pre-boot period.

Trusted data from Flash means when there is no BCFG* fail status during Flash configuration word reads. If accompanied by fail status BCFGFAIL (RCON[26]) or blank/erase indication, then reset values (described in the following register description) are retained and new values from FBCFG4 are not loaded.

Bit	31	30	29	28	27	26	25	24
	RTCNTM_CSEL	LPOSCEN	UVREGROVR	DSBITEN	DSWDTEN	DSWDTLPRC	DSWDTPS[4:3]	
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	1	0	0	0	0	1	0	0

Bit	23	22	21	20	19	18	17	16
	DSWDTPS[2:0]			DSZPBOREN	CPEN_DLY[2:0]			RTCEVTYPE
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	0	0	0	1	1	1	0

Bit	15	14	13	12	11	10	9	8
	MLPCLK_MOD	VBKP_DIVSEL	VBKP_32KCSEL[1:0]		VBKP_1KCSEL	RTCEVENT_EN	RTCEVENTSEL[1:0]	
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	1	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SOSC_CFG[7:0]							
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	0	0	0	0	0	0	0

Bit 31 – RTCNTM_CSEL RTCC Counter Mode Clock Select
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Raw 32 KHz clock
0	Processed 32 KHz clock

Bit 30 – LPOSCEN Low Power (Secondary) Oscillator Enable
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Low Power (Secondary) Oscillator, also at Reset is enabled
0	Low Power (Secondary) Oscillator is disabled

Bit 29 – UVREGROVR ULPVREG Retention Mode Override
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	ULPVREG forced in the Retention mode
0	ULPVREG controlled by XDS/DS FSM

Bit 28 – DSBITEN Deep Sleep Bit Enable
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	DS bit in DSCON is enabled
0	DS bit in DSCON is disabled

Bit 27 – DSWDTEN Deep Sleep Watchdog Timer Enable
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	DSWDT during deep sleep is enabled
0	DSWDT during deep sleep is disabled

Bit 26 – DSWDTLPRC Deep Sleep Watchdog Timer Reference Clock Select
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Select LPRC as DSWDT reference clock
0	Select SOSC as DSWDT reference clock

Bits 25:21 – DSWDTPS[4:0] Deep Sleep Watchdog Timer Postscale Select
The DS WDT prescaler is 32; this creates an approximate base time unit of 1 ms.
Note: These bits are only writable when CFGLOCK[1:0] is '00'.

Value	Description
11111	$1:2^{36}$ (25.7 days)
11110	$1:2^{35}$ (12.8 days)
11101	$1:2^{34}$ (6.4 days)
11100	$1:2^{33}$ (77.0 hours)
11011	$1:2^{32}$ (38.5 hours)
11010	$1:2^{31}$ (19.2 hours)
11001	$1:2^{30}$ (9.6 hours)
11000	$1:2^{29}$ (4.8 hours)
10111	$1:2^{28}$ (2.4 hours)
10110	$1:2^{27}$ (72.2 minutes)
10101	$1:2^{26}$ (36.1 minutes)
10100	$1:2^{25}$ (18.0 minutes)
10011	$1:2^{24}$ (9.0 minutes)
10010	$1:2^{23}$ (4.5 minutes)
10001	$1:2^{22}$ (135.3 s)
10000	$1:2^{21}$ (67.7 s)
01111	$1:2^{20}$ (33.825 s)
01110	$1:2^{19}$ (16.912 s)
01101	$1:2^{18}$ (8.456 s)
01100	$1:2^{17}$ (4.228 s)
01011	1:65536 (2.114 s)
01010	1:32768 (1.057 s)
01001	1:16384 (528.5 ms)
01000	1:8192 (264.3 ms)
00111	1:4096 (132.1 ms)
00110	1:2048 (66.1 ms)
00101	1:1024 (33 ms)
00100	1:512 (16.5 ms)
00011	1:256 (8.3 ms)
00010	1:128 (4.1 ms)
00001	1:64 (2.1 ms)
00000	1:32 (1 ms)

Bit 20 – DSZPBOREN Deep Sleep Zero-Power BOR Enable
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	ZPBOR during deep sleep is enabled
0	ZPBOR during deep sleep is disabled

Bits 19:17 – CPEN_DLY[2:0] Charge-pump Ready Digital Delay (Safety delay to Analog CP Ready)
n = (n+1) LPRC Clock Cycle Delay
Note: These bits are only writable when CFGLOCK[1:0] is '00'.

Bit 16 – RTCEVTYPE RTCC Event Type
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	RTC_EVENT
0	RTC_OUT

Bit 15 – MLPCLK_MOD LPCLK Modifier in Counter/Delay Mode
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Divide-by 1.024 (Recommended, when LPCLK = 32.768 KHz)
0	Divide-by 1 (Recommended, when LPCLK = 32 KHz)

Bit 14 – VBKP_DIVSEL VDDBUKPCORE LPCLK Clock Divider Selection
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Divide by 31.25 (Recommended, when LPCLK = 32 KHz)
0	Divide-by 32 (Recommended, when LPCLK = 32.768 KHz)

Bits 13:12 – VBKP_32KCSEL[1:0] VDDBUKPCORE 32 KHz Clock Source Selection
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
11	LPRC
10	SOSC
01	POSC
00	FRC

Bit 11 – VBKP_1KCSEL VDDBUKPCORE LPCLK Clock Selection
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Divide by 32 or 31.25 clock depending on VBKP_DIVSEL
0	32 KHz low power clock

Bit 10 – RTCEVENT_EN Output Enable for RTCC Event Output
Note: This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	RTCC-Event output is enabled
0	RTCC-Event output is disabled

Bits 9:8 – RTCEVENTSEL[1:0] RTCC Event Selection
Note: These bits are only writable when CFGLOCK[1:0] is '00'.

Value	Description
00	1-Second clock
01	Alarm pulse
1x	32 KHz clock

Bits 7:0 – SOSC_CFG[7:0] SOSC Configuration Bits

Note: These bits are only writable when CFGLOCK[1:0] is '00'.

18.4.5 Permission Group Configuration

Name: CFGPGQOS
Offset: 0x50
Reset: 0xe040004c
Property: -

All bits in this register are writable only when CFGCON0.PGLOCK = 0.

There is no Flash location for this register because the purpose of this register is to provide a software-based protection mechanism to a device memory-mapped region, which is typically handled by a trusted boot/OScode.

Note: Ensure this register is programmed to the values shown: 0xE040_004C if you are not using Microchip-provided boot code.

Bit	31	30	29	28	27	26	25	24
	WISIBQOS[1:0]		FCQOS[1:0]				DSUPG[1:0]	
Access	R/W/L	R/W/L	R/W/L	R/W/L			R/W/L	R/W/L
Reset	1	1	1	0			0	0

Bit	23	22	21	20	19	18	17	16
	ICMQOS[1:0]		ICMPG[1:0]		ADCQOS[1:0]		ADCPG[1:0]	
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
							DMAPG[1:0]	
Access							R/W/L	R/W/L
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	ICDJQOS[1:0]		ICDJPG[1:0]		CPUQOS[1:0]		CPUPG[1:0]	
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	1	0	0	1	1	0	0

Bits 31:30 – WISIBQOS[1:0] Wireless SIB QOS Control bits
Note: This field is only writable when CFGCON0.PGLOCK = 0.

Value	Description
00	Disable; Background
01	Low; Sensitive bandwidth
10	Medium; Sensitive latency
11	High; Critical latency

Bits 29:28 – FCQOS[1:0] FC Controller QOS Control bits
Note: This field is only writable when CFGCON0.PGLOCK = 0.

Value	Description
00	Disable; Background
01	Low; Sensitive bandwidth
10	Medium; Sensitive latency
11	High; Critical latency

Bits 25:24 – DSUPG[1:0] DSU Permission Group, drive the inputs cfg_dsu_pg[1:0] directly to the SSX
The DSU bus host has access to Access Controlled memory regions as defined by the bit-fields RDPER[3:0] and WRPER[3:0] in the Bus Structure's Permission Groups SFRs for these memory regions. The encoding works as follows:

- DSUPG[1:0] == 2'b11 : Read Access if RDPER[3]==1, Write Access if WRPER[3]==1 (Perm. Grp. 3)

- DSUPG[1:0] == 2'b10 : Read Access if RDPER[2]==1, Write Access if WRPER[2]==1 (Perm. Grp. 2)
- DSUPG[1:0] == 2'b01 : Read Access if RDPER[1]==1, Write Access if WRPER[1]==1 (Perm. Grp. 1)
- DSUPG[1:0] == 2'b00 : Read Access if RDPER[0]==1, Write Access if WRPER[0]==1 (Perm. Grp. 0)

Note: This field is only writable when CFGCON0.PGLOCK = 0.

Bits 23:22 – ICMQOS[1:0] ICM QOS Control bits

Note: This field is only writable when CFGCON0.PGLOCK = 0.

Value	Description
00	Disable; Background
01	Low; Sensitive bandwidth
10	Medium; Sensitive latency
11	High; Critical latency

Bits 21:20 – ICMPG[1:0] ICM Permission Group, drive the inputs cfg_icm_pg[1:0] directly to the SSX

The ICM bus host has access to Access Controlled memory regions as defined by the bit-fields RDPER[3:0] and WRPER[3:0] in the Bus Structure's Permission Groups SFRs for these memory regions. The encoding works as follows:

- ICMPG[1:0] == 2'b11 : Read Access if RDPER[3]==1, Write Access if WRPER[3]==1 (Perm. Grp. 3)
- ICMPG[1:0] == 2'b10 : Read Access if RDPER[2]==1, Write Access if WRPER[2]==1 (Perm. Grp. 2)
- ICMPG[1:0] == 2'b01 : Read Access if RDPER[1]==1, Write Access if WRPER[1]==1 (Perm. Grp. 1)
- ICMPG[1:0] == 2'b00 : Read Access if RDPER[0]==1, Write Access if WRPER[0]==1 (Perm. Grp. 0)

Note: This field is only writable when CFGCON0.PGLOCK = 0.

Bits 19:18 – ADCQOS[1:0] ADC Controller QOS Control bits

Note: This field is only writable when CFGCON0.PGLOCK = 0.

Value	Description
00	Disable; Background
01	Low; Sensitive bandwidth
10	Medium; Sensitive latency
11	High; Critical latency

Bits 17:16 – ADCPG[1:0] ADC Controller Permission Group, drive the inputs cfg_adc_pg[1:0] directly to the SSX

The ADC bus host has access to Access Controlled memory regions as defined by the bit-fields RDPER[3:0] and WRPER[3:0] in the Bus Structure's Permission Groups SFRs for these memory regions. The encoding works as follows:

- ADCPG[1:0] == 2'b11 : Read Access if RDPER[3]==1, Write Access if WRPER[3]==1 (Perm. Grp. 3)
- ADCPG[1:0] == 2'b10 : Read Access if RDPER[2]==1, Write Access if WRPER[2]==1 (Perm. Grp. 2)
- ADCPG[1:0] == 2'b01 : Read Access if RDPER[1]==1, Write Access if WRPER[1]==1 (Perm. Grp. 1)
- ADCPG[1:0] == 2'b00 : Read Access if RDPER[0]==1, Write Access if WRPER[0]==1 (Perm. Grp. 0)

Note: This field is only writable when CFGCON0.PGLOCK = 0.

Bits 9:8 – DMAPG[1:0] DMA (Rd/Wr) Permission Group, drive the inputs cfg_dma_pg[1:0] directly to the SSX

The DMA bus host has access to Access Controlled memory regions as defined by the bit-fields RDPER[3:0] and WRPER[3:0] in the Bus Structure's Permission Groups SFRs for these memory regions. The encoding works as follows:

- DMAPG[1:0] == 2'b11 : Read Access if RDPER[3]==1, Write Access if WRPER[3]==1 (Perm. Grp. 3)
- DMAPG[1:0] == 2'b10 : Read Access if RDPER[2]==1, Write Access if WRPER[2]==1 (Perm. Grp. 2)
- DMAPG[1:0] == 2'b01 : Read Access if RDPER[1]==1, Write Access if WRPER[1]==1 (Perm. Grp. 1)
- DMAPG[1:0] == 2'b00 : Read Access if RDPER[0]==1, Write Access if WRPER[0]==1 (Perm. Grp. 0)

Note: This field is only writable when CFGCON0.PGLOCK = 0.

Bits 7:6 – ICDJQOS[1:0] ICD-JTAG Bus QOS Control bits
Note: This field is only writable when CFGCON0.PGLOCK = 0.

Value	Description
00	Disable; Background
01	Low; Sensitive bandwidth
10	Medium; Sensitive latency
11	High; Critical latency

Bits 5:4 – ICDJPG[1:0] ICD-JTAG Permission Group, drive the inputs cfg_icdj_pg[1:0] directly to the SSX
The ICD-JTAG bus host has access to Access Controlled memory regions as defined by the bit-fields RDPER[3:0] and WRPER[3:0] in the Bus Structure's Permission Groups SFRs for these memory regions. The encoding works as follows:

- ICDJPG[1:0] == 2'b11 : Read Access if RDPER[3]==1, Write Access if WRPER[3]==1 (Perm. Grp. 3)
- ICDJPG[1:0] == 2'b10 : Read Access if RDPER[2]==1, Write Access if WRPER[2]==1 (Perm. Grp. 2)
- ICDJPG[1:0] == 2'b01 : Read Access if RDPER[1]==1, Write Access if WRPER[1]==1 (Perm. Grp. 1)
- ICDJPG[1:0] == 2'b00 : Read Access if RDPER[0]==1, Write Access if WRPER[0]==1 (Perm. Grp. 0)

Note: This field is only writable when CFGCON0.PGLOCK = 0.

Bits 3:2 – CPUQOS[1:0] CPU I/D and System Bus QOS Control bits
Note: This field is only writable when CFGCON0.PGLOCK = 0.

Value	Description
00	Disable; Background
01	Low; Sensitive bandwidth
10	Medium; Sensitive latency
11	High; Critical latency

Bits 1:0 – CPUPG[1:0] CPU (Code) Permission Group, drive the inputs cfg_cpu_pg[1:0] directly to the SSX
The CPU Bus host has access to Access Controlled memory regions as defined by the bit fields RDPER[3:0] and WRPER[3:0] in the Bus Structure's Permission Groups SFRs for these memory regions. The encoding works as follows:

- CPUPG[1:0] == 2'b11 : Read Access if RDPER[3]==1, Write Access if WRPER[3]==1 (Perm. Grp. 3)
- CPUPG[1:0] == 2'b10 : Read Access if RDPER[2]==1, Write Access if WRPER[2]==1 (Perm. Grp. 2)
- CPUPG[1:0] == 2'b01 : Read Access if RDPER[1]==1, Write Access if WRPER[1]==1 (Perm. Grp. 1)
- CPUPG[1:0] == 2'b00 : Read Access if RDPER[0]==1, Write Access if WRPER[0]==1 (Perm. Grp. 0)

Notes:

- CPUPG[1:0] automatically reverts to 2'b00 when the CPU acknowledges entering into an NMI exception as indicated by its STAUS[NMI] bit, which is carried by the cpu1_si_nmitaken system signal.
- This field is only writable when CFGCON0.PGLOCK = 0.

18.4.6 Peripheral Clock Generator 1

Name: CFGPCLKGEN1
Offset: 0x60
Reset: 0x00
Property: -

The CFGPCLKGEN1 dictates the peripheral clock selection described in the *Clock System* chapter.

There is no Flash location for this register because the purpose of this register is to provide an application-based peripheral clocking selection. This is best handled in the application software drivers.

Bit	31	30	29	28	27	26	25	24
	CM4TD	CM4TCSEL[2:0]			TC23CD	TC23CSEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TCC12CD	TCC12CSEL[2:0]			S23D	SERCOM23CSEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	S01D	SERCOM01CSEL[2:0]			MCD	FREQMMCSEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RCD	FREQMRCSEL[2:0]			EICCD	EICCSEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – CM4TD Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 30:28 – CM4TCSEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 27 – TC23CD Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 26:24 – TC23CSEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected

Value	Description
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 23 – TCC12CD Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 22:20 – TCC12CSEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 19 – S23D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 18:16 – SERCOM23CSEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 15 – S01D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 14:12 – SERCOM01CSEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 11 – MCD Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 10:8 – FREQMMCSEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 7 – RCD Peripheral Clock Disable

Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 6:4 – FREQMRCSEL[2:0] Peripheral Clock Selection

Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 3 – EICCD Peripheral Clock Disable

Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 2:0 – EICCSEL[2:0] Peripheral Clock Selection

Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

18.4.7 Peripheral Clock Generator 2

Name: CFGPCLKGEN2
Offset: 0x70
Reset: 0x00
Property: -

The CFGPCLKGEN2 dictates the peripheral clock selection described in the *Clock System* chapter.

There is no Flash location for this register because the purpose of this register is to provide an application-based peripheral clocking selection. This is best handled in the application software drivers.

Bit	31	30	29	28	27	26	25	24
	C8D	EVSYSC8SEL[2:0]			C7D	EVSYSC7SEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	C6D	EVSYSC6SEL[2:0]			C5D	EVSYSC5SEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	C4D	EVSYSC4SEL[2:0]			C3D	EVSYSC3SEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	C2D	EVSYSC2SEL[2:0]			C1D	EVSYSC1SEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – C8D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 30:28 – EVSYSC8SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 27 – C7D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 26:24 – EVSYSC7SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected

Value	Description
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 23 – C6D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 22:20 – EVSYSC6SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 19 – C5D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 18:16 – EVSYSC5SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 15 – C4D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 14:12 – EVSYSC4SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 11 – C3D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 10:8 – EVSYSC3SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 7 – C2D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 6:4 – EVSYSC2SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 3 – C1D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 2:0 – EVSYSC1SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

18.4.8 Peripheral Clock Generator 3

Name: CFGPCLKGEN3
Offset: 0x80
Reset: 0x00
Property: -

The CFGPCLKGEN3 dictates the peripheral clock selection described in the *Clock System* chapter.

There is no Flash location for this register because the purpose of this register is to provide an application-based peripheral clocking selection. This is best handled in the application software drivers.

Bit	31	30	29	28	27	26	25	24
	TC1CD	TC1CSEL[2:0]			TC0CD	TC0CSEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TCC0CD	TCC0CSEL[2:0]			ACCD	ACCLKSEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	C12D	EVSYSC12SEL[2:0]			C11D	EVSYSC11SEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	C10D	EVSYSC10SEL[2:0]			C9D	EVSYSC9SEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – TC1CD Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 30:28 – TC1CSEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 27 – TC0CD Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 26:24 – TC0CSEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected

Value	Description
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 23 – TCC0CD Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 22:20 – TCC0CSEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 19 – ACCD Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 18:16 – ACCLKSEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 15 – C12D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 14:12 – EVSYSC12SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 11 – C11D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 10:8 – EVSYSC11SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 7 – C10D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 6:4 – EVSYSC10SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

Bit 3 – C9D Peripheral Clock Disable
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	Clock is disabled
1	Clock is enabled

Bits 2:0 – EVSYSC9SEL[2:0] Peripheral Clock Selection
Note: This field is only writable when CFGLOCK[1:0] = 00.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

18.4.9 User Unique ID

Name: USER_ID
Offset: 0xA0
Reset: 0x00
Property: -

The User ID is a 16-bit ID that may be programmed to differentiate products that use the same device. The User ID value may be read directly out of the USER_ID register or through the JTAG interface via the MCHP_CMD.USER ID command.
There is no dedicated JTAG status bit to indicate when the User ID value is loaded into the USER_ID register and is ready to be read from JTAG. It is assumed that a non-zero value for the User ID will be used to indicate that the User ID is loaded.

The USER_ID register is reset on power-up, then is loaded with trusted data from FBCFG5 during the pre-boot period and it is controlled.

Trusted data from Flash means when there is no BCFG* fail status during Flash configuration word reads. If accompanied by fail status or blank/erase indication, then Reset values (described in the register description below) are retained and new values from FBCFG5 are not loaded.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	USER_ID[15:8]							
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	USER_ID[7:0]							
Access	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – USER_ID[15:0] User unique ID, readable using the JTAG USER_ID instruction
Note: This field is only writable when CFGLOCK[1:0] = 00.

18.4.10 Boot Configuration 0

Name: BCFG0
Offset: 0x200
Reset: 0x00
Property: -

Note: Safe value of BCFG0 is 0xFFFF_FFFF as applicable only to the implemented bits.

Bit	31	30	29	28	27	26	25	24
	BINFOVALID0		SIGN	CP				
Access	R		R	R				
Reset	c		c	c				

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							PCSCMODE	
Access							R	
Reset							c	

Bit 31 – BINFOVALID0 First 256-bit BCFG information is valid

Notes:

1. This bit is added to provide a mechanism to determine if information coming from Flash is valid or invalid. The BCFG area is critical to the device boot-up.
2. Trusted FBCFG* data = (BINFOVALID = 0) and (BCFGFAIL = 0).
3. It is recommended that the application program this bit to zero for proper operation.

Value	Description
1	FBCFG0 to FBCFG5 is not valid (Untrusted, Flash values are ignored and safe values are used)
0	FBCFG0 to FBCFG5 is valid (Trusted and loaded from Flash)

Bit 29 – SIGN Flash SIGN bit

Value	Description
1	Unsigned
0	Signed

Bit 28 – CP Boot Code Protect (~FCPN0.CPN && ~FSIGN0.SIGN).

Value	Description
1	Protection is enabled
0	Protection is disabled

Bit 1 – PCSCMODE PCHE Single Cache mode

Value	Description
1	PCHE ICache Only. CPU Instructions (code, data) go to PCHE ICache only.
0	PCHE ICache and DCache. CPU opcodes go to PCEHE ICache port and data goes to PCHE DCache port.

19. Register Locking

This device provides several different types of register-level locking:

Locking using the System Lock Register – This mechanism, described in System Lock Register (see *System Lock Register* from Related Links), provides for a 2-way (locking and unlocking) write lock of system critical registers. It includes protection for the following registers:

- CRU.OSCCON
- CRU.OSCTRM
- CRU.SPLLCON
- CRU.RSWRST
- CRU.RNMICON
- CRU.PB1DIV
- CRU.PB2DIV
- CRU.PB3DIV
- CRU.SLEWCON
- CRU.CLK_DIAG

Locking using the CFGCON0.IOLOCK, CFGCON0.PMDLOCK, CFGCON0.PMUCLOCK and CFGCON0.PGLOCK register bits – This mechanism provides for a 1-way lock (once locked, only a reset can unlock) of the following registers:

- All PPS registers (IOLOCK bit)
- All PMD registers (PMDLOCK bit)
- CFGPG register (PGLOCK bit)
- All PMU registers (PMULOCK bit)

Locking using the CFGCON0.CFGLOCK[1:0] register bits – This mechanism provides for a 1-way or 2-way lock (software selectable). It applies to the following registers and memories:

- BCFG0
- CFGCON0
- CFGCON1
- CFGCON2
- CFGCON3
- USER_ID
- CFGCON4
- CFGPCLKGENx

Related Links

19.1. System Lock Register

19.1 System Lock Register

Several modules contain registers that are protected from errant code causing unwanted changes by the system lock feature. When the system lock is in effect, which is the default state, registers protected by it are not writable. The system lock feature protects registers that are system critical such as the boot time option.

Each module that uses the system lock feature describes which register bits and functions it affects. A specific sequence of writes to the SYSKEY register unlock the access to those register bits and features.

19.1.1 Unlock Requirements

The unlock sequence must be atomic. If any other peripheral bus access occurs on the same peripheral bus on which SYSKEY resides during the unlock attempt sequence, the unlock fails. Therefore, turn off all bus initiators like DMA, USB and so on, and disable interrupts.

19.1.2 Unlock Sequence

It is recommended that application code performs step 2 and 3 before step 5 and 6. The unlock sequencer, however, only looks for step 5 and 6 to be atomic. For this sequence, atomic means that there is no other activity on the peripheral bus between step 5 and 6. Step 4 is only needed to ensure that the sequence starts from a known locked state.

1. Suspend all other Peripheral Bus accesses.
2. Load 0xAA996655 to CPU register X.
3. Load 0x556699AA to CPU register Y.
4. Store CPU register r0 to SYSKEY.
5. Store CPU register X to SYSKEY.
6. Store CPU register Y to SYSKEY.

19.1.3 Lock Sequence

When the system is unlocked, any write to the SYSKEY register causes the system lock to become active.

19.1.4 Lock/Unlock Indication

The SYSKEY register read value indicates the status of the unlock sequence. A value of 0x00000000 indicates the system is still locked. A value of 0x00000001 indicates the sequence succeeded and the system is unlocked.

19.2 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00 ... 0xAF	Reserved									
0xB0	SYSKEY	7:0	SYSKEY7	SYSKEY6	SYSKEY5	SYSKEY4	SYSKEY3	SYSKEY2	SYSKEY1	SYSKEY0
		15:8	SYSKEY15	SYSKEY14	SYSKEY13	SYSKEY12	SYSKEY11	SYSKEY10	SYSKEY9	SYSKEY8
		23:16	SYSKEY23	SYSKEY22	SYSKEY21	SYSKEY20	SYSKEY19	SYSKEY18	SYSKEY17	SYSKEY16
		31:24	SYSKEY31	SYSKEY30	SYSKEY29	SYSKEY28	SYSKEY27	SYSKEY26	SYSKEY25	SYSKEY24

19.3 Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the “PAC Write-Protection” property in each individual register description.

Some registers are synchronized when read and/or written. Synchronization is denoted by the “Write-Synchronized” or the “Read-Synchronized” property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable protection is denoted by the “Enable-Protected” property in each individual register description.

Note: All registers in this table have corresponding CLR, SET and INV registers at its virtual address, plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET, and INV Registers* from Related Links.

Related Links

6.1.9. CLR, SET and INV Registers

19.3.1 System Key Register

Name: SYSKEY
Offset: 0xB0
Reset: 0x0

Bit	31	30	29	28	27	26	25	24
	SYSKEY31	SYSKEY30	SYSKEY29	SYSKEY28	SYSKEY27	SYSKEY26	SYSKEY25	SYSKEY24
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SYSKEY23	SYSKEY22	SYSKEY21	SYSKEY20	SYSKEY19	SYSKEY18	SYSKEY17	SYSKEY16
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SYSKEY15	SYSKEY14	SYSKEY13	SYSKEY12	SYSKEY11	SYSKEY10	SYSKEY9	SYSKEY8
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SYSKEY7	SYSKEY6	SYSKEY5	SYSKEY4	SYSKEY3	SYSKEY2	SYSKEY1	SYSKEY0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 – SYSKEY
System Key

Keys are written to this register as part of a sequence to unlock system critical registers. A successful key write to this register will set the system signal.

20. Peripheral Module Disable Register (PMD)

20.1 Overview

This section describes the following peripheral module disable functions:

- Device peripheral configuration
- Configuration defined by product variants
- Peripheral disable for power conservation

20.2 Enabling Peripherals

Peripheral Module Disable (PMD) register bits control the operation of individual peripherals on the device. When a peripheral's associated PMD bit is zero (0), the peripheral is enabled and operates as programmed. However, when the associated PMD bit is one (1), the peripheral logic, memory map and SFR bits are completely removed from visibility and the peripheral is held in Reset. This disabled state provides for the lowest power state of the peripheral.

Before a peripheral may be configured or used, it must be enabled by clearing the corresponding PMD register bit.

There are some caveats to using PMD bits. The following must be observed:

1. Disabling a peripheral while its ON bit is zero (0) results in undefined behavior of the external interface.
2. For bus initiators, software must verify the module is not busy after setting the ON bit to zero (0) before disabling it.
3. Setting the PMD bit when there is a pending interrupt results in undefined behavior. Therefore, all Interrupt Flags must be cleared prior to setting the associated PMD.

20.3 Registers and Bits

Note: PMD registers can be write-locked using the CFGCON0.PMDLOCK register bit. If this bit is set, then writing of PMD registers has no effect.

See *PMD1*, *PMD2* and *PMD3* in the *PMD Register Summary* from Related Links for a description of the PMD registers, and to identify the location of the register, see *Device Configuration Map* in the *Product Memory Mapping Overview* from Related Links.

Related Links

8. Product Memory Mapping Overview
20.4.1. PMD Register Summary

20.4 PMD Register

20.4.1 PMD Register Summary

Note: All registers in this table have corresponding CLR, SET and INV registers at its virtual address, plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET and INV Registers* from Related Links.

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00 ... 0xBF	Reserved									
0xC0	PMD1	7:0	ADCMD	ACMD		PLVDMD	LPAMD	MPAMD	BTMD	ZBMD
		15:8								ADCSARMD
		23:16								RTCCMD
		31:24			SQIMD					

Related Links

6.1.9. CLR, SET and INV Registers

20.4.2 Register Description

Some peripherals include module enable bits internally. The PMD bit is used for clock gating of the PBx_CLK and GCLK for all peripherals. If the peripheral also includes the internal enable bit, the PMD bit and internal enable configuration bit must be configured by software for that peripheral.

The following table summarizes each peripheral's enable and disable controls. For more details on the internal enable/disable control, see *Peripheral Access Controller (PAC)* from Related Links.

Table 20-1. Module Enable/Disable Controls

Module	PMD control	Module control	Enable/Disable Strategy
AC	Present	Present	Disable at PMD or Module
AES	Present	Present	Disable at PMD or Module
CCL	NA	Present	Disable at Module
CMCC	NA	Present	Disable at Module
DMAC	NA	Present	Disable at Module
DSU	NA	NA	Always Enabled (Dynamic On/Off)
EIC	NA	Present	Disable at Module
EVSYS	NA	NA	Always Enabled (Dynamic On/Off)
FREQM	NA	Present	Disable at Module
ICM	Present	Present	Enable both/Disable at PMD or Module
PAC	NA	NA	Always Enabled (Dynamic On/Off)
PUKCC	Present	NA	Disable at PMD
QSPI	Present	Present	Enable both/Disable at PMD or Module
RAMECC	NA	NA	Disable using fuse bit
RTCC	Present	Present	Enable both/Disable at PMD or Module
SERCOM	Present	Present	Enable both/Disable at PMD or Module
TC	Present	Present	Enable both/Disable at PMD or Module
TCC	Present	Present	Enable both/Disable at PMD or Module
TRNG	Present	Present	Enable both/Disable at PMD or Module

Note: For Modules with both PMD control and Module control, Enable = PMDx=0 AND Module Enable=1, Disable =PMDx=1 OR Module Enable=0.

Related Links

26. Peripheral Access Controller (PAC)

20.4.3 PMD1 – Peripheral Module Disable 1 Register

Name: PMD1
Offset: 0x00C0
Reset: 0x00000000
Property: -

Notes:

- This bit is only writable when CFGCON0.PMDLOCK = 0.
- All registers in this table have corresponding CLR, SET and INV registers at its virtual address, plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET and INV Registers* from Related Links.

Bit	31	30	29	28	27	26	25	24
			SQIMD					
Access			R/W/L					
Reset			0					

Bit	23	22	21	20	19	18	17	16
								RTCCMD
Access								R/W/L
Reset								0

Bit	15	14	13	12	11	10	9	8
								ADCSARMD
Access								R/W/L
Reset								0

Bit	7	6	5	4	3	2	1	0
	ADCMD	ACMD		PLVDMD	LPAMD	MPAMD	BTMD	ZBMD
Access	R/W/L	R/W/L		R/W/L	R/W/L	R/W/L	R/W/L	R/W/L
Reset	0	0		0	0	0	0	0

Bit 29 – SQIMD SQI Module Disable

Value	Description
1	SQI module is disabled
0	SQI module is enabled

Bit 16 – RTCCMD RTCC Module Disable (Unused at top level, part of XDS controller SFR)

Value	Description
1	RTCC module is disabled
0	RTCC module is enabled

Bit 8 – ADCSARMD Shared ADC SAR Core Module Disable bit

Value	Description
1	ADC SAR Core module is disabled. When disabled, the corresponding ADC SAR SHARED will be disabled.
0	ADC SAR Core module is enabled

Bit 7 – ADCMD ADC Controller Module Disable

Value	Description
1	ADC Controller module is disabled
0	ADC Controller module is enabled

Bit 6 – ACMD AC Module Disable

Value	Description
1	AC module is disabled
0	AC module is enabled

Bit 4 – PLVDMD PLVD Module Disable bit

Value	Description
1	PLVD module is disabled. When disabled, the corresponding PLVD will be disabled.

Bit 3 – LPAMD RF LPA Module Disable bit

Value	Description
1	RF LPA module is disabled
0	RF LPA module is enabled

Bit 2 – MPAMD RF MPA Module Disable bit

Value	Description
1	RF MPA module is disabled
0	RF MPA module is enabled

Bit 1 – BTMD Bluetooth Module Disable bit

Value	Description
1	Bluetooth module is disabled
0	Bluetooth module is enabled

Bit 0 – ZBMD Zigbee Module Disable bit

Value	Description
1	Zigbee module is disabled
0	Zigbee module is enabled

Related Links

6.1.9. CLR, SET and INV Registers

20.5 PMDx Initialization Values by Variant Name

Table 20-2. PMDx Initialization Values

PartName	31							24		23							16		15							8	7							0	Hex
	PMD1																																			
			SQIMD														RTCCMD								ADCSARMD	ADCMD	ADCMD			PLVDMD	LPAMD	MPAMD	BTMD	ZBMD	
PIC32CX1012BZ25048	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0[1]	0[1]	01	01	0000_0001

PartName	31							24		23							16		15							8	7							0	Hex
	PMD2																																			
	REFO4MD	REFO3MD	REFO2MD	REFO1MD			REFO6MD	REFO5MD																												
PIC32CX1012BZ25048	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000_0001

PartName	31							24		23							16	15								8	7									0	Hex
	PMD3																																				
																			TCC2MD	TCC1MD	TCC0MD	TC3MD	TC2MD	TC1MD	TC0MD	AESMD	RNGMD		PUKCCMD	ICMMD	SER3MD	SER2MD	SER1MD	SER0MD	
PIC32CX1012BZ25048	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0[1]	0[1]		0[1]	0[1]	0[1]	0[1]	0 1	0 1	0000_0001

Note:

1. The value depends on the specific Device Variant, as specified in Configuration Summary, see *Configuration Summary* from Related Links.

21. Real-Time Counter and Calendar (RTCC)

21.1 Overview

The Real-Time Counter and Calendar (RTCC) is a 32-bit counter with a 10-bit programmable prescaler that typically runs continuously to keep track of time. The RTCC can wake up the device from sleep modes using the alarm/compare wake up, periodic wake up, or overflow wake up mechanisms.

The RTCC can generate periodic peripheral events from outputs of the prescaler, as well as alarm/compare interrupts and peripheral events, which can trigger at any counter value. Additionally, the timer can trigger an overflow interrupt and peripheral event, and can be reset on the occurrence of an alarm/compare match. This allows periodic interrupts and peripheral events at very long and accurate intervals.

The 10-bit programmable prescaler can scale down the clock source. By this, a wide range of resolutions and time-out periods can be configured. With a 32.768 kHz clock source, the minimum counter tick interval is 30.5µs, and time-out periods can range up to 36 hours. For a counter tick interval of 1s, the maximum time-out period is more than 136 years.

21.2 Features

- 32-bit counter with 10-bit prescaler
- Multiple clock sources
- 32-bit or 16-bit counter mode
- Clock/Calendar mode
  - Time in seconds, minutes, and hours (12/24)
  - Date in day of month, month, and year
  - Leap year correction
- Digital prescaler correction/tuning for increased accuracy
- Overflow, alarm/compare match and prescaler interrupts and events
  - Optional clear on alarm/compare match
- 4 general purpose registers
- 1 backup register with retention capability
- Tamper Detection
  - Timestamp on event or up to 4 inputs with debouncing
  - Active layer protection

21.3 Block Diagram

Figure 21-1. RTCC Block Diagram (Mode 0 — 32-Bit Counter)

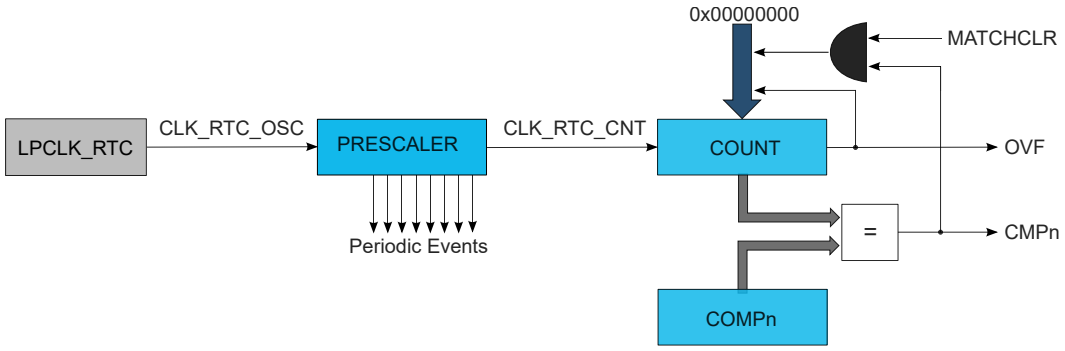


Figure 21-2. RTCC Block Diagram (Mode 1 — 16-Bit Counter)

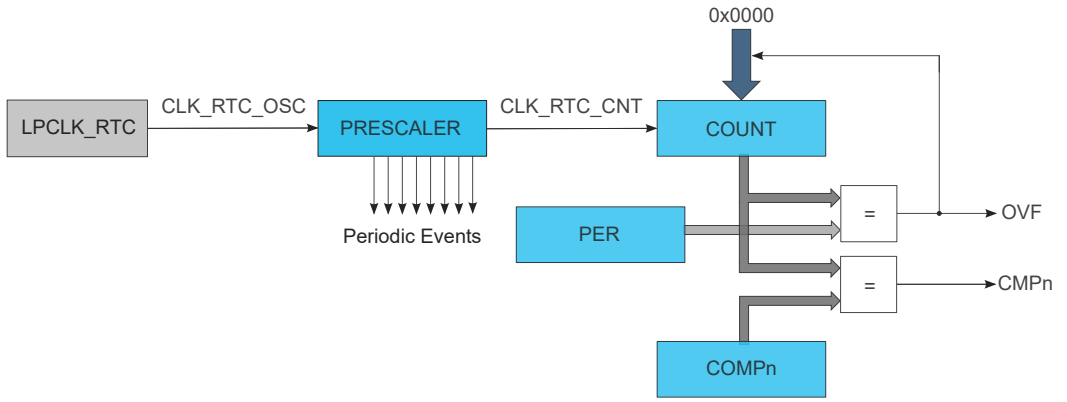


Figure 21-3. RTCC Block Diagram (Mode 2 — Clock/Calendar)

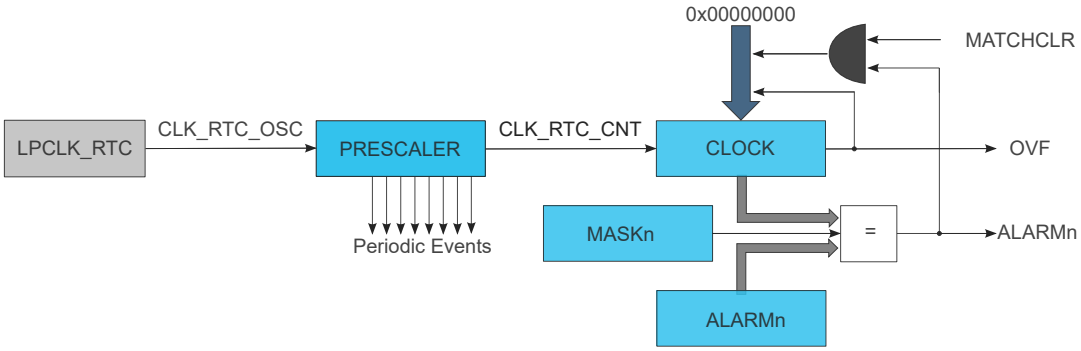
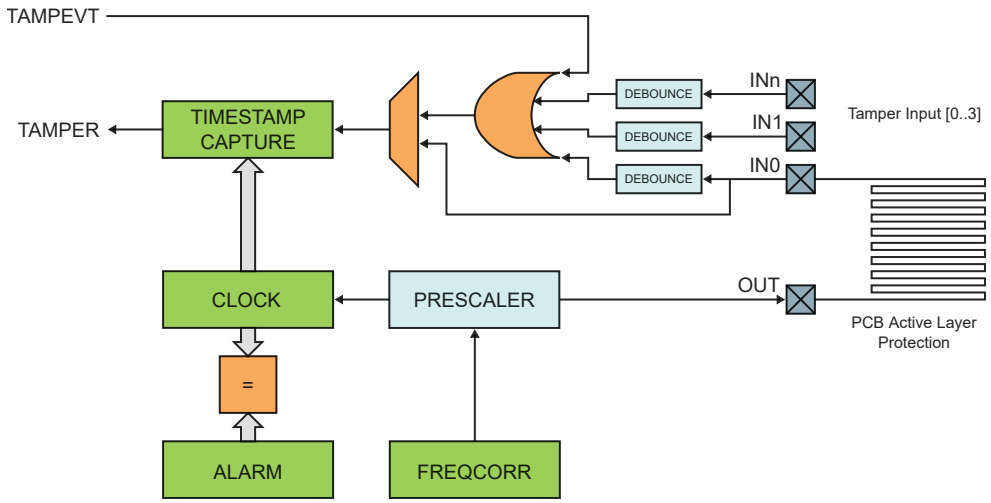


Figure 21-4. RTCC Block Diagram (Tamper Detection)



21.4 Signal Description

Table 21-1. Signal Description

Signal	Description	Type
INn [n=0..3]	Tamper Detection Input	Digital input
OUT	Tamper Detection Output	Digital output

.....continued		
Signal	**Description**	**Type**
RTC_EVENT	RTC Event Output	Digital output

21.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

21.5.1 I/O Lines

In order to use the I/O lines of this peripheral, the RTC must be enabled and no higher priority peripherals for the RTC pins can be enabled. See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links.

Related Links

6. I/O Ports and Peripheral Pin Select (PPS)

21.5.2 Power Management

The RTC will continue to operate in any sleep modes where the selected source clock is running. The RTC interrupts can be used to wake-up the device from sleep modes. Events connected to the event system can trigger other operations in the system without exiting sleep modes. See *Power Management Unit (PMU)* from Related Links for details on the different sleep modes.

The RTCC can only be reset by a power on reset (POR) or by setting the Software Reset bit in the Control A register (CTRLA.SWRST = 1).

Related Links

15. Power Management Unit (PMU)

21.5.3 Clocks

A 32 KHz or 1 KHz oscillator clock (CLK_RTC_OSC) is required to clock the RTC. The 32 KHz clock source can be FRC, POSC, SOSC or LPRC based on the mux selection controlled by the CFG.CFGCON4.VBKP_32KCSEL bit. The 1 KHz clock source is based on the mux selection controlled by the CFG.CFGCON4.VBKP_1KCSEL bit.

This oscillator clock is asynchronous to the bus clock (PB3_CLK). Due to this asynchronicity, writing to certain registers will require synchronization between the clock domains.

21.5.4 DMA

21.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the RTC interrupt requires the Interrupt Controller to be configured first.

21.5.6 Events

The events are connected to the *Event System*. See *Event System (EVSYS)* from Related Links.

Related Links

28. Event System (EVSYS)

21.5.7 Debug Operation

When the CPU is halted in debug mode the RTC will halt normal operation. The RTC can be forced to continue operation during debugging. See *DBGCTRL* from Related Links.

Related Links

21.8.7. DBGCTRL

21.6 Functional Description

21.6.1 Principle of Operation

The RTC keeps track of time in the system and enables periodic events, as well as interrupts and events at a specified time. The RTC consists of a 10-bit prescaler that feeds a 32-bit counter. The actual format of the 32-bit counter depends on the RTC operating mode.

The RTC can function in one of these modes:

- Mode 0 - COUNT32: RTC serves as 32-bit counter
- Mode 1 - COUNT16: RTC serves as 16-bit counter
- Mode 2 - CLOCK: RTC serves as clock/calendar with alarm functionality

21.6.2 Basic Operation

21.6.2.1 Initialization

The following bits are enable-protected, meaning that they can only be written when the RTC is disabled (CTRLA.ENABLE=0):

- Operating Mode bits in the Control A register (CTRLA.MODE)
- Prescaler bits in the Control A register (CTRLA.PRESCALER)
- Clear on Match bit in the Control A register (CTRLA.MATCHCLR)
- Clock Representation bit in the Control A register (CTRLA.CLKREP)
- BKUP Registers Reset On Tamper bit in Control A register (CTRLA.BKTRST)
- GP Registers Reset On Tamper Enable in Control A register (CTRLA.GPTRST)

The following registers are enable-protected:

- Event Control register (EVCTRL)
- Control B register (CTRLB)

Enable-protected bits and registers can be changed only when the RTC is disabled (CTRLA.ENABLE=0). If the RTC is enabled (CTRLA.ENABLE=1), these operations are necessary: first write CTRLA.ENABLE=0 and check whether the write synchronization has finished, then change the desired bit field value. Enable-protected bits in CTRLA register can be written at the same time as CTRLA.ENABLE is written to '1', but not at the same time as CTRLA.ENABLE is written to '0'.

Enable-protection is denoted by the "Enable-Protected" property in the register description.

The RTC prescaler divides the source clock for the RTC counter.

Note: In Clock/Calendar mode, the prescaler must be configured to provide a 1.024 kHz clock to the counter for correct operation.

The frequency of the RTC clock (CLK_RTC_CNT) is given by the following formula:

$$f_{\mathrm{CLK\_RTC\_CNT}} = \frac{f_{\mathrm{CLK\_RTC\_OSC}}}{2^{\mathrm{PRESCALER}}}$$

The frequency of the oscillator clock, CLK_RTC_OSC, is given by $f_{CLK\_RTC\_OSC}$, and $f_{CLK\_RTC\_CNT}$ is the frequency of the internal prescaled RTC clock, CLK_RTC_CNT.

21.6.2.2 Enabling, Disabling, and Resetting

The RTC is enabled by setting the Enable bit in the Control A register (CTRLA.ENABLE=1). The RTC is disabled by writing CTRLA.ENABLE=0.

The RTC is reset by setting the Software Reset bit in the Control A register (CTRLA.SWRST=1). All registers in the RTC, except DEBUG, will be reset to their initial state, and the RTC will be disabled. The RTC must be disabled before resetting it.

21.6.2.3 32-Bit Counter (Mode 0)

When the RTC Operating Mode bits in the Control A register (CTRLA.MODE) are written to 0x0, the counter operates in 32-bit Counter mode. See *RTC Block Diagram (Mode 0 — 32-Bit Counter)* figure in the *Block Diagram* from

Related Links. When the RTC is enabled, the counter will increment on every 0-to-1 transition of CLK_RTC_CNT. The counter will increment until it reaches the top value of 0xFFFFFFFF, and then wrap to 0x00000000. This sets the Overflow Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF).

The RTC counter value can be read from or written to the Counter Value register (COUNT) in 32-bit format.

The counter value is continuously compared with the 32-bit Compare register (COMP). When a compare match occurs, the Compare Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMP) is set on the next 0-to-1 transition of CLK_RTC_CNT.

If the Clear on Match bit in the Control A register (CTRLA.MATCHCLR) is '1', the counter is cleared on the next counter cycle when a compare match with COMP occurs. This allows the RTC to generate periodic interrupts or events with longer periods than the prescaler events. Note that when CTRLA.MATCHCLR is '1', INTFLAG.CMP and INTFLAG.OVF will both be set simultaneously on a compare match with COMP.

Related Links

21.3. Block Diagram

21.6.2.4 16-Bit Counter (Mode 1)

When the RTC Operating Mode bits in the Control A register (CTRLA.MODE) are written to 0x1, the counter operates in 16-bit Counter mode. See *RTC Block Diagram (Mode 1 — 16-Bit Counter)* figure in the *Block Diagram* from Related Links. When the RTC is enabled, the counter will increment on every 0-to-1 transition of CLK_RTC_CNT. In 16-bit Counter mode, the 16-bit Period register (PER) holds the maximum value of the counter. The counter will increment until it reaches the PER value, and then wrap to 0x0000. This sets the Overflow Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF).

The RTC counter value can be read from or written to the Counter Value register (COUNT) in 16-bit format.

The counter value is continuously compared with the 16-bit Compare registers (COMPn, n=0..). When a compare match occurs, the Compare n Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMPn, n=0..) is set on the next 0-to-1 transition of CLK_RTC_CNT.

Related Links

21.3. Block Diagram

21.6.2.5 Clock/Calendar (Mode 2)

When the RTC Operating Mode bits in the Control A register (CTRLA.MODE) are written to 0x2, the counter operates in Clock/Calendar mode. See *RTC Block Diagram (Mode 2 — Clock/Calendar)* figure in the *Block Diagram* from Related Links. When the RTC is enabled, the counter will increment on every 0-to-1 transition of CLK_RTC_CNT. The selected clock source and RTC prescaler must be configured to provide a 1Hz clock to the counter for correct operation in this mode.

The time and date can be read from or written to the Clock Value register (CLOCK) in a 32-bit time/date format. Time is represented as:

- Seconds
- Minutes
- Hours

Hours can be represented in either 12- or 24-hour format, selected by the Clock Representation bit in the Control A register (CTRLA.CLKREP). This bit can be changed only while the RTC is disabled.

The date is represented in this form:

- Day as the numeric day of the month (starting at 1)
- Month as the numeric month of the year (1 = January, 2 = February, etc.)
- Year as a value from 0x00 to 0x3F. This value must be added to a user-defined reference year. The reference year must be a leap year (2016, 2020 etc). Example: the year value 0x2D, added to a reference year 2016, represents the year 2061.

The RTC will increment until it reaches the top value of 23:59:59 December 31 of year value 0x3F, and then wrap to 00:00:00 January 1 of year value 0x00. This will set the Overflow Interrupt flag in the Interrupt Flag Status and Clear registers (INTFLAG.OVF).

The clock value is continuously compared with the 32-bit Alarm register (ALARM). When an alarm match occurs, the Alarm Interrupt flag in the Interrupt Flag Status and Clear registers (INTFLAG.ALARM) is set on the next 0-to-1 transition of CLK_RTC_CNT. E.g. For a 1Hz clock counter, it means the Alarm 0 Interrupt flag is set with a delay of 1s after the occurrence of alarm match.

A valid alarm match depends on the setting of the Alarm Mask Selection bits in the Alarm Mask register (MASK.SEL). These bits determine which time/date fields of the clock and alarm values are valid for comparison and which are ignored.

If the Clear on Match bit in the Control A register (CTRLA.MATCHCLR) is set, the counter is cleared on the next counter cycle when an alarm match with ALARM occurs. This allows the RTC to generate periodic interrupts or events with longer periods than it would be possible with the prescaler events only (see *Periodic Intervals* from Related Links).
Note: When CTRLA.MATCHCLR is 1, INTFLAG.ALARMn and INTFLAG.OVF will both be set simultaneously on an alarm match with ALARM.

Related Links
21.3. Block Diagram
21.6.8.1. Periodic Intervals

21.6.3 DMA Operation

21.6.4 Interrupts

The RTC has the following interrupt sources:

- Overflow (OVF): Indicates that the counter has reached its top value and wrapped to zero.
- Compare (CMPn): Indicates a match between the counter value and the compare register.
- Alarm (ALARM): Indicates a match between the clock value and the alarm register.
- Period n (PERn): The corresponding bit in the prescaler has toggled, see *Periodic Intervals* from Related Links.

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear (INTFLAG) register is set when the interrupt condition occurs. Each interrupt can be individually enabled by setting the corresponding bit in the Interrupt Enable Set register (INTENSET=1), and disabled by setting the corresponding bit in the Interrupt Enable Clear register (INTENCLR=1). The status of enabled interrupts can be read from either INTENSET or INTENCLR.

An interrupt request is generated when the interrupt flag is raised and the corresponding interrupt is enabled. The interrupt request remains active until either the interrupt flag is cleared, the interrupt is disabled or the RTC is reset. See the description of the INTFLAG registers for details on how to clear interrupt flags.

All interrupt requests from the peripheral are ORed together on system level to generate one combined interrupt request to the NVIC, see *Nested Vector Interrupt Controller (NVIC)* from Related Links. The user must read the INTFLAG register to determine which interrupt condition is present.

Note: Interrupts must be globally enabled for interrupt requests to be generated, see *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links
10.2. Nested Vector Interrupt Controller (NVIC)
21.6.8.1. Periodic Intervals

21.6.5 Events

The RTC can generate the following output events and these events can be used by EVSYS module:

- Overflow (OVF): Generated when the counter has reached its top value and wrapped to zero.
- Compare (CMPn): Indicates a match between the counter value and the compare register.
- Alarm (ALARMn): Indicates a match between the clock value and the alarm register.
- Period n (PERn): The corresponding bit in the prescaler has toggled, see *Periodic Intervals* from Related Links.
- Periodic Daily (PERD): Generated when the COUNT/CLOCK has incremented at a fixed period of time.
- RTC Event (RTC_EVENT): Generates specific external signal on the RTC EVENT I/O pin.

Setting the Event Output bit in the Event Control Register (EVCTRL.xxxEO=1) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event. See *Event System (EVSYS)* from Related Links for more details on configuring the event system.

The RTC can take the following actions on an input event:

- Tamper (TAMPEVT): Capture the RTC counter to the timestamp register. See *Tamper Detection* from Related Links.

Writing a one to an Event Input bit into the Event Control register (EVCTRL.xxxEI) enables the corresponding action on input event. Writing a zero to this bit disables the corresponding action on input event.

RTC Event (RTC_EVENT): Other than the above events, which are mapped to the EVSYS module, the following events can generate specific external signal on the RTC EVENT I/O pin.

- 32 KHz clock
- Alarm pulse
- 1-second clock

These event signals are configured using CFGCON4.RTCEVENTSEL[1:0] bits.

Note: The RTC_OUT and RTC_EVENT signals are multiplexed and any one of the signal can be out at a time in pin limited variants. See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links. The selection between RTC_OUT and RTC_EVENT is configurable through CFGCON4.RTCEVTYPE bit.

Related Links

6. I/O Ports and Peripheral Pin Select (PPS)

21.6.8.1. Periodic Intervals

21.6.8.5. Tamper Detection

28. Event System (EVSYS)

21.6.6 Sleep Mode Operation

The RTC will continue to operate in any sleep mode where the source clock is active. The RTC *interrupts* can be used to wake up the device from a sleep mode. RTC *events* can trigger other operations in the system without exiting the sleep mode.

An interrupt request will be generated after the wake-up if the Interrupt Controller is configured accordingly. Otherwise the CPU will wake up directly, without triggering any interrupt. In this case, the CPU continues executing right from the first instruction that followed the entry into sleep.

The periodic events can also wake up the CPU through the interrupt function of the Event System. In this case, the event must be enabled and connected to an event channel with its interrupt enabled. See *Event System (EVSYS)* from Related Links.

Related Links

28. Event System (EVSYS)

21.6.7 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset bit in Control A register, CTRLA.SWRST
- Enable bit in Control A register, CTRLA.ENABLE

The following registers are synchronized when written:

- Counter Value register, COUNT
- Clock Value register, CLOCK
- Counter Period register, PER
- Compare n Value registers, COMPn
- Alarm n Value registers, ALARMn

- Frequency Correction register, FREQCORR
- Alarm n Mask register, MASKn

The following registers are synchronized when read:

- The Counter Value register, COUNT, if the Counter Read Sync Enable bit in CTRLA (CTRLA.COUNTSYNC) is '1'
- The Clock Value register, CLOCK, if the Clock Read Sync Enable bit in CTRLA (CTRLA.CLOCKSYNC) is '1'

Required write synchronization is denoted by the "Write-Synchronized" property in the register description.

Required read synchronization is denoted by the "Read-Synchronized" property in the register description.

21.6.8 Additional Features

21.6.8.1 Periodic Intervals

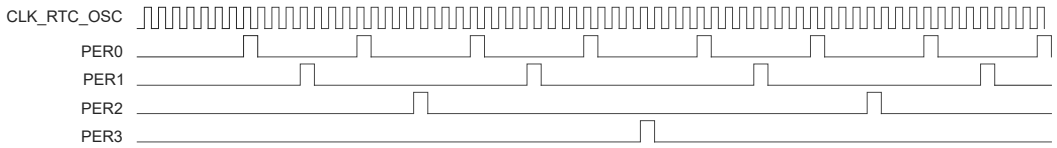
The RTC prescaler can generate interrupts and events at periodic intervals, allowing flexible system tick creation. Any of the upper eight bits of the prescaler (bits 2 to 9) can be the source of an interrupt/event. When one of the eight Periodic Event Output bits in the Event Control register (EVCTRL.PEREO[n=0..7]) is '1', an event is generated on the 0-to-1 transition of the related bit in the prescaler, resulting in a periodic event frequency of:

$$f_{\mathrm{PERIODIC(n)}} = \frac{f_{\mathrm{CLK\_RTC\_OSC}}}{2^{n+3}}$$

$f_{CLK\_RTC\_OSC}$ is the frequency of the internal prescaler clock CLK_RTC_OSC, and n is the position of the EVCTRL.PEREOn bit. For example, PER0 will generate an event every eight CLK_RTC_OSC cycles, PER1 every 16 cycles, etc. This is shown in the figure below.

Periodic events are independent of the prescaler setting used by the RTC counter, except if CTRLA.PRESCALER is zero. Then, no periodic events will be generated.

Figure 21-5. Example Periodic Events



Note: This example also applies to interrupts. Just replace EVCTRL.PEREOn with the PERn fields of INTENCLR, INTENSET, and INTFLAG. For Modes 0 and 2, n = 0,..7. For Mode 1 n = 2...7.

21.6.8.2 Frequency Correction

The RTC Frequency Correction module employs periodic counter corrections to compensate for a too-slow or too-fast oscillator. Frequency correction requires that CTRLA.PRESCALER is greater than 1.

The digital correction circuit adds or subtracts cycles from the RTC prescaler to adjust the frequency in approximately 1ppm steps. Digital correction is achieved by adding or skipping a single count in the prescaler once every CLK_RTC_OSC cycles. The Value bit group in the Frequency Correction register (FREQCORR.VALUE) determines the number of times the adjustment is applied over of these periods. The resulting correction is as follows:

This results in a resolution of ppm.

The Sign bit in the Frequency Correction register (FREQCORR.SIGN) determines the direction of the correction. A positive value will add counts and increase the period (reducing the frequency), and a negative value will reduce counts per period (speeding up the frequency).

Digital correction also affects the generation of the periodic events from the prescaler. When the correction is applied at the end of the correction cycle period, the interval between the previous periodic event and the next occurrence may also be shortened or lengthened depending on the correction value.

21.6.8.3 Backup Registers

The RTC includes one Backup register (BKUP0). This register maintain its content in Backup/Deep Sleep mode. It can be used to store user-defined values.

If more user-defined data must be stored than the Backup register can hold, the General Purpose registers (GPn) can be used.

21.6.8.4 General Purpose Registers

The RTC includes four General Purpose registers (GPn). These registers are reset only when the RTC is reset, and remain powered while the RTC is powered. They can be used to store user-defined values while other parts of the system are powered off.

The general purpose registers 2*n and 2*n+1 are enabled by writing a '1' to the General Purpose Enable bit n in the Control B register (CTRLB.GPnEN).

The GP registers share internal resources with the COMPARE/ALARM features. Each COMPARE/ALARM register have a separate read buffer and write buffer. When the general purpose feature is enabled the even GP uses the read buffer while the odd GP uses the write buffer.

When the COMPARE/ALARM register is written, the write buffer hold temporarily the COMPARE/ALARM value until the synchronisation is complete (bit SYNCBUSY.COMPn going to 0). After the write is completed the write buffer can be used as a odd general purpose register without affecting the COMPARE/ALARM function.

If the COMPARE/ALARM function is not used, the read buffer can be used as an even general purpose register. In this case, writing the even GP will temporarily use the write buffer until the synchronisation is complete (bit SYNCBUSY.GPn going to 0). Thus an even GP must be written before writing the odd GP. Changing or writing an even GP needs to temporarily save the value of the odd GP.

Before using an even GP, the associated COMPARE/ALARM feature must be disabled by writing a '1' to the General Purpose Enable bit in the Control B register (CTRLB.GPnEN). To re-enable the compare/alarm, CTRLB.GPnEN must be written to zero and the associated COMPn/ALARMn must be written with the correct value.

It is recommended to use the Backup register (BKUPn) first to store user-defined values, and use the GPn only when the user-defined values exceed the capacity of the provided BKUPn.

An example procedure to write the general purpose registers GP0 and GP1 is:

1. Wait for any ongoing write to COMP0 to complete (SYNCBUSY.COMP0 = 0). If the RTC is operating in Mode 1, wait for any ongoing write to COMP1 to complete as well (SYNCBUSY.COMP1 = 0).
2. Write CTRLB.GP0EN = 1 if GP0 is needed.
3. Write GP0 if needed.
4. Wait for any ongoing write to GP0 to complete (SYNCBUSY.GP0 = 0). Note that GP1 will also show as busy when GP0 is busy.
5. Write GP1 if needed.

The following table provides the correspondence of General Purpose Registers and the COMPARE/ALARM read or write buffer in all RTC modes.

Table 21-2. General Purpose Registers Versus Compare/Alarm Registers: n in 0, 2, 4, 6...

Register	Mode 0	Mode 1	Mode 2	Write Before
GPn	COMPn/2 write buffer	(COMPn , COMPn+1) write buffer	ALARMn/2 write buffer	GPn+1
GPn+1	COMPn/2 read buffer	(COMPn , COMPn+1) read buffer	ALARMn/2 read buffer	—

21.6.8.5 Tamper Detection

The RTC provides four tamper channels that can be used for tamper detection.

The action of each tamper channel is configured using the Input n Action bits in the Tamper Control register (TAMPCTRL.INnACT):

- Off: Detection for tamper channel n is disabled.

- Wake: A transition on INn input (tamper channel n) matching TAMPCTRL.TAMPLVLn will be detected and the tamper interrupt flag (INTFLAG.TAMPER) will be set. The RTC value will not be captured in the TIMESTAMP register.
- Capture: A transition on INn input (tamper channel n) matching TAMPCTRL.TAMPLVLn will be detected and the tamper interrupt flag (INTFLAG.TAMPER) will be set. The RTC value will be captured in the TIMESTAMP register.
- Active Layer Protection: A mismatch of an internal RTC signal routed between INn and OUTn pins will be detected and the tamper interrupt flag (INTFLAG.TAMPER) will be set. The RTC value will be captured in the TIMESTAMP register.

In order to determine which tamper source caused a tamper event, the Tamper ID register (TAMPID) provides the detection status of each tamper channel. These bits remain active until cleared by software.

A single interrupt request (TAMPER) is available for all tamper channels.

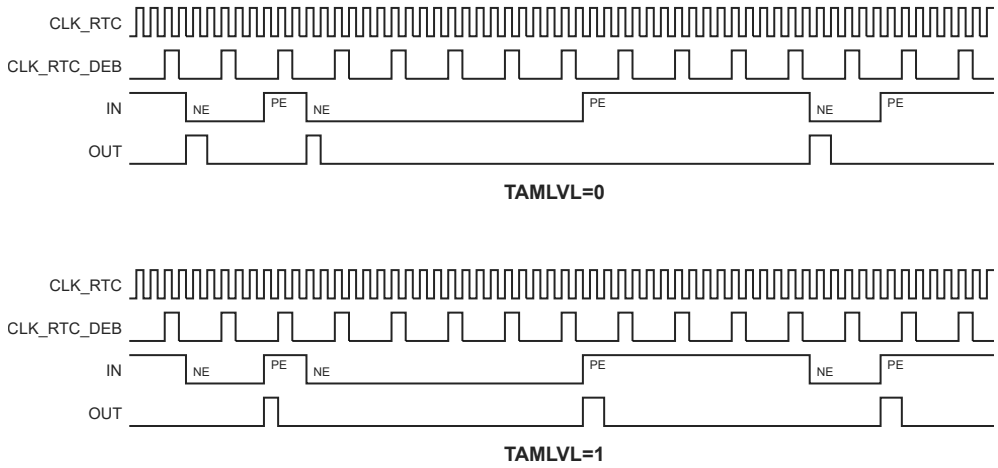
The RTC also supports an input event (TAMPEVT) for generating a tamper condition within the Event System. The tamper input event is enabled by the Tamper Input Event Enable bit in the Event Control register (EVCTRL.TAMPEVTEI).

Up to four polarity external inputs (INn) can be used for tamper detection. The polarity for each input is selected with the Tamper Level bits in the Tamper Control register (TAMPCTRL.TAMPLVLn).

Separate debouncers are embedded for each external input. The debouncer for each input is enabled/disabled with the Debounce Enable bits in the Tamper Control register (TAMPCTRL.DEBNCn). The debouncer configuration is fixed for all inputs as set by the Control B register (CTRLB). The debouncing period duration is configurable using the Debounce Frequency field in the Control B register (CTRLB.DEBF). The period is set for all debouncers (i.e., the duration cannot be adjusted separately for each debouncer).

When TAMPCTRL.DEBNCn = 0, INn is detected asynchronously. See the following figure for an example.

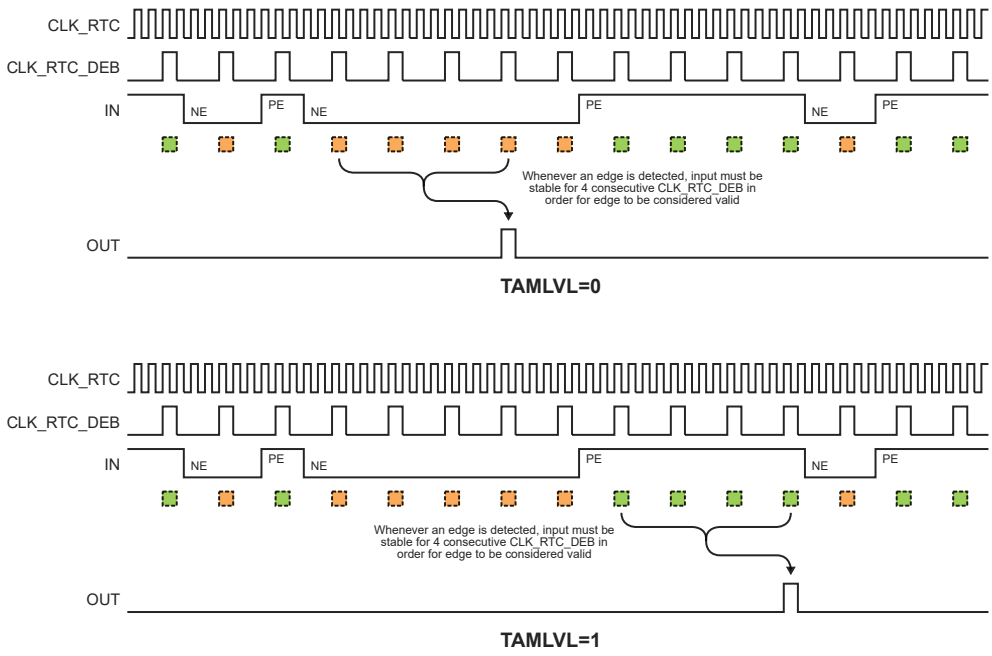
Figure 21-6. Edge Detection with Debouncer Disabled



When TAMPCTRL.DEBNCn = 1, the detection time depends on whether the debouncer operates synchronously or asynchronously, and whether majority detection is enabled or not. For more details, refer to the following table. Synchronous versus asynchronous stability debouncing is configured by the Debounce Asynchronous Enable bit in the Control B register (CTRLB.DEBASYNC):

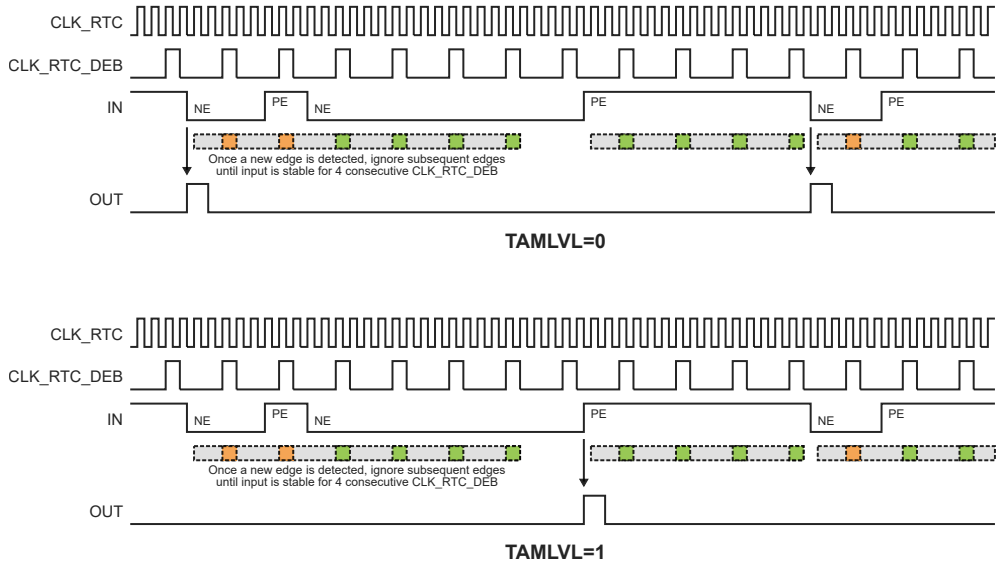
- Synchronous (CTRLB.DEBASYNC = 0): INn is synchronized in two CLK_RTC periods and then must remain stable for four CLK_RTC_DEB periods before a valid detection occurs. See the following figure for an example.

Figure 21-7. Edge Detection with Synchronous Stability Debouncing



- Asynchronous (CTRLB.DEBASYNC = 1): The first edge on INn is detected. Further detection is blanked until INn remains stable for four CLK_RTC_DEB periods. See the following figure for an example.

Figure 21-8. Edge Detection with Asynchronous Stability Debouncing



Majority debouncing is configured by the Debounce Majority Enable bit in the Control B register (CTRLB.DEBMAJ). INn must be valid for two out of three CLK_RTC_DEB periods. See the following figure for an example.

Figure 21-9. Edge Detection with Majority Debouncing

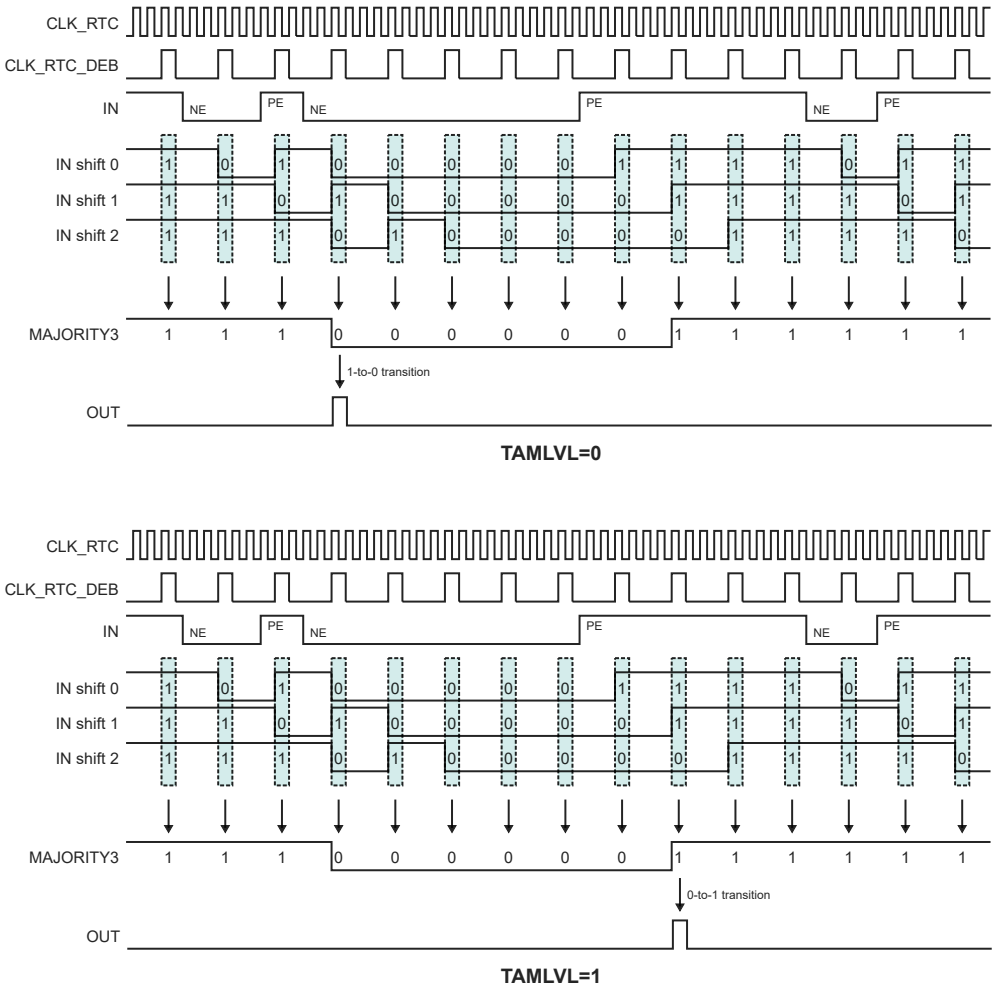


Table 21-3. Debouncer Configuration

TAMPCTRL. DEBNCn	CTRLB. DEBMAJ	CTRLB. DEBASYNC	Description
0	X	X	Detect edge on INn with no debouncing. Every edge detected is immediately triggered.
1	0	0	Detect edge on INn with synchronous stability debouncing. Edge detected is only triggered when INn is stable for 4 consecutive CLK_RTC_DEB periods.
1	0	1	Detect edge on INn with asynchronous stability debouncing. First detected edge is triggered immediately. All subsequent detected edges are ignored until INn is stable for 4 consecutive CLK_RTC_DEB periods.
1	1	X	Detect edge on INn with majority debouncing. Pin INn is sampled for 3 consecutive CLK_RTC_DEB periods. Signal level is determined by majority-rule (LLL, LLH, LHL, HLL = '0' and LHH, HLH, HHL, HHH = '1').

21.6.8.5.1 Timestamp

As part of tamper detection the RTC can capture the counter value (COUNT/CLOCK) into the TIMESTAMP register. Three CLK_RTC periods are required to detect the tampering condition and capture the value. The TIMESTAMP value can be read once the Tamper flag in the Interrupt Flag register (INTFLAG.TAMPER) is set. If the DMA Enable bit in the Control B register (CTRLB.DMAEN) is '1', a DMA request will be triggered by the timestamp. In order to determine which tamper source caused a capture, the Tamper ID register (TAMPID) provides the detection status of each tamper channel and the tamper input event. A DMA transfer can then read both TIMESTAMP and TAMPID in succession.

A new timestamp value cannot be captured until the Tamper flag is cleared, either by reading the timestamp or by writing a '1' to INTFLAG.TAMPER. If several tamper conditions occur in a short window before the flag is cleared, only the first timestamp may be logged. However, the detection of each tamper will still be recorded in TAMPID.

The Tamper Input Event (TAMPEVT) will always perform a timestamp capture. To capture on the external inputs (INn), the corresponding Input Action field in the Tamper Control register (TAMPCTRL.INnACT) must be written to '1'. If an input is set for wake functionality it does not capture the timestamp; however the Tamper flag and TAMPID will still be updated.

Note: Once the value from the TIMESTAMP register is read, the INTFLAG.TAMPER bit must be cleared. The next value from this register must be read only after the INTFLAG.TAMPER bit is set again.

21.6.8.5.2 Active Layer Protection

The RTC provides a mean of detecting broken traces on the PCB, also known as Active layer Protection. In this mode, a generated internal RTC signal can be directly routed over critical components on the board using the RTC_OUT output pin to one RTC INn input pin. A tamper condition is detected if there is a mismatch on the generated RTC signal.

The Active Layer Protection mode and the generation of the RTC signal is enabled by setting the RTCOUT bit in the Control B register (CTRLB.RTCOUT).

Note: The Active Layer Protection works with one output pin (RTC_OUT) and multiple input pin INn. This is achieved by clearing the Separate Tamper Output bit CTRLB.SEPTO.

Enabling active layer protection requires the following steps:

- Enable the RTC prescaler output by writing a '1' to the RTC Out bit in the Control B register (CTRLB.RTCOUT). The I/O pins must also be configured to correctly route the signal to the external pins.
- Select the frequency of the output signal by configuring the RTC Active Layer Frequency field in the Control B register (CTRLB.ACTF).

$$\text{CLK\_RTC\_OUT} = \frac{\text{CLK\_RTC}}{2^{\text{CTRLB.ACTF}+1}}$$

- Enable the tamper input n (INn) in Active Layer mode by writing '3' to the corresponding Input Action field in the Tamper Control register (TAMPCTRL.INnACT). When active layer protection is enabled and INn and OUTn pin are used, the value of INn is sampled on the falling edge of CLK_RTC and compared to the expected value of OUTn. Therefore up to one half of a CLK_RTC period is available for propagation delay through the trace.
- Enable Active Layer Protection by setting CTRLB.RTCOUT bit.

21.7 Register Summary - Mode 0 - 32-Bit Counter

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	MATCHCLR				MODE[1:0]		ENABLE	SWRST
		15:8	COUNTSYNC		BKTRST		PRESCALER[3:0]			
0x02	CTRLB	7:0	DMAEN	RTCOUT	DEBASYNC	DEBMAJ				GP0EN
		15:8		ACTF[2:0]				DEBF[2:0]		
0x04	EVCTRL	7:0	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
		15:8	OVFEO							
		23:16								
		31:24								
0x08	INTENCLR	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF							
0x0A	INTENSET	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF							
0x0C	INTFLAG	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF	TAMPER						
0x0E	DBGCTRL	7:0								DBGRUN
0x0F	Reserved									
0x10	SYNCBUSY	7:0					COUNT	FREQCORR	ENABLE	SWRST
		15:8	COUNTSYNC							
		23:16								
		31:24								
0x14	FREQCORR	7:0	SIGN	VALUE[6:0]						
0x15 ... 0x17	Reserved									
0x18	COUNT	7:0	COUNT[7:0]							
		15:8	COUNT[15:8]							
		23:16	COUNT[23:16]							
		31:24	COUNT[31:24]							
0x1C ... 0x1F	Reserved									
0x20	COMP0	7:0	COMP[7:0]							
		15:8	COMP[15:8]							
		23:16	COMP[23:16]							
		31:24	COMP[31:24]							
0x24	COMP1	7:0	COMP[7:0]							
		15:8	COMP[15:8]							
		23:16	COMP[23:16]							
		31:24	COMP[31:24]							
0x28 ... 0x3F	Reserved									
0x40	GP0	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							
0x44	GP1	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							
0x48	GP2	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x4C	GP3	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							
0x50 ... 0x5F	Reserved									
0x60	TAMPCTRL	7:0				IN3ACT[1:0]		IN2ACT[1:0]		IN0ACT[1:0]
		15:8								
		23:16					TAMLVL3	TAMLVL2	TAMLVL1	TAMLVL0
		31:24					DEBNC3	DEBNC2	DEBNC1	DEBNC0
0x64 ... 0x67	Reserved									
0x68	TAMPID	7:0					TAMPID3	TAMPID2	TAMPID1	TAMPID0
		15:8								
		23:16								
		31:24	TAMPEVT							
0x6C ... 0x7F	Reserved									
0x80	BKUP0	7:0	BKUP[7:0]							
		15:8	BKUP[15:8]							
		23:16	BKUP[23:16]							
		31:24	BKUP[31:24]							

21.8 Register Description - Mode 0 - 32-Bit Counter

This Register Description section is valid if the RTC is in COUNT32 mode (CTRLA.MODE=0).

21.8.1 Control A in COUNT32 mode (CTRLA.MODE=0)

Name: CTRLA
Offset: 0x00
Reset: 0x0000
Property: Enable-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	COUNTSYNC		BKTRST		PRESCALER[3:0]			
Access	R/W		R/W		R/W	R/W	R/W	R/W
Reset	0		0		0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MATCHCLR				MODE[1:0]		ENABLE	SWRST
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0

Bit 15 – COUNTSYNC COUNT Read Synchronization Enable
The COUNT register requires synchronization when reading. Disabling the synchronization will prevent reading valid values from the COUNT register.
This bit is not enable-protected.

Value	Description
0	COUNT read synchronization is disabled
1	COUNT read synchronization is enabled

Bit 13 – BKTRST BKUP Registers Reset On Tamper Enable
All BKUPn registers are affected. This bit can be written only when the peripheral is disabled.
This bit is not synchronized.

Value	Description
0	BKUPn registers will not reset when a tamper condition occurs.
1	BKUPn registers will reset when a tamper condition occurs.

Bits 11:8 – PRESCALER[3:0] Prescaler
These bits define the prescaling factor for the RTC clock source (GCLK_RTC) to generate the counter clock (CLK_RTC_CNT). Periodic events and interrupts are not available when the prescaler is off. These bits are not synchronized.

Value	Name	Description
0x0	OFF	CLK_RTC_CNT = GCLK_RTC/1
0x1	DIV1	CLK_RTC_CNT = GCLK_RTC/1
0x2	DIV2	CLK_RTC_CNT = GCLK_RTC/2
0x3	DIV4	CLK_RTC_CNT = GCLK_RTC/4
0x4	DIV8	CLK_RTC_CNT = GCLK_RTC/8
0x5	DIV16	CLK_RTC_CNT = GCLK_RTC/16
0x6	DIV32	CLK_RTC_CNT = GCLK_RTC/32
0x7	DIV64	CLK_RTC_CNT = GCLK_RTC/64
0x8	DIV128	CLK_RTC_CNT = GCLK_RTC/128
0x9	DIV256	CLK_RTC_CNT = GCLK_RTC/256
0xA	DIV512	CLK_RTC_CNT = GCLK_RTC/512
0xB	DIV1024	CLK_RTC_CNT = GCLK_RTC/1024
0xC-0xF	-	Reserved

Bit 7 – MATCHCLR Clear on Match
This bit defines if the counter is cleared or not on a match.
This bit is not synchronized.

Value	Description
0	The counter is not cleared on a Compare/Alarm match

Value	Description
1	The counter is cleared on a Compare/Alarm match

Bits 3:2 – MODE[1:0] Operating Mode

This bit group defines the operating mode of the RTC.
This bit is not synchronized.

Value	Name	Description
0x0	COUNT32	Mode 0: 32-bit counter
0x1	COUNT16	Mode 1: 16-bit counter
0x2	CLOCK	Mode 2: Clock/calendar
0x3	-	Reserved

Bit 1 – ENABLE Enable

Due to synchronization there is a delay between writing CTRLA.ENABLE and until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the Enable bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

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

Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the RTC (except DBGCTRL) to their initial state, and the RTC will be disabled.
Writing a '1' to CTRLA.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded.
Due to synchronization there is a delay between writing CTRLA.SWRST and until the reset is complete.
CTRLA.SWRST will be cleared when the reset is complete.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	There is not reset operation ongoing
1	The reset operation is ongoing

21.8.2 Control B in COUNT32 mode (CTRLA.MODE=0)

Name: CTRLB
Offset: 0x02
Reset: 0x0000
Property: Enable-Protected

Bit	15	14	13	12	11	10	9	8
		ACTF[2:0]				DEBF[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
	DMAEN	RTCOUT	DEBASYNC	DEBMAJ				GP0EN
Access	R/W	R/W	R/W	R/W				R/W
Reset	0	0	0	0				0

Bits 14:12 – ACTF[2:0] Active Layer Frequency
These bits define the prescaling factor for the RTC clock output (OUT) used during active layer protection in terms of the CLK_RTC.

Value	Name	Description
0x0	DIV2	CLK_RTC_OUT = CLK_RTC / 2
0x1	DIV4	CLK_RTC_OUT = CLK_RTC / 4
0x2	DIV8	CLK_RTC_OUT = CLK_RTC / 8
0x3	DIV16	CLK_RTC_OUT = CLK_RTC / 16
0x4	DIV32	CLK_RTC_OUT = CLK_RTC / 32
0x5	DIV64	CLK_RTC_OUT = CLK_RTC / 64
0x6	DIV128	CLK_RTC_OUT = CLK_RTC / 128
0x7	DIV256	CLK_RTC_OUT = CLK_RTC / 256

Bits 10:8 – DEBF[2:0] Debounce Frequency
These bits define the prescaling factor for the input debouncers in terms of the CLK_RTC.

Value	Name	Description
0x0	DIV2	CLK_RTC_DEB = CLK_RTC / 2
0x1	DIV4	CLK_RTC_DEB = CLK_RTC / 4
0x2	DIV8	CLK_RTC_DEB = CLK_RTC / 8
0x3	DIV16	CLK_RTC_DEB = CLK_RTC / 16
0x4	DIV32	CLK_RTC_DEB = CLK_RTC / 32
0x5	DIV64	CLK_RTC_DEB = CLK_RTC / 64
0x6	DIV128	CLK_RTC_DEB = CLK_RTC / 128
0x7	DIV256	CLK_RTC_DEB = CLK_RTC / 256

Bit 7 – DMAEN DMA Enable
The RTC can trigger a DMA request when the timestamp is ready in the TIMESTAMP register.

Value	Description
0	Tamper DMA request is disabled. Reading TIMESTAMP has no effect on INTFLAG.TAMPER.
1	Tamper DMA request is enabled. Reading TIMESTAMP will clear INTFLAG.TAMPER.

Bit 6 – RTCOUT RTC Output Enable

Value	Description
0	The RTC active layer output is disabled.
1	The RTC active layer output is enabled.

Bit 5 – DEBASYNC Debouncer Asynchronous Enable

Value	Description
0	The tamper input debouncers operate synchronously.
1	The tamper input debouncers operate asynchronously.

Bit 4 – DEBMAJ Debouncer Majority Enable

Value	Description
0	The tamper input debouncers match three equal values.
1	The tamper input debouncers match majority two of three values.

Bit 0 – GP0EN General Purpose 0 Enable

Value	Description
0	COMP0 compare function enabled. GP0/GP1 disabled.
1	COMP0 compare function disabled. GP0/GP1 enabled.

21.8.3 Event Control in COUNT32 mode (CTRLA.MODE=0)

Name: EVCTRL
Offset: 0x04
Reset: 0x00000000
Property: Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	OVFEO							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVFEO Overflow Event Output Enable

Value	Description
0	Overflow event is disabled and will not be generated.
1	Overflow event is enabled and will be generated for every overflow.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PEREOn Periodic Interval n Event Output Enable [n = 7..0]

Value	Description
0	Periodic Interval n event is disabled and will not be generated.
1	Periodic Interval n event is enabled and will be generated.

21.8.4 Interrupt Enable Clear in COUNT32 mode (CTRLA.MODE=0)

Name: INTENCLR
Offset: 0x08
Reset: 0x0000

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set (INTENSET) register.

Bit	15	14	13	12	11	10	9	8
	OVF							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVF Overflow Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Overflow Interrupt Enable bit, which disables the Overflow interrupt.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PERn Periodic Interval n Interrupt Enable [n = 7..0]
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Periodic Interval n Interrupt Enable bit, which disables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.

21.8.5 Interrupt Enable Set in COUNT32 mode (CTRLA.MODE=0)

Name: INTENSET
Offset: 0x0A
Reset: 0x0000

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

Bit	15	14	13	12	11	10	9	8
	OVF							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVF Overflow Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Overflow Interrupt Enable bit, which enables the Overflow interrupt.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PERn Periodic Interval n Interrupt Enable [n = 7..0]
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Periodic Interval n Interrupt Enable bit, which enables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.

21.8.6 Interrupt Flag Status and Clear in COUNT32 mode (CTRLA.MODE=0)

Name: INTFLAG
Offset: 0x0C
Reset: 0x0000
Property: -

Bit	15	14	13	12	11	10	9	8
	OVF	TAMPER						
Access	R/W	R/W						
Reset	0	0						

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVF Overflow
This flag is cleared by writing a '1' to the flag.
This flag is set on the next CLK_RTC_CNT cycle after an overflow condition occurs, and an interrupt request will be generated if INTENCLR/SET.OVF is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Overflow interrupt flag.

Bit 14 – TAMPER Tamper event
This flag is set after a tamper condition occurs, and an interrupt request will be generated if INTENCLR.TAMPER/INTENSET.TAMPER is '1'. Writing a '0' to this bit has no effect. Writing a '1' to this bit clears the Tamper interrupt flag.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PERn Periodic Interval n [n = 7..0]
This flag is cleared by writing a '1' to the flag.
This flag is set on the 0-to-1 transition of prescaler bit [n+2], and an interrupt request will be generated if INTENCLR/SET.PERn is one.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Periodic Interval n interrupt flag.

21.8.7 Debug Control

Name: DBGCTRL
Offset: 0x0E
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN Debug Run

This bit is not reset by a software reset.

This bit controls the functionality when the CPU is halted by an external debugger.

Value	Description
0	The RTC is halted when the CPU is halted by an external debugger.
1	The RTC continues normal operation when the CPU is halted by an external debugger.

21.8.8 Synchronization Busy in COUNT32 mode (CTRLA.MODE=0)

Name: SYNCBUSY
Offset: 0x10
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	COUNTSYNC							
Access	R							
Reset	0							

Bit	7	6	5	4	3	2	1	0
					COUNT	FREQCORR	ENABLE	SWRST
Access					R	R	R	R
Reset					0	0	0	0

Bit 15 – COUNTSYNC Count Read Sync Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.COUNTSYNC bit is complete.
1	Write synchronization for CTRLA.COUNTSYNC bit is ongoing.

Bit 3 – COUNT Count Value Synchronization Busy Status

Value	Description
0	Read/write synchronization for COUNT register is complete.
1	Read/write synchronization for COUNT register is ongoing.

Bit 2 – FREQCORR Frequency Correction Synchronization Busy Status

Value	Description
0	Write synchronization for FREQCORR register is complete.
1	Write synchronization for FREQCORR register is ongoing.

Bit 1 – ENABLE Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.ENABLE bit is complete.
1	Write synchronization for CTRLA.ENABLE bit is ongoing.

Bit 0 – SWRST Software Reset Synchronization Busy Status

Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	Write synchronization for CTRLA.SWRST bit is complete.
1	Write synchronization for CTRLA.SWRST bit is ongoing.

21.8.9 Frequency Correction

Name: FREQCORR
Offset: 0x14
Reset: 0x00
Property: Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	SIGN	VALUE[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – SIGN Correction Sign

Value	Description
0	The correction value is positive, i.e., frequency will be decreased.
1	The correction value is negative, i.e., frequency will be increased.

Bits 6:0 – VALUE[6:0] Correction Value

These bits define the amount of correction applied to the RTC prescaler.

Value	Description
0	Correction is disabled and the RTC frequency is unchanged.
1 - 127	The RTC frequency is adjusted according to the value.

21.8.10 Counter Value in COUNT32 mode (CTRLA.MODE=0)

Name: COUNT
Offset: 0x18
Reset: 0x00000000
Property: Write-Synchronized, Read-Synchronized

Bit	31	30	29	28	27	26	25	24
	COUNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	COUNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COUNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – COUNT[31:0] Counter Value
These bits define the value of the 32-bit RTC counter in mode 0.

21.8.11 Compare 0 Value in COUNT32 mode (CTRLA.MODE=0)

Name: COMP0
Offset: 0x20
Reset: 0x00000000
Property: Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	COMP[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	COMP[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COMP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – COMP[31:0] Compare Value

The 32-bit value of COMPn is continuously compared with the 32-bit COUNT value. When a match occurs, the Compare n interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMPn) is set on the next counter cycle, and the counter value is cleared if CTRLA.MATCHCLR is one.

21.8.12 Compare 1 Value in COUNT32 mode (CTRLA.MODE=1)

Name: COMP1
Offset: 0x24
Reset: 0x00000000
Property: Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	COMP[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	COMP[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COMP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – COMP[31:0] Compare Value

The 32-bit value of COMPn is continuously compared with the 32-bit COUNT value. When a match occurs, the Compare n interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMPn) is set on the next counter cycle, and the counter value is cleared if CTRLA.MATCHCLR is one.

21.8.13 General Purpose n

Name: GPn
Offset: 0x40 + n*0x04 [n=0..3]
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	GP[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	GP[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	GP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – GP[31:0] General Purpose
These bits are for user-defined general purpose use, see *General Purpose Registers* from Related Links.
Related Links
21.6.8.4. General Purpose Registers

21.8.14 Tamper Control

Name: TAMPCTRL
Offset: 0x60
Reset: 0x00000000
Property: Enable-Protected

Bit	31	30	29	28	27	26	25	24
					DEBNC3	DEBNC2	DEBNC1	DEBNC0
Access								
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
					TAMLVL3	TAMLVL2	TAMLVL1	TAMLVL0
Access								
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				IN3ACT[1:0]		IN2ACT[1:0]		IN0ACT[1:0]
Access								
Reset				0	0	0	0	0

Bits 24, 25, 26, 27 – DEBNCn Debounce Enable of Tamper Input INn [n=0..3]
Note: Debounce feature does not apply to the Active Layer Protection mode (TAMPCTRL.INACT = ACTL).

Value	Description
0	Debouncing is disabled for Tamper input INn
1	Debouncing is enabled for Tamper input INn

Bits 16, 17, 18, 19 – TAMLVLn Tamper Level Select of Tamper Input INn [n=0..3]
Note: Tamper Level feature does not apply to the Active Layer Protection mode (TAMPCTRL.INACT = ACTL).

Value	Description
0	A falling edge condition will be detected on Tamper input INn.
1	A rising edge condition will be detected on Tamper input INn.

Bits 0:1, 1:2, 2:3, 3:4 – INnACT Tamper Channel n Action [n=0..3]
These bits determine the action taken by Tamper Channel n.

Value	Name	Description
0x0	OFF	Off (Disabled)
0x1	WAKE	Wake and set Tamper flag
0x2	CAPTURE	Capture timestamp and set Tamper flag
0x3	ACTL	Compare RTC signal routed between INn and OUT pins . When a mismatch occurs, capture timestamp and set Tamper flag

21.8.15 Tamper ID

Name: TAMPID
Offset: 0x68
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
	TAMPEVT							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					TAMPID3	TAMPID2	TAMPID1	TAMPID0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 31 – TAMPEVT Tamper Event Detected

Writing a '0' to this bit has no effect. Writing a '1' to this bit clears the tamper detection bit.

Value	Description
0	A tamper input event has not been detected
1	A tamper input event has been detected

Bits 0, 1, 2, 3 – TAMPIDn Tamper on Channel n Detected [n=0..3]

Writing a '0' to this bit has no effect. Writing a '1' to this bit clears the tamper detection bit.

Value	Description
0	A tamper condition has not been detected on Channel n
1	A tamper condition has been detected on Channel n

21.8.16 Backup0

Name: BKUP0
Offset: 0x80
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
	BKUP[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BKUP[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BKUP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – BKUP[31:0] Backup
These bits are user-defined for general purpose use in the Backup domain.

21.9 Register Summary - Mode 1 - 16-Bit Counter

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0					MODE[1:0]		ENABLE	SWRST
		15:8	COUNTSYNC		BKTRST		PRESCALER[3:0]			
0x02	CTRLB	7:0	DMAEN	RTCOUT	DEBASYNC	DEBMAJ				GP0EN
		15:8		ACTF[2:0]				DEBF[2:0]		
0x04	EVCTRL	7:0	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
		15:8	OVFEO							
		23:16								
		31:24								
0x08	INTENCLR	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF							
0x0A	INTENSET	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF							
0x0C	INTFLAG	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF							
0x0E	DBGCTRL	7:0								DBGRUN
0x0F	Reserved									
0x10	SYNCBUSY	7:0				PER	COUNT	FREQCORR	ENABLE	SWRST
		15:8	COUNTSYNC							
		23:16								
		31:24								
0x14	FREQCORR	7:0	SIGN	VALUE[6:0]						
0x15 ... 0x17	Reserved									
0x18	COUNT	7:0	COUNT[7:0]							
		15:8	COUNT[15:8]							
0x1A ... 0x1B	Reserved									
0x1C	PER	7:0	PER[7:0]							
		15:8	PER[15:8]							
0x1E ... 0x1F	Reserved									
0x20	COMP0	7:0	COMP[7:0]							
		15:8	COMP[15:8]							
0x22	COMP1	7:0	COMP[7:0]							
		15:8	COMP[15:8]							
0x24	COMP2	7:0	COMP[7:0]							
		15:8	COMP[15:8]							
0x26	COMP3	7:0	COMP[7:0]							
		15:8	COMP[15:8]							
0x28 ... 0x3F	Reserved									
0x40	GP0	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							
0x44	GP1	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x48	GP2	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							
0x4C	GP3	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							
0x50 ... 0x5F	Reserved									
0x60	TAMPCTRL	7:0				IN3ACT[1:0]		IN2ACT[1:0]		IN0ACT[1:0]
		15:8								
		23:16					TAMLVL3	TAMLVL2	TAMLVL1	TAMLVL0
		31:24					DEBNC3	DEBNC2	DEBNC1	DEBNC0
0x64 ... 0x67	Reserved									
0x68	TAMPID	7:0					TAMPID3	TAMPID2	TAMPID1	TAMPID0
		15:8								
		23:16								
		31:24	TAMPEVT							
0x6C ... 0x7F	Reserved									
0x80	BKUP0	7:0	BKUP[7:0]							
		15:8	BKUP[15:8]							
		23:16	BKUP[23:16]							
		31:24	BKUP[31:24]							

21.10 Register Description - Mode 1 - 16-Bit Counter

This Register Description section is valid if the RTC is in COUNT16 mode (CTRLA.MODE=1).

21.10.1 Control A in COUNT16 mode (CTRLA.MODE=1)

Name: CTRLA
Offset: 0x00
Reset: 0x0000
Property: Enable-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	COUNTSYNC		BKTRST		PRESCALER[3:0]			
Access	R/W		R/W		R/W	R/W	R/W	R/W
Reset	0		0		0	0	0	0

Bit	7	6	5	4	3	2	1	0
					MODE[1:0]		ENABLE	SWRST
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 15 – COUNTSYNC COUNT Read Synchronization Enable
The COUNT register requires synchronization when reading. Disabling the synchronization will prevent reading valid values from the COUNT register.
This bit is not enable-protected.

Value	Description
0	COUNT read synchronization is disabled
1	COUNT read synchronization is enabled

Bit 13 – BKTRST BKUP Registers Reset On Tamper Enable
All BKUPn registers are affected. This bit can be written only when the peripheral is disabled.
This bit is not synchronized.

Value	Description
0	BKUPn registers will not reset when a tamper condition occurs.
1	BKUPn registers will reset when a tamper condition occurs.

Bits 11:8 – PRESCALER[3:0] Prescaler
These bits define the prescaling factor for the RTC clock source (GCLK_RTC) to generate the counter clock (CLK_RTC_CNT). Periodic events and interrupts are not available when the prescaler is off. These bits are not synchronized.

Value	Name	Description
0x0	OFF	CLK_RTC_CNT = GCLK_RTC/1
0x1	DIV1	CLK_RTC_CNT = GCLK_RTC/1
0x2	DIV2	CLK_RTC_CNT = GCLK_RTC/2
0x3	DIV4	CLK_RTC_CNT = GCLK_RTC/4
0x4	DIV8	CLK_RTC_CNT = GCLK_RTC/8
0x5	DIV16	CLK_RTC_CNT = GCLK_RTC/16
0x6	DIV32	CLK_RTC_CNT = GCLK_RTC/32
0x7	DIV64	CLK_RTC_CNT = GCLK_RTC/64
0x8	DIV128	CLK_RTC_CNT = GCLK_RTC/128
0x9	DIV256	CLK_RTC_CNT = GCLK_RTC/256
0xA	DIV512	CLK_RTC_CNT = GCLK_RTC/512
0xB	DIV1024	CLK_RTC_CNT = GCLK_RTC/1024
0xC-0xF	-	Reserved

Bits 3:2 – MODE[1:0] Operating Mode
This field defines the operating mode of the RTC. This bit is not synchronized.

Value	Name	Description
0x0	COUNT32	Mode 0: 32-bit counter
0x1	COUNT16	Mode 1: 16-bit counter

Value	Name	Description
0x2	CLOCK	Mode 2: Clock/calendar
0x3	-	Reserved

Bit 1 – ENABLE Enable

Due to synchronization there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the Enable bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

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

Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the RTC (except DBGCTRL) to their initial state, and the RTC will be disabled.
Writing a '1' to CTRLA.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded.
Due to synchronization there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST will be cleared when the reset is complete.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	There is not reset operation ongoing
1	The reset operation is ongoing

21.10.2 Control B in COUNT16 mode (CTRLA.MODE=1)

Name: CTRLB
Offset: 0x02
Reset: 0x0000
Property: Enable-Protected

Bit	15	14	13	12	11	10	9	8
		ACTF[2:0]				DEBF[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
	DMAEN	RTCOUT	DEBASYNC	DEBMAJ				GP0EN
Access	R/W	R/W	R/W	R/W				R/W
Reset	0	0	0	0				0

Bits 14:12 – ACTF[2:0] Active Layer Frequency
These bits define the prescaling factor for the RTC clock output (OUT) used during active layer protection in terms of the CLK_RTC.

Value	Name	Description
0x0	DIV2	CLK_RTC_OUT = CLK_RTC / 2
0x1	DIV4	CLK_RTC_OUT = CLK_RTC / 4
0x2	DIV8	CLK_RTC_OUT = CLK_RTC / 8
0x3	DIV16	CLK_RTC_OUT = CLK_RTC / 16
0x4	DIV32	CLK_RTC_OUT = CLK_RTC / 32
0x5	DIV64	CLK_RTC_OUT = CLK_RTC / 64
0x6	DIV128	CLK_RTC_OUT = CLK_RTC / 128
0x7	DIV256	CLK_RTC_OUT = CLK_RTC / 256

Bits 10:8 – DEBF[2:0] Debounce Frequency
These bits define the prescaling factor for the input debouncers in terms of the CLK_RTC.

Value	Name	Description
0x0	DIV2	CLK_RTC_DEB = CLK_RTC / 2
0x1	DIV4	CLK_RTC_DEB = CLK_RTC / 4
0x2	DIV8	CLK_RTC_DEB = CLK_RTC / 8
0x3	DIV16	CLK_RTC_DEB = CLK_RTC / 16
0x4	DIV32	CLK_RTC_DEB = CLK_RTC / 32
0x5	DIV64	CLK_RTC_DEB = CLK_RTC / 64
0x6	DIV128	CLK_RTC_DEB = CLK_RTC / 128
0x7	DIV256	CLK_RTC_DEB = CLK_RTC / 256

Bit 7 – DMAEN DMA Enable
The RTC can trigger a DMA request when the timestamp is ready in the TIMESTAMP register.

Value	Description
0	Tamper DMA request is disabled. Reading TIMESTAMP has no effect on INTFLAG.TAMPER.
1	Tamper DMA request is enabled. Reading TIMESTAMP will clear INTFLAG.TAMPER.

Bit 6 – RTCOUT RTC Output Enable

Value	Description
0	The RTC active layer output is disabled.
1	The RTC active layer output is enabled.

Bit 5 – DEBASYNC Debouncer Asynchronous Enable

Value	Description
0	The tamper input debouncers operate synchronously.
1	The tamper input debouncers operate asynchronously.

Bit 4 – DEBMAJ Debouncer Majority Enable

Value	Description
0	The tamper input debouncers match three equal values.
1	The tamper input debouncers match majority two of three values.

Bit 0 – GP0EN General Purpose 0 Enable

Value	Description
0	COMP0 compare function enabled. GP0/GP1 disabled.
1	COMP0 compare function disabled. GP0/GP1 enabled.

21.10.3 Event Control in COUNT16 mode (CTRLA.MODE=1)

Name: EVCTRL
Offset: 0x04
Reset: 0x00000000
Property: Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	OVFEO							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVFEO Overflow Event Output Enable

Value	Description
0	Overflow event is disabled and will not be generated.
1	Overflow event is enabled and will be generated for every overflow.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PEREOn Periodic Interval n Event Output Enable [n = 7..0]

Value	Description
0	Periodic Interval n event is disabled and will not be generated.
1	Periodic Interval n event is enabled and will be generated.

21.10.4 Interrupt Enable Clear in COUNT16 mode (CTRLA.MODE=1)

Name: INTENCLR
Offset: 0x08
Reset: 0x0000

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set (INTENSET) register.

Bit	15	14	13	12	11	10	9	8
	OVF							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVF Overflow Interrupt Enable
Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Overflow Interrupt Enable bit, which disables the Overflow interrupt.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PERn Periodic Interval n Interrupt Enable [n = 7..0]
Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Periodic Interval n Interrupt Enable bit, which disables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.

21.10.5 Interrupt Enable Set in COUNT16 mode (CTRLA.MODE=1)

Name: INTENSET
Offset: 0x0A
Reset: 0x0000

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

Bit	15	14	13	12	11	10	9	8
	OVF							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVF Overflow Interrupt Enable
Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Overflow Interrupt Enable bit, which enables the Overflow interrupt.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PERn Periodic Interval n Interrupt Enable [n = 7..0]
Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Periodic Interval n Interrupt Enable bit, which enables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.

21.10.6 Interrupt Flag Status and Clear in COUNT16 mode (CTRLA.MODE=1)

Name: INTFLAG
Offset: 0x0C
Reset: 0x0000
Property: -

Bit	15	14	13	12	11	10	9	8
	OVF							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVF Overflow
This flag is cleared by writing a '1' to the flag.
This flag is set on the next CLK_RTC_CNT cycle after an overflow condition occurs, and an interrupt request will be generated if INTENCLR/SET.OVF is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Overflow interrupt flag.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PERn Periodic Interval n [n = 7..0]
This flag is cleared by writing a '1' to the flag.
This flag is set on the 0-to-1 transition of prescaler bit [n+2], and an interrupt request will be generated if INTENCLR/SET.PERx is one.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Periodic Interval n interrupt flag.

21.10.7 Debug Control

Name: DBGCTRL
Offset: 0x0E
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN Debug Run

This bit is not reset by a software reset.

This bit controls the functionality when the CPU is halted by an external debugger.

Value	Description
0	The RTC is halted when the CPU is halted by an external debugger.
1	The RTC continues normal operation when the CPU is halted by an external debugger.

21.10.8 Synchronization Busy in COUNT16 mode (CTRLA.MODE=1)

Name: SYNCBUSY
Offset: 0x10
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	COUNTSYNC							
Access	R							
Reset	0							

Bit	7	6	5	4	3	2	1	0
				PER	COUNT	FREQCORR	ENABLE	SWRST
Access				R	R	R	R	R
Reset				0	0	0	0	0

Bit 15 – COUNTSYNC Count Read Sync Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.COUNTSYNC bit is complete.
1	Write synchronization for CTRLA.COUNTSYNC bit is ongoing.

Bit 4 – PER Period Synchronization Busy Status

Value	Description
0	Write synchronization for PER register is complete.
1	Write synchronization for PER register is ongoing.

Bit 3 – COUNT Count Value Synchronization Busy Status

Value	Description
0	Read/write synchronization for COUNT register is complete.
1	Read/write synchronization for COUNT register is ongoing.

Bit 2 – FREQCORR Frequency Correction Synchronization Busy Status

Value	Description
0	Write synchronization for FREQCORR register is complete.
1	Write synchronization for FREQCORR register is ongoing.

Bit 1 – ENABLE Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.ENABLE bit is complete.
1	Write synchronization for CTRLA.ENABLE bit is ongoing.

Bit 0 – SWRST Software Reset Synchronization Busy Status

Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	Write synchronization for CTRLA.SWRST bit is complete.
1	Write synchronization for CTRLA.SWRST bit is ongoing.

21.10.9 Frequency Correction

Name: FREQCORR
Offset: 0x14
Reset: 0x00
Property: Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	SIGN	VALUE[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – SIGN Correction Sign

Value	Description
0	The correction value is positive, i.e., frequency will be decreased.
1	The correction value is negative, i.e., frequency will be increased.

Bits 6:0 – VALUE[6:0] Correction Value

These bits define the amount of correction applied to the RTC prescaler.

Value	Description
0	Correction is disabled and the RTC frequency is unchanged.
1 - 127	The RTC frequency is adjusted according to the value.

21.10.10 Counter Value in COUNT16 mode (CTRLA.MODE=1)

Name: COUNT
Offset: 0x18
Reset: 0x0000
Property: Write-Synchronized, Read-Synchronized

Bit	15	14	13	12	11	10	9	8
	COUNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:0 – COUNT[15:0] Counter Value
These bits define the value of the 16-bit RTC counter in COUNT16 mode (CTRLA.MODE=1).

21.10.11 Counter Period in COUNT16 mode (CTRLA.MODE=1)

Name: PER
Offset: 0x1C
Reset: 0x0000
Property: Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	PER[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:0 – PER[15:0] Counter Period
These bits define the value of the 16-bit RTC period in COUNT16 mode (CTRLA.MODE=1).

21.10.12 Compare n Value in COUNT16 mode (CTRLA.MODE=1)

Name: COMP
Offset: 0x20 + n*0x02 [n=0..3]
Reset: 0x0000
Property: PAC Write-Protection, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	COMP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:0 – COMP[15:0] Compare Value
The 16-bit value of COMPn is continuously compared with the 16-bit COUNT value. When a match occurs, the Compare n interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMPn) is set on the next counter cycle.

21.10.13 General Purpose n

Name: GPn
Offset: 0x40 + n*0x04 [n=0..3]
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	GP[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	GP[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	GP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – GP[31:0] General Purpose
These bits are for user-defined general purpose use, see *General Purpose Registers* from Related Links.
Related Links
21.6.8.4. General Purpose Registers

21.10.14 Tamper Control

Name: TAMPCTRL
Offset: 0x60
Reset: 0x00000000
Property: Enable-Protected

Bit	31	30	29	28	27	26	25	24
					DEBNC3	DEBNC2	DEBNC1	DEBNC0
Access								
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
					TAMLVL3	TAMLVL2	TAMLVL1	TAMLVL0
Access								
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				IN3ACT[1:0]		IN2ACT[1:0]		IN0ACT[1:0]
Access								
Reset				0	0	0	0	0

Bits 24, 25, 26, 27 – DEBNCn Debounce Enable of Tamper Input INn [n=0..3]
Note: Debounce feature does not apply to the Active Layer Protection mode (TAMPCTRL.INACT = ACTL).

Value	Description
0	Debouncing is disabled for Tamper input INn
1	Debouncing is enabled for Tamper input INn

Bits 16, 17, 18, 19 – TAMLVLn Tamper Level Select of Tamper Input INn [n=0..3]
Note: Tamper Level feature does not apply to the Active Layer Protection mode (TAMPCTRL.INACT = ACTL).

Value	Description
0	A falling edge condition will be detected on Tamper input INn.
1	A rising edge condition will be detected on Tamper input INn.

Bits 0:1, 1:2, 2:3, 3:4 – INnACT Tamper Channel n Action [n=0..3]
These bits determine the action taken by Tamper Channel n.

Value	Name	Description
0x0	OFF	Off (Disabled)
0x1	WAKE	Wake and set Tamper flag
0x2	CAPTURE	Capture timestamp and set Tamper flag
0x3	ACTL	Compare RTC signal routed between INn and OUT pins . When a mismatch occurs, capture timestamp and set Tamper flag

21.10.15 Tamper ID

Name: TAMPID
Offset: 0x68
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
	TAMPEVT							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					TAMPID3	TAMPID2	TAMPID1	TAMPID0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 31 – TAMPEVT Tamper Event Detected

Writing a '0' to this bit has no effect. Writing a '1' to this bit clears the tamper detection bit.

Value	Description
0	A tamper input event has not been detected
1	A tamper input event has been detected

Bits 0, 1, 2, 3 – TAMPIDn Tamper on Channel n Detected [n=0..3]

Writing a '0' to this bit has no effect. Writing a '1' to this bit clears the tamper detection bit.

Value	Description
0	A tamper condition has not been detected on Channel n
1	A tamper condition has been detected on Channel n

21.10.16 Backup0

Name: BKUP0
Offset: 0x80
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
	BKUP[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BKUP[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BKUP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – BKUP[31:0] Backup
These bits are user-defined for general purpose use in the Backup domain.

21.11 Register Summary - Mode 2 - Clock/Calendar

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	MATCHCLR	CLKREP			MODE[1:0]		ENABLE	SWRST
		15:8	CLOCKSYNC		BKTRST		PRESCALER[3:0]			
0x02	CTRLB	7:0	DMAEN	RTCOUT	DEBASYNC	DEBMAJ				GP0EN
		15:8		ACTF[2:0]				DEBF[2:0]		
0x04	EVCTRL	7:0	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
		15:8	OVFEO						ALARMEO1	ALARMEO0
		23:16								
		31:24								
0x08	INTENCLR	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF							
0x0A	INTENSET	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF							
0x0C	INTFLAG	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF							
0x0E	DBGCTRL	7:0								DBGRUN
0x0F	Reserved									
0x10	SYNCBUSY	7:0					CLOCK	FREQCORR	ENABLE	SWRST
		15:8	CLOCKSYNC							
		23:16								
		31:24								
0x14	FREQCORR	7:0	SIGN	VALUE[6:0]						
0x15 ... 0x17	Reserved									
0x18	CLOCK	7:0	MINUTE[1:0]		SECOND[5:0]					
		15:8	HOUR[3:0]				MINUTE[5:2]			
		23:16	MONTH[1:0]		DAY[4:0]					HOUR[4]
		31:24	YEAR[5:0]						MONTH[3:2]	
0x1C ... 0x1F	Reserved									
0x20	ALARM0	7:0	MINUTE[1:0]		SECOND[5:0]					
		15:8	HOUR[3:0]				MINUTE[5:2]			
		23:16	MONTH[1:0]		DAY[4:0]					HOUR[4]
		31:24	YEAR[5:0]						MONTH[3:2]	
0x24	MASK0	7:0						SEL[2:0]		
0x25 ... 0x27	Reserved									
0x28	ALARM1	7:0	MINUTE[1:0]		SECOND[5:0]					
		15:8	HOUR[3:0]				MINUTE[5:2]			
		23:16	MONTH[1:0]		DAY[4:0]					HOUR[4]
		31:24	YEAR[5:0]						MONTH[3:2]	
0x2C	MASK1	7:0						SEL[2:0]		
0x2D ... 0x3F	Reserved									
0x40	GP0	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x44	GP1	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							
0x48	GP2	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							
0x4C	GP3	7:0	GP[7:0]							
		15:8	GP[15:8]							
		23:16	GP[23:16]							
		31:24	GP[31:24]							
0x50 ... 0x5F	Reserved									
0x60	TAMPCTRL	7:0				IN3ACT[1:0]		IN2ACT[1:0]		IN0ACT[1:0]
		15:8								
		23:16					TAMLVL3	TAMLVL2	TAMLVL1	TAMLVL0
		31:24					DEBNC3	DEBNC2	DEBNC1	DEBNC0
0x64 ... 0x67	Reserved									
0x68	TAMPID	7:0					TAMPID3	TAMPID2	TAMPID1	TAMPID0
		15:8								
		23:16								
		31:24	TAMPEVT							
0x6C ... 0x7F	Reserved									
0x80	BKUP0	7:0	BKUP[7:0]							
		15:8	BKUP[15:8]							
		23:16	BKUP[23:16]							
		31:24	BKUP[31:24]							

21.12 Register Description - Mode 2 - Clock/Calendar

This Register Description section is valid if the RTC is in Clock/Calendar mode (CTRLA.MODE=2).

21.12.1 Control A in Clock/Calendar mode (CTRLA.MODE=2)

Name: CTRLA
Offset: 0x00
Reset: 0x0000
Property: Enable-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	CLOCKSYNC		BKTRST		PRESCALER[3:0]			
Access	R/W		R/W		R/W	R/W	R/W	R/W
Reset	0		0		0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MATCHCLR	CLKREP			MODE[1:0]		ENABLE	SWRST
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit 15 – CLOCKSYNC CLOCK Read Synchronization Enable
The CLOCK register requires synchronization when reading. Disabling the synchronization will prevent reading valid values from the CLOCK register.
This bit is not enable-protected.

Value	Description
0	CLOCK read synchronization is disabled
1	CLOCK read synchronization is enabled

Bit 13 – BKTRST BKUP Registers Reset On Tamper Enable
All BKUPn registers are affected. This bit can be written only when the peripheral is disabled.
This bit is not synchronized.

Value	Description
0	BKUPn registers will not reset when a tamper condition occurs.
1	BKUPn registers will reset when a tamper condition occurs.

Bits 11:8 – PRESCALER[3:0] Prescaler
These bits define the prescaling factor for the RTC clock source (GCLK_RTC) to generate the counter clock (CLK_RTC_CNT). Periodic events and interrupts are not available when the prescaler is off. These bits are not synchronized.

Value	Name	Description
0x0	OFF	CLK_RTC_CNT = GCLK_RTC/1
0x1	DIV1	CLK_RTC_CNT = GCLK_RTC/1
0x2	DIV2	CLK_RTC_CNT = GCLK_RTC/2
0x3	DIV4	CLK_RTC_CNT = GCLK_RTC/4
0x4	DIV8	CLK_RTC_CNT = GCLK_RTC/8
0x5	DIV16	CLK_RTC_CNT = GCLK_RTC/16
0x6	DIV32	CLK_RTC_CNT = GCLK_RTC/32
0x7	DIV64	CLK_RTC_CNT = GCLK_RTC/64
0x8	DIV128	CLK_RTC_CNT = GCLK_RTC/128
0x9	DIV256	CLK_RTC_CNT = GCLK_RTC/256
0xA	DIV512	CLK_RTC_CNT = GCLK_RTC/512
0xB	DIV1024	CLK_RTC_CNT = GCLK_RTC/1024
0xC-0xF	-	Reserved

Bit 7 – MATCHCLR Clear on Match
This bit is valid only in Mode 0 (COUNT32) and Mode 2 (CLOCK). This bit can be written only when the peripheral is disabled. This bit is not synchronized.

Value	Description
0	The counter is not cleared on a Compare/Alarm 0 match

Value	Description
1	The counter is cleared on a Compare/Alarm 0 match

Bit 6 – CLKREP Clock Representation

This bit is valid only in Mode 2 and determines how the hours are represented in the Clock Value (CLOCK) register. This bit can be written only when the peripheral is disabled. This bit is not synchronized.

Value	Description
0	24 Hour
1	12 Hour (AM/PM)

Bits 3:2 – MODE[1:0] Operating Mode

This field defines the operating mode of the RTC. This bit is not synchronized.

Value	Name	Description
0x0	COUNT32	Mode 0: 32-bit counter
0x1	COUNT16	Mode 1: 16-bit counter
0x2	CLOCK	Mode 2: Clock/calendar
0x3	-	Reserved

Bit 1 – ENABLE Enable

Due to synchronization there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the Enable bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

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

Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the RTC, except DBGCTRL, to their initial state, and the RTC will be disabled.
Writing a '1' to CTRLA.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded.
Due to synchronization there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST will be cleared when the reset is complete.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	There is not reset operation ongoing
1	The reset operation is ongoing

21.12.2 Control B in Clock/Calendar mode (CTRLA.MODE=2)

Name: CTRLB
Offset: 0x2
Reset: 0x0000
Property: Enable-Protected

Bit	15	14	13	12	11	10	9	8
		ACTF[2:0]				DEBF[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
	DMAEN	RTCOUT	DEBASYNC	DEBMAJ				GP0EN
Access	R/W	R/W	R/W	R/W				R/W
Reset	0	0	0	0				0

Bits 14:12 – ACTF[2:0] Active Layer Frequency
These bits define the prescaling factor for the RTC clock output (OUT) used during active layer protection in terms of the CLK_RTC.

Value	Name	Description
0x0	DIV2	CLK_RTC_OUT = CLK_RTC / 2
0x1	DIV4	CLK_RTC_OUT = CLK_RTC / 4
0x2	DIV8	CLK_RTC_OUT = CLK_RTC / 8
0x3	DIV16	CLK_RTC_OUT = CLK_RTC / 16
0x4	DIV32	CLK_RTC_OUT = CLK_RTC / 32
0x5	DIV64	CLK_RTC_OUT = CLK_RTC / 64
0x6	DIV128	CLK_RTC_OUT = CLK_RTC / 128
0x7	DIV256	CLK_RTC_OUT = CLK_RTC / 256

Bits 10:8 – DEBF[2:0] Debounce Frequency
These bits define the prescaling factor for the input debouncers in terms of the CLK_RTC.

Value	Name	Description
0x0	DIV2	CLK_RTC_DEB = CLK_RTC / 2
0x1	DIV4	CLK_RTC_DEB = CLK_RTC / 4
0x2	DIV8	CLK_RTC_DEB = CLK_RTC / 8
0x3	DIV16	CLK_RTC_DEB = CLK_RTC / 16
0x4	DIV32	CLK_RTC_DEB = CLK_RTC / 32
0x5	DIV64	CLK_RTC_DEB = CLK_RTC / 64
0x6	DIV128	CLK_RTC_DEB = CLK_RTC / 128
0x7	DIV256	CLK_RTC_DEB = CLK_RTC / 256

Bit 7 – DMAEN DMA Enable
The RTC can trigger a DMA request when the timestamp is ready in the TIMESTAMP register.

Value	Description
0	Tamper DMA request is disabled. Reading TIMESTAMP has no effect on INTFLAG.TAMPER.
1	Tamper DMA request is enabled. Reading TIMESTAMP will clear INTFLAG.TAMPER.

Bit 6 – RTCOUT RTC Out Enable

Value	Description
0	The RTC active layer output is disabled.
1	The RTC active layer output is enabled.

Bit 5 – DEBASYNC Debouncer Asynchronous Enable

Value	Description
0	The tamper input debouncers operate synchronously.
1	The tamper input debouncers operate asynchronously.

Bit 4 – DEBMAJ Debouncer Majority Enable

Value	Description
0	The tamper input debouncers match three equal values.
1	The tamper input debouncers match majority two of three values.

Bit 0 – GP0EN General Purpose 0 Enable

Value	Description
0	COMP0 compare function enabled. GP0 disabled.
1	COMP0 compare function disabled. GP0 enabled.

21.12.3 Event Control in Clock/Calendar mode (CTRLA.MODE=2)

Name: EVCTRL
Offset: 0x04
Reset: 0x00000000
Property: Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	OVFEO						ALARMEO1	ALARMEO0
Access	R/W						R/W	R/W
Reset	0						0	0

Bit	7	6	5	4	3	2	1	0
	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVFEO Overflow Event Output Enable

Value	Description
0	Overflow event is disabled and will not be generated.
1	Overflow event is enabled and will be generated for every overflow.

Bit 9 – ALARMEO1 Alarm 1 Event Output Enable

Value	Description
0	Alarm 1 event is disabled and will not be generated.
1	Alarm 1 event is enabled and will be generated for every compare match.

Bit 8 – ALARMEO0 Alarm 0 Event Output Enable

Value	Description
0	Alarm 0 event is disabled and will not be generated.
1	Alarm 0 event is enabled and will be generated for every compare match.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PEREOn Periodic Interval n Event Output Enable [n = 7..0]

Value	Description
0	Periodic Interval n event is disabled and will not be generated.
1	Periodic Interval n event is enabled and will be generated.

21.12.4 Interrupt Enable Clear in Clock/Calendar mode (CTRLA.MODE=2)

Name: INTENCLR
Offset: 0x08
Reset: 0x0000

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set (INTENSET) register.

Bit	15	14	13	12	11	10	9	8
	OVF							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVF Overflow Interrupt Enable
Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Overflow Interrupt Enable bit, which disables the Overflow interrupt.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PERn Periodic Interval n Interrupt Enable [n = 7..0]
Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Periodic Interval n Interrupt Enable bit, which disables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.

21.12.5 Interrupt Enable Set in Clock/Calendar mode (CTRLA.MODE=2)

Name: INTENSET
Offset: 0x0A
Reset: 0x0000

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

Bit	15	14	13	12	11	10	9	8
	OVF							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVF Overflow Interrupt Enable
Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Overflow Interrupt Enable bit, which enables the Overflow interrupt.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PERn Periodic Interval n Interrupt Enable [n = 7..0]
Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Periodic Interval n Interrupt Enable bit, which enables the Periodic Interval n interrupt.

Value	Description
0	Periodic Interval n interrupt is disabled.
1	Periodic Interval n interrupt is enabled.

21.12.6 Interrupt Flag Status and Clear in Clock/Calendar mode (CTRLA.MODE=2)

Name: INTFLAG
Offset: 0x0C
Reset: 0x0000
Property: -

Bit	15	14	13	12	11	10	9	8
	OVF							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 15 – OVF Overflow
This flag is cleared by writing a '1' to the flag.
This flag is set on the next CLK_RTC_CNT cycle after an overflow condition occurs, and an interrupt request will be generated if INTENCLR/SET.OVF is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Overflow interrupt flag.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PERn Periodic Interval n [n = 7..0]
This flag is cleared by writing a '1' to the flag.
This flag is set on the 0-to-1 transition of prescaler bit [n+2], and an interrupt request will be generated if INTENCLR/SET.PERx is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Periodic Interval n interrupt flag.

21.12.7 Debug Control

Name: DBGCTRL
Offset: 0x0E
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN Debug Run

This bit is not reset by a software reset.

This bit controls the functionality when the CPU is halted by an external debugger.

Value	Description
0	The RTC is halted when the CPU is halted by an external debugger.
1	The RTC continues normal operation when the CPU is halted by an external debugger.

21.12.8 Synchronization Busy in Clock/Calendar mode (CTRLA.MODE=2)

Name: SYNCBUSY
Offset: 0x10
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CLOCKSYNC							
Access	R							
Reset	0							

Bit	7	6	5	4	3	2	1	0
					CLOCK	FREQCORR	ENABLE	SWRST
Access					R	R	R	R
Reset					0	0	0	0

Bit 15 – CLOCKSYNC Clock Read Sync Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.CLOCKSYNC bit is complete.
1	Write synchronization for CTRLA.CLOCKSYNC bit is ongoing.

Bit 3 – CLOCK Clock Register Synchronization Busy Status

Value	Description
0	Read/write synchronization for CLOCK register is complete.
1	Read/write synchronization for CLOCK register is ongoing.

Bit 2 – FREQCORR Frequency Correction Synchronization Busy Status

Value	Description
0	Write synchronization for FREQCORR register is complete.
1	Write synchronization for FREQCORR register is ongoing.

Bit 1 – ENABLE Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.ENABLE bit is complete.
1	Write synchronization for CTRLA.ENABLE bit is ongoing.

Bit 0 – SWRST Software Reset Synchronization Busy Status

Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	Write synchronization for CTRLA.SWRST bit is complete.
1	Write synchronization for CTRLA.SWRST bit is ongoing.

21.12.9 Frequency Correction

Name: FREQCORR
Offset: 0x14
Reset: 0x00
Property: Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	SIGN	VALUE[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – SIGN Correction Sign

Value	Description
0	The correction value is positive, i.e., frequency will be decreased.
1	The correction value is negative, i.e., frequency will be increased.

Bits 6:0 – VALUE[6:0] Correction Value

These bits define the amount of correction applied to the RTC prescaler.

Value	Description
0	Correction is disabled and the RTC frequency is unchanged.
1 - 127	The RTC frequency is adjusted according to the value.

21.12.10 Clock Value in Clock/Calendar mode (CTRLA.MODE=2)

Name: CLOCK
Offset: 0x18
Reset: 0x00000000
Property: Write-Synchronized, Read-Synchronized

Bit	31	30	29	28	27	26	25	24
	YEAR[5:0]						MONTH[3:2]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MONTH[1:0]		DAY[4:0]					HOUR[4]
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HOUR[3:0]				MINUTE[5:2]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MINUTE[1:0]		SECOND[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:26 – YEAR[5:0] Year
The year offset with respect to the reference year (defined in software).
The year is considered a leap year if YEAR[1:0] is zero.

Bits 25:22 – MONTH[3:0] Month
1 – January
2 – February
...
12 – December

Bits 21:17 – DAY[4:0] Day
Day starts at 1 and ends at 28, 29, 30, or 31, depending on the month and year.

Bits 16:12 – HOUR[4:0] Hour
When CTRLA.CLKREP=0, the Hour bit group is in 24-hour format, with values 0-23. When CTRLA.CLKREP=1, HOUR[3:0] has values 1-12, and HOUR[4] represents AM (0) or PM (1).

Bits 11:6 – MINUTE[5:0] Minute
0 – 59

Bits 5:0 – SECOND[5:0] Second
0 – 59

21.12.11 Alarm n Value in Clock/Calendar mode (CTRLA.MODE=2)

Name: ALARM
Offset: 0x20 + n*0x08 [n=0..1]
Reset: 0x00000000
Property: Write-Synchronized

The 32-bit value of ALARMn is continuously compared with the 32-bit CLOCK value, based on the masking set by MASKn.SEL. When a match occurs, the Alarm n interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.ALARMn) is set on the next counter cycle, and the counter is cleared if CTRLA.MATCHCLR is '1'.

Bit	31	30	29	28	27	26	25	24
	YEAR[5:0]						MONTH[3:2]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MONTH[1:0]		DAY[4:0]					HOUR[4]
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HOUR[3:0]				MINUTE[5:2]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MINUTE[1:0]		SECOND[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:26 – YEAR[5:0] Year
The alarm year. Years are only matched if MASKn.SEL is 6

Bits 25:22 – MONTH[3:0] Month
The alarm month. Months are matched only if MASKn.SEL is greater than 4.

Bits 21:17 – DAY[4:0] Day
The alarm day. Days are matched only if MASKn.SEL is greater than 3.

Bits 16:12 – HOUR[4:0] Hour
The alarm hour. Hours are matched only if MASKn.SEL is greater than 2.

Bits 11:6 – MINUTE[5:0] Minute
The alarm minute. Minutes are matched only if MASKn.SEL is greater than 1.

Bits 5:0 – SECOND[5:0] Second
The alarm second. Seconds are matched only if MASKn.SEL is greater than 0.

21.12.12 Alarm n Mask in Clock/Calendar mode (CTRLA.MODE=2)

Name: MASK
Offset: 0x24 + n*0x08 [n=0..1]
Reset: 0x00
Property: Write-Synchronized

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

Bits 2:0 – SEL[2:0] Alarm Mask Selection
These bits define which bit groups of Alarm n are valid.

Value	Name	Description
0x0	OFF	Alarm Disabled
0x1	SS	Match seconds only
0x2	MMSS	Match seconds and minutes only
0x3	HHMMSS	Match seconds, minutes, and hours only
0x4	DDHHMMSS	Match seconds, minutes, hours, and days only
0x5	MMDDHHMMSS	Match seconds, minutes, hours, days, and months only
0x6	YYMMDDHHMMSS	Match seconds, minutes, hours, days, months, and years
0x7	-	Reserved

21.12.13 General Purpose n

Name: GPn
Offset: 0x40 + n*0x04 [n=0..3]
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	GP[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	GP[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	GP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – GP[31:0] General Purpose
These bits are for user-defined general purpose use, see *General Purpose Registers* from Related Links.
Related Links
21.6.8.4. General Purpose Registers

21.12.14 Tamper Control

Name: TAMPCTRL
Offset: 0x60
Reset: 0x00000000
Property: Enable-Protected

Bit	31	30	29	28	27	26	25	24
					DEBNC3	DEBNC2	DEBNC1	DEBNC0
Access								
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
					TAMLVL3	TAMLVL2	TAMLVL1	TAMLVL0
Access								
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				IN3ACT[1:0]		IN2ACT[1:0]		IN0ACT[1:0]
Access								
Reset				0	0	0	0	0

Bits 24, 25, 26, 27 – DEBNCn Debounce Enable of Tamper Input INn [n=0..3]
Note: Debounce feature does not apply to the Active Layer Protection mode (TAMPCTRL.INACT = ACTL).

Value	Description
0	Debouncing is disabled for Tamper input INn
1	Debouncing is enabled for Tamper input INn

Bits 16, 17, 18, 19 – TAMLVLn Tamper Level Select of Tamper Input INn [n=0..3]
Note: Tamper Level feature does not apply to the Active Layer Protection mode (TAMPCTRL.INACT = ACTL).

Value	Description
0	A falling edge condition will be detected on Tamper input INn.
1	A rising edge condition will be detected on Tamper input INn.

Bits 0:1, 1:2, 2:3, 3:4 – INnACT Tamper Channel n Action [n=0..3]
These bits determine the action taken by Tamper Channel n.

Value	Name	Description
0x0	OFF	Off (Disabled)
0x1	WAKE	Wake and set Tamper flag
0x2	CAPTURE	Capture timestamp and set Tamper flag
0x3	ACTL	Compare RTC signal routed between INn and OUT pins . When a mismatch occurs, capture timestamp and set Tamper flag

21.12.15 Tamper ID

Name: TAMPID
Offset: 0x68
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
	TAMPEVT							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					TAMPID3	TAMPID2	TAMPID1	TAMPID0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 31 – TAMPEVT Tamper Event Detected

Writing a '0' to this bit has no effect. Writing a '1' to this bit clears the tamper detection bit.

Value	Description
0	A tamper input event has not been detected
1	A tamper input event has been detected

Bits 0, 1, 2, 3 – TAMPIDn Tamper on Channel n Detected [n=0..3]

Writing a '0' to this bit has no effect. Writing a '1' to this bit clears the tamper detection bit.

Value	Description
0	A tamper condition has not been detected on Channel n
1	A tamper condition has been detected on Channel n

21.12.16 Backup0

Name: BKUP0
Offset: 0x80
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
	BKUP[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BKUP[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BKUP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – BKUP[31:0] Backup
These bits are user-defined for general purpose use in the Backup domain.

22. Direct Memory Access Controller (DMAC)

22.1 Overview

The Direct Memory Access Controller (DMAC) contains both a Direct Memory Access engine and a Cyclic Redundancy Check (CRC) engine. The DMAC can transfer data between memories and peripherals, and thus off-load these tasks from the CPU. It enables high data transfer rates with minimum CPU intervention, and frees up CPU time. With access to all peripherals, the DMAC can handle automatic transfer of data between communication modules.

The DMA part of the DMAC has several DMA channels which all can receive different types of transfer triggers to generate transfer requests from the DMA channels to the arbiter (see *DMAC Block Diagram* in the *Block Diagram* from Related Links). The arbiter will grant one DMA channel at a time to act as the active channel. When an active channel has been granted, the fetch engine of the DMAC will fetch a transfer descriptor from the SRAM and store it in the internal memory of the active channel, which will execute the data transmission.

An ongoing data transfer of an active channel can be interrupted by a higher prioritized DMA channel. The DMAC will write back the updated transfer descriptor from the internal memory of the active channel to SRAM, and grant the higher prioritized channel to start transfer as the new active channel. Once a DMA channel is done with its transfer, interrupts and events can be generated optionally.

The DMAC has four bus interfaces:

- The *data transfer bus* is used for performing the actual DMA transfer.
- The *AHB/APB Bridge bus* is used when writing and reading the I/O registers of the DMAC.
- The *descriptor fetch bus* is used by the fetch engine to fetch transfer descriptors before data transfer can be started or continued.
- The *write-back bus* is used to write the transfer descriptor back to SRAM.

All buses are AHB Manager interfaces except for the AHB/APB Bridge bus, which is an APB Subordinate interface.

Burst transfer options, buffered active channel to pre-fetch descriptors and advance quality of service features ensure low-latency transfers for high-speed peripherals or high-speed operations.

The CRC engine can be used by software to detect an accidental error in the transferred data and to take corrective action, such as requesting the data to be sent again or simply not using the incorrect data.

Note: Traditional Direct Memory Access Controller (DMAC) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client” respectively.

Related Links

22.3. Block Diagram

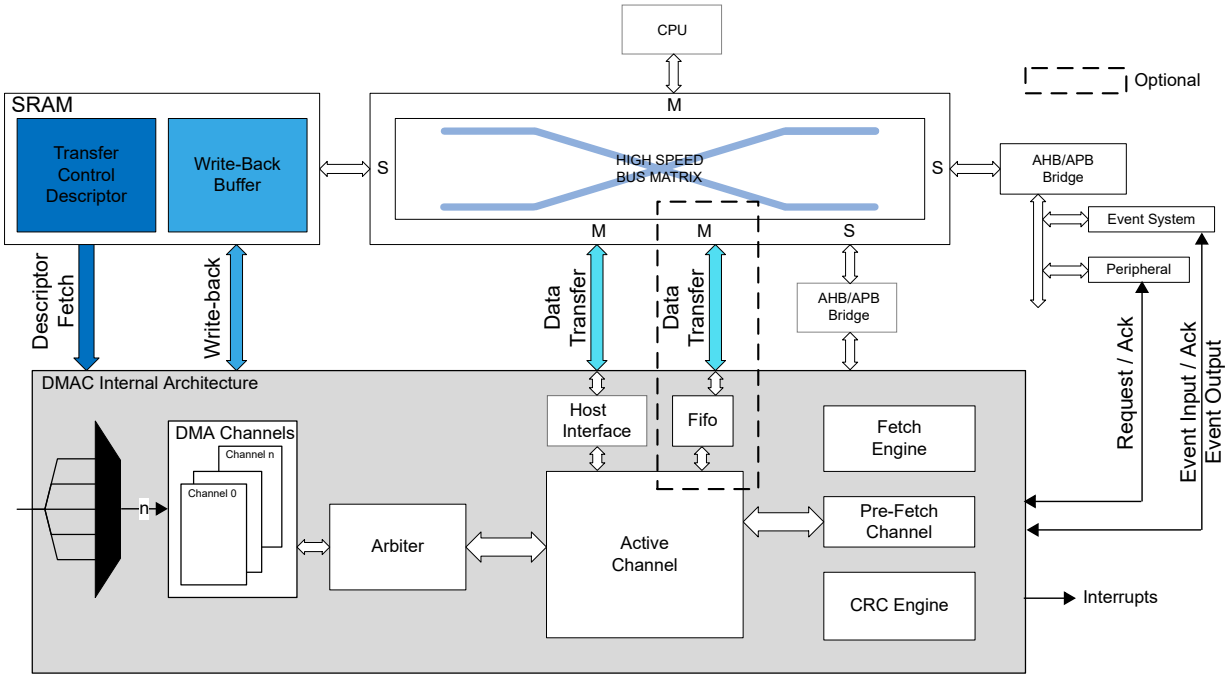
22.2 Features

- Data transfer from:
  - Peripheral to peripheral
  - Peripheral to memory
  - Memory to peripheral
  - Memory to memory
- Transfer trigger sources
  - Software
  - Events from Event System
  - Dedicated requests from peripherals
- SRAM based transfer descriptors
  - Single transfer using one descriptor
  - Multi-buffer or circular buffer modes by linking multiple descriptors

- Up to 16 channels
  - Enable 16 independent transfers
  - Automatic descriptor fetch for each channel
  - Suspend/resume operation support for each channel
- Flexible arbitration scheme
  - 4 configurable priority levels for each channel
  - Fixed or round-robin priority scheme within each priority level
- From 1 to 256KB data transfer in a single block transfer
- Multiple addressing modes
  - Static
  - Configurable increment scheme
- Optional interrupt generation
  - On block transfer complete
  - On error detection
  - On channel suspend
- 8 event inputs
  - One event input for each of the 8 least significant DMA channels
  - Can be selected to trigger normal transfers, periodic transfers or conditional transfers
  - Can be selected to suspend or resume channel operation
- 4 event outputs
  - One output event for each of the 4 least significant DMA channels
  - Selectable generation on AHB, block, or transaction transfer complete
- Error management supported by write-back function
  - Dedicated Write-Back memory section for each channel to store ongoing descriptor transfer
- CRC polynomial software selectable to
  - CRC-16 (CRC-CCITT)
  - CRC-32 (IEEE® 802.3)

22.3 Block Diagram

Figure 22-1. DMAC Block Diagram



22.4 Signal Description

Not applicable.

22.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

22.5.1 I/O Lines

Not applicable.

22.5.2 Power Management

The DMAC will continue to operate in any Sleep mode where the selected source clock is running. The DMAC's interrupts can be used to wake up the device from Sleep modes. Events connected to the event system can trigger other operations in the system without exiting Sleep modes. On hardware or software Reset, all registers are set to their Reset value.

22.5.3 DMA

Not applicable.

22.5.4 Interrupts

The interrupt request line is connected to the interrupt controller. Using the DMAC interrupt requires the interrupt controller to be configured first. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

22.5.5 Events

The events are connected to the event system.

22.5.6 Debug Operation

When the CPU is halted in Debug mode the DMAC will halt normal operation. The DMAC can be forced to continue operation during debugging. See *DBGCTRL* from Related Links.

Related Links

22.8.6. DBGCTRL

22.5.7 Register Access Protection

All registers with write access can be write-protected optionally by the Peripheral Access Controller (PAC), except for the following registers:

- Interrupt Pending register (INTPEND)
- Channel ID register (CHID)
- Channel Interrupt Flag Status and Clear register (CHINTFLAG)

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

PAC write protection does not apply to accesses through an external debugger.

22.5.8 Analog Connections

Not applicable.

22.6 Functional Description

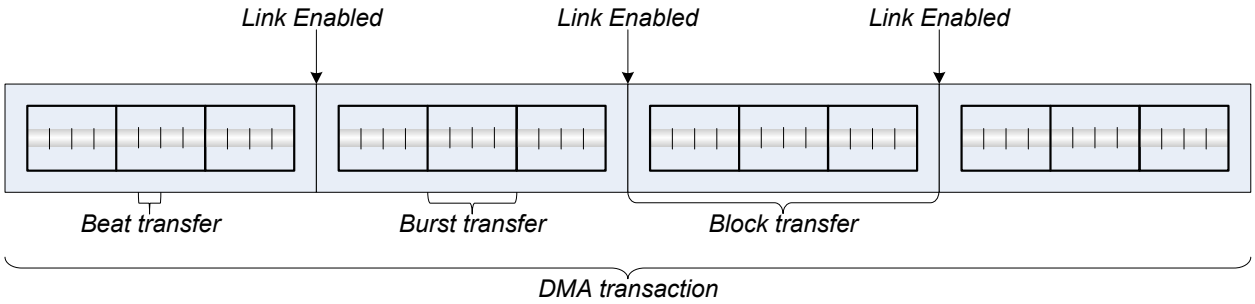
22.6.1 Principle of Operation

The DMAC consists of a DMA module and a CRC module.

22.6.1.1 DMA

The DMAC can transfer data between memories and peripherals without interaction from the CPU. The data transferred by the DMAC are called transactions, and these transactions can be split into smaller data transfers. The following figure shows the relationship between the different transfer sizes:

Figure 22-2. DMA Transfer Sizes



- Beat transfer: The size of one data transfer bus access, and the size is selected by writing the Beat Size bit group in the Block Transfer Control register (BTCTRL.BEATSIZE)
- Block transfer: The amount of data one transfer descriptor can transfer, and the amount can range from 1 to 64k beats. A block transfer can be interrupted.
- Transaction: The DMAC can link several transfer descriptors by having the first descriptor pointing to the second and so forth, as shown in the figure above. A DMA transaction is the complete transfer of all blocks within a linked list.

A transfer descriptor describes how a block transfer must be carried out by the DMAC, and it must remain in SRAM (see *Transfer Descriptors* from Related Links).

The figure above shows several block transfers linked together, which are called linked descriptors (see *Linked Descriptors* from Related Links).

A DMA transfer is initiated by an incoming transfer trigger on one of the DMA channels. This trigger can be configured to be either a software trigger, an event trigger, or one of the dedicated peripheral triggers. The transfer trigger will result in a DMA transfer request from the specific channel to the arbiter. If there are several DMA channels with pending transfer requests, the arbiter chooses which channel is granted access to become the active channel. The DMA channel granted access as the active channel will carry out the transaction as configured in the transfer descriptor. A current transaction can be interrupted by a higher prioritized channel, but will resume the block transfer when the according DMA channel is granted access as the active channel again.

For each beat transfer, an optional output event can be generated. For each block transfer, optional interrupts and an optional output event can be generated. When a transaction is completed, depending on the configuration, the DMA channel will either be suspended or disabled.

Related Links

22.6.2.3. Transfer Descriptors
22.6.3.1. Linked Descriptors

22.6.1.2 CRC

The internal CRC engine supports two commonly used CRC polynomials: CRC-16 (CRC-CCITT) and CRC-32 (IEEE 802.3). It can be used on a selectable DMA channel, or on the I/O interface. See *CRC Operation* from Related Links.

Related Links

22.6.3.8. CRC Operation

22.6.2 Basic Operation

22.6.2.1 Initialization

DMAC Initialization

Before DMAC is enabled, it must be configured as defined below:

- The SRAM address of where the descriptor memory section is located must be written to the Description Base Address (BASEADDR) register.
- The SRAM address of where the write-back section must be located must be written to the Write-Back Memory Base Address (WRBADDR) register.
- Priority level x of the arbiter can be enabled by setting the Priority Level x Enable bit in the Control register (CTRL.LVLENx=1)

DMA Channel Initialization

Before a DMA channel is enabled, the DMA channel and the corresponding first transfer descriptor must be configured, as defined below:

- DMA Channel Configuration:
  - The channel number of the DMA channel to configure must be written to the Channel Control A (CHCTRLA) register.
  - Trigger action must be selected by writing the Trigger Action bit field in the Channel Control A (CHCTRLA.TRIGACT) register.
  - Trigger source must be selected by writing the Trigger Source bit field in the Channel Control A (CHCTRLA.TRIGSRC) register.
- Transfer Descriptor
  - The size of each access of the data transfer bus must be selected by writing the Beat Size bit group in the Block Transfer Control (BTCTRL.BEATSIZE) register.
  - The transfer descriptor must be made valid by writing a one to the Valid bit in the Block Transfer Control (BTCTRL.VALID) register.
  - Number of beats in the block transfer must be selected by writing the Block Transfer Count (BTCNT) register.

– Source address for the block transfer must be selected by writing the Block Transfer Source Address (SRCADDR) register.
– Destination address for the block transfer must be selected by writing the Block Transfer Destination Address (DSTADDR) register.

CRC Calculation
If CRC calculation is needed, the CRC engine must be configured before it is enabled, as described below:

- The CRC input source must selected by writing the CRC Input Source bit group in the CRC Control (CRCCTRL.CRCSRC) register.
- The type of CRC calculation must be selected by writing the CRC Polynomial Type bit group in the CRC Control (CRCCTRL.CRCPOLY) register.
- If I/O is selected as input source, the beat size must be selected by writing the CRC Beat Size bit group in the CRC Control (CRCCTRL.CRCBEATSIZE) register.

Register Properties
The following DMAC registers are enable-protected, that is, they can only be written when the DMAC is disabled (CTRL.DMAENABLE=0):

- The Descriptor Base Memory Address (BASEADDR) register
- The Write-Back Memory Base Address (WRBADDR) register

The following DMAC bit is enable-protected, that is, it can only be written when the DMAC and CRC are disabled (CTRL.DMAENABLE=0 and CRCCTRL.CRCSRC=0):

- The Software Reset bit in the Control (CTRL.SWRST) register

The following DMA channel bit is enable-protected, meaning that it can only be written when the corresponding DMA channel is disabled:

- The Channel Software Reset bit in the Channel Control A (CHCTRLA.SWRST) register

The following CRC registers are enable-protected, that is, they can only be written when the CRC is disabled (CRCCTRL.CRCSRC=0):

- The CRC Control (CRCCTRL) register
- CRC Checksum (CRCCHKSUM) register

Enable-protection is denoted by the ‘Enable-Protected’ property in the register description.

22.6.2.2 Enabling, Disabling, and Resetting

The DMAC is enabled by writing the DMA Enable bit in the Control register (CTRL.DMAENABLE) to '1'. The DMAC is disabled by writing a '0' to the CTRL.DMAENABLE bit.

A DMA channel is enabled by writing the Enable bit in the Channel Control A register (CHCTRLA.ENABLE) to '1', after the corresponding channel ID to the channel is configured. A DMA channel is disabled by writing a '0' to CHCTRLAn.ENABLE.

The CRC is enabled by writing a ‘1’ to the CRC Enable bit in the Control register (CTRL.CRCENABLE). The CRC is disabled by writing a '0' to CTRL.CRCENABLE.

The DMAC is reset by writing a '1' to the Software Reset bit in the Control register (CTRL.SWRST) while the DMAC and CRC are disabled. All registers in the DMAC except DBGCTRL will be reset to their initial state.

A DMA channel is reset by writing a '1' to the Software Reset bit in the Channel Control A register (CHCTRLAn.SWRST), after writing the corresponding channel ID to the Channel ID bit group in the Channel ID register (CHID.ID). The channel registers will be reset to their initial state. The corresponding DMA channel must be disabled in order for the Reset to take effect.

22.6.2.3 Transfer Descriptors

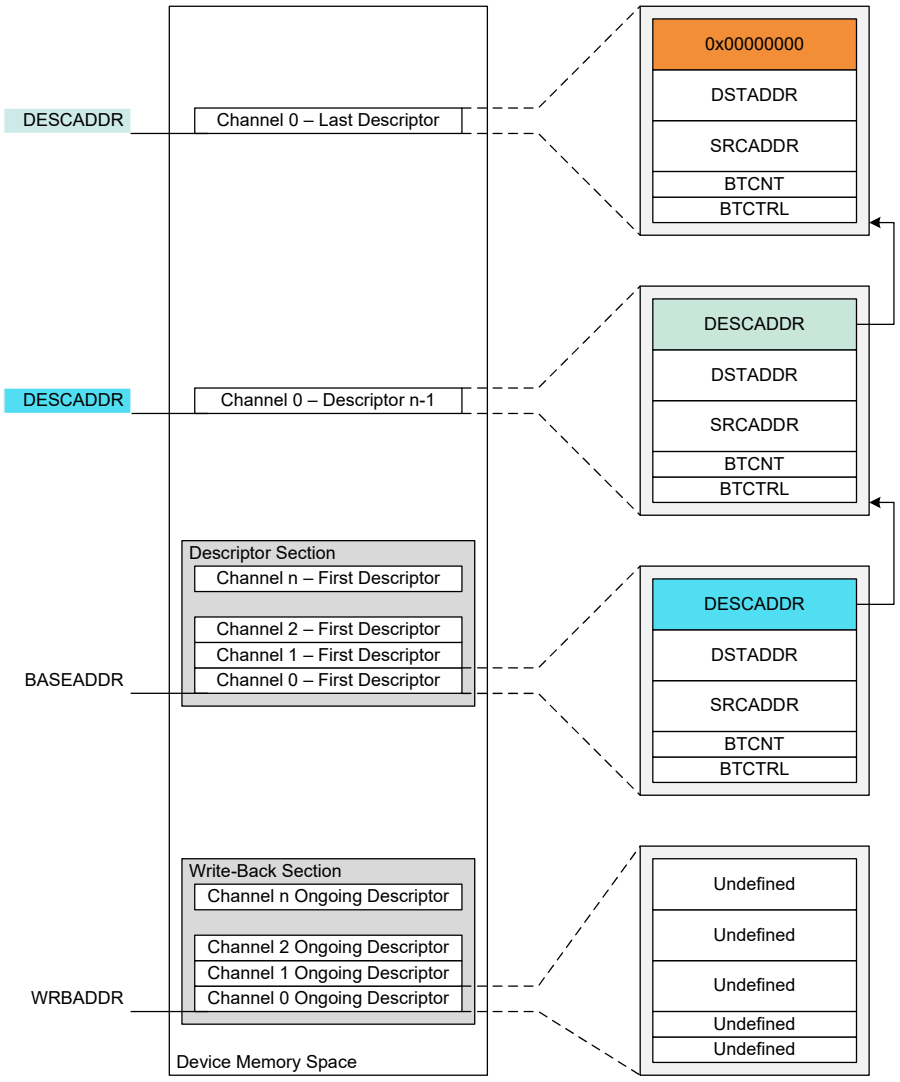
The transfer descriptors, together with the channel configurations, decide how a block transfer must be executed. Before a DMA channel is enabled (CHCTRLA.ENABLE is written to one) and receives a transfer trigger, its first transfer descriptor must be initialized and valid (BTCTRL.VALID). The first transfer descriptor describes the first block transfer of a transaction.

All transfer descriptors must reside in SRAM. The addresses stored in the Descriptor Memory Section Base Address (BASEADDR) and Write-Back Memory Section Base Address (WRBADDR) registers tell the DMAC where to find the descriptor memory section and the write-back memory section.

The descriptor memory section is where the DMAC expects to find the first transfer descriptors for all DMA channels. As BASEADDR points only to the first transfer descriptor of channel '0' (see the following figure). All first transfer descriptors must be stored in a contiguous memory section, where the transfer descriptors must be ordered according to their channel number (see *Linked Descriptors* from Related Links).

The write-back memory section is where the DMAC stores the transfer descriptors for the ongoing block transfers. WRBADDR points to the ongoing transfer descriptor of channel '0'. All ongoing transfer descriptors are stored in a contiguous memory section where the transfer descriptors are ordered according to their channel number. The figure below shows an example of linked descriptors on DMA channel '0' (see *Linked Descriptors* from Related Links).

Figure 22-3. Memory Sections



The size of the descriptor and write-back memory sections are dependent on the number of the most significant enabled DMA channel *m*, as shown below:

$Size = 128\text{bits} \cdot (m + 1)$

For memory optimization, it is recommended to use the less significant DMA channels, if not all channels are required.

The descriptor and write-back memory sections can either be two separate memory sections, or they can share a memory section (BASEADDR=WRBADDR). The benefit of having them in two separate sections, is that the same transaction for a channel can be repeated without having to modify the first transfer descriptor.

Related Links

22.6.3.1. Linked Descriptors

22.6.2.4 Arbitration

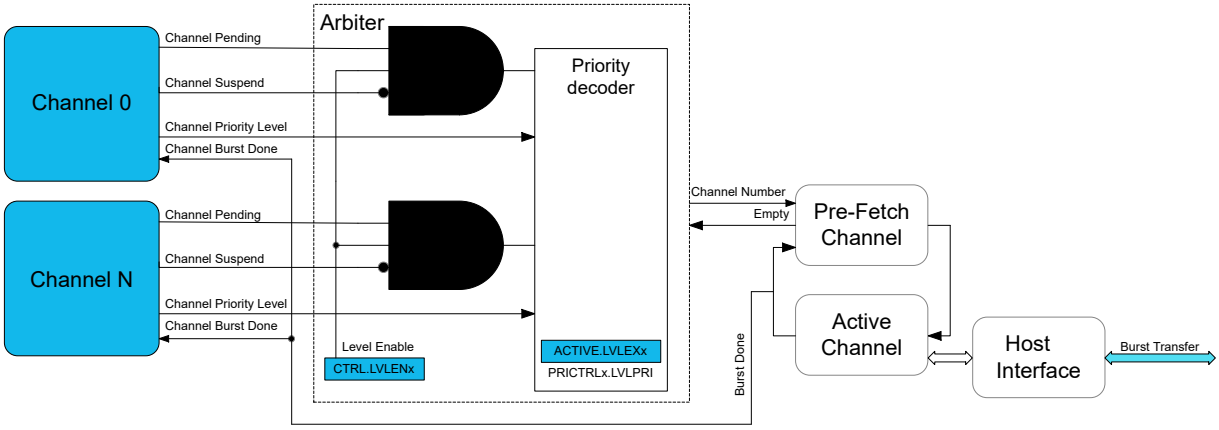
If a DMA channel is enabled and not suspended when it receives a transfer trigger, it will send a transfer request to the arbiter. When the arbiter receives the transfer request it will include the DMA channel in the queue of channels having pending transfers, and the corresponding Pending Channel x bit in the Pending Channels registers (PENDCH.PENDCHx) will be set. Depending on the arbitration scheme, the arbiter will choose which DMA channel will be the next active channel. The next transfer descriptor will be fetched from SRAM memory and stored internally in the Pre-Fetch Channel. The active channel is the DMA channel being granted access to perform its next burst transfer. When the Active Channel has completed a burst transfer, the descriptor stored in the Pre-Fetch Channel is transferred to the Active Channel and a new burst will take place.

When the descriptor stored in the Pre-Fetch Channel is transferred to the Active Channel, the corresponding PENDCH.PENDCHx will be cleared. In the same way, depending on trigger action settings and if the upcoming burst transfer is the first for the transfer request or not, the corresponding Busy Channel x bit in the Busy Channels register (BUSYCH.BUSYCHx), will either be set or remain '1'. When the channel has performed its granted burst transfer(s) it will be either fed into the queue of channels with pending transfers, set to be waiting for a new transfer trigger, suspended, or disabled. This depends on the channel and block transfer configuration. If the DMA channel is set to wait for a new transfer trigger, suspended or disabled, the corresponding BUSYCH.BUSYCHx will be cleared.

If a DMA channel is suspended while it has a pending transfer, it will be removed from the queue of pending channels, but the corresponding PENDCH.PENDCHx will remain set. The status will also be indicated in CHINTFLAGn.SUSP. When the same DMA channel is resumed, it will be added to the queue of pending channels again.

If a DMA channel gets disabled (CHCTRLA.ENABLE=0) while it has a pending transfer, it will be removed from the queue of pending channels, and the corresponding PENDCH.PENDCHx will be cleared.

Figure 22-4. Arbiter Overview



Priority Levels

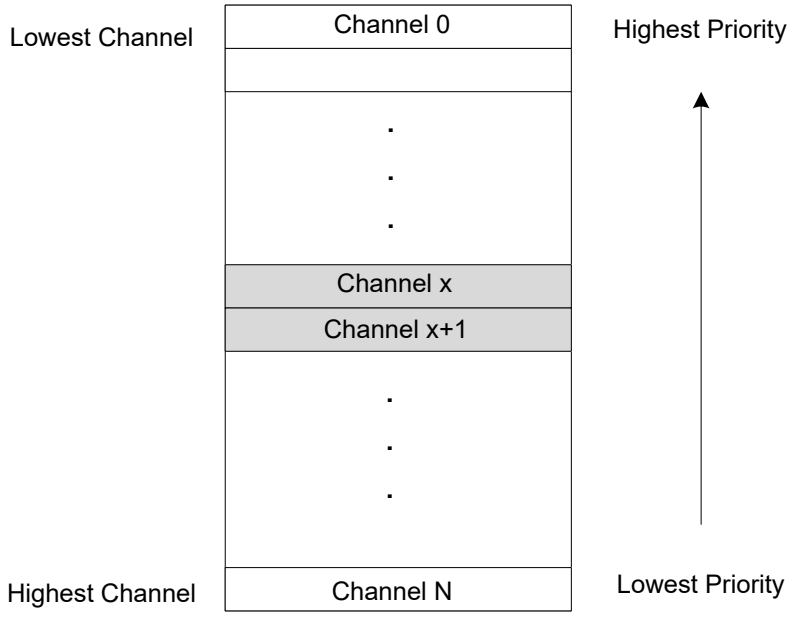
When a channel level is pending or the channel is transferring data, the corresponding Level Executing bit is set in the Active Channel and Levels register (ACTIVE.LVLEXx).

Each DMA channel supports up to 4-level priority scheme.

The priority level for a channel is configured by writing to the Channel Arbitration Level bit group in the Channel Priority Level register (CHPRILVL.PRILVL). As long as all priority levels are enabled, a channel with a higher priority level number will have priority over a channel with a lower priority level number. A priority level is enabled by writing the Priority Level x Enable bit in the Control register (CTRL.LVLENx) to '1', for the corresponding level.

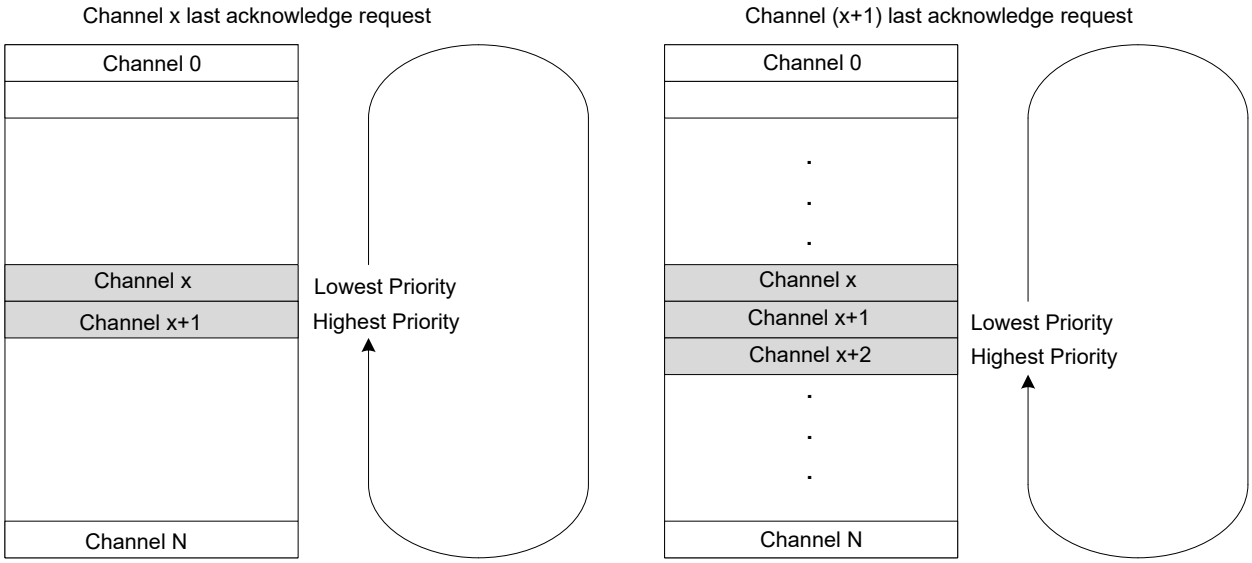
Within each priority level, the DMAC's arbiter can be configured to prioritize statically or dynamically. For the arbiter to perform static arbitration within a priority level, the Level X Round-Robin Scheduling Enable bit in the Priority Control x register (PRICTRL0.RRLVLENx) has to be written to '0'. When static arbitration is enabled (PRICTRL0.RRLVLENx is '0'), the arbiter will prioritize a low channel number over a high channel number as shown in the following figure. When using the static scheme, there is a risk of high channel numbers never being granted access as the active channel. This can be avoided using a dynamic arbitration scheme.

Figure 22-5. Static Priority Scheduling



The dynamic arbitration scheme in the DMAC is round-robin. Round-robin arbitration is enabled by writing PRICTRL0.RRLVLEN to '1', for a given priority level x. With the round-robin scheme, the channel number of the last channel being granted access will have the lowest priority the next time the arbiter has to grant access to a channel within the same priority level, as shown in the following figure. The channel number of the last channel being granted access as the active channel is stored in the Level x Channel Priority Number bit group in the Priority Control 0 register (PRICTRL0.LVLPRIx) for the corresponding priority level.

Figure 22-6. Dynamic (Round-Robin) Priority Scheduling



22.6.2.5 Data Transmission

Before the DMAC can perform a data transmission, a DMA channel has to be configured and enabled, its corresponding transfer descriptor has to be initialized, and the arbiter has to grant the DMA channel access as the active channel.

Once the arbiter has granted a DMA channel access as the active channel (see *DMAC Block Diagram* in the *Block Diagram* from Related Links) the transfer descriptor for the DMA channel will be fetched from SRAM using the fetch bus, and stored in the internal memory for the active channel. For a new block transfer, the transfer descriptor will be fetched from the descriptor memory section (BASEADDR); For an ongoing block transfer, the descriptor will be fetched from the write-back memory section (WRBADDR). By using the data transfer bus, the DMAC will read the data from the current source address and write it to the current destination address. For further details on how the current source and destination addresses are calculated (see *Addressing* from the Related Links).

The arbitration procedure is performed after each transfer. If the current DMA channel is granted access again, the block transfer counter (BTCNT) of the internal transfer descriptor will be decremented by the number of beats in a transfer, the optional output event Beat will be generated if configured and enabled, and the active channel will perform a new transfer. If a different DMA channel than the current active channel is granted access, the block transfer counter value will be written to the write-back section before the transfer descriptor of the newly granted DMA channel is fetched into the internal memory of the active channel.

When a block transfer has come to its end (BTCNT is zero), the Valid bit in the Block Transfer Control register will be cleared (BTCTRL.VALID=0) before the entire transfer descriptor is written to the write-back memory. The optional interrupts, Channel Transfer Complete and Channel Suspend, and the optional output event Block, will be generated if configured and enabled. After the last block transfer in a transaction, the Next Descriptor Address register (DESCADDR) will hold the value 0x00000000, and the DMA channel will either be suspended or disabled, depending on the configuration in the Block Action bit group in the Block Transfer Control register (BTCTRL.BLOCKACT). If the transaction has further block transfers pending, DESCADDR will hold the SRAM address to the next transfer descriptor to be fetched. The DMAC will fetch the next descriptor into the internal memory of the active channel and write its content to the write-back section for the channel, before the arbiter gets to choose the next active channel.

Related Links

22.6.2.7. Addressing
22.3. Block Diagram

22.6.2.6 Transfer Triggers and Actions

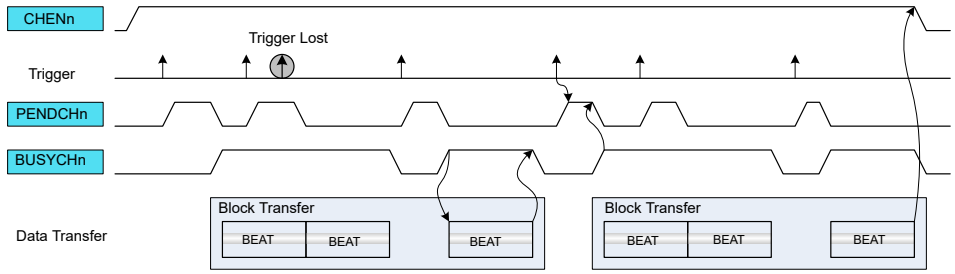
A DMA transfer through a DMA channel can be started only when a DMA transfer request is detected, and the DMA channel has been granted access to the DMA. A transfer request can be triggered from software, from a peripheral, or from an event. There are dedicated Trigger Source selections for each DMA Channel n Control A (CHCTRLAn.TRIGSRC).

The trigger actions are available in the Trigger Action bit group in the Channel n Control A register (CHCTRLAn.TRIGACT). By default, a trigger generates a request for a block transfer operation. If a single descriptor is defined for a channel, the channel is automatically disabled when a block transfer has been completed. If a list of linked descriptors is defined for a channel, the channel is automatically disabled when the last descriptor in the list is executed. As long as the list still has descriptors to execute, the channel will be waiting for the next block transfer trigger. When enabled again, the channel will wait for the next block transfer trigger. The trigger actions can also be configured to generate a request for a burst transfer (CHCTRLAn.TRIGACT=0x2) or transaction transfer (CHCTRLAn.TRIGACT=0x3) instead of a block transfer (CHCTRLAn.TRIGACT=0x0).

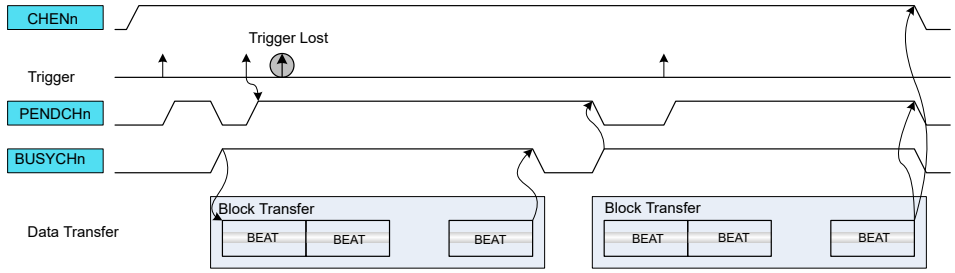
The following figure shows an example where triggers are used with two linked block descriptors.

Figure 22-7. Trigger Action and Transfers

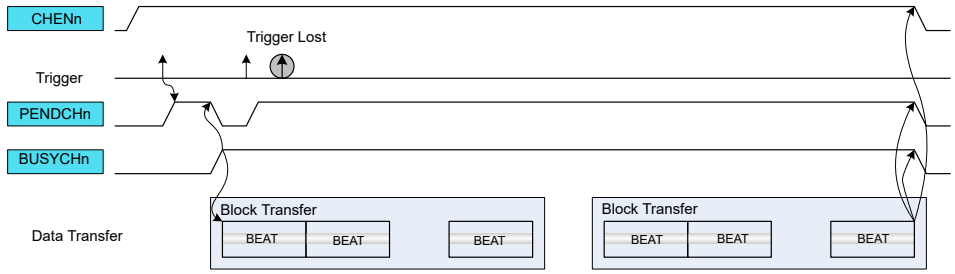
Beat Trigger Action



Block Trigger Action



Transaction Trigger Action



If the trigger source generates a transfer request for a channel during an ongoing transfer, the new transfer request will be kept pending (CHSTATUSn.PEND=1), and the new transfer can start after the ongoing one is done. Only one pending transfer can be kept per channel. If the trigger source generates more transfer requests while one is already pending, the additional ones will be lost. All channels pending status flags are also available in the Pending Channels register (PENDCH).

When the transfer starts, the corresponding Channel Busy status flag is set in Channel n Status register (CHSTATUSn.BUSY). When the trigger action is complete, the Channel Busy status flag is cleared. All channel busy status flags are also available in the Busy Channels register (BUSYCH) in DMAC.

22.6.2.7 Addressing

Each block transfer needs to have both a source address and a destination address defined. The source address is set by writing the Transfer Source Address (SRCADDR) register, the destination address is set by writing the Transfer Destination Address (SRCADDR) register.

The addressing of this DMAC module can be static or incremental, for either source or destination of a block transfer, or both.

Incrementation for the source address of a block transfer is enabled by writing the Source Address Incrementation Enable bit in the Block Transfer Control register (BTCTRL.SRCINC=1). The step size of the incrementation is configurable and can be chosen by writing the Step Selection bit in the Block Transfer Control register (BTCTRL.STEPSEL=1) and writing the desired step size in the Address Increment Step Size bit group in the Block Transfer Control register (BTCTRL.STEPSIZE). If BTCTRL.STEPSEL=0, the step size for the source incrementation will be the size of one beat.

When source address incrementation is configured (BTCTRL.SRCINC=1), SRCADDR is calculated as follows:

If BTCTRL.STEPSEL=1:

$$\mathrm{SRCADDR} = \mathrm{SRCADDR}_{START} + BTCNT \cdot \left(BEATSIZE + 1\right) \cdot 2^{\mathrm{STEPSIZE}}$$

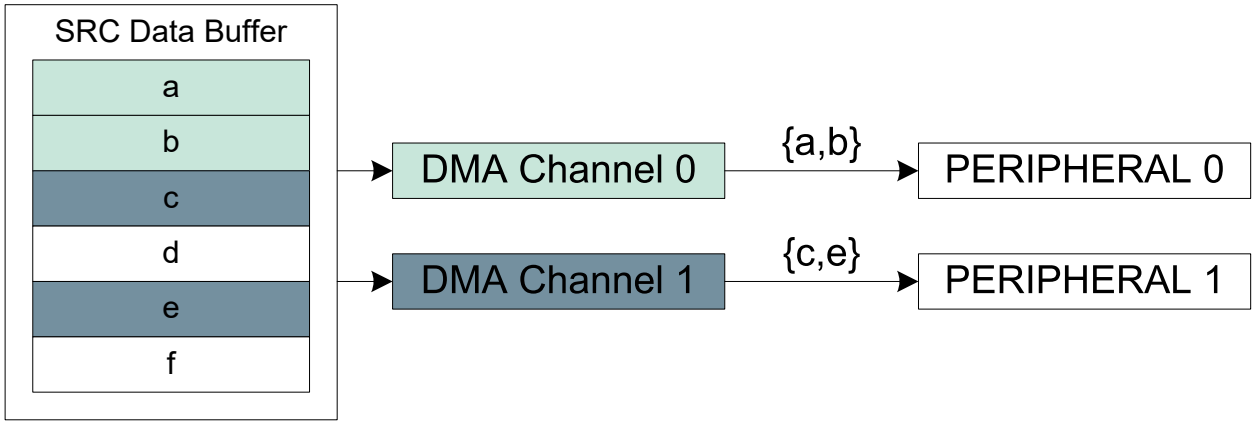
If BTCTRL.STEPSEL=0:

$$\mathrm{SRCADDR} = \mathrm{SRCADDR}_{START} + BTCNT \cdot (BEATSIZE + 1)$$

- $\mathrm{SRCADDR}_{START}$ is the source address of the first beat transfer in the block transfer
- BTCNT is the initial number of beats remaining in the block transfer
- BEATSIZE is the configured number of bytes in a beat
- STEPSIZE is the configured number of beats for each incrementation

The following figure shows an example where DMA channel 0 is configured to increment the source address by one beat after each beat transfer (BTCTRL.SRCINC=1), and DMA channel 1 is configured to increment the source address by two beats (BTCTRL.SRCINC=1, BTCTRL.STEPSEL=1, and BTCTRL.STEPSIZE=0x1). As the destination address for both channels are peripherals, destination incrementation is disabled (BTCTRL.DSTINC=0).

Figure 22-8. Source Address Increment



Incrementation for the destination address of a block transfer is enabled by setting the Destination Address Incrementation Enable bit in the Block Transfer Control register (BTCTRL.DSTINC=1). The step size of the incrementation is configurable by clearing BTCTRL.STEPSEL=0 and writing BTCTRL.STEPSIZE to the desired step size. If BTCTRL.STEPSEL=1, the step size for the destination incrementation will be the size of one beat.

When the destination address incrementation is configured (BTCTRL.DSTINC=1), DSTADDR must be set and calculated as follows:

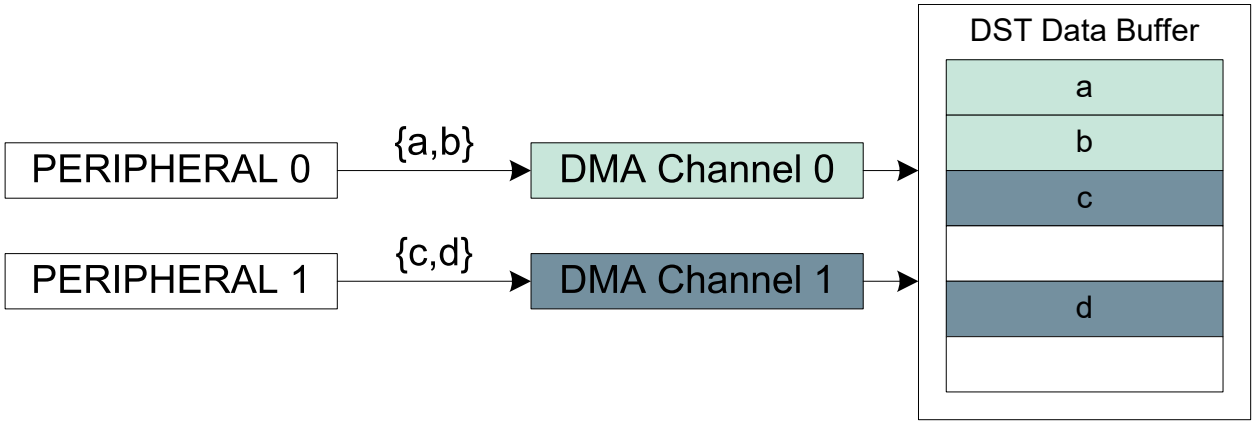
$DSTADDR = DSTADDR_{START} + BTCNT \bullet \left(BEATSIZE + 1\right) \bullet 2^{STEPSIZE}$	where BTCTRL.STEPSEL is zero

$DSTADDR = DSTADDR_{START} + BTCNT \bullet (BEATSIZE + 1)$	where BTCTRL.STEPSEL is one

- $DSTADDR_{START}$ is the destination address of the first beat transfer in the block transfer
- BTCNT is the initial number of beats remaining in the block transfer
- BEATSIZE is the configured number of bytes in a beat
- STEPSIZE is the configured number of beats for each incrementation

The following figure shows an example where DMA channel 0 is configured to increment destination address by one beat (BTCTRL.DSTINC=1) and DMA channel 1 is configured to increment destination address by two beats (BTCTRL.DSTINC=1, BTCTRL.STEPSEL=0, and BTCTRL.STEPSIZE=0x1). As the source address for both channels are peripherals, source incrementation is disabled (BTCTRL.SRCINC=0).

Figure 22-9. Destination Address Increment



22.6.2.8 Internal FIFO

To improve the bandwidth, the DMAC can support FIFO operation. When single-beat burst configuration is selected (CHCTRALx.BURSTLEN = SINGLE), the channel waits until the FIFO can transmit or accept a single beat transfer before it requests a bus access to write to the destination address. In all other cases, the channel waits until the FIFO threshold is reached before it requests a bus access to write to the destination address. The threshold is configurable and can be set by writing the THRESHOLD bits in the Channel x Control A register.

If the DMAC completes the read operations before the threshold is reached, the write to the destination is automatically enabled. If the FIFO is empty and the read from source is ongoing, the DMA will wait again until the FIFO threshold is reached before it requests a bus access to write the destination.

22.6.2.9 Error Handling

If a bus error is received from an AHB subordinate during a DMA data transfer, the corresponding active channel is disabled and the corresponding Channel Transfer Error Interrupt flag in the Channel Interrupt Status and Clear register (CHINTFLAG.TERR) is set. If enabled, the optional transfer error interrupt is generated. The transfer counter will not be decremented and its current value is written-back in the write-back memory section before the channel is disabled.

When the DMAC fetches an invalid descriptor (BTCTRL.VALID=0) or when the channel is resumed and the DMA fetches the next descriptor with null address (DESCADDR=0x00000000), the corresponding channel operation is suspended, the Channel Suspend Interrupt Flag in the Channel Interrupt Flag Status and Clear register (CHINTFLAG.SUSP) is set, and the Channel Fetch Error bit in the Channel Status register (CHSTATUS.FERR) is set. If enabled, the optional suspend interrupt is generated.

22.6.3 Additional Features

22.6.3.1 Linked Descriptors

A transaction can consist of either a single block transfer or of several block transfers. When a transaction consists of several block transfers it is done with the help of linked descriptors.

Memory Sections illustrates how linked descriptors work (see *Memory Sections* figure in the *Transfer Descriptors* from Related Links). When the first block transfer is completed on DMA channel 0, the DMAC fetches the next transfer descriptor, which is pointed to by the value stored in the Next Descriptor Address (DESCADDR) register of the first transfer descriptor. Fetching the next transfer descriptor (DESCADDR) is continued until the last transfer descriptor. When the block transfer for the last transfer descriptor is executed and DESCADDR = 0x00000000, the transaction is terminated. For further details on how the next descriptor is fetched from SRAM (see *Data Transmission* from Related Links).

Related Links

22.6.2.3. Transfer Descriptors
22.6.2.5. Data Transmission

22.6.3.1.1 Adding Descriptor to the End of a List

To add a new descriptor at the end of the descriptor list, create the descriptor in SRAM, with DESCADDR = 0x00000000 indicating that it is the new last descriptor in the list, and modify the DESCADDR value of the current last descriptor to the address of the newly created descriptor.

22.6.3.1.2 Modifying a Descriptor in a List

In order to add descriptors to a linked list, the following actions must be performed:

1. Enable the Suspend interrupt for the DMA channel.
2. Enable the DMA channel.
3. Reserve memory space in SRAM to configure a new descriptor.
4. Configure the new descriptor:
   - Set the next descriptor address (DESCADDR)
   - Set the destination address (DESCADDR)
   - Set the source address (SRCADDR)
   - Configure the block transfer control (BTCTRL) including
     - Optionally enable the suspend block action
     - Set the descriptor VALID bit
5. Clear the VALID bit for the existing list and for the descriptor which has to be updated.
6. Read DESCADDR from the write-back memory.
   - If the DMA has not already fetched the descriptor that requires changes (in other words, DESCADDR is wrong):
     - Update the DESCADDR location of the descriptor from the list
     - Optionally clear the suspend block action
     - Set the descriptor VALID bit to '1'
     - Optionally enable the Resume Software command
   - If the DMA is executing the same descriptor as the one that requires changes:
     - Set the Channel Suspend Software command and wait for the suspend interrupt
     - Update the next descriptor address (DESCADDR) in the write-back memory
     - Clear the interrupt sources and set the Resume Software command
     - Update the DESCADDR location of the descriptor from the list
     - Optionally clear the suspend block action
     - Set the descriptor VALID bit to '1'
7. Go to step 4 if needed.

22.6.3.1.3 Adding a Descriptor Between Existing Descriptors

To insert a new descriptor 'C' between two existing descriptors ('A' and 'B'), the descriptor currently executed by the DMA must be identified.

1. If DMA is executing descriptor B, descriptor C cannot be inserted.
2. If DMA has not started to execute descriptor A, follow the steps:
   a. Set the descriptor A VALID bit to '0'.

b. Set the DESCADDR value of descriptor A to point to descriptor C instead of descriptor B (see *DESCADDR* in the *DMAC Register Summary* from Related Links).
c. Set the DESCADDR value of descriptor C to point to descriptor B (see *DESCADDR* in the *DMAC Register Summary* from Related Links).
d. Set the descriptor A VALID bit to '1'.

3. If DMA is executing descriptor A:
   a. Apply the software suspend command to the channel and
   b. Perform steps 2.1 through 2.4.
   c. Apply the software resume command to the channel.

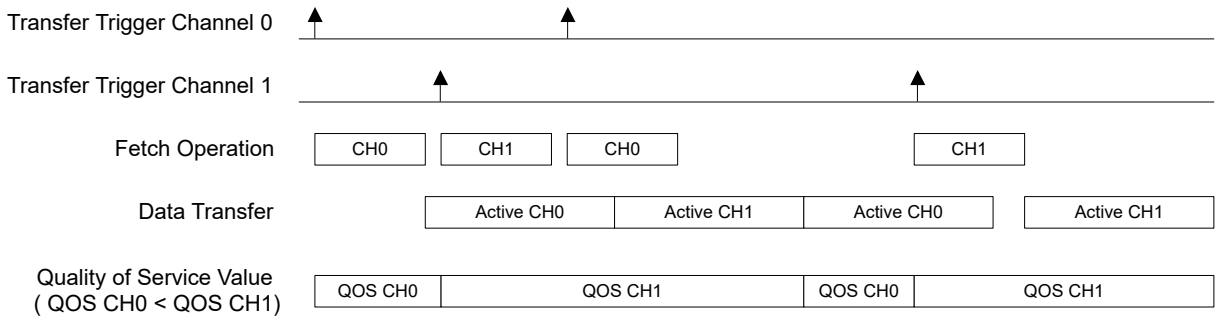
Related Links

22.9. DMAC Register Summary (SRAM)

22.6.3.2 Transfer Quality of Service

Each priority level group has dedicated quality of service settings. The setting can be written in the corresponding Quality of Service bit group in the Priority Control x register (PRICTRL0.QOSn).

Figure 22-10. Quality of Service



When a channel is stored in the Pre-Fetch or Active Channel, the corresponding PRICTRLx.QOS bits value is stored in the respective channel. As shown in Quality of Service, the DMAC will select the highest QOS value between Active and Pre-Fetch channels. This value will apply to all DMAC buses.

22.6.3.3 Channel Suspend

The channel operation can be suspended at any time by software by writing a '1' to the Suspend command in the Command bit field of Channel Control B register (CHCTRLB.CMD). After the ongoing burst transfer is completed, the channel operation is suspended and the suspend command is automatically cleared.

When suspended, the Channel Suspend Interrupt flag in the Channel Interrupt Status and Clear register is set (CHINTFLAG.SUSP=1) and the optional suspend interrupt is generated.

By configuring the block action to suspend by writing Block Action bit group in the Block Transfer Control register (BTCTRL.BLOCKACT is 0x2 or 0x3), the DMA channel will be suspended after it has completed a block transfer. The DMA channel will be kept enabled and will be able to receive transfer triggers, but it will be removed from the arbitration scheme.

If an invalid transfer descriptor (BTCTRL.VALID=0) is fetched from SRAM, the DMA channel will be suspended, and the Channel Fetch Error bit in the Channel Status register (CHASTATUS.FERR) will be set.

Note: Only enabled DMA channels can be suspended. If a channel is disabled when it is attempted to be suspended, the internal suspend command will be ignored.

For more details on transfer descriptors (see *Transfer Descriptors* from Related Links).

Related Links

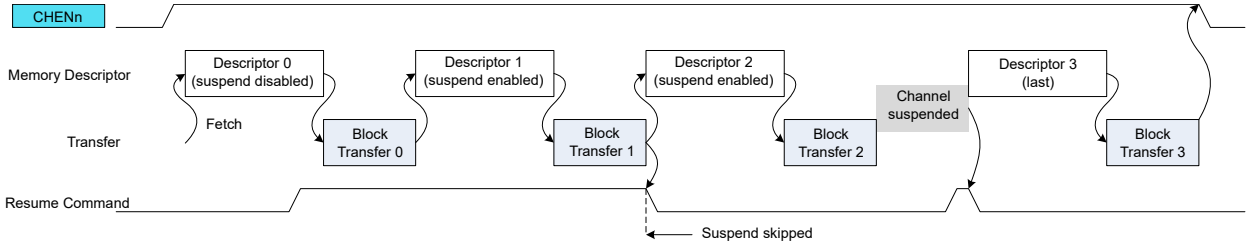
22.6.2.3. Transfer Descriptors

22.6.3.4 Channel Resume and Next Suspend Skip

A channel operation can be resumed by software by setting the Resume command in the Command bit field of the Channel Control B register (CHCTRLB.CMD). If the channel is already suspended, the channel operation

resumes from where it previously stopped when the Resume command is detected. When the Resume command is issued before the channel is suspended, the next suspend action is skipped and the channel continues the normal operation.

Figure 22-11. Channel Suspend/Resume Operation



22.6.3.5 Event Input Actions

The event input actions are available only on the least significant DMA channels. For more details on channels with event input support (see *Event System (EVSYS)* from Related Links).

Before using event input actions, the event controller must be configured first according to the following table, and the Channel Event Input Enable bit in the Channel Event Control register (CHEVCTRL.EVIE) must be written to '1'. See *Events* from Related Links.

Table 22-1. Event Input Action

Action	CHEVCTRL.EVACT	CHCTRLA.TRIGSRC
None	NOACT	—
Normal Transfer	TRIG	DISABLE
Conditional Transfer on Strobe	TRIG	Any peripheral
Conditional Transfer	CTRIG	
Conditional Block Transfer	CBLOCK	
Channel Suspend	SUSPEND	
Channel Resume	RESUME	
Skip Next Block Suspend	SSKIP	
Increase priority	INCPRI	

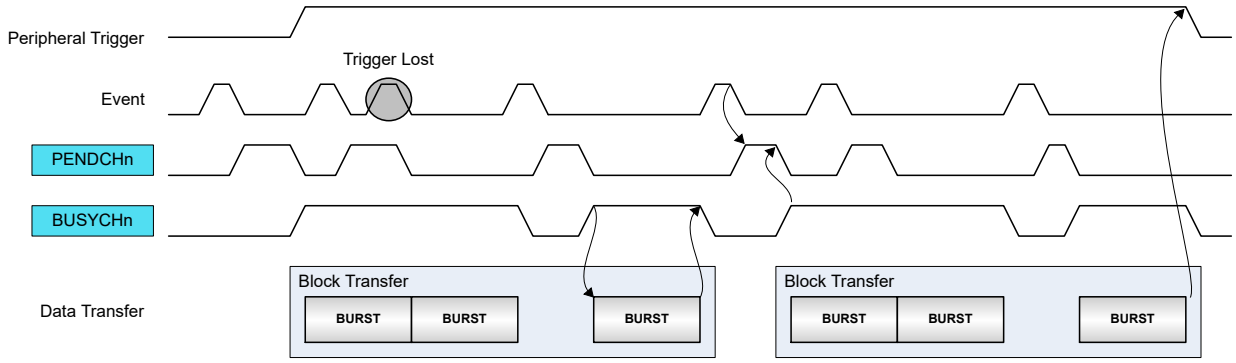
Normal Transfer

The event input is used to trigger a beat or burst transfer on peripherals.

The event is acknowledged as soon as the event is received. When received, both the Channel Pending status bit in the Channel Status register (CHSTATUS.PEND) and the corresponding Channel n bit in the Pending Channels register (PENDCH.PENDCHn) are set. If the event is received while the channel is pending, the event trigger is lost.

The following figure shows an example where beat transfers are enabled by internal events.

Figure 22-12. Burst Event Trigger Action



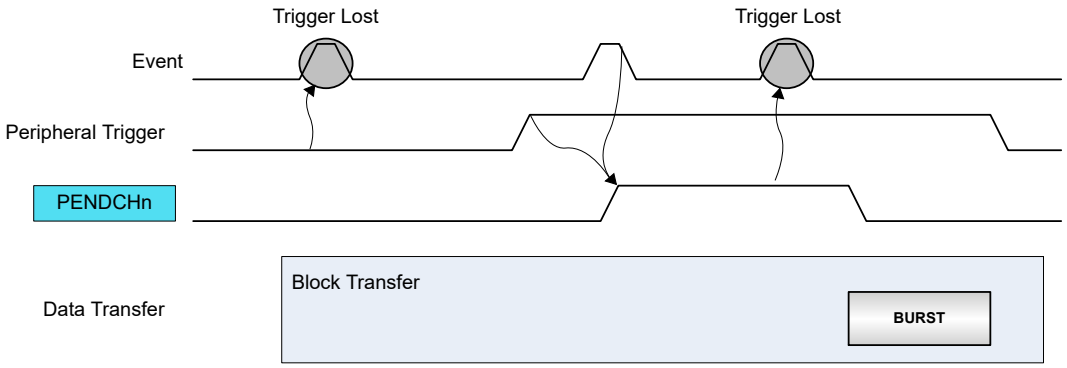
Conditional Transfer on Strobe

The event input is used to trigger a transfer on peripherals with pending transfer requests. This event action is intended to be used with peripheral triggers, for example, for timed communication protocols or periodic transfers between peripherals: only when the peripheral trigger coincides with the occurrence of a (possibly cyclic) event the transfer is issued.

The event is acknowledged as soon as the event is received. The peripheral trigger request is stored internally when the previous trigger action is completed (in other words, the channel is not pending) and when an active event is received. If the peripheral trigger is active, the DMA waits for an event before the peripheral trigger is internally registered. When both event and peripheral transfer trigger are active, both CHSTATUS.PEND and PENDCH.PENDCHn are set. A software trigger will now trigger a transfer.

The following figure shows an example where the peripheral beat transfer is started by a conditional strobe event action.

Figure 22-13. Periodic Event with Burst Peripheral Triggers



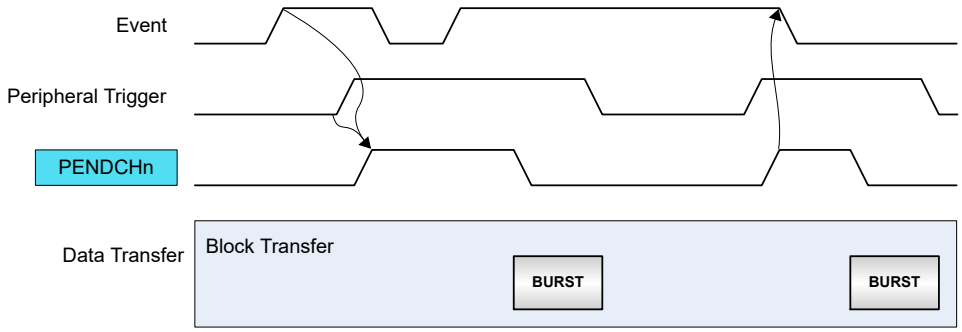
Conditional Transfer

The event input is used to trigger a conditional transfer on peripherals with pending transfer requests. As example, this type of event can be used for peripheral-to-peripheral transfers, where one peripheral is the source of event and the second peripheral is the source of the trigger.

Each peripheral trigger is stored internally when the event is received. When the peripheral trigger is stored internally, the Channel Pending status bit is set (CHSTATUS.PEND), the respective Pending Channel n Bit in the Pending Channels register is set (PENDCH.PENDCHn), and the event is acknowledged. A software trigger will now trigger a transfer.

The following figure shows an example where conditional event is enabled with peripheral beat trigger requests.

Figure 22-14. Conditional Event with Burst Peripheral Triggers

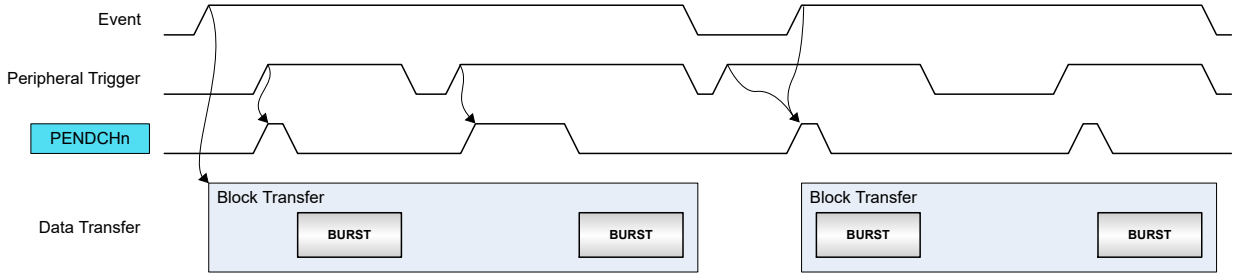


Conditional Block Transfer
The event input is used to trigger a conditional block transfer on peripherals.

Before starting transfers within a block, an event must be received. When received, the event is acknowledged when the block transfer is completed. A software trigger will trigger a transfer.

The following figure shows an example where conditional event block transfer is started with peripheral beat trigger requests.

Figure 22-15. Conditional Block Transfer with Burst Peripheral Triggers



Channel Suspend
The event input is used to suspend an ongoing channel operation. The event is acknowledged when the current AHB access is completed. For more details on Channel Suspend (see *Channel Suspend* from Related Links).

Channel Resume
The event input is used to resume a suspended channel operation. The event is acknowledged as soon as the event is received and the Channel Suspend Interrupt Flag (CHINTFLAG.SUSP) is cleared. See *Channel Suspend* from Related Links.

Skip Next Block Suspend
This event can be used to skip the next block suspend action. If the channel is suspended before the event rises, the channel operation is resumed and the event is acknowledged. If the event rises before a suspend block action is detected, the event is kept until the next block suspend detection. When the block transfer is completed, the channel continues the operation (not suspended) and the event is acknowledged.

Increase priority
This event can be used to increase a channel priority and to request higher quality of service (QOS), when critical transfers must be done. When the event is detected, the channel will have the highest priority and the output Quality of Service value is internally forced to the maximum value. The event is acknowledged when the trigger action execution is completed. When acknowledged, the channel will recover its initial priority level and quality of service settings.

Related Links

22.6.3.3. Channel Suspend
22.6.6. Events
28. Event System (EVSYS)

22.6.3.6 Event Output Selection

The event output selections are available only for channels supporting event outputs.

The Channel Event Output Enable can be set in the corresponding Channel n Event Control register (CHEVCTRL.EVOE). The Event Output Mode bits in the Channel n Event Control register (CHEVCTRL.EVOMODE) selects the event type the channel will generate.

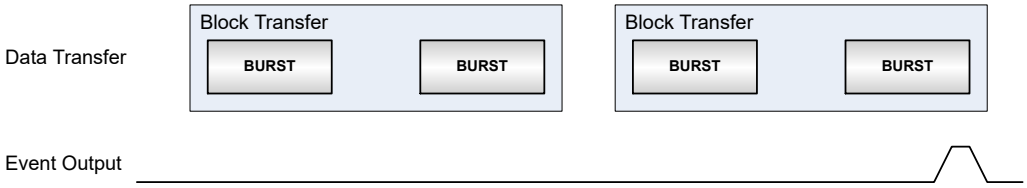
The transfer events (CHEVCTRL.EVOMODE = DEFAULT) are strobe events and their duration is one CLK_DMAC_AHB clock period. The transfer event type selection is available in each Descriptor Block Control location (BTCTRL.EVOSEL). Block or burst event output generation is supported.

The trigger action event (CHEVCTRL.EVOMODE = TRIGACT) is a level, active while the trigger action execution is not completed.

Block Event Output

When the block event output is selected, an event strobe is generated when the block transfer is completed. The pulse width of a block event output from a channel is one AHB clock cycle. It is also possible to use this event type to generate an event when the transaction is complete. For this type of application, the block event selection must be set in the last transfer descriptor only, as shown below.

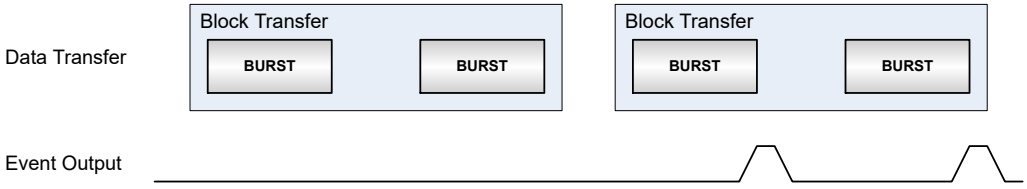
Figure 22-16. Block Event Output Generation



Burst Event Output

When the burst event output is selected, an event strobe is generated when each burst transfer within the corresponding block is completed. The pulse width of a burst event output from a channel is one AHB clock cycle. The figure below shows an example where the burst event output is set in the second descriptor of a linked list.

Figure 22-17. Burst Event Output Generation

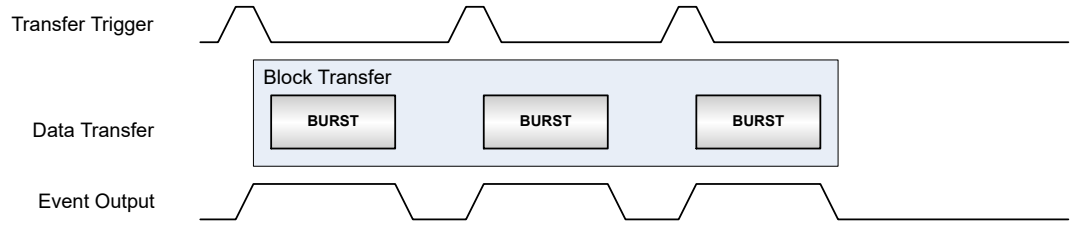


Trigger Action Event Output

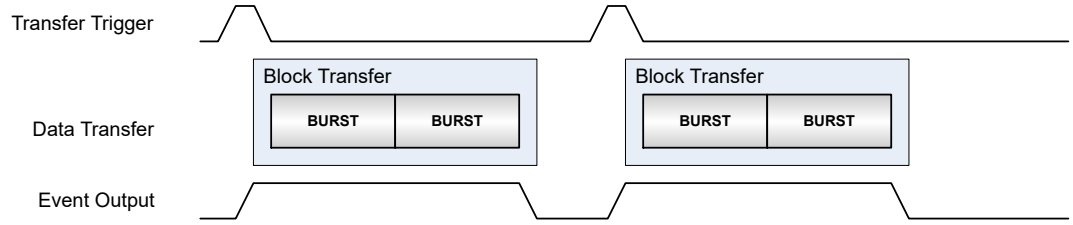
When the trigger action event output is selected, an event level is generated. The event output is set when the transfer trigger occurred, and cleared when the corresponding trigger action is completed. The following figure shows an example for each trigger action type.

Figure 22-18. Trigger Action Event Output Generation

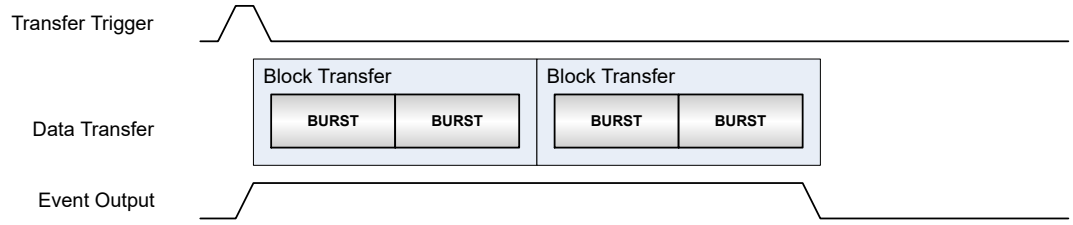
Burst Trigger Action Event Output



Block Trigger Action Event Output



Transaction Trigger Action Event Output



22.6.3.7 Aborting Transfers

Transfers on any channel can be aborted gracefully by software by disabling the corresponding DMA channel. It is also possible to abort all ongoing or pending transfers by disabling the DMAC.

When a DMA channel disable request or DMAC disable request is detected:

- Ongoing transfers of the active channel will be disabled when the ongoing beat transfer is completed and the write-back memory section is updated. This prevents transfer corruption before the channel is disabled.
- All other enabled channels will be disabled in the next clock cycle.

The corresponding Channel Enable bit in the Channel Control A register is cleared (CHCTRLA.ENABLE=0) when the channel is disabled.

The corresponding DMAC Enable bit in the Control register is cleared (CTRL.DMAENABLE=0) when the entire DMAC module is disabled.

22.6.3.8 CRC Operation

A Cyclic Redundancy Check (CRC) is an error detection technique used to find errors in data. It is commonly used to determine whether the data during a transmission, or data present in data and program memories has been corrupted or not. A CRC takes a data stream or a block of data as input and generates a 16- or 32-bit output that can be appended to the data and used as a checksum.

When the data is received, the device or application repeats the calculation: If the new CRC result does not match the one calculated earlier, the block contains a data error. The application will, then, detect this and may take a corrective action, such as requesting the data to be sent again or simply not using the incorrect data.

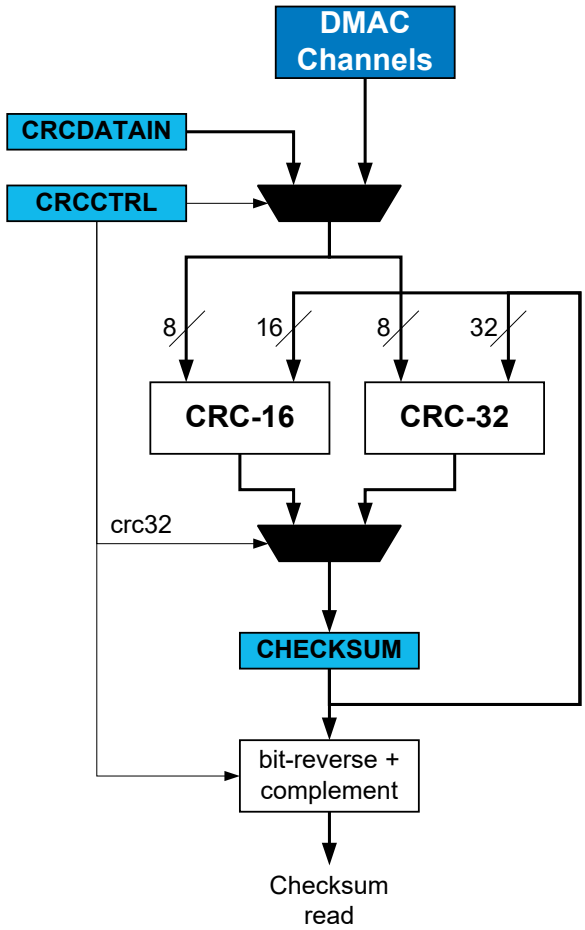
The CRC engine in DMAC supports two commonly used CRC polynomials: CRC-16 (CRC-CCITT) and CRC-32 (IEEE 802.3). Typically, applying CRC-n (CRC-16 or CRC-32) to a data block of arbitrary length will detect any single alteration that is ≤n bits in length, and will detect the fraction 1-2-n of all longer error bursts.

- CRC-16:
  - Polynomial: $x^{16}+ x^{12}+ x^5+ 1$
  - Hex value: 0x1021
- CRC-32:
  - Polynomial: $x^{32}+x^{26}+ x^{23}+ x^{22}+x^{16}+ x^{12}+ x^{11}+ x^{10}+ x^8+ x^7+ x^5+ x^4+ x^2+ x + 1$
  - Hex value: 0x04C11DB7

The data source for the CRC engine can either be one of the DMA channels or the APB bus interface, and must be selected by writing to the CRC Input Source bits in the CRC Control register (CRCCTRL.CRCSRC). The CRC engine then takes data input from the selected source and generates a checksum based on these data. The checksum is available in the CRC Checksum register (CRCCHKSUM). When the CRC-32 polynomial is used, the final checksum read is bit reversed and complemented, as shown in the following figure.

The CRC polynomial is selected by writing to the CRC Polynomial Type bit in the CRC Control register (CRCCTRL.CRCPOLY); the default is CRC-16. The CRC engine operates on byte only. When the DMA is used as a data source for the CRC engine, the DMA channel beat size setting will be used. When used with the APB bus interface, the application must select the CRC Beat Size bit field of the CRC Control register (CRCCTRL.CRCBEATSIZE). 8-, 16- or 32-bit bus transfer access type is supported. The corresponding number of bytes will be written in the CRCDATAIN register and the CRC engine will operate on the input data in a byte-by-byte manner.

Figure 22-19. CRC Generator Block Diagram



CRC on DMA Data

CRC-16 or CRC-32 calculations can be performed on data passing through any DMA channel. Once a DMA channel is selected as the source, the CRC engine will continuously generate the CRC on the data passing through the DMA channel. The checksum is available for readout once the DMA transaction is completed or aborted. A CRC can also be generated on SRAM, Flash or I/O memory by passing these

data through a DMA channel. If the latter is done, the destination register for the DMA data can be the data input (CRCDATAIN) register in the CRC engine.

CRC Using the I/O Interface Before using the CRC engine with the I/O interface, the application must set the CRC Beat Size bits in the CRC Control register (CRCCTRL.CRCBEATSIZE). 8/16/32-bit bus transfer type can be selected.

CRC can be performed on any data by loading them into the CRC engine using the CPU and writing the data to the CRCDATAIN register. Using this method, an arbitrary number of bytes can be written to the register by the CPU, and CRC is done continuously for each byte. This means if a 32-bit data is written to the CRCDATAIN register, the CRC engine takes four cycles to calculate the CRC. The CRC complete is signaled by a set CRCBUSY bit in the CRCSTATUS register. New data can be written only when the CRCBUSY flag is not set.

22.6.3.9 Memory CRC Generation

When enabled, it is possible to automatically calculate a memory block checksum. When the channel is enabled and the descriptor is fetched, the CRC Checksum register (CRCCHKSUM) is reloaded with the initial checksum value (CHKINIT) stored in the Block Transfer Destination Address register (DSTADDR). The DMA read and calculate the checksum over the data from the source address.When the checksum calculation is completed, the CRC value is stored in the CRC Checksum register (CRCCHKSUM), the Transfer Complete interrupt flag is set (CHINTFLAGn.TCMPL) and optional interrupt is generated.

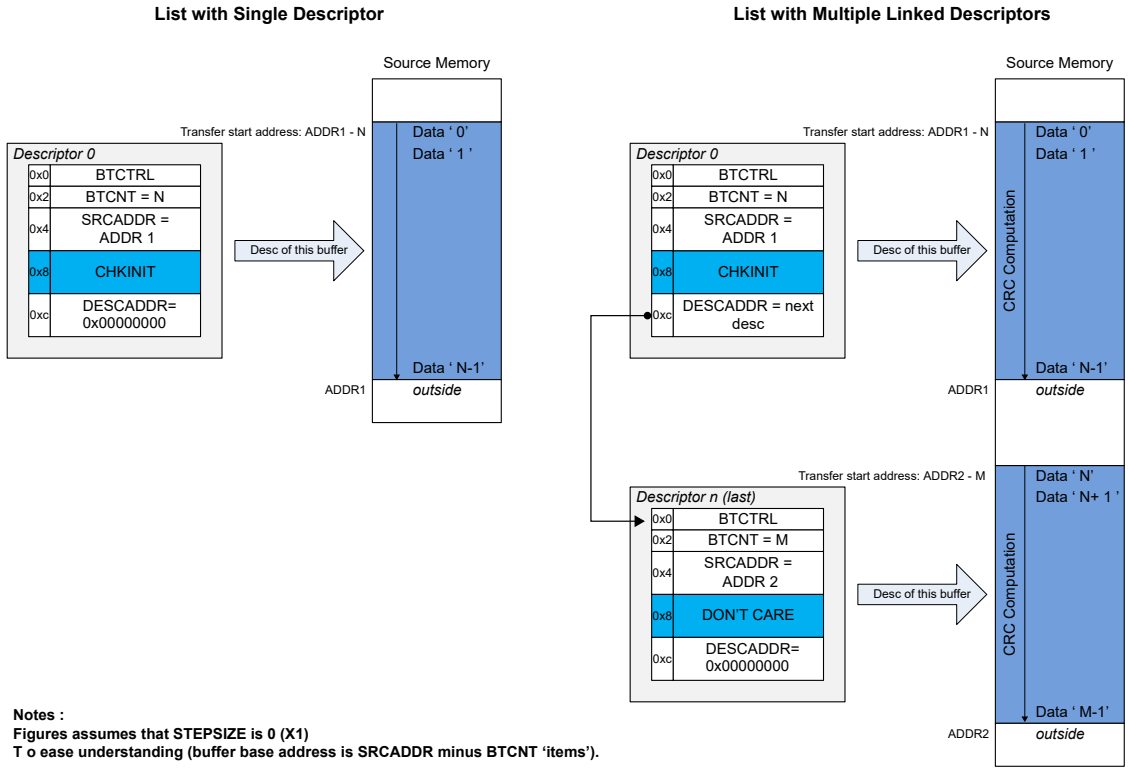
If the linked descriptor is in the list (DESCADDR !=0), the DMA will fetch the next descriptor and CRC calculation continues as described above. When the last list descriptor is executed, the channel is automatically disabled.

In order to enable the memory CRC generation, the following actions must be performed:

1. The CRC module must be set to be used with a DMA channel (CRCCTRL.CRCSRC)
2. Reserve memory space addresses to configure a descriptor or a list of descriptors
3. Configure each descriptor:
   - Set the next descriptor address (DESCADDR)
   - Set the destination address with the initial checksum value (DSTADDR = CHKINIT) in the first descriptior in a list
   - Set the transfer source address (SRCADDR)
   - Set the block transfer count (BTCNT)
   - Set the memory CRC generation operation mode (CRCCTRL.CRCMODE = CRCGEN)
   - Enable optional interrupts
4. Enable the corresponding DMA channel (CHCTRLAn.ENABLE)

The figure below shows the CRC computation slots and descriptor configuration when single or linked-descriptors transfers are enabled.

Figure 22-20. CRC Computation with Single Linked Transfers



Notes :
Figures assumes that STEPSIZE is 0 (X1)
T o ease understanding (buffer base address is SRCADDR minus BTCNT 'items').

22.6.3.10 Memory CRC Monitor

When enabled, it is possible to continuously check a a memory block data integrity by calculating and checking the CRC checksum. The expected CRC checksum value must be located in the last memory block location, as shown in the table below:

CRCCTRL.CRCPOLY	CRCCTRL.CRCBEATSIZE	Last Memory Block Byte Locations Value (MSB Byte First)	CHECKSUM Result
CRC-16	Byte	Expected CRC[7:0]	0x00000000
	Half-word	Expected CRC[15:8]	
	Word	0x00 0x00 Expected CRC[7:0] Expected CRC[15:8]	
CRC-32	Byte	Expected CRC[31:24]	CRC Magic Number (0x2144DF1C)
	Half-word	Expected CRC[23:16]	
	Word	Expected CRC[15:8] Expected CRC[7:0]	

When the channel is enabled and the descriptor is fetched, the CRC Checksum register (CRCCHKSUM) is reloaded with the initial checksum value (CHKINIT), stored in the DSTADDR location of the first descriptor. The DMA read and calculate the checksum over the entire data from the source address.When the checksum calculation is completed the DMA read the last beat from the memory, the calculated CRC value from the CRC Checksum register is compared to zero or CRC magic number, depending on CRC polynomial selection.

If the CHECKSUM does not match the comparison value the DMA channel is disabled, and both and the CRC Error bit in the Channel n Status register (CHSTATUSn.CRCERR) and Transfer Error interrupt flag (CHINTFLAGn.TERR) are set. If enabled, the Transfer Error interrupt is generated.

If the calculated checksum value matches the compare value, the Transfer Complete interrupt flag (CHINTFLAGn.TCMPL) is set, optional interrupt is generated and the DMA will perform the following actions, depending on the descriptor list settings:

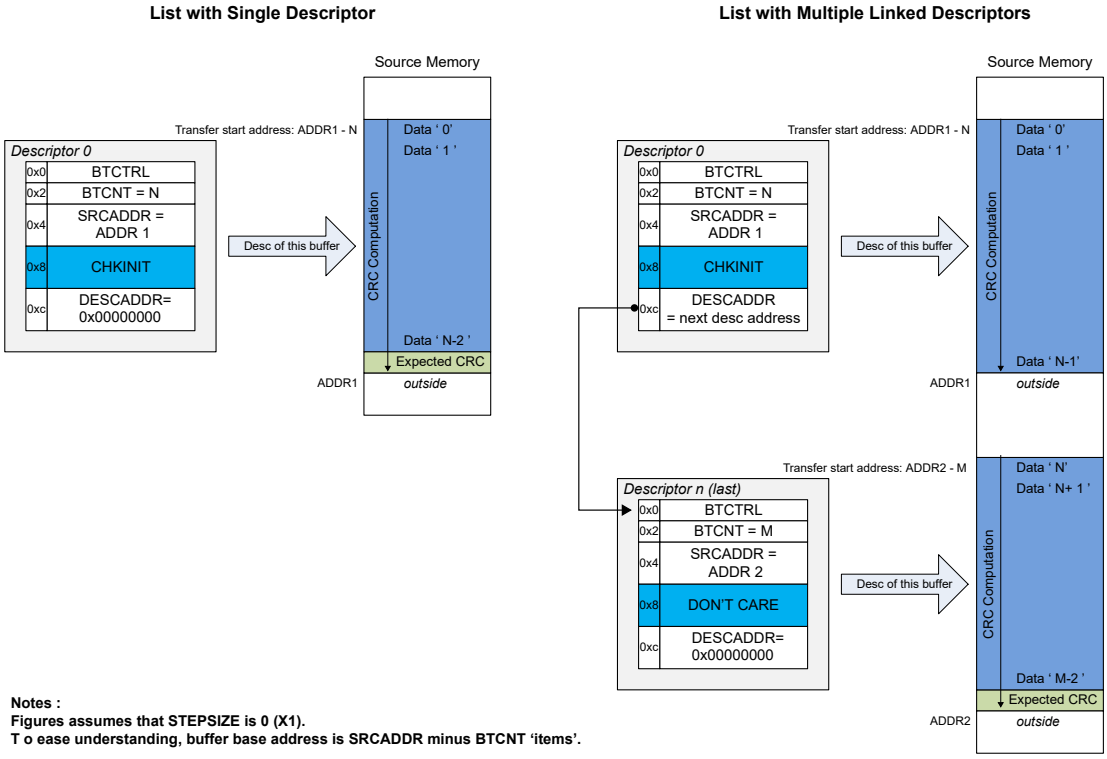
- If the list has only one descriptor, the DMA will re-fetch the descriptor
- If the current descriptor is the last descriptor from the list, the DMA will fetch the first descriptor from the list

When the fetch is completed, the DMA restarts the operations described above when new triggers are detected.

In order to enable the memory CRC monitor, the following actions must be performed:

1. The CRC module must be set to be used with a DMA channel (CRCCTRL.CRCSRC)
2. Reserve memory space addresses to configure a descriptor or a list of descriptors
3. Configure each descriptor
   - Set the next descriptor address (DESCADDR)
   - In the first list descriptor, set the destination address with the initial checksum value (DSTADDR = CHKINIT)
   - Set the transfer source address (SRCADDR)
   - Set the block transfer count (BTCNT)
   - Set the memory CRC monitor operation mode (CRCCTRL.CRCMODE = CRCMON)
   - Enable optional interrupts
4. Enable the corresponding DMA channel (CHCTRLAn.ENABLE)

Figure 22-21. CRC Computation and Check with Single or Linked Transfers



Notes :
Figures assumes that STEPSIZE is 0 (X1).
T o ease understanding, buffer base address is SRCADDR minus BTCNT 'items'.

22.6.4 DMA Operation

Not applicable.

22.6.5 Interrupts

The DMAC channels have the following interrupt sources:

- Transfer Complete (TCMPL): Indicates that a block transfer is completed on the corresponding channel. See *Data Transmission* from Related Links.
- Transfer Error (TERR): Indicates that a bus error has occurred during a burst transfer, or that an invalid descriptor has been fetched. See *Error Handling* from Related Links.
- Channel Suspend (SUSP): Indicates that the corresponding channel has been suspended. See *Channel Suspend* and *Data Transmission* from Related Links.

Each interrupt source has an Interrupt flag associated with it. The Interrupt flag in the Channel Interrupt Flag Status and Clear (CHINTFLAG) register is set when the Interrupt condition occurs. Each interrupt can be individually enabled by setting the corresponding bit in the Channel Interrupt Enable Set register (CHINTENSET=1), and disabled by setting the corresponding bit in the Channel Interrupt Enable Clear register (CHINTENCLR=1). The status of enabled interrupts can be read from either INTENSET or INTENCLR.

An interrupt request is generated when the Interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the Interrupt flag is cleared, the interrupt is disabled, the DMAC is reset or the corresponding DMA channel is reset. See CHINTFLAG for details on how to clear Interrupt flags. All interrupt requests are ORed together on system level to generate one combined interrupt request to the NVIC. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

The user must read the Channel Interrupt Status (INTSTATUS) register to identify the channels with pending interrupts and must read the Channel Interrupt Flag Status and Clear (CHINTFLAG) register to determine which Interrupt condition is present for the corresponding channel. It is also possible to read the Interrupt Pending register (INTPEND), which provides the lowest channel number with pending interrupt and the respective Interrupt flags.

Note: Interrupts must be globally enabled for interrupt requests to be generated.

Related Links

22.6.2.5. Data Transmission
22.6.3.3. Channel Suspend
10.2. Nested Vector Interrupt Controller (NVIC)

22.6.6 Events

The DMAC can generate the following output events:

- Channel (CH): Generated when a block transfer for a given channel has been completed, or when a beat transfer within a block transfer for a given channel has been completed. See *Event Output Selection* from Related Links.

Setting the Channel Event Output Enable bit (CHEVCTRLx.EVOE = 1) enables the corresponding output event configured in the Event Output Selection bit group in the Block Transfer Control register (BTCTRL.EVOSEL). Clearing CHEVCTRLx.EVOE = 0 disables the corresponding output event.

The DMAC can take the following actions on an input event:

- Transfer and Periodic Transfer Trigger (TRIG): normal transfer or periodic transfers on peripherals are enabled
- Conditional Transfer Trigger (CTRIG): conditional transfers on peripherals are enabled
- Conditional Block Transfer Trigger (CBLOCK): conditional block transfers on peripherals are enabled
- Channel Suspend Operation (SUSPEND): suspend a channel operation
- Channel Resume Operation (RESUME): resume a suspended channel operation
- Skip Next Block Suspend Action (SSKIP): skip the next block suspend transfer condition
- Increase Priority (INCPRI): increase channel priority

Setting the Channel Event Input Enable bit (CHEVCTRLx.EVIE = 1) enables the corresponding action on input event. Clearing this bit disables the corresponding action on input event. Note that several actions can be enabled for incoming events. If several events are connected to the peripheral, any enabled action will be taken for any of the incoming events. See *Event Input Actions* from Related Links for more details on event input actions.
Note: Event input and outputs are not available for every channel. See *Features* from Related Links.

Related Links

22.6.3.6. Event Output Selection
22.6.3.5. Event Input Actions
22.2. Features

22.6.7 Sleep Mode Operation

22.6.8 Synchronization

Not applicable.

22.7 DMAC Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRL	7:0							DMAENABLE	SWRST
		15:8					LVLENx3	LVLENx2	LVLENx1	LVLENx0
0x02	CRCCTRL	7:0					CRCPOLY[1:0]		CRCBEATSIZE[1:0]	
		15:8	CRCMODE[1:0]		CRCSRC[5:0]					
0x04	CRCDATAIN	7:0	CRCDATAIN[7:0]							
		15:8	CRCDATAIN[15:8]							
		23:16	CRCDATAIN[23:16]							
		31:24	CRCDATAIN[31:24]							
0x08	CRCCHKSUM	7:0	CRCCHKSUM[7:0]							
		15:8	CRCCHKSUM[15:8]							
		23:16	CRCCHKSUM[23:16]							
		31:24	CRCCHKSUM[31:24]							
0x0C	CRCSTATUS	7:0						CRCERR	CRCZERO	CRCBUSY
0x0D	DBGCTRL	7:0								DBGRUN
0x0E ... 0x0F	Reserved									
0x10	SWTRIGCTRL	7:0	SWTRIGn[7:0]							
		15:8	SWTRIGn[15:8]							
		23:16								
		31:24								
0x14	PRICTRL0	7:0	RRLVLEN0	QOS00[1:0]		LVLPRI0[4:0]				
		15:8	RRLVLEN1	QOS01[1:0]		LVLPRI1[4:0]				
		23:16	RRLVLEN2	QOS02[1:0]		LVLPRI2[4:0]				
		31:24	RRLVLEN3	QOS03[1:0]		LVLPRI3[4:0]				
0x18 ... 0x1F	Reserved									
0x20	INTPEND	7:0				ID[4:0]				
		15:8	PEND	BUSY	FERR	CRCERR		SUSP	TCMPL	TERR
0x22 ... 0x23	Reserved									
0x24	INTSTATUS	7:0								
		15:8								
		23:16								
		31:24								
0x28	BUSYCH	7:0								
		15:8								
		23:16								
		31:24								
0x2C	PENDCH	7:0								
		15:8								
		23:16								
		31:24								
0x30	ACTIVE	7:0								
		15:8	ABUSY							
		23:16	BTCNT[7:0]							
		31:24	BTCNT[15:8]							
0x34	BASEADDR	7:0	BASEADDR[7:0]							
		15:8	BASEADDR[15:8]							
		23:16	BASEADDR[23:16]							
		31:24	BASEADDR[31:24]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x38	WRBADDR	7:0	WRBADDR[7:0]							
		15:8	WRBADDR[15:8]							
		23:16	WRBADDR[23:16]							
		31:24	WRBADDR[31:24]							
0x3C ... 0x3F	Reserved									
0x40	CHCTRLA0	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0x44	CHCTRLB0	7:0							CMD[1:0]	
0x45	CHPRILVL0	7:0								
0x46	CHEVCTRL0	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0x47 ... 0x4B	Reserved									
0x4C	CHINTENCLR0	7:0						SUSP	TCMPL	TERR
0x4D	CHINTENSET0	7:0						SUSP	TCMPL	TERR
0x4E	CHINTFLAG0	7:0						SUSP	TCMPL	TERR
0x4F	CHSTATUS0	7:0					CRCERR	FERR	BUSY	PEND
0x50	CHCTRLA1	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0x54	CHCTRLB1	7:0							CMD[1:0]	
0x55	CHPRILVL1	7:0								
0x56	CHEVCTRL1	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0x57 ... 0x5B	Reserved									
0x5C	CHINTENCLR1	7:0						SUSP	TCMPL	TERR
0x5D	CHINTENSET1	7:0						SUSP	TCMPL	TERR
0x5E	CHINTFLAG1	7:0						SUSP	TCMPL	TERR
0x5F	CHSTATUS1	7:0					CRCERR	FERR	BUSY	PEND
0x60	CHCTRLA2	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0x64	CHCTRLB2	7:0							CMD[1:0]	
0x65	CHPRILVL2	7:0								
0x66	CHEVCTRL2	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0x67 ... 0x6B	Reserved									
0x6C	CHINTENCLR2	7:0						SUSP	TCMPL	TERR
0x6D	CHINTENSET2	7:0						SUSP	TCMPL	TERR
0x6E	CHINTFLAG2	7:0						SUSP	TCMPL	TERR
0x6F	CHSTATUS2	7:0					CRCERR	FERR	BUSY	PEND
0x70	CHCTRLA3	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0x74	CHCTRLB3	7:0							CMD[1:0]	
0x75	CHPRILVL3	7:0								
0x76	CHEVCTRL3	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0x77 ... 0x7B	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7C	CHINTENCLR3	7:0						SUSP	TCMPL	TERR
0x7D	CHINTENSET3	7:0						SUSP	TCMPL	TERR
0x7E	CHINTFLAG3	7:0						SUSP	TCMPL	TERR
0x7F	CHSTATUS3	7:0					CRCERR	FERR	BUSY	PEND
0x80	CHCTRLA4	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0x84	CHCTRLB4	7:0							CMD[1:0]	
0x85	CHPRILVL4	7:0								
0x86	CHEVCTRL4	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0x87 ... 0x8B	Reserved									
0x8C	CHINTENCLR4	7:0						SUSP	TCMPL	TERR
0x8D	CHINTENSET4	7:0						SUSP	TCMPL	TERR
0x8E	CHINTFLAG4	7:0						SUSP	TCMPL	TERR
0x8F	CHSTATUS4	7:0					CRCERR	FERR	BUSY	PEND
0x90	CHCTRLA5	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0x94	CHCTRLB5	7:0							CMD[1:0]	
0x95	CHPRILVL5	7:0								
0x96	CHEVCTRL5	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0x97 ... 0x9B	Reserved									
0x9C	CHINTENCLR5	7:0						SUSP	TCMPL	TERR
0x9D	CHINTENSET5	7:0						SUSP	TCMPL	TERR
0x9E	CHINTFLAG5	7:0						SUSP	TCMPL	TERR
0x9F	CHSTATUS5	7:0					CRCERR	FERR	BUSY	PEND
0xA0	CHCTRLA6	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0xA4	CHCTRLB6	7:0							CMD[1:0]	
0xA5	CHPRILVL6	7:0								
0xA6	CHEVCTRL6	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0xA7 ... 0xAB	Reserved									
0xAC	CHINTENCLR6	7:0						SUSP	TCMPL	TERR
0xAD	CHINTENSET6	7:0						SUSP	TCMPL	TERR
0xAE	CHINTFLAG6	7:0						SUSP	TCMPL	TERR
0xAF	CHSTATUS6	7:0					CRCERR	FERR	BUSY	PEND
0xB0	CHCTRLA7	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0xB4	CHCTRLB7	7:0							CMD[1:0]	
0xB5	CHPRILVL7	7:0								
0xB6	CHEVCTRL7	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0xB7 ... 0xBB	Reserved									
0xBC	CHINTENCLR7	7:0						SUSP	TCMPL	TERR
0xBD	CHINTENSET7	7:0						SUSP	TCMPL	TERR
0xBE	CHINTFLAG7	7:0						SUSP	TCMPL	TERR

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0xBF	CHSTATUS7	7:0					CRCERR	FERR	BUSY	PEND
0xC0	CHCTRLA8	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0xC4	CHCTRLB8	7:0							CMD[1:0]	
0xC5	CHPRILVL8	7:0								
0xC6	CHEVCTRL8	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0xC7 ... 0xCB	Reserved									
0xCC	CHINTENCLR8	7:0						SUSP	TCMPL	TERR
0xCD	CHINTENSET8	7:0						SUSP	TCMPL	TERR
0xCE	CHINTFLAG8	7:0						SUSP	TCMPL	TERR
0xCF	CHSTATUS8	7:0					CRCERR	FERR	BUSY	PEND
0xD0	CHCTRLA9	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0xD4	CHCTRLB9	7:0							CMD[1:0]	
0xD5	CHPRILVL9	7:0								
0xD6	CHEVCTRL9	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0xD7 ... 0xDB	Reserved									
0xDC	CHINTENCLR9	7:0						SUSP	TCMPL	TERR
0xDD	CHINTENSET9	7:0						SUSP	TCMPL	TERR
0xDE	CHINTFLAG9	7:0						SUSP	TCMPL	TERR
0xDF	CHSTATUS9	7:0					CRCERR	FERR	BUSY	PEND
0xE0	CHCTRLA10	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0xE4	CHCTRLB10	7:0							CMD[1:0]	
0xE5	CHPRILVL10	7:0								
0xE6	CHEVCTRL10	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0xE7 ... 0xEB	Reserved									
0xEC	CHINTENCLR10	7:0						SUSP	TCMPL	TERR
0xED	CHINTENSET10	7:0						SUSP	TCMPL	TERR
0xEE	CHINTFLAG10	7:0						SUSP	TCMPL	TERR
0xEF	CHSTATUS10	7:0					CRCERR	FERR	BUSY	PEND
0xF0	CHCTRLA11	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0xF4	CHCTRLB11	7:0							CMD[1:0]	
0xF5	CHPRILVL11	7:0								
0xF6	CHEVCTRL11	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0xF7 ... 0xFB	Reserved									
0xFC	CHINTENCLR11	7:0						SUSP	TCMPL	TERR
0xFD	CHINTENSET11	7:0						SUSP	TCMPL	TERR
0xFE	CHINTFLAG11	7:0						SUSP	TCMPL	TERR
0xFF	CHSTATUS11	7:0					CRCERR	FERR	BUSY	PEND

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0100	CHCTRLA12	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0x0104	CHCTRLB12	7:0							CMD[1:0]	
0x0105	CHPRILVL12	7:0								
0x0106	CHEVCTRL12	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0x0107 ... 0x010B	Reserved									
0x010C	CHINTENCLR12	7:0						SUSP	TCMPL	TERR
0x010D	CHINTENSET12	7:0						SUSP	TCMPL	TERR
0x010E	CHINTFLAG12	7:0						SUSP	TCMPL	TERR
0x010F	CHSTATUS12	7:0					CRCERR	FERR	BUSY	PEND
0x0110	CHCTRLA13	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0x0114	CHCTRLB13	7:0							CMD[1:0]	
0x0115	CHPRILVL13	7:0								
0x0116	CHEVCTRL13	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0x0117 ... 0x011B	Reserved									
0x011C	CHINTENCLR13	7:0						SUSP	TCMPL	TERR
0x011D	CHINTENSET13	7:0						SUSP	TCMPL	TERR
0x011E	CHINTFLAG13	7:0						SUSP	TCMPL	TERR
0x011F	CHSTATUS13	7:0					CRCERR	FERR	BUSY	PEND
0x0120	CHCTRLA14	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0x0124	CHCTRLB14	7:0							CMD[1:0]	
0x0125	CHPRILVL14	7:0								
0x0126	CHEVCTRL14	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0x0127 ... 0x012B	Reserved									
0x012C	CHINTENCLR14	7:0						SUSP	TCMPL	TERR
0x012D	CHINTENSET14	7:0						SUSP	TCMPL	TERR
0x012E	CHINTFLAG14	7:0						SUSP	TCMPL	TERR
0x012F	CHSTATUS14	7:0					CRCERR	FERR	BUSY	PEND
0x0130	CHCTRLA15	7:0		RUNSTDBY					ENABLE	SWRST
		15:8	TRIGSRC[7:0]							
		23:16			TRIGACT[1:0]					
		31:24			THRESHOLD[1:0]		BURSTLEN[3:0]			
0x0134	CHCTRLB15	7:0							CMD[1:0]	
0x0135	CHPRILVL15	7:0								
0x0136	CHEVCTRL15	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
0x0137 ... 0x013B	Reserved									
0x013C	CHINTENCLR15	7:0						SUSP	TCMPL	TERR
0x013D	CHINTENSET15	7:0						SUSP	TCMPL	TERR
0x013E	CHINTFLAG15	7:0						SUSP	TCMPL	TERR
0x013F	CHSTATUS15	7:0					CRCERR	FERR	BUSY	PEND

22.8 Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the “PAC Write-Protection” property in each individual register description. See *Register Access Protection* from Related Links.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

Related Links

22.5.7. Register Access Protection

22.8.1 Control

Name: CTRL
Offset: 0x00
Reset: 0x0000
Property: PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
					LVLENx3	LVLENx2	LVLENx1	LVLENx0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
							DMAENABLE	SWRST
Access							R/W	R/W
Reset							0	0

Bits 8, 9, 10, 11 – LVLENxx Priority Level x Enable
When this bit is set, all requests with the corresponding level will be fed into the arbiter block. When cleared, all requests with the corresponding level will be ignored.
For details on arbitration schemes, see *Arbitration* from Related Links.
These bits are not enable-protected.

Value	Description
0	Transfer requests for Priority level x will not be handled.
1	Transfer requests for Priority level x will be handled.

Bit 1 – DMAENABLE DMA Enable
Setting this bit will enable the DMA module.
Writing a '0' to this bit will disable the DMA module. When writing a '0' during an ongoing transfer, the bit will not be cleared until the internal data transfer buffer is empty and the DMA transfer is aborted. The internal data transfer buffer will be empty once the ongoing burst transfer is completed.
This bit is not enable-protected.

Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

Bit 0 – SWRST Software Reset
Writing a '0' to this bit has no effect.
Writing a '1' to this bit when the DMAC module is disabled (DMAENABLE bit set to '0'), resets all registers in the DMAC (except DBGCTRL) to their initial state. If either the DMAC or CRC module is enabled, the Reset request will be ignored and the DMAC will return an access error.

Value	Description
0	There is no Reset operation ongoing.
1	A Reset operation is ongoing.

Related Links
22.6.2.4. Arbitration

22.8.2 CRC Control

Name: CRCCTRL
Offset: 0x02
Reset: 0x0000
Property: PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
	CRCMODE[1:0]		CRCSRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
					CRCPOLY[1:0]		CRCBEATSIZE[1:0]	
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 15:14 – CRCMODE[1:0] CRC Operating Mode
These bits define the block transfer mode.

Value	Name	Description
0x0	DEFAULT	Default operating mode
0x1	-	Reserved
0x2	CRCMON	Memory CRC monitor operating mode
0x3	CRCGEN	Memory CRC generation operating mode

Bits 13:8 – CRCSRC[5:0] CRC Input Source
These bits select the input source for generating the CRC. The selected source is locked until either the CRC generation is completed or the CRC module is disabled. This means the CRCSRC cannot be modified when the CRC operation is ongoing. The lock is signaled by the CRCBUSY status bit. CRC generation complete is generated and signaled from the selected source when used with the DMA channel.

Value	Name	Description
0x00	DISABLE	No action
0x01	IO	I/O interface
0x02 – 0x1F	-	Reserved
0x20	CH0	DMA channel 0
0x21	CH1	DMA channel 1
0x22	CH2	DMA channel 2
0x23	CH3	DMA channel 3
0x24	CH4	DMA channel 4
0x25	CH5	DMA channel 5
0x26	CH6	DMA channel 6
0x27	CH7	DMA channel 7
0x28	CH8	DMA channel 8
0x29	CH9	DMA channel 9
0x2A	CH10	DMA channel 10
0x2B	CH11	DMA channel 11
0x2C	CH12	DMA channel 12
0x2D	CH13	DMA channel 13
0x2E	CH14	DMA channel 14
0x2F	CH15	DMA channel 15
0x30	CH16	DMA channel 16
0x31	CH17	DMA channel 17
0x32	CH18	DMA channel 18
0x33	CH19	DMA channel 19

Value	Name	Description
0x34	CH20	DMA channel 20
0x35	CH21	DMA channel 21
0x36	CH22	DMA channel 22
0x37	CH23	DMA channel 23
0x38	CH24	DMA channel 24
0x39	CH25	DMA channel 25
0x3A	CH26	DMA channel 26
0x3B	CH27	DMA channel 27
0x3C	CH28	DMA channel 28
0x3D	CH29	DMA channel 29
0x3E	CH30	DMA channel 30
0x3F	CH31	DMA channel 31

Bits 3:2 – CRCPOLY[1:0] CRC Polynomial Type

These bits select the CRC polynomial type.

Value	Name	Description
0x0	CRC16	CRC-16 (CRC-CCITT)
0x1	CRC32	CRC32 (IEEE 802.3)
0x2-0x3	-	Reserved

Bits 1:0 – CRCBEATSIZE[1:0] CRC Beat Size

These bits define the size of the data transfer for each bus access when the CRC is used with I/O interface.

Value	Name	Description
0x0	BYTE	8-bit bus transfer
0x1	HWORD	16-bit bus transfer
0x2	WORD	32-bit bus transfer
0x3	-	Reserved

22.8.3 CRC Data Input

Name: CRCDATAIN
Offset: 0x04
Reset: 0x00000000
Property: PAC Write Protection

Bit	31	30	29	28	27	26	25	24
	CRCDATAIN[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CRCDATAIN[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCDATAIN[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – CRCDATAIN[31:0] CRC Data Input
These bits store the data for which the CRC checksum is computed. A new CRC checksum is ready (CRCBEAT+ 1) clock cycles after the CRCDATAIN register is written.

22.8.4 CRC Checksum

Name: CRCCHKSUM
Offset: 0x08
Reset: 0x00000000
Property: PAC Write Protection, Enable-Protected

The CRCCHKSUM represents the 16- or 32-bit checksum value and the generated CRC. The register is reset to zero by default, but it is possible to reset all bits to one by writing the CRCCHKSUM register directly. It is possible to write this register only when the CRC module is disabled. If CRC-32 is selected and the CRC Status Busy flag is cleared (i.e., CRC generation is completed or aborted), the bit reversed (bit 31 is swapped with bit 0, bit 30 with bit 1, etc.) and complemented result will be read from CRCCHKSUM. If CRC-16 is selected or the CRC Status Busy flag is set (i.e., CRC generation is ongoing), CRCCHKSUM will contain the actual content.

Bit	31	30	29	28	27	26	25	24
	CRCCHKSUM[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CRCCHKSUM[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCCHKSUM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – CRCCHKSUM[31:0] CRC Checksum
These bits store the generated CRC result. The 16 MSB bits are always read zero when CRC-16 is enabled.

22.8.5 CRC Status

Name: CRCSTATUS
Offset: 0x0C
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
						CRCERR	CRCZERO	CRCBUSY
Access						R	R	R/W
Reset						0	0	0

Bit 2 – CRCERR CRC Error
This bit is read '1' when the memory CRC monitor detects data corruption.

Bit 1 – CRCZERO CRC Zero
This bit is cleared when a new CRC source is selected.
This bit is set when the CRC generation is complete and the CRC Checksum is zero.

Bit 0 – CRCBUSY CRC Module Busy
When used with an I/O interface (CRCCTRL.CRCSRC=0x1):

- This bit is cleared by writing a '1' to it
- This bit is set when the CRC Data Input (CRCDATAIN) register is written
- Writing a '1' to this bit will clear the CRC Module Busy bit
- Writing a '0' to this bit has no effect

When used with a DMA channel (CRCCTRL.CRCSRC=0x20..,0x3F):

- This bit is cleared when the corresponding DMA channel is disabled
- This bit is set when the corresponding DMA channel is enabled
- Writing a '1' to this bit has no effect
- Writing a '0' to this bit has no effect

Related Links
22.7. DMAC Register Summary

22.8.6 Debug Control

Name: DBGCTRL
Offset: 0x0D
Reset: 0x00
Property: PAC Write Protection

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN Debug Run
This bit is not reset by a Software Reset.
This bit controls the functionality when the CPU is halted by an external debugger.

Value	Description
0	The DMAC is halted when the CPU is halted by an external debugger.
1	The DMAC continues normal operation when the CPU is halted by an external debugger.

22.8.7 Software Trigger Control

Name: SWTRIGCTRL
Offset: 0x10
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	SWTRIGn[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:0 – SWTRIGn[15:0] Channel n Software Trigger [n = 15..0]

This bit is cleared when the Channel Pending bit in the Channel Status register (CHSTATUS.PEND) for the corresponding channel is either set, or by writing a '1' to it. See *CHSTATUS* in the *DMAC Register Summary* from Related Links.

This bit is set if CHSTATUS.PEND is already '1' when writing a '1' to that bit. See *CHSTATUS* in the *DMAC Register Summary* from Related Links.

Writing a '0' to this bit will clear the bit.

Writing a '1' to this bit will generate a DMA software trigger on channel x, if CHSTATUS.PEND=0 for channel x. CHSTATUS.PEND will be set and SWTRIGn will remain cleared. See *CHSTATUS* in the *DMAC Register Summary* from Related Links.

Related Links

22.7. DMAC Register Summary

22.8.8 Priority Control 0

Name: PRICTRL0
Offset: 0x14
Reset: 0x40404040
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	RRLVLEN3	QOS03[1:0]		LVLPRI3[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RRLVLEN2	QOS02[1:0]		LVLPRI2[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RRLVLEN1	QOS01[1:0]		LVLPRI1[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RRLVLEN0	QOS00[1:0]		LVLPRI0[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	0	0	0	0	0	0

Bits 7, 15, 23, 31 – RRLVLEN Level Round-Robin Scheduling Enable
For details on arbitration schemes, see *Arbitration* from Related Links.

Value	Description
0	Static arbitration scheme for channels with level 0 priority.
1	Round-robin arbitration scheme for channels with level 0 priority.

Bits 5:6, 13:14, 21:22, 29:30 – QOS Level Quality of Service

0x0	DISABLE Background (no sensitive operation)
0x1	LOW Sensitive to bandwidth
0x2	MEDIUM Sensitive to latency
0x3	Critical Latency

Bits 0:4, 8:12, 16:20, 24:28 – LVLPRI Level Channel Priority Number
When round-robin arbitration is enabled (PRICTRL0.RRLVLEN0=1) for priority level 0, this register holds the channel number of the last DMA channel being granted access as the active channel with priority level 0.
When static arbitration is enabled (PRICTRL0.RRLVLEN0=0) for priority level 0, and the value of this bit group is non-zero, it will not affect the static priority scheme.
This bit group is not reset when round-robin arbitration gets disabled (PRICTRL0.RRLVLEN0 written to '0').

Related Links
22.6.2.4. Arbitration

22.8.9 Interrupt Pending

Name: INTPEND
Offset: 0x20
Reset: 0x0000
Property: -

This register allows the user to identify the lowest DMA channel with pending interrupt.
An interrupt that handles several channels must consult the INTPEND register to find out which channel number has priority (ignoring/filtering each channel that has its own interrupt line). An interrupt dedicated to only one channel must not use the INTPEND register.

Bit	15	14	13	12	11	10	9	8
	PEND	BUSY	FERR	CRCERR		SUSP	TCMPL	TERR
Access	R	R	R	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

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

Bit 15 – PEND Pending
This bit will read '1' when the channel selected by Channel ID field (ID) is pending.

Bit 14 – BUSY Busy
This bit will read '1' when the channel selected by Channel ID field (ID) is busy.

Bit 13 – FERR Fetch Error
This bit will read '1' when the channel selected by Channel ID field (ID) fetched an invalid descriptor.

Bit 12 – CRCERR CRC Error
This bit will read '1' when the channel selected by Channel ID field (ID) has a CRC Error Status Flag bit set, and is set when the CRC monitor detects data corruption.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear it. It will also clear the corresponding flag in the Channel n Interrupt Flag Status and Clear register (CHINTFLAGn), where n is determined by the Channel ID bit field (ID).

Bit 10 – SUSP Channel Suspend
This bit will read '1' when the channel selected by Channel ID field (ID) has pending Suspend interrupt.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear it. It will also clear the corresponding flag in the Channel n Interrupt Flag Status and Clear register (CHINTFLAGn), where n is determined by the Channel ID bit field (ID).

Bit 9 – TCMPL Transfer Complete
This bit will read '1' when the channel selected by Channel ID field (ID) has pending Transfer Complete interrupt.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear it. It will also clear the corresponding flag in the Channel n Interrupt Flag Status and Clear register (CHINTFLAGn), where n is determined by the Channel ID bit field (ID).

Bit 8 – TERR Transfer Error
This bit will read '1' when the channel selected by Channel ID field (ID) has pending Transfer Error interrupt.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear it. It will also clear the corresponding flag in the Channel n Interrupt Flag Status and Clear register (CHINTFLAGn), where n is determined by the Channel ID bit field (ID).

Bits 4:0 – ID[4:0] Channel ID

These bits store the lowest channel number with pending interrupts. The number is valid if Suspend (SUSP), Transfer Complete (TCMPL) or Transfer Error (TERR) bits are set. The Channel ID field is refreshed when a new channel (with channel number less than the current one) with pending interrupts is detected, or when the application clears the corresponding channel interrupt sources. When no pending channels interrupts are available, these bits will always return zero value when read.

When the bits are written, indirect access to the corresponding Channel Interrupt Flag register is enabled.

22.8.10 Interrupt Status

Name: INTSTATUS
Offset: 0x24
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

22.8.11 Busy Channels

Name: BUSYCH
Offset: 0x28
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

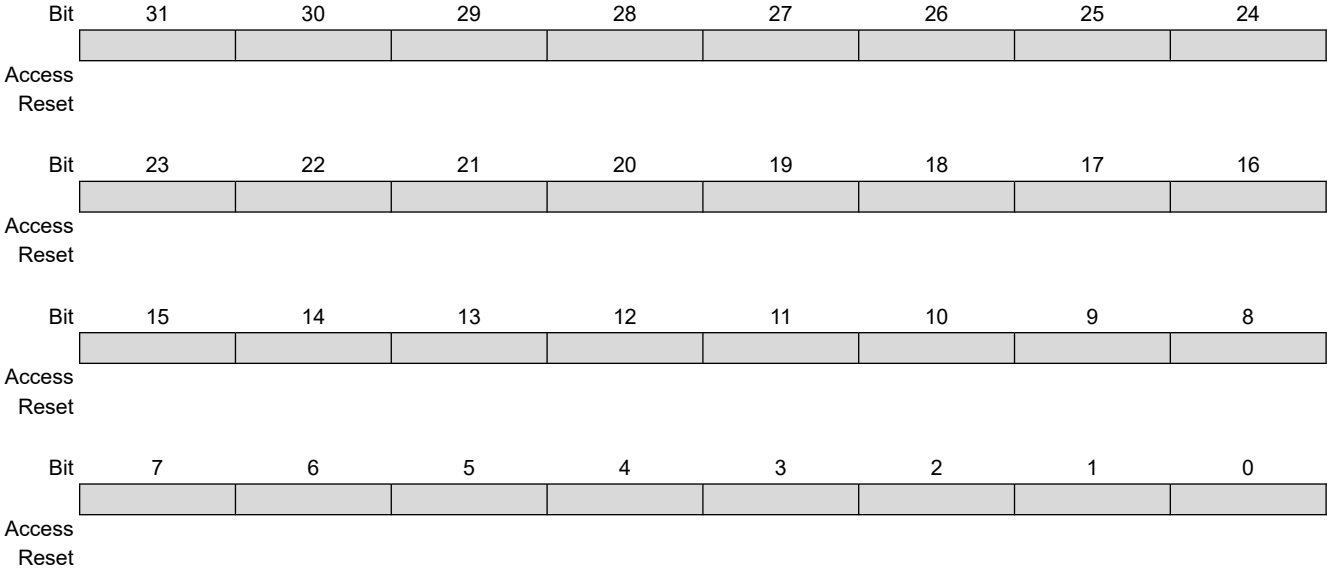
Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

22.8.12 Pending Channels

Name: PENDCH
Offset: 0x2C
Reset: 0x00000000
Property: -



22.8.13 Active Channel and Levels

Name: ACTIVE
Offset: 0x30
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	BTCNT[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BTCNT[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ABUSY							
Access	R							
Reset	0							

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 31:16 – BTCNT[15:0] Active Channel Block Transfer Count

These bits hold the 16-bit block transfer count of the ongoing transfer. This value is stored in the active channel and written back in the corresponding Write-Back channel memory location when the arbiter grants a new channel access. The value is valid only when the active channel Active Busy flag (ABUSY) is set.

Bit 15 – ABUSY Active Channel Busy

This bit is cleared when the active transfer count is written back in the write-back memory section.
This bit is set when the next descriptor transfer count is read from the write-back memory section.

22.8.14 Descriptor Memory Section Base Address

Name: BASEADDR
Offset: 0x34
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	BASEADDR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BASEADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BASEADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – BASEADDR[31:0] Descriptor Memory Base Address
These bits store the Descriptor memory section base address. The value must be 64-bit aligned.

22.8.15 Write-Back Memory Section Base Address

Name: WRBADDR
Offset: 0x38
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	WRBADDR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	WRBADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	WRBADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – WRBADDR[31:0] Write-Back Memory Base Address
These bits store the Write-Back memory base address. The value must be 64-bit aligned.

22.8.16 Channel Control A

Name: CHCTRLA
Offset: 0x40 + n*0x10 [n=0..15]
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
			THRESHOLD[1:0]		BURSTLEN[3:0]			
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

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

Bit	15	14	13	12	11	10	9	8
	TRIGSRC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RUNSTDBY					ENABLE	SWRST
Access		R/W					R/W	R/W
Reset		0					0	0

Bits 29:28 – THRESHOLD[1:0] FIFO Threshold
These bits define the threshold from which the DMA starts to write to the destination. These bits have no effect in the case of single beat transfers.
These bits are not enable-protected.

Value	Name	Description
0x0	1BEAT	Destination write starts after each beat source addess read
0x1	2BEATS	Destination write starts after 2-beats source address read
0x2	4BEATS	Destination write starts after 4-beats source address read
0x3	8BEATS	Destination write starts after 8-beats source address read

Bits 27:24 – BURSTLEN[3:0] Burst Length
These bits define the burst mode.
These bits are not enable-protected.

Value	Name	Description
0x0	SINGLE	Single-beat burst
0x1	2BEAT	2-beats burst length
0x2	3BEAT	3-beats burst length
0x3	4BEAT	4-beats burst length
0x4	5BEAT	5-beats burst length
0x5	6BEAT	6-beats burst length
0x6	7BEAT	7-beats burst length
0x7	8BEAT	8-beats burst length
0x8	9BEAT	9-beats burst length
0x9	10BEAT	10-beats burst length
0xA	11BEAT	11-beats burst length
0xB	12BEAT	12-beats burst length
0xC	13BEAT	13-beats burst length
0xD	14BEAT	14-beats burst length

Value	Name	Description
0xE	15BEAT	15-beats burst length
0xF	16BEAT	16-beats burst length

Bits 21:20 – TRIGACT[1:0] Trigger Action

These bits define the trigger action used for a transfer.
These bits are not enable-protected.

Value	Name	Description
0x0	BLOCK	One trigger required for each block transfer
0x1		Reserved
0x2	BURST	One trigger required for each burst transfer
0x3	TRANSACTION	One trigger required for each transaction

Bits 15:8 – TRIGSRC[7:0] Trigger Source

These bits define the peripheral that will be the source of a trigger.

Table 22-2. Triggers Map

Number	Name
0	Unused (Tied to 1'b0)
1	RTC_DMAC_ID_TIMESTAMP
2	DSU_DMAC_ID_DCC0
3	DSU_DMAC_ID_DCC1
4	SERCOM0_DMAC_ID_RX
5	SERCOM0_DMAC_ID_TX
6	SERCOM1_DMAC_ID_RX
7	SERCOM1_DMAC_ID_TX
8	SERCOM2_DMAC_ID_RX
9	SERCOM2_DMAC_ID_TX
10	SERCOM3_DMAC_ID_RX
11	SERCOM3_DMAC_ID_TX
12	TCC0_DMAC_ID_OVF
13	TCC0_DMAC_ID_MC_0
14	TCC0_DMAC_ID_MC_1
15	TCC0_DMAC_ID_MC_2
16	TCC0_DMAC_ID_MC_3
17	TCC0_DMAC_ID_MC_4
18	TCC0_DMAC_ID_MC_5
19	TCC1_DMAC_ID_OVF
20	TCC1_DMAC_ID_MC_0
21	TCC1_DMAC_ID_MC_1
22	TCC1_DMAC_ID_MC_2
23	TCC1_DMAC_ID_MC_3
24	TCC1_DMAC_ID_MC_4
25	TCC1_DMAC_ID_MC_5
26	TCC2_DMAC_ID_OVF
27	TCC2_DMAC_ID_MC_0
28	TCC2_DMAC_ID_MC_1
29	TC0_DMAC_ID_OVF
30	TC0_DMAC_ID_MC_0
31	TC0_DMAC_ID_MC_1
32	TC1_DMAC_ID_OVF
33	TC1_DMAC_ID_MC_0
34	TC1_DMAC_ID_MC_1
35	TC2_DMAC_ID_OVF
36	TC2_DMAC_ID_MC_0
37	TC2_DMAC_ID_MC_1

.....continued	
Number	**Name**
38	TC3_DMAC_ID_OVF
39	TC3_DMAC_ID_MC_0
40	TC3_DMAC_ID_MC_1
41	AES_DMAC_ID_WR
42	AES_DMAC_ID_RD
43	QSPI_DMAC_ID_RX
44	QSPI_DMAC_ID_TX

Bit 6 – RUNSTDBY Channel run in standby

This bit is used to keep the DMAC channel running in standby mode.
This bit is not enable-protected.

Value	Description
0	The DMAC channel is halted in standby.
1	The DMAC channel continues to run in standby.

Bit 1 – ENABLE Channel Enable

Writing a '0' to this bit during an ongoing transfer, the bit will not be cleared until the internal data transfer buffer is empty and the DMA transfer is aborted. The internal data transfer buffer will be empty once the ongoing burst transfer is completed.
Writing a '1' to this bit will enable the DMA channel.
This bit is not enable-protected.

Value	Description
0	DMA channel is disabled.
1	DMA channel is enabled.

Bit 0 – SWRST Channel Software Reset

Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets the channel registers to their initial state. The bit can be set when the channel is disabled (ENABLE=0). Writing a '1' to this bit will be ignored as long as ENABLE=1. This bit is automatically cleared when the reset is completed.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

22.8.17 Channel Control B

Name: CHCTRLB
Offset: 0x44 + n*0x10 [n=0..15]
Reset: 0x00
Property: PAC Write-Protection

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

Bits 1:0 – CMD[1:0] Software Command
These bits define the software commands. See *Channel Suspend* and *Channel Resume and Next Suspend Skip* from Related Links.
These bits are not enable-protected.

CMD[1:0]	Name	Description
0x0	NOACT	No action
0x1	SUSPEND	Channel suspend operation
0x2	RESUME	Channel resume operation
0x3	-	Reserved

Related Links
22.6.3.3. Channel Suspend
22.6.3.4. Channel Resume and Next Suspend Skip

22.8.18 Channel Priority Level

Name: CHPRILVL
Offset: 0x45 + n*0x10 [n=0..15]
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

22.8.19 Channel Event Control

Name: CHEVCTRL
Offset: 0x46 + n*0x10 [n=0..15]
Reset: 0x00
Property: PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]		
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit 7 – EVOE Channel Event Output Enable
This bit indicates if the Channel event generation is enabled. The event will be generated for every condition defined in the Channel Event Output Selection bits (CHEVCTRL.EVOMODE).

Value	Description
0	Channel event generation is disabled.
1	Channel event generation is enabled.

Bit 6 – EVIE Channel Event Input Enable

Value	Description
0	Channel event action will not be executed on any incoming event.
1	Channel event action will be executed on any incoming event.

Bits 5:4 – EVOMODE[1:0] Channel Event Output Mode
These bits define the channel event output selection. For more details on event output generation, see *Event Output Selection* from Related Links.

Value	Name	Description
0x0	DEFAULT	Block event output selection. See BTCTRL.EVOSEL for available selections.
0x1	TRIGACT	Ongoing trigger action
0x2-0x3		Reserved

Bits 2:0 – EVACT[2:0] Channel Event Input Action
These bits define the event input action. The action is executed only if the corresponding EVIE bit in the CHEVCTRL register of the channel is set. For more details on event actions, see *Event Input Actions* from Related Links. These bits are available only for channels with event input support.

Value	Name	Description
0x0	NOACT	No action
0x1	TRIG	Transfer and periodic transfer trigger
0x2	CTRIG	Conditional transfer trigger
0x3	CBLOCK	Conditional block transfer
0x4	SUSPEND	Channel suspend operation
0x5	RESUME	Channel resume operation
0x6	SSKIP	Skip next block suspend action
0x7	INCPRI	Increase priority

Related Links
22.6.3.5. Event Input Actions
22.6.3.6. Event Output Selection

22.8.20 Channel Interrupt Enable Clear

Name: CHINTENCLR
Offset: 0x4C + n*0x10 [n=0..15]
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Channel Interrupt Enable Set (CHINTENSET) register.

Bit	7	6	5	4	3	2	1	0
						SUSP	TCMPL	TERR
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – SUSP Channel Suspend Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Channel Suspend Interrupt Enable bit, which disables the Channel Suspend interrupt.

Value	Description
0	The Channel Suspend interrupt is disabled.
1	The Channel Suspend interrupt is enabled.

Bit 1 – TCMPL Channel Transfer Complete Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Channel Transfer Complete Interrupt Enable bit, which disables the Channel Transfer Complete interrupt.

Value	Description
0	The Channel Transfer Complete interrupt is disabled. When block action is set to none, the TCMPL flag will not be set when a block transfer is completed.
1	The Channel Transfer Complete interrupt is enabled.

Bit 0 – TERR Channel Transfer Error Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Channel Transfer Error Interrupt Enable bit, which disables the Channel Transfer Error interrupt.

Value	Description
0	The Channel Transfer Error interrupt is disabled.
1	The Channel Transfer Error interrupt is enabled.

22.8.21 Channel Interrupt Enable Set

Name: CHINTENSET
Offset: 0x4D + n*0x10 [n=0..15]
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Channel Interrupt Enable Clear (CHINTENCLR) register.

Bit	7	6	5	4	3	2	1	0
						SUSP	TCMPL	TERR
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – SUSP Channel Suspend Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Channel Suspend Interrupt Enable bit, which enables the Channel Suspend interrupt.

Value	Description
0	The Channel Suspend interrupt is disabled.
1	The Channel Suspend interrupt is enabled.

Bit 1 – TCMPL Channel Transfer Complete Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Channel Transfer Complete Interrupt Enable bit, which enables the Channel Transfer Complete interrupt.

Value	Description
0	The Channel Transfer Complete interrupt is disabled.
1	The Channel Transfer Complete interrupt is enabled.

Bit 0 – TERR Channel Transfer Error Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Channel Transfer Error Interrupt Enable bit, which enables the Channel Transfer Error interrupt.

Value	Description
0	The Channel Transfer Error interrupt is disabled.
1	The Channel Transfer Error interrupt is enabled.

22.8.22 Channel Interrupt Flag Status and Clear

Name: CHINTFLAG
Offset: 0x4E + n*0x10 [n=0..15]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						SUSP	TCMPL	TERR
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – SUSP Channel Suspend
This flag is cleared by writing a '1' to it.
This flag is set when a block transfer with suspend block action is completed, when a software suspend command is executed, when a suspend event is received or when an invalid descriptor is fetched by the DMA.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Channel Suspend interrupt flag for the corresponding channel.
For details on available software commands, see *CHCTRLB* in the *DMAC Register Summary* from Related Links.
For details on available event input actions, see *CHCTRLB* in the *DMAC Register Summary* from Related Links.
For details on available block actions, see *BTCTRL* in the *DMAC Register Summary (SRAM)* from Related Links.

Bit 1 – TCMPL Channel Transfer Complete
This flag is cleared by writing a '1' to it.
This flag is set when a block transfer is completed and the corresponding interrupt block action is enabled.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Transfer Complete interrupt flag for the corresponding channel.

Bit 0 – TERR Channel Transfer Error
This flag is cleared by writing a '1' to it.
This flag is set when a bus error is detected during a beat transfer or when the DMAC fetches an invalid descriptor.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Transfer Error interrupt flag for the corresponding channel.

Related Links
22.9. DMAC Register Summary (SRAM)
22.7. DMAC Register Summary

22.8.23 Channel Status

Name: CHSTATUS
Offset: 0x4F + n*0x10 [n=0..15]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CRCERR	FERR	BUSY	PEND
Access					R/W	R	R	R
Reset					0	0	0	0

Bit 3 – CRCERR Channel CRC Error
This bit is set when the CRC monitor detects data corruption. This bit is cleared by writing '1' to it, or by clearing the CRC Error bit in the INTPEND register (INTPEND.CRCERR). See *INTPEND* in the *DMAC Register Summary* from Related Links.

Bit 2 – FERR Channel Fetch Error
This bit is cleared when a software resume command is executed.
This bit is set when an invalid descriptor is fetched.

Bit 1 – BUSY Channel Busy
This bit is cleared when the channel trigger action is completed, when a bus error is detected or when the channel is disabled.
This bit is set when the DMA channel starts a DMA transfer.

Bit 0 – PEND Channel Pending
This bit is cleared when the channel trigger action is started, when a bus error is detected or when the channel is disabled. For details on trigger action settings, see *CHCTRLB* in the *DMAC Register Summary* from Related Links.
This bit is set when a transfer is pending on the DMA channel, as soon as the transfer request is received.

Related Links
22.7. DMAC Register Summary

22.9 DMAC Register Summary (SRAM)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	BTCTRL	7:0				BLOCKACT[1:0]		EVOSEL[1:0]		VALID
		15:8	STEPSIZE[2:0]			STEPSEL	DSTINC	SRCINC	BEATSIZE[1:0]	
0x02	BTCNT	7:0	BTCNT[7:0]							
		15:8	BTCNT[15:8]							
0x04	SRCADDR	7:0	SRCADDR[7:0]							
		15:8	SRCADDR[15:8]							
		23:16	SRCADDR[23:16]							
		31:24	SRCADDR[31:24]							
0x08	DSTADDR	7:0	DSTADDR[7:0]							
		15:8	DSTADDR[15:8]							
		23:16	DSTADDR[23:16]							
		31:24	DSTADDR[31:24]							
0x0C	DESCADDR	7:0	DESCADDR[7:0]							
		15:8	DESCADDR[15:8]							
		23:16	DESCADDR[23:16]							
		31:24	DESCADDR[31:24]							

22.10 Register Description - SRAM

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the “PAC Write-Protection” property in each individual register description. See *Register Access Protection* from Related Links.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

Related Links

22.5.7. Register Access Protection

22.10.1 Block Transfer Control

Name: BTCTRL
Offset: 0x00
Reset: 0x0000
Property: -

The BTCTRL register offset is relative to (BASEADDR or WRBADDR) + Channel Number * 0x10

Bit	15	14	13	12	11	10	9	8
	STEPSIZE[2:0]			STEPSEL	DSTINC	SRCINC	BEATSIZE[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				BLOCKACT[1:0]		EVOSEL[1:0]		VALID
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 15:13 – STEPSIZE[2:0] Address Increment Step Size
These bits select the address increment step size. The setting apply to source or destination address, depending on STEPSEL setting.

Value	Name	Description
0x0	X1	Next ADDR = ADDR + (Beat size in byte) * 1
0x1	X2	Next ADDR = ADDR + (Beat size in byte) * 2
0x2	X4	Next ADDR = ADDR + (Beat size in byte) * 4
0x3	X8	Next ADDR = ADDR + (Beat size in byte) * 8
0x4	X16	Next ADDR = ADDR + (Beat size in byte) * 16
0x5	X32	Next ADDR = ADDR + (Beat size in byte) * 32
0x6	X64	Next ADDR = ADDR + (Beat size in byte) * 64
0x7	X128	Next ADDR = ADDR + (Beat size in byte) * 128

Bit 12 – STEPSEL Step Selection
This bit selects if source or destination addresses are using the step size settings.

Value	Name	Description
0x0	DST	Step size settings apply to the destination address
0x1	SRC	Step size settings apply to the source address

Bit 11 – DSTINC Destination Address Increment Enable
Writing a '0' to this bit will disable the destination address incrementation. The address will be kept fixed during the data transfer.
Writing a '1' to this bit will enable the destination address incrementation. By default, the destination address is incremented by 1. If the STEPSEL bit is cleared, flexible step-size settings are available in the STEPSIZE register.

Value	Description
0	The Destination Address Increment is disabled
1	The Destination Address Increment is enabled

Bit 10 – SRCINC Source Address Increment Enable
Writing a '0' to this bit will disable the source address incrementation. The address will be kept fixed during the data transfer.
Writing a '1' to this bit will enable the source address incrementation. By default, the source address is incremented by 1. If the STEPSEL bit is set, flexible step-size settings are available in the STEPSIZE register.

Value	Description
0	The Source Address Increment is disabled
1	The Source Address Increment is enabled

Bits 9:8 – BEATSIZE[1:0] Beat Size

These bits define the size of one beat. A beat is the size of one data transfer bus access, and the setting apply to both read and write accesses.

Value	Name	Description
0x0	BYTE	8-bit bus transfer
0x1	HWORD	16-bit bus transfer
0x2	WORD	32-bit bus transfer
other		Reserved

Bits 4:3 – BLOCKACT[1:0] Block Action

These bits define what actions the DMAC must take after a block transfer has completed.

BLOCKACT[1:0]	Name	Description
0x0	NOACT	Channel will be disabled if it is the last block transfer in the transaction
0x1	INT	Channel will be disabled if it is the last block transfer in the transaction and block interrupt
0x2	SUSPEND	Channel suspend operation is completed
0x3	BOTH	Both channel suspend operation and block interrupt

Bits 2:1 – EVOSEL[1:0] Event Output Selection

These bits define the event output selection.

EVOSEL[1:0]	Name	Description
0x0	DISABLE	Event generation disabled
0x1	BLOCK	Event strobe when block transfer complete
0x2		Reserved
0x3	BEAT	Event strobe when beat transfer complete

Bit 0 – VALID Descriptor Valid

Writing a '0' to this bit in the Descriptor or Write-Back memory will suspend the DMA channel operation when fetching the corresponding descriptor.

The bit is automatically cleared in the Write-Back memory section when channel is aborted, when an error is detected during the block transfer, or when the block transfer is completed.

Value	Description
0	The descriptor is not valid
1	The descriptor is valid

22.10.2 Block Transfer Count

Name: BTCNT
Offset: 0x02
Property: -

The BTCNT register offset is relative to (BASEADDR or WRBADDR) + Channel Number * 0x10

Bit	15	14	13	12	11	10	9	8
	BTCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:0 – BTCNT[15:0] Block Transfer Count
This bit group holds the 16-bit block transfer count.
During a transfer, the internal counter value is decremented by one after each beat transfer. The internal counter is written to the corresponding write-back memory section for the DMA channel when the DMA channel loses priority, is suspended or gets disabled. The DMA channel can be disabled by a complete transfer, a transfer error or by software.

22.10.3 Block Transfer Source Address

Name: SRCADDR
Offset: 0x04
Property: -

The SRCADDR register offset is relative to (BASEADDR or WRBADDR) + Channel Number * 0x10

Bit	31	30	29	28	27	26	25	24
	SRCADDR[31:24]							
Access	-	-	-	-	-	-	-	-
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SRCADDR[23:16]							
Access	-	-	-	-	-	-	-	-
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SRCADDR[15:8]							
Access	-	-	-	-	-	-	-	-
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SRCADDR[7:0]							
Access	-	-	-	-	-	-	-	-
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – SRCADDR[31:0] Transfer Source Address

This bit field holds the block transfer source address.
When source address incrementation is disabled (BTCTRL.SRCINC=0), SRCADDR corresponds to the last beat transfer address in the block transfer.
When source address incrementation is enabled (BTCTRL.SRCINC=1), SRCADDR is calculated as follows:
If BTCTRL.STEPSEL = 1:

$$\mathrm{SRCADDR} = \mathrm{SRCADDR}_{START} + BTCNT \cdot \left(BEATSIZE + 1\right) \cdot 2^{\mathrm{STEPSIZE}}$$

If BTCTRL.STEPSEL= 0:

$$\mathrm{SRCADDR} = \mathrm{SRCADDR}_{START} + BTCNT \cdot (BEATSIZE + 1)$$

- $\mathrm{SRCADDR}_{START}$ is the source address of the first beat transfer in the block transfer
- BTCNT is the initial number of beats remaining in the block transfer
- BEATSIZE is the configured number of bytes in a beat
- STEPSIZE is the configured number of beats for each incrementation

22.10.4 Block Transfer Destination Address

Name: DSTADDR
Offset: 0x08
Property: -

The DSTADDR register offset is relative to (BASEADDR or WRBADDR) + Channel Number * 0x10

Bit	31	30	29	28	27	26	25	24
	DSTADDR[31:24]							
Access	-	-	-	-	-	-	-	-
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DSTADDR[23:16]							
Access	-	-	-	-	-	-	-	-
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DSTADDR[15:8]							
Access	-	-	-	-	-	-	-	-
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DSTADDR[7:0]							
Access	-	-	-	-	-	-	-	-
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DSTADDR[31:0] Transfer Destination Address
This bit field holds the block transfer destination address.
When destination address incrementation is disabled (BTCTRL.DSTINC = 0), DSTADDR corresponds to the last beat transfer address in the block transfer.
When destination address incrementation is enabled (BTCTRL.DSTINC = 1), DSTADDR is calculated as follows:
If BTCTRL.STEPSEL = 1:
$DSTADDR = DSTADDR_{START} + BTCNT \bullet (BEATSIZE + 1)$
If BTCTRL.STEPSEL = 0:
$DSTADDR = DSTADDR_{START} + BTCNT \bullet \left(BEATSIZE + 1\right) \bullet 2^{STEPSIZE}$

- $DSTADDR_{START}$ is the destination address of the first beat transfer in the block transfer
- BTCNT is the initial number of beats remaining in the block transfer
- BEATSIZE is the configured number of bytes in a beat
- STEPSIZE is the configured number of beats for each incrementation

22.10.5 Next Descriptor Address

Name: DESCADDR
Offset: 0x0C
Property: -

The DESCADDR register offset is relative to (BASEADDR or WRBADDR) + Channel Number * 0x10

Bit	31	30	29	28	27	26	25	24
	DESCADDR[31:24]							
Access	-	-	-	-	-	-	-	-
Reset								

Bit	23	22	21	20	19	18	17	16
	DESCADDR[23:16]							
Access	-	-	-	-	-	-	-	-
Reset								

Bit	15	14	13	12	11	10	9	8
	DESCADDR[15:8]							
Access	-	-	-	-	-	-	-	-
Reset								

Bit	7	6	5	4	3	2	1	0
	DESCADDR[7:0]							
Access	-	-	-	-	-	-	-	-
Reset								

Bits 31:0 – DESCADDR[31:0] Next Descriptor Address
This bit group holds the SRAM address of the next descriptor. The value must be 128-bit aligned. If the value of this SRAM register is 0x00000000, the transaction will be terminated when the DMAC tries to load the next transfer descriptor.

23. External Interrupt Controller (EIC)

23.1 Overview

The External Interrupt Controller (EIC) allows external pins to be configured as interrupt lines. Each interrupt line can be individually masked and can generate an interrupt on rising, falling, both edges, or on high or low levels. Each external pin has a configurable filter to remove spikes. Also, each external pin can be configured to be asynchronous in order to wake-up the device from Sleep modes where all clocks have been disabled. External pins can generate an event.

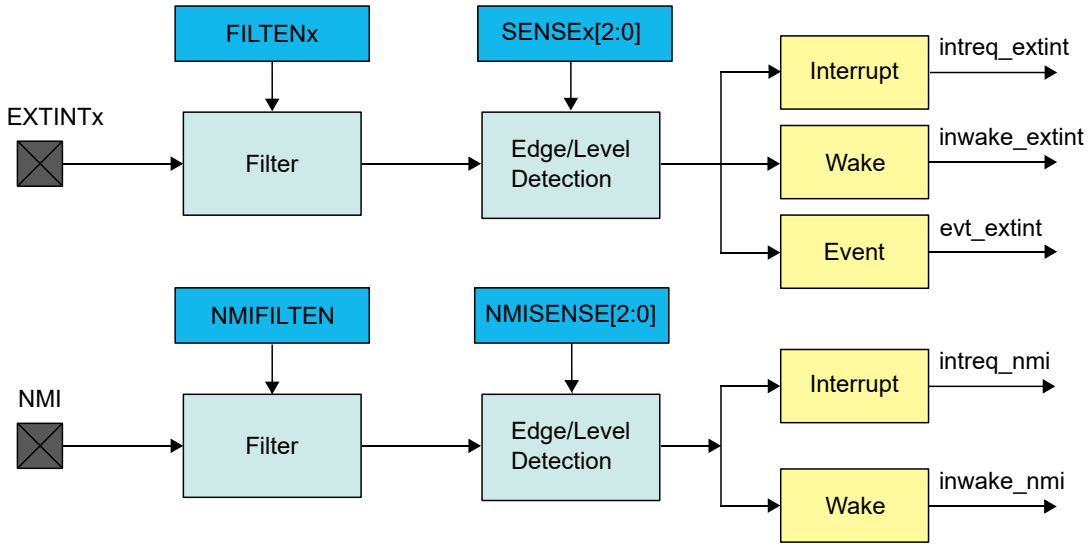
A separate Non-Maskable Interrupt (NMI) is supported. It has properties similar to the other external interrupts, but is connected to the NMI request of the CPU, enabling it to interrupt any other Interrupt mode.

23.2 Features

- Up to four external pins (EXTINTx), plus one non-maskable pin (NMI)
- Dedicated, Individually Maskable Interrupt for Each Pin
- Interrupt on Rising, Falling, or Both Edges
- Interrupt on High or Low Levels
- Asynchronous Interrupts for Sleep Modes Without Clock
- Filtering of External Pins
- Event Generation from EXTINTx

23.3 Block Diagram

Figure 23-1. EIC Block Diagram



23.4 Signal Description

Signal Name	Type	Description
EXTINT[3..0]	Digital Input	External interrupt pin

.....continued		
Signal Name	**Type**	**Description**
NMI	Digital Input	Non-maskable interrupt pin

One signal may be available on several pins.

23.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

23.5.1 I/O Lines

Using the EIC's I/O lines requires the I/O pins to be configured.

23.5.2 Power Management

All interrupts are available down to STANDBY Sleep mode, but the EIC can be configured to automatically mask some interrupts in order to prevent device wake-up.

The EIC will continue to operate in any Sleep mode where the selected source clock is running. The EIC's interrupts can be used to wake up the device from Sleep modes. Events connected to the Event System can trigger other operations in the system without exiting Sleep modes.

23.5.3 Clocks

The EIC bus clock (PB1_CLK) can be enabled and disabled by the CRU, the default state of PB1_CLK can be found in the CRU and PMD registers.

Some optional functions need a peripheral clock, which can either be a generic clock (GCLK_EIC, for wider frequency selection) or a Ultra Low-Power 32 KHz clock (CLK_ULP32K, for highest power efficiency). One of the clock sources must be configured and enabled before using the peripheral.

GCLK_EIC is configured and enabled in the CRU registers (see *Clock and Reset (CRU)* from Related Links).

CLK_ULP32K is provided by the internal Ultra Low-Power (OSCULP32K) Oscillator in the CRU module.

Both GCLK_EIC and CLK_ULP32K are asynchronous to the user interface clock (PB1_CLK). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains.

Related Links

13. Clock and Reset Unit (CRU)

23.5.4 DMA

Not applicable.

23.5.5 Interrupts

There are several interrupt request lines, some (the number depends on the product variant) for the external interrupts (EXTINT) and one for Non-Maskable Interrupt (NMI).

Each EXTINT interrupt request line is connected to the interrupt controller. Using the EIC interrupt requires the interrupt controller to be configured first.

The NMI interrupt request line is connected to the interrupt controller, but does not require the interrupt to be configured.

23.5.6 Events

The events are connected to the Event System. Using the events requires the Event System to be configured first.

Related Links

28. Event System (EVSYS)

23.5.7 Debug Operation

When the CPU is halted in Debug mode, the EIC continues normal operation. If the EIC is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

23.5.8 Register Access Protection

All registers with write access can be write-protected optionally by the Peripheral Access Controller (PAC), except for the following registers:

- Interrupt Flag Status and Clear register (INTFLAG)
- Non-Maskable Interrupt Flag Status and Clear register (NMIFLAG)

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

PAC write protection does not apply to accesses through an external debugger.

23.5.9 Analog Connections

Not applicable.

23.6 Functional Description

23.6.1 Principle of Operation

The EIC detects edge or level condition to generate interrupts to the CPU interrupt controller or events to the Event System. Each external interrupt pin (EXTINT) can be filtered using majority vote filtering, clocked by GCLK_EIC or by CLK_ULP32K.

23.6.2 Basic Operation

23.6.2.1 Initialization

The EIC must be initialized in the following order:

1. If required, configure the NMI by writing the Non-Maskable Interrupt Control register (NMICTRL).
2. Enable GCLK_EIC or CLK_ULP32K when one of the following configurations is selected:
   – The NMI uses edge detection or filtering
   – One EXTINT uses filtering
   – One EXTINT uses edge detection
   – One EXTINT uses debouncing

   GCLK_EIC is used when a frequency higher than 32 KHz is required for filtering.

   CLK_ULP32K is recommended when power consumption is the priority. For CLK_ULP32K, write a '1' to the Clock Selection bit in the Control A register (CTRLA.CKSEL).
3. Configure the EIC input sense and filtering by writing the Configuration register (CONFIG).
4. Enable the EIC by writing a '1' to CTRLA.ENABLE.

The following bits are enable-protected, meaning that it can only be written when the EIC is disabled (CTRLA.ENABLE=0):

- Clock Selection bit in Control A register (CTRLA.CKSEL)

The following registers are enable-protected:

- Event Control register (EVCTRL)
- Configuration register (CONFIG)
- External Interrupt Asynchronous Mode register (ASYNCH)
- Debouncer Enable register (DEBOUNCEN)
- Debounce Prescaler register (DPRESCALER)

Enable-protected bits in the CTRLA register can be written at the same time when setting CTRLA.ENABLE to '1', but not at the same time as CTRLA.ENABLE is being cleared.

Enable-protection is denoted by the "Enable-Protected" property in the register description.

See *NMICTRL*, *CTRLA*, *CONFIG*, *ASYNCH*, *DEBOUNCEN*, *DPRESCALER*, *EVCTRL* registers in the *EIC Register Summary* from Related Links.

Related Links

23.7. EIC Register Summary

23.6.2.2 Enabling, Disabling and Resetting

The EIC is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The EIC is disabled by writing CTRLA.ENABLE to '0'.

The EIC is reset by setting the Software Reset bit in the Control register (CTRLA.SWRST). All registers in the EIC will be reset to their initial state, and the EIC will be disabled.

23.6.3 External Pin Processing

Each external pin can be configured to generate an interrupt/event on edge detection (rising, falling or both edges) or level detection (high or low). The sense of external interrupt pins is configured by writing the Input Sense x bits in the Config n register (CONFIG.SENSEx). The corresponding interrupt flag (INTFLAG.EXTINT[x]) in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition is met.

When the interrupt flag has been cleared in edge-sensitive mode, INTFLAG.EXTINT[x] will only be set if a new interrupt condition is met.

In level-sensitive mode, when the interrupt has been cleared, INTFLAG.EXTINT[x] will be set immediately if the EXTINTx pin still matches the interrupt condition.

Each external pin can be filtered by a majority vote filtering, clocked by GCLK_EIC or CLK_ULP32K. Filtering is enabled if the bit Filter Enable x in the Configuration n register (CONFIG.FILTENx) is written to '1'. The majority vote filter samples the external pin three times with GCLK_EIC or CLK_ULP32K and outputs the value when two or more samples are equal.

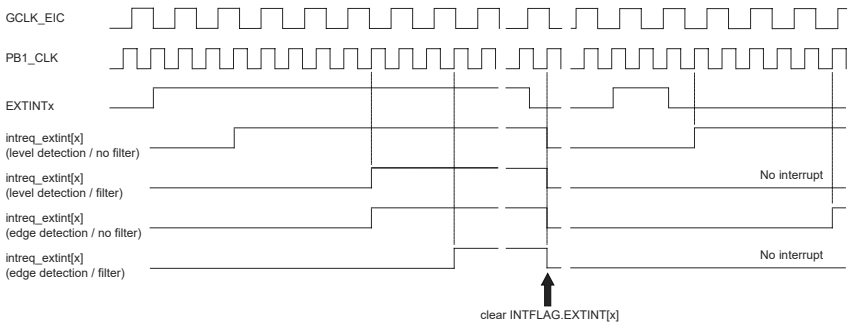
Table 23-1. Majority Vote Filter

Samples [0, 1, 2]	Filter Output
[0,0,0]	0
[0,0,1]	0
[0,1,0]	0
[0,1,1]	1
[1,0,0]	0
[1,0,1]	1
[1,1,0]	1
[1,1,1]	1

When an external interrupt is configured for level detection and when filtering is disabled, detection is done asynchronously. Level detection does not require GCLK_EIC or CLK_ULP32K, but interrupt and events can still be generated.

If filtering or edge detection is enabled, the EIC automatically requests GCLK_EIC or CLK_ULP32K to operate. The selection between these two clocks is done by writing the Clock Selection bits in the Control A register (CTRLA.CKSEL). GCLK_EIC must be enabled in the CRU. In these modes the external pin is sampled at the EIC clock rate, thus pulses with duration lower than two EIC clock periods may not be properly detected.

Figure 23-2. Interrupt Detection Latency by Modes (Rising Edge)



The detection latency depends on the detection mode.

Table 23-2. Detection Latency

Detection Mode	Latency (Worst Case)
Level without filter	Five PB1_CLK periods
Level with filter	Four GCLK_EIC/CLK_ULP32K periods + five PB1_CLK periods
Edge without filter	Four GCLK_EIC/CLK_ULP32K periods + five PB1_CLK periods
Edge with filter	Six GCLK_EIC/CLK_ULP32K periods + five PB1_CLK periods

23.6.4 Additional Features

23.6.4.1 Non-Maskable Interrupt (NMI)

The non-maskable interrupt pin can also generate an interrupt on edge or level detection, but it is configured with the dedicated NMI Control register (NMICTRL). To select the sense for NMI, write to the NMISENSE bit group in the NMI Control register (NMICTRL.NMISENSE). NMI filtering is enabled by writing a '1' to the NMI Filter Enable bit (NMICTRL.NMIFILTEN).

If edge detection or filtering is required, enable GCLK_EIC or CLK_ULP32K.

NMI detection is enabled only by the NMICTRL.NMISENSE value, and the EIC module is not required to be enabled.

When an NMI is detected, the Non-maskable Interrupt flag in the NMI Flag Status and Clear register is set (NMIFLAG.NMI). NMI interrupt generation is always enabled, and NMIFLAG.NMI generates an interrupt request when set.

23.6.4.2 Asynchronous Edge Detection Mode

The EXTINT edge detection operates synchronously or asynchronously, as selected by the Asynchronous Control Mode bit for external pin x in the External Interrupt Asynchronous Mode register (ASYNCH.ASYNCH[x]). The EIC edge detection is operated synchronously when the Asynchronous Control Mode bit (ASYNCH.ASYNCH[x]) is '0' (default value). It is operated asynchronously when ASYNCH.ASYNCH[x] is written to '1'.

In Synchronous Edge Detection Mode, the external interrupt (EXTINT) or the non-maskable interrupt (NMI) pins are sampled using the EIC clock as defined by the Clock Selection bit in the Control A register (CTRLA.CKSEL). The External Interrupt flag (INTFLAG.EXTINT[x]) or Non-Maskable Interrupt flag (NMIFLAG.NMI) is set when the last sampled state of the pin differs from the previously sampled state. The EIC clock is needed in this mode.

The Synchronous Edge Detection Mode can be used in Idle and Standby sleep modes.

In Asynchronous Edge Detection Mode, the external interrupt (EXTINT) pins or the non-maskable interrupt (NMI) pins set the External Interrupt flag or Non-Maskable Interrupt flag (INTFLAG.EXTINT[x] or NMIFLAG) directly. The EIC clock is not needed in this mode.

Note: The asynchronous edge detection mode can be used in Idle and Standby sleep modes.

23.6.4.3 Interrupt Pin Debouncing

The external interrupt pin (EXTINT) edge detection can use a debouncer to improve input noise immunity. When selected, the debouncer can work in the synchronous mode or the asynchronous mode, depending on the

configuration of the ASYNCH.ASYNCH[x] bit for the pin. The debouncer uses the EIC clock as defined by the bit CTRLA.CKSEL to clock the debouncing circuitry. The debouncing time frame is set with the debouncer prescaler DPRESCALER.PRESCALERn, which provides the *low frequency clock* tick that is used to reject higher frequency signals.

The debouncing mode for pin EXTINT x can be selected only if the Sense bits in the Configuration y register (CONFIG.SENSEx) are set to RISE, FALL or BOTH. If the debouncing mode for pin EXTINT x is selected, the filter mode for that pin (CONFIG.FILTENx) can not be selected.

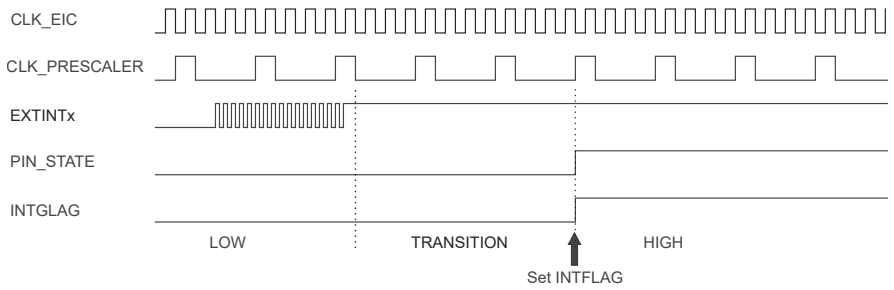
The debouncer manages an internal “valid pin state” that depends on the external interrupt (EXTINT) pin transitions, the debouncing mode and the debouncer prescaler frequency. The valid pin state reflects the pin value after debouncing. The external interrupt pin (EXTINT) is sampled continously on EIC clock. The sampled value is evaluated on each *low frequency clock* tick to detect a transitional edge when the sampled value is different of the current valid pin state. The sampled value is evaluated on each EIC clock when DPRESCALER.TICKON=0 or on each *low frequency clock* tick when DPRESCALER.TICKON=1, to detect a bounce when the sampled value is equal to the current valid pin state. Transitional edge detection increments the transition counter of the EXTINT pin, while bounce detection resets the transition counter. The transition counter must exceed the transition count threshold as defined by the DPRESCALER.STATESn bitfield. In the synchronous mode the threshold is 4 when DPRESCALER.STATESn=0 or 8 when DPRESCALER.STATESn=1. In the asynchronous mode the threshold is 4.

The valid pin state for the pins can be accessed by reading the register PINSTATE for both synchronous or asynchronous debouncing mode.

Synchronous edge detection In this mode the external interrupt (EXTINT) pin is sampled continously on EIC clock.

1. A pin edge transition will be validated when the sampled value is consistently different of the current valid pin state for 4 (or 8 depending on bit DPRESCALER.STATESn) consecutive ticks of the low frequency clock.
2. Any pin sample, at the *low frequency clock* tick rate, with a value opposite to the current valid pin state will increment the transition counter.
3. Any pin sample, at EIC clock rate (when DPRESCALER.TICKON=0) or the *low frequency clock* tick (when DPRESCALER.TICKON=1), with a value identical to the current valid pin state will return the transition counter to zero.
4. When the transition counter meets the count threshold, the pin edge transition is validated and the pin state PINSTATE.PINSTATE[x] is changed to the detected level.
5. The external interrupt flag (INTFLAG.EXTINT[x]) is set when the pin state PINSTATE.PINSTATE[x] is changed.

Figure 23-3. EXTINT Pin Synchronous Debouncing (Rising Edge)

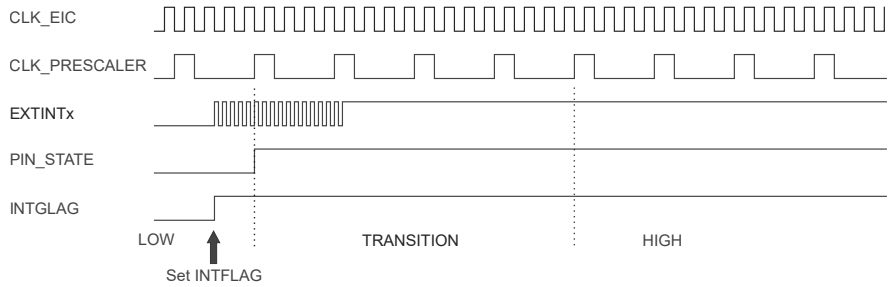


In the synchronous edge detection mode, the EIC clock is required. The synchronous edge detection mode can be used in Idle and Standby sleep modes.

Asynchronous edge detection In this mode, the external interrupt (EXTINT) pin directly drives an asynchronous edges detector which triggers any rising or falling edge on the pin:

1. Any edge detected that indicates a transition from the current valid pin state will immediately set the valid pin state PINSTATE.PINSTATE[x] to the detected level.
2. The external interrupt flag (INTFLAG.EXTINT[x] is immediately changed.
3. The edge detector will then be idle until no other rising or falling edge transition is detected during 4 consecutive ticks of the low frequency clock.
4. Any rising or falling edge transition detected during the idle state will return the transition counter to 0.
5. After 4 consecutive ticks of the low frequency clock without bounce detected, the edge detector is ready for a new detection.

Figure 23-4. EXTINT Pin Asynchronous Debouncing (Rising Edge)



In this mode, the EIC clock is requested. The asynchronous edge detection mode can be used in Idle and Standby sleep modes.

23.6.5 DMA Operation

Not applicable.

23.6.6 Interrupts

The EIC has the following interrupt sources:

- External interrupt (EXTINTx) pins. See *Basic Operation* from Related Links.
- Non-maskable interrupt (NMI) pin. See *Non-Maskable Interrupt (NMI)* from Related Links

Each interrupt source has an associated Interrupt flag. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when an Interrupt condition occurs (NMIFLAG for NMI). Each interrupt, except NMI, can be individually enabled by setting the corresponding bit in the Interrupt Enable Set register (INTENSET = 1), and disabled by setting the corresponding bit in the Interrupt Enable Clear register (INTENCLR = 1). The status of enabled interrupts can be read from either INTENSET or INTENCLR.

An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled, or the EIC is reset. See the INTFLAG register for details on how to clear Interrupt flags. The EIC has one interrupt request line for each external interrupt (EXTINTx) and one line for NMI. The user must read the INTFLAG (or NMIFLAG) register to determine which Interrupt condition is present.

Notes:

1. Interrupts must be globally enabled for interrupt requests to be generated.
2. If an external interrupt (EXTINT) is common on two or more I/O pins, only one will be active (the first one programmed).

Related Links

23.6.2. Basic Operation

23.6.4.1. Non-Maskable Interrupt (NMI)

23.6.7 Events

The EIC can generate the following output events:

- External event from pin (EXTINT0-3)

Setting an Event Output Control register (EVCTRL.EXTINTEO) enables the corresponding output event. Clearing this bit disables the corresponding output event. For more details on configuring the event system, see *Event System (EVSYS)* from Related Links.

When the condition on pin EXTINTx matches the configuration in the CONFIG register, the corresponding event is generated, if enabled.

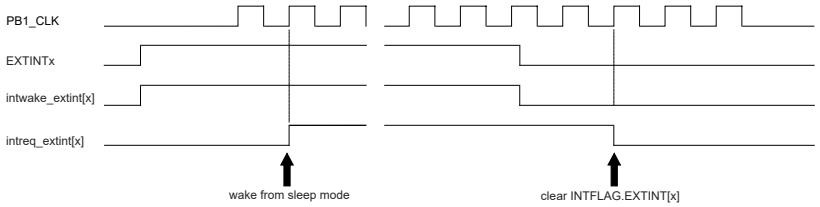
Related Links

28. Event System (EVSYS)

23.6.8 Sleep Mode Operation

In sleep modes, an EXTINTx pin can wake up the device if the corresponding condition matches the configuration in the CONFIG register, and the corresponding bit in the Interrupt Enable Set register (INTENSET) is written to '1'.

Figure 23-5. Wake-up Operation Example (High-Level Detection, No Filter, Interrupt Enable Set)



23.6.9 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset bit in control register (CTRLA.SWRST)
- Enable bit in control register (CTRLA.ENABLE)

Required write synchronization is denoted by the "Write-Synchronized" property in the register description.

See *CTRLA* from Related Links.

Related Links

23.8.1. CTRLA

23.7 EIC Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0				CKSEL			ENABLE	SWRST
0x01	NMICTRL	7:0				NMIASYNCH	NMIFILTEN	NMISENSE[2:0]		
0x02	NMIFLAG	7:0								NMI
0x03	Reserved									
0x04	SYNCBUSY	7:0							ENABLE	SWRST
		15:8								
		23:16								
		31:24								
0x08	EVCTRL	7:0								
		15:8								
		23:16								
		31:24								
0x0C	INTENCLR	7:0								
		15:8								
		23:16								
		31:24								
0x10	INTENSET	7:0								
		15:8								
		23:16								
		31:24								
0x14	INTFLAG	7:0								
		15:8								
		23:16								
		31:24								
0x18	ASYNCH	7:0								
		15:8								
		23:16								
		31:24								
0x1C	CONFIG	7:0	FILTEN1	SENSE1[2:0]			FILTEN0	SENSE0[2:0]		
		15:8	FILTEN3	SENSE3[2:0]			FILTEN2	SENSE2[2:0]		
		23:16								
		31:24								
0x20 ... 0x2F	Reserved									
0x30	DEBOUNCEN	7:0								
		15:8								
		23:16								
		31:24								
0x34	DPRESCALER	7:0								
		15:8								
		23:16								TICKON
		31:24								
0x38	PINSTATE	7:0								
		15:8								
		23:16								
		31:24								

23.8 Register Description

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

Some registers require synchronization when read and/or written. Synchronization is denoted by the “Read-Synchronized” and/or “Write-Synchronized” property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the “Enable-Protected” property in each individual register description.

23.8.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
				CKSEL			ENABLE	SWRST
Access				RW			RW	W
Reset				0			0	0

Bit 4 – CKSEL Clock Selection
The EIC can be clocked either by GCLK_EIC (when a frequency higher than 32.768 KHz is required for filtering) or by CLK_ULP32K (when power consumption is the priority).
This bit is not Write-Synchronized.

Value	Description
0	The EIC is clocked by GCLK_EIC.
1	The EIC is clocked by CLK_ULP32K.

Bit 1 – ENABLE Enable
Due to synchronization there is a delay between writing to CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the Enable bit in the Synchronization Busy register will be set (SYNCBUSY.ENABLE=1). SYNCBUSY.ENABLE will be cleared when the operation is complete.
This bit is not Enable-Protected.
This bit is Write-Synchronized.

Value	Description
0	The EIC is disabled.
1	The EIC is enabled.

Bit 0 – SWRST Software Reset
Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the EIC to their initial state, and the EIC will be disabled.
Writing a '1' to CTRLA.SWRST will always take precedence, meaning that all other writes in the same write operation will be discarded.
Due to synchronization there is a delay from writing CTRLA.SWRST until the Reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the Reset is complete.
This bit is not Enable-Protected.
This bit is Write-Synchronized.

Value	Description
0	There is no ongoing reset operation.
1	The reset operation is ongoing.

23.8.2 Non-Maskable Interrupt Control

Name: NMICTRL
Offset: 0x01
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
				NMIASYNCH	NMIFILTEN	NMISENSE[2:0]		
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit 4 – NMIASYNCH NMI Asynchronous Edge Detection Mode
The NMI edge detection can be operated synchronously or asynchronously to the EIC clock.

Value	Description
0	The NMI edge detection is synchronously operated.
1	The NMI edge detection is asynchronously operated.

Bit 3 – NMIFILTEN Non-Maskable Interrupt Filter Enable

Value	Description
0	NMI filter is disabled.
1	NMI filter is enabled.

Bits 2:0 – NMISENSE[2:0] Non-Maskable Interrupt Sense Configuration
These bits define on which edge or level the NMI triggers.

Value	Name	Description
0x0	NONE	No detection
0x1	RISE	Rising-edge detection
0x2	FALL	Falling-edge detection
0x3	BOTH	Both-edge detection
0x4	HIGH	High-level detection
0x5	LOW	Low-level detection
0x6 - 0x7	-	Reserved

23.8.3 Non-Maskable Interrupt Flag Status and Clear

Name: NMIFLAG
Offset: 0x2
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
								NMI
Access								RW
Reset								0

Bit 0 – NMI Non-Maskable Interrupt
This flag is cleared by writing a '1' to it.
This flag is set when the NMI pin matches the NMI sense configuration, and will generate an interrupt request.
Writing a '0' to this bit has no effect.

23.8.4 Synchronization Busy

Name: SYNCBUSY
Offset: 0x04
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							ENABLE	SWRST
Access							R	R
Reset							0	0

Bit 1 – ENABLE Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.ENABLE bit is complete.
1	Write synchronization for CTRLA.ENABLE bit is ongoing.

Bit 0 – SWRST Software Reset Synchronization Busy Status

Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	Write synchronization for CTRLA.SWRST bit is complete.
1	Write synchronization for CTRLA.SWRST bit is ongoing.

23.8.5 Event Control

Name: EVCTRL
Offset: 0x08
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

23.8.6 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x0C
Reset: 0x00000000
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

23.8.7 Interrupt Enable Set

Name: INTENSET
Offset: 0x10
Reset: 0x00000000
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

23.8.8 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x14
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

23.8.9 External Interrupt Asynchronous Mode

Name: ASYNCH
Offset: 0x18
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

23.8.10 External Interrupt Sense Configuration

Name: CONFIG
Offset: 0x1C
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	FILTEN3	SENSE3[2:0]			FILTEN2	SENSE2[2:0]		
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FILTEN1	SENSE1[2:0]			FILTEN0	SENSE0[2:0]		
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bits 3, 7, 11, 15 – FILTENx Filter Enable x [x=3..0]

Note: The filter must be disabled if the asynchronous detection is enabled.

Value	Description
0	Filter is disabled for EXTINT[x] input.
1	Filter is enabled for EXTINT[x] input.

Bits 0:2, 4:6, 8:10, 12:14 – SENSEx Input Sense Configuration x [x=3..0]

These bits define on which edge or level the interrupt or event for EXTINT[x] will be generated.

Value	Name	Description
0x0	NONE	No detection
0x1	RISE	Rising-edge detection
0x2	FALL	Falling-edge detection
0x3	BOTH	Both-edge detection
0x4	HIGH	High-level detection
0x5	LOW	Low-level detection
0x6 – 0x7	-	Reserved

23.8.11 Debouncer Enable

Name: DEBOUNCEN
Offset: 0x30
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

23.8.12 Debouncer Prescaler

Name: DPRESCALER
Offset: 0x34
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
								TICKON
Access								RW
Reset								0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

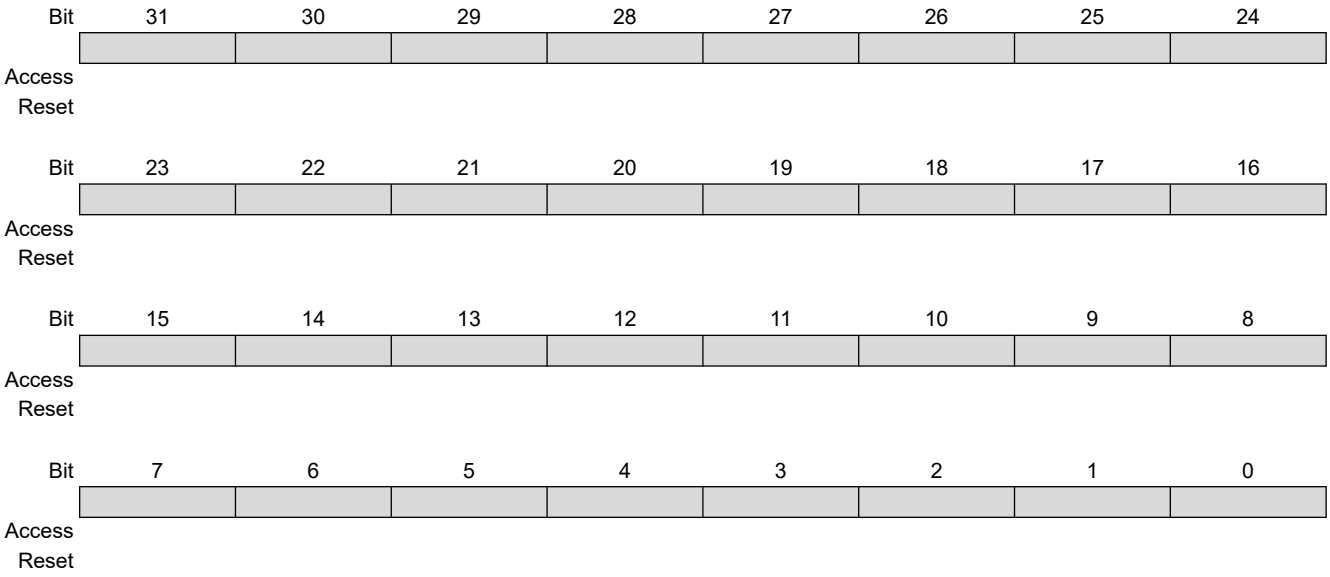
Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 16 – TICKON Pin Sampler frequency selection
This bit selects the clock used for the sampling of bounce during transition detection.

Value	Description
0	The bounce sampler is using GCLK_EIC.
1	The bounce sampler is using the low frequency clock.

23.8.13 Pin State

Name: PINSTATE
Offset: 0x38
Reset: 0x00000000



24. Flash Memory

24.1 Overview

The PIC32CX-BZ2 devices contain a single bank of Flash memory with their Program Flash Memory (PFM) partition and Boot Flash Memory (BFM) partition for storing user code or non-volatile data. The Flash controller is used to access the Flash memory. The peripheral bus interface is used for commands and configuration of the Flash controller.

24.2 Features

Flash Controller

- PB-Bridge-D interface that provides access to the Flash controller registers
- AHB Initiator for bus hosted reads the row programming data from SRAM
- Write Protect for Program Flash (PFM)
  - Single page protection resolution
  - Protect “Less Than” Address
  - Protect “Greater Than or Equal to” Address
- Individual page write protection for boot Flash (BFM)
- Error-correction code (ECC) support
- Supports chip and page erase
- Supports Single Word, Quad Word and row program options
- Supports flash Erase/Retry to increase Retention and Endurance

Flash Memory

- 128-bit wide Flash Memory Access
- 4 Kbytes page size
- Row size is 1 KB (256 IW)
- Flash-based OTP (one-time-programmable) page

The Flash controller allows the Flash memory to be accessed through the following methods:

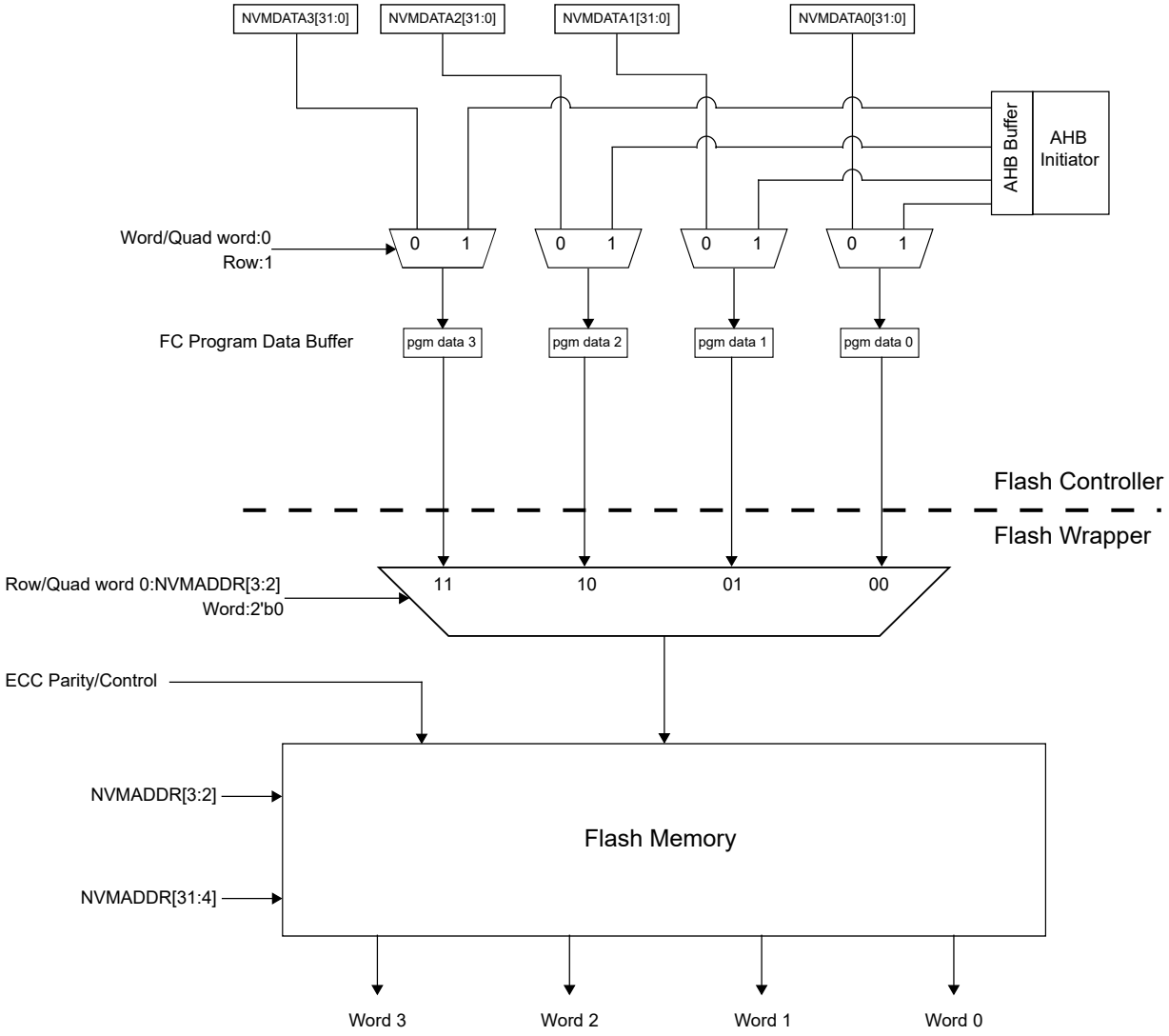
1. Run-Time Self-Programming (RTSP)
2. Serial Wire Debug (SWD) programming using DSU (See *Device Service Unit (DSU)* from Related Links and *PIC32CX-BZ2 Programming Specification*.)

Related Links

12. Device Service Unit (DSU)

24.3 Functional Block Diagram

Figure 24-1. Flash Memory Block Diagram



24.4 Flash Memory Addressing

Flash memory addressing uses physical addresses only. For more information on addressing, see *Product Memory Mapping Overview* from Related Links.

Related Links

8. Product Memory Mapping Overview

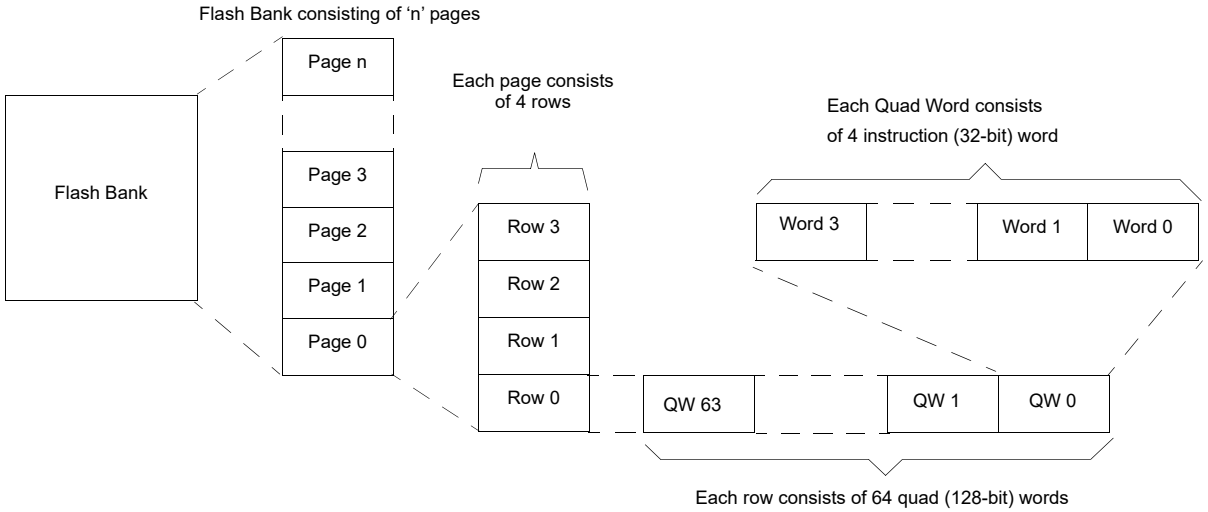
24.5 Memory Configuration

24.5.1 Flash Memory Construction

Flash memory is divided into pages. A page is the smallest unit of memory that can be erased at one time. Each page of memory is segmented into four rows. A row is the largest unit of memory that can be programmed at one

time. A row consists of 64 Quad (128-bit) Word. Each Quad Word consists of a four instruction (32-bit) Word. Flash memory can be programmed in rows, Quad Word (128-bit) or Single Word (32-bit) units.

Figure 24-2. Flash Construction



24.5.2 Flash Memory Organization

The Device Flash memory is divided into two logical Flash partitions:

1. Main Program Flash Memory (PFM)
2. Boot/Configuration Flash Memory (BFM)
   a. Boot Flash
   b. Device/Boot Configuration – Device and boot configuration data
   c. OTP (One Time Programmable) – User system calibration data

Each Flash section has a different protection status; refer to the following table.

Table 24-1. Protection Status

Flash Partition	Memory Region	Write Protection	Erase Protection	Chip Erase through DSU
BFM	Boot Flash	Yes. Page-wise Configurable	Yes. Page-wise Configurable	Erased
	Device/Boot Configuration	Yes. Configurable	Yes. Configurable	Erased
	OTP (One-Time-Programmable)	Yes. Configurable	Always Erase protected. Can not be erased	Not Erased
PFM	Program Flash	Yes. Configurable	Yes. Configurable	Erased

24.6 Boot Flash Memory (BFM) Partitions

24.6.1 BFM Write Protection

Pages in the BFM regions can be protected individually using bits in the NVMLBWP register. At Reset, all pages are in a write-protected state and must be disabled prior to performing any programming operations on the BFM regions. There is also an unlock bit, ULOCK(NVMLBWP[31]), that is set at Reset and can be cleared by the user software. When cleared, changes to write protection for that region can no longer be made. Once cleared, the ULOCK bit can only be set by a Reset.

The NVMLBWP write-protect register can only be changed when the unlock sequence is followed. See *NVMKEY Register Unlocking Sequence* from Related Links.

Related Links

24.11. NVMKEY Register Unlocking Sequence

24.7 Program Flash Memory (PFM) Partitions

24.7.1 PFM Write Protection

Write protection for the PFM region is implemented by pages, defined by the NVMPWPLT and NVMPWPGTE registers. The NVMPWP* registers define an area within the program space (PFM) that is write-protected. This write-protected address resolves to Flash page boundaries; therefore, the 12 LSBs for a 4 KB page Flash of any address written to the NVMPWP* registers are ignored. The width of each NVMPWP* address register is determined by the size of the Flash. The NVMPWPLT register is used to set the Program Flash pages lower than the provided address as write-protected. The NVMPWPGTE register is used to set the Program Flash pages greater than or equal to the provided address as write-protected. Therefore, a value of all `0`s in the NVMPWPLT register and all `1`s in the NVMPWPGTE register results in no region of Flash being write-protected (default state at Reset).

There is also an unlock bit, ULOCK (NVMPWPLT [31] and NVMPWPGTE[31]), that is set at Reset and can be cleared by the user software. When cleared, changes to the write-protection of the PFM can no longer be made, including the ULOCK bit. The NVMPWPLT and NVMPWPGTE write-protected register can only be changed when the unlock sequence is followed. See *NVMKEY Register Unlocking Sequence* from Related Links.

Related Links

24.11. NVMKEY Register Unlocking Sequence

24.8 Error Correcting Code (ECC) and Flash Programming

The PIC32CX-BZ2 devices incorporate Error Correcting Code (ECC) features that detect and correct errors resulting in extended Flash memory life. For more details on this feature, see *Prefetch Cache* from Related Links.

ECC is implemented in 128-bit Quad Flash Words or 32-bit Single Word. As a result, when programming Flash memory on a device where ECC is employed, the programming operation must be, at minimum, four instruction Words or in groups of four instruction Words. This is the reason that the Quad Word programming command exists and why row programming always programs multiples of four Words.

For a given software application, ECC can be enabled at all times, disabled at all times or dynamically enabled using the ECCCTL Configuration bits in the CFGCON0 register. When ECC is enabled at all times, the Single Word NVMOP programming command does not function and the Quad Word is the smallest unit of memory that can be programmed. When ECC is disabled or enabled dynamically, both the Single Word and Quad Word programming NVMOP commands are functional and the programming method used determines how ECC is handled.

In the case of dynamic ECC, if the memory was programmed with the Singe Word command, ECC is turned off for that Word, and, when it is read, no error correction is performed. If the memory was programmed with the Quad Word or Row Programming commands, ECC data is written and tested for errors (and corrected if needed) when read. The following table describes the different ECC scenarios.

Table 24-2. ECC Programming Summary

ECCCTL Setting	Programming Operation			Data Read
	Single Word Write	Quad Word Write	Row Write	
Disabled	Allowed	Allowed	Allowed	ECC is never applied on a Flash read
Enabled	Not allowed	Allowed	Allowed	ECC is applied on every Flash Word read

.....continued				
ECCCTL Setting	Programming Operation			Data Read
	Single Word Write	Quad Word Write	Row Write	
Dynamic	Allowed but when used, the programmed word is flagged to NOT USE ECC	Writes ECC data and flags programmed words to USE ECC	Writes ECC data and flags programmed words to USE ECC	ECC is only applied on words that are flagged to USE ECC

Note: When using dynamic ECC, all non-ECC locations must be programmed with the 32-bit Word programming command, while all ECC-enabled locations must be programmed with a 128-bit Quad Word or Row programming command. Divisions between ECC and non-ECC memory must be on even Quad Word boundaries (address bits 0 through 3 are equal to '0').

Related Links

9. Prefetch Cache (PCHE)

24.9 Interrupts

An interrupt is generated when the WR bit is cleared by the Flash Controller upon completion of a Flash program or erase operation. The interrupt event will cause a CPU interrupt if it was configured and enabled in the Nested Interrupt Vector Controller. See *Nested Vector Interrupt Controller (NVIC)* from Related Links for the vector mapping table. The interrupt occurs regardless of the outcome of the program or erase operation, successful or unsuccessful. The only exception is the No Operation (NOP) programming operation (NVMOP = 0), which is used to manually clear the error flags and does not create an interrupt event on completion but does clear the WR bit.

The Flash Controller interrupts are not persistent, and, therefore, no additional steps are required to clear the cause or source of the interrupt.

Once the Interrupt Controller is configured, the Flash event causes the CPU to jump to the vector assigned to the Flash event. The CPU starts executing the code at the vector address. The user software at this vector address must perform the required operations and, then, exit.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

24.9.1 Interrupts and CPU Stalling

Code cannot be fetched by the CPU from the same Flash bank, either BFM or PFM, that is the target of the programming operation. When this operation is attempted, the CPU will cease to execute code (stall) while the programming operation is in progress. CPU code execution does not resume until the programming operation is complete, and, when this occurs, any pending interrupts, including those from the Flash Controller, will be processed in order of priority.

Note: Code that is already loaded into the processor cache will continue to execute up to the point where an attempt is made to fetch code or data from the same Flash bank as the active programming operation. At this point the CPU will stall.

The stalling of the CPU can also be avoided by placing any needed executable code in SRAM during Flash programming.

24.10 Error Detection

The NVMCON register includes two bits for detecting error conditions during a program or erase operation. They are Low-Voltage detect error, LVDERR bit (NVMCON[12]), and Write Error, WRERR bit (NVMCON[13]).

The WRERR is set each time the WR bit (NVMCON[15]) is set, initiating a programming operation. When the Flash operation is complete, indicated by hardware clearing the value of the WR bit (i.e., WR bit is set to '0'), hardware will update the value in the WRERR bit to indicate if an error occurred. Firmware must check the value of the WR bit to

see if the Flash operation completed before checking the value of the WRERR bit. When the WRERR bit is set, any future attempt to initiate programming or erase operation is ignored. WRERR must be cleared before commencing Flash program or erase operations.

The LVDERR bit is set when a Brown-out Reset (BOR) occurs during a programming operation. The only Reset that clears the LVDERR bit is a Power-on Reset (POR). Other Reset types do not affect the LVDERR bit. When the LVDERR bit is set, any attempt to initiate programming or erase operation is ignored. The LVDERR bit must be cleared before commencing Flash program or erase operations.

Both the WRERR and LVDERR bits must be cleared manually in software by initiating a Flash operation (setting WR) referred to as NOP (0x00) (see the NVMOP bit fields).
Note: Executing the NVMOP NOP command clears WRERR, LVDERR and WR bits, but does not generate an interrupt event on completion.

Table 24-3. Programming Error Cause and Effects

Cause of Error	Effect on Programming Erase Operation	Indication
A low-voltage event occurred during a programming sequence.	The last programming or erase operation may not have completed.	LVDERR = 1, WRERR = 1
A non-POR Reset occurred during programming.	Programming or erase operation is aborted.	WRERR = 1
Attempt to program or erase a page out of the Flash memory range.	Erase or programming operation is not initiated.	WRERR = 1
Attempt to erase or program a write-protected PFM page.	Erase or programming operation is not initiated.	WRERR = 1
Attempt to erase or program a write-protected BFM page.	Operation occurs, but the page is not programmed or erased.	WRERR = 0
Bus host error or row programming data underrun error during programming.	Programming or erase operation is aborted.	WRERR = 1

24.11 NVMKEY Register Unlocking Sequence

Important register settings that could compromise the Flash memory if inadvertently changed are protected by a register unlocking sequence. This feature is implemented using the NVMKEY register. The NVMKEY register is a write-only register that is used to implement an unlock sequence to help prevent accidental writes or erasures of Flash memory.

In some instances, the operation is also dependent on the setting of the WREN bit (NVMCON[14]), as shown in the following table.

Table 24-4. NVMKEY Register Unlocking and WREN

Operation	WREN Setting	Unlock Sequence Required
Changing value of NVMOP[3:0] (NVMCON[3:0])	0	No
Setting WR (NVMCON[15]) to start a write or erase operation	1	Yes
Changing any fields in the NVMPWP* register	—	Yes
Changing any fields in the NVMLBWP register	—	Yes

The following steps must be followed in the exact order as shown to enable writes to registers that require this unlock sequence:

1. Write 0x00000000 to NVMKEY.
2. Write 0xAA996655 to NVMKEY.
3. Write 0x556699AA to NVMKEY.
4. Write the value to the register NVMCON, NVMCON2, NVMPWP* or NVMLBWP requiring the unlock sequence.

When using the unlock sequence to set or clear bits in the NVMCON register, as shown in Step 4, Steps 2 through 4 must be executed without any other activity on the peripheral bus that is in use by the Flash Controller. Interrupts and DMA transfers that access the same peripheral bus as the Flash Controller must be disabled. In addition, the operation in Step 4 must be atomic. The Set, Clear and Invert registers may be used, where applicable, for the target register in Step 4.

The following code shows code written in the C language to initiate a NVM Operation (NVMOP) command. In this particular example, the WR bit is being set in the NVMCON register and, therefore, must include the unlock sequence.

Initiate NVM Operation (System Unlock Sequence Example):

```
void NVMInitiateOperation(void)
{
  // Disable Interrupts
  asm volatile("di%0" : "=r"(int_status));
  uint32_t globalInterruptState= __get_PRIMASK();
// Disable Interrupts
__disable_irq();
  NVMKEY = 0x0;
  NVMKEY = 0xAA996655;
  NVMKEY = 0x556699AA;
  NVMCONSET = 1 << 15;// must be an atomic instruction

  // Restore Interrupts
  __set_PRIMASK(globalInterruptState);
}
```

Note: Once the unlock codes are written to the NVMKEY register, the next activity on the same peripheral bus as the Flash Controller will Reset the lock. As a result, only atomic operations can be used. Use of the NVMCONSET register sets the WR bit in a single instruction without changing other bits in the register. Using NVMCONbits.WR = 1 will fail, as this line of code compiles to a read-modify-write sequence.

24.12 Word Programming

The smallest block of data that can be programmed in a single operation is one Flash write Word (32-bit). The data to be programmed must be written to the NVMDATA0 register, and the address of the Word must be loaded into the NVMADDR register before the programming sequence is initiated. The instruction Word at the physical location pointed to by the NVMADDR register is, then, programmed. Programming occurs on 32-bit Word boundaries; therefore, bits '0' and '1' of the NVMADDR register are ignored.

When a Word is programmed, it must be erased before it can be programmed again, even if changing a bit from an erased '1' state to a '0' state.

Word programming will only succeed if the target address is in a page that is not write-protected. Programming to a write-protected PFM page will fail and result in the WRERR bit being set in the NVMCON register. Programming a write-protected BFM page will fail but does not set the WRERR bit.

A programming sequence consists of the following steps:

1. Write 32-bit data to be programmed to the NVMDATA0 register.
2. Load the NVMADDR register with the address to be programmed.
3. Set the WREN bit = 1 and NVMOP bits = 1 in the NVMCON register. This defines and enables the programming operation.
4. Initiate the programming operation. (See *NVMKEY Register Unlocking Sequence* from Related Links.)

5. Monitor the WR bit of the NVMCON register to flag completion of the operation.
6. Clear the WREN bit in the NVMCON register.
7. Check for errors and process accordingly.

The following code shows code for Word programming, where a value of 0x12345678 is programmed into location 0x1008000.

Word Programming Code Example:

```
…
// Set up Address and Data Registers
  NVMADDR= 0x1008000;          // physical address
  NVMDATA0 = 0x12345678;        // value

// set the operation, assumes WREN = 0
  NVMCONbits.NVMOP = 0x1;         // NVMOP for Word programming

// Enable Flash for write operation and set the NVMOP
  NVMCONbits.WREN = 1;

// Start programming
  NVMInitiateOperation();               // see Initiate NVM Operation (Unlock Sequence
Example)

// Wait for WR bit to clear
  while (NVMCONbits.WR);

// Disable future Flash Write/Erase operations
  NVMCONbits.WREN = 0;

// Check Error Status
  if(NVMCON & 0x3000)            // mask for WRERR and LVDERR
  {
        // process errors
    }
…
```

Related Links

24.11. NVMKEY Register Unlocking Sequence

24.13 Quad Word Programming

The process for Quad Word programming is identical to Word programming except that all four of the NVMDATAx registers are used. The value of the NVMDATA0 register is programmed at address NVMADDR, NVMDATA1 at NVMADDR + 0x4, NVMDATA2 at NVMADDR + 0x8, and NVMDATA3 at address NVMDATA + 0xC.

Quad Word programming is always performed on a Quad Word boundary; therefore, NVMADDR address bits 3 through 0 are ignored.

Quad Word programming will only succeed if the target address is in a page that is not write-protected. When a Quad Word is programmed, it must be erased before any Word in it can be programmed again, even if changing a bit from an erased '1' state to a '0' state.

Where a value of 0x11111111 is programmed into location 0x1008000, 0x22222222 into 0x1008004, 0x33333333 into 0x1008008, and 0x44444444 into location 0x100800C. Refer to the following code example for details.

Quad Word Programming Code Example:

```
…

// Set up Address and Data Registers
    NVMADDR  = 0x1008000;      // physical address
    NVMDATA0 = 0x11111111;     // value written to 0x1008000
    NVMDATA1 = 0x22222222;     // value written to 0x1008004
    NVMDATA2 = 0x33333333;     // value written to 0x1008008
    NVMDATA3 = 0x44444444;     // value written to 0x100800C

// set the operation, assumes WREN = 0
    NVMCONbits.NVMOP = 0x2;    // NVMOP for Quad Word programming
```

```
// Enable Flash for write operation and set the NVMOP
    NVMCONbits.WREN = 1;

// Start programming
    NVMInitiateOperation();    // see Initiate NVM Operation (Unlock Sequence Example)

// Wait for WR bit to clear
    while(NVMCON & NVMCON_WR);

// Disable future Flash Write/Erase operations
    NVMCONbits.WREN = 0;

// Check Error Status
    if(NVMCON & 0x3000)    // mask for WRERR and LVDERR bits
```

24.14 Row Programming

The largest block of data that can be programmed is a row.

Unlike Word and Quad Word Programming where the data source is stored in SFR memory, Row programming source data is stored in SRAM. The NVMSRCADDR register is a pointer to the physical location of the source data for Row programming.

Like other Non-Volatile Memory (NVM) programming commands, the NVMADDR register points to the target address of the operation. Row programming always occurs on row boundaries with the row size of 1024, bits 0 through 9 of the NVMADDR register are ignored.

Row Word programming will only succeed if the target address is in a page that is not write-protected. When a row is programmed, it must be erased before any Word in it can be programmed again, even if changing a bit from an erased '1' state to a '0' state.

Array `rowbuff` is populated with data and programmed into a row located at physical address `0x1008000`.

Note: When assigning the value to the NVMSRCADDR register, it must be converted to a physical address.

Row Programming Code Example:

```
…

unsigned long rowbuff[256]; // example is for a 256 Word row size.
int x;                      // loop counter

// put some data in the source buffer
      for (x = 0; x < (sizeof(rowbuff) * sizeof (int)); x++)
            ((char *)rowbuff)[x] = x;

// set destination row address
     NVMADDR = 0x1008000;             // row physical address

// set source address. Must be converted to a physical address.
     NVMSRCADDR = (unsigned int)((int)rowbuff & 0x1FFFFFF);

// define Flash operation
      NVMCONbits.NVMOP = 0x3;            // NVMOP for Row programming

// Enable Flash Write
      NVMCONbits.WREN = 1;

// commence programming
      NVMInitiateOperation();            // see Initiate NVM Operation (Unlock Sequence
Example)

// Wait for WR bit to clear
      while(NVMCONbits.WR);

// Disable future Flash Write/Erase operations
       NVMCONbits.WREN = 0;

// Check Error Status
       if(NVMCON & 0x3000)            // mask for WRERR and LVDERR bits
      {
         // process errors
```

```
        }
…
```

24.15 Page Erase

A Page Erase performs an erase of a single page of either PFM or BFM.

The page to be erased is selected using the NVMADDR register. Pages are always erased on page boundaries; therefore, for a device with an instruction Word page size of 4096, bits 0 through 11 of the NVMADDR register are ignored.

A Page Erase will only succeed if the target address is a page that is not write-protected. Erasing a write-protected page will fail and result in the WRERR bit being set in the NVMCON register.

The following code shows the code for a single Page Erase operation at address 0x1008000.

Page Erase Code Example:

```
…

// set destination page address
    NVMADDR = 0x1008000;     // page physical address

// define Flash operation
    NVMCONbits.NVMOP = 0x4;    // NVMOP for Page Erase

// Enable Flash Write
    NVMCONbits.WREN = 1;

// commence programming
    NVMInitiateOperation();    // see Initiate NVM Operation (Unlock Sequence Example)

// Wait for WR bit to clear
    while(NVMCONbits.WR);

// Disable future Flash Write/Erase operations
    NVMCONbits.WREN = 0;

// Check Error Status
    if(NVMCON & 0x3000)    // mask for WRERR and LVDERR bits
    {
        // process errors
    }

…
```

24.15.1 Page Erase Retry

Page Erase Retry is a method to improve the life of a Flash by attempting to erase again if the Page Erase was not successful. Page Erase Retry can only be used for a Page Erase.

Page Erase Retry works by increasing the voltage used on the Flash when erasing. Initially, the minimum voltage necessary is applied by setting the RETRY[1:0] bits (NVMCON2[9:8]) = 00. If the page erase is not successful, the voltage may be increased by incrementing the setting of the RETRY[1:0] bits.
Note: Each Flash page, as it ages and wears, may have different voltage requirements; therefore, a higher setting on one Flash page does not indicate that the same setting must be used on all pages.

The maximum voltage for Page Erase is used when the RETRY[1:0] bits = 11. If Page Erase is not successful after 7 trials, this means that the Flash for that page, or the Words that did not erase, must be considered “non-functional”.

Together with the normal Page Erase controls, Page Erase Retry also uses the WS[4:0], CREAD1, VREAD1 and RETRY[1:0] bits in the NVMCON2 register. The ERS[3:0] bits (NVMCON2[31:28]) are for the benefit of software performing the programming sequence in the event that a drop in power causes a BOR event but not a POR event.

Perform the following steps to set up a Page Erase Retry:

1. Set the NVMADDR register with the address of the page to be erased.
2. Execute the write unlock sequence.
3. Save the value of the NVMCON2 register.

4. Do the following in the NVMCON2 register:
   a. Set the ERS[3:0] bits as desired.
   b. Set the WS[4:0] bits per the description.
   c. Set the VREAD1 bit to '1'.
   d. Set the CREAD1 bit to '1'.
   e. Set the RETRY[1:0] bits to '00'.
5. Run the unlock sequence using the Page Erase command to start the sequence.
6. Wait for the WR bit (NVMCON[15]) to be cleared by hardware.
7. Clear the WREN bit (NVMCON[14]).
8. Verify the erase using the CPU. To shorten the verify time, use CREAD1 = 1 to perform a hardware compare to logic '1' of each bit in the Flash Word including ECC. A successful compare yields a read of 0x00000001 in the lowest addressed word in a Flash Word (128 bits). This is the Compare Word. All other Words are 0x00010000. If any bit is logic '0', all Words in the Flash Word read 0x00000000. Remember to increment the address by the number of bytes in a Flash Word between reads.
9. If all Compare Words verify correctly, the Page Erase Retry process is complete. Go to step 11.
10. If a Compare Word yields a read of 0x00000000, perform steps 4 through 9 up to six more times with the following change to step 4:
    a. Increment the RETRY[1:0] bits by one if the bit has not already reached the '11' setting.
    b. Maintain all other fields.
11. Restore the value of the NVMCON2 register, which was saved in step 3.

Notes:

1. When the VREAD1 = 1, the Flash uses the WS[3:0] bits for Flash access wait state generation to the panel selected by NVMADDR. Software is responsible for writing the VREAD1 bit back to '0' when both erase and verify is complete.
2. The device configuration boot page (the page containing the DEVCFGx values) does not support Page Erase Retry.

The following code provides code for a single page erase operation at address 0x1008000, where Page Erase Retry is used.

Page Erase Retry Code Example:

```
uint32_tsaveNVMCON2;
uint32_t*cmpPtr;
uint8_terased;
uint8_ttryCount;

// set destination page address
    NVMADDR = 0x1008000;    // Page physical address

// define flash operation
    NVMCONbits.NVMOP = 0x4;    // NVMOP for Page Erase

// Unlock sequence
    NVMKEY = 0x0;
    NVMKEY = 0xAA996655;
    NVMKEY = 0x556699AA;

// save NVMCON2
    saveNVMCON2 = NVMCON2;

// set up Page Erase Retry
    NVMCON2bits.ERS = 0;    // Stage 0 - SW use only
    NVMCON2bits.VREAD1 = 1;
    NVMCON2bits.CREAD1 = 1;
    NVMCON2bits.RETRY = 0b00;

    tryCount = 0;        // Up to 4 attempts

    do {
        tryCount++;

        // commence programming
```

```
        NVMInitiateOperation();

        // Wait for WR bit to clear
        while(NVMCONbits.WR);

        // Turn off WREN
        NVMCONbits.WREN = 0;

        // Check that the page was erased
        erased = 1;
        cmpPtr = (uint32_t *)NVMADDR;
        erased &= (*cmpPtr == 0x00000001);
        cmpPtr++;
        erased &= (*cmpPtr == 0x00010000);
        cmpPtr++;
        erased &= (*cmpPtr == 0x00010000);
        cmpPtr++;
        erased &= (*cmpPtr == 0x00010000);

        if (!erased) {
            // Erase failed. Try with different settings.
            NVMCON2bits.RETRY++;

            NVMCONbits.NVMOP = 0x4;
            NVMCONbits.WREN = 1;
        }
    } while (!erased && (tryCount < 4));

// Restore settings
    NVMCON2 = saveNVMCON2;
```

24.16 Program Flash Memory (PFM) Erase

Program Flash memory can be erased entirely. All three discrete NVMOP values, 0111, 0110, 0101, do the same operation of erase of entire Flash. When erasing the entire PFM area, in case of RTSP (Run Time Self Programming), the code must be executing from BFM. When erasing the entire PFM area, PFM write-protection must be completely disabled.

The following code shows code for erasing the entire Flash bank.

Program Flash Erase Code Example:

```
…

// define Flash operation
    NVMCONbits.NVMOP = 0x7;        // NVMOP for entire PFM erase

// Enable Flash Write
    NVMCONbits.WREN = 1;

// commence programming
    NVMInitiateOperation();        // see Initiate NVM Operation (Unlock Sequence Example)

// Wait for WR bit to clear
    while(NVMCONbits.WR);

// Disable future Flash Write/Erase operations
    NVMCONbits.WREN = 0;

// Check Error Status
    if(NVMCON & 0x3000)        // mask for WRERR and LVDERR bits
    {
         // process errors
    }

…
```

24.17 Pre-Program

The PIC32CX-BZ2 Flash supports an option to programming that increases endurance and retention. This feature is called Pre-Program, and it requires the user to perform the programming operation twice, first, with NVMCON2.NVMPREPG = 1 and, secondly, with NVMCON2.NVMPREPG = 0. Any of the programming operations (Single, Quad, Row) can be performed with this method. In all other respects, the SFR setup is identical. To use this feature, set or clear the NVMCON2.NVMPREPG SFR bit prior to setting the NVMWR bit. Pre-Program, typically double, the native Endurance and Retention of the Flash.

24.18 Device Code Protection bit (CP)

The PIC32CX-BZ2 family of devices features code protection, which, when enabled, prevents reading of the Flash memory by an external programming device (SWD through DSU).

When code protection is enabled, it can only be disabled by erasing the device with the Chip Erase command through an external programmer. See *Device Service Unit (DSU)* from Related Links.

When programming a device that has opted to utilize code protection, the external programming device must perform verification prior to enabling code protection. Enabling code protection must be the last step of the programming process. For the location of the code protection enable bits, refer to *PIC32CX-BZ2 Programming Specification* and *System Configuration Registers (CFG)* from Related Links.

Related Links

12. Device Service Unit (DSU)
18. System Configuration and Register Locking (CFG)

24.19 Operation in Power-Saving Modes

The Flash Controller does not operate in power-saving modes. If a WAIT instruction is encountered when programming, the CPU will stop execution (stall), wait for the programming operation to complete, then enter the Power-Saving mode.

24.20 Operation in Debug Mode

Programming operations will continue to completion if the processor execution is halted in Debug mode.

24.21 Effects of Various Resets

Device Resets, other than a Power-on Reset (POR), reset the entire contents of the NVMPWP and NVMLBWP registers. All other register content persists through a non-POR reset.

All Flash Controller registers are forced to their reset states upon a POR.

24.22 Control Registers

Note: The following conventions are used in the following registers:

- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- 1= Bit is set
- 0= Bit is cleared
- x = Bit is unknown
- -n = Value at POR

- HS = Hardware Set
- HC = Hardware Cleared

Note: All registers in this table have corresponding CLR, SET and INV registers at its virtual address, plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET and INV Registers* from Related Links.

Related Links

6.1.9. CLR, SET and INV Registers

24.22.1 Register Summary

The following registers provides a brief summary of the Flash programming-related registers.

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	NVMCON	7:0					NVMOP[3:0]			
		15:8	WR	WREN	WRERR	LVDERR				HTDPGM
		23:16								
		31:24								
0x04 ... 0x0F	Reserved									
0x10	NVMCON2	7:0								NVMPREPG
		15:8		TEMP	CREAD1	VREAD1			RETRY[1:0]	
		23:16				WS[4:0]				
		31:24	ERS[3:0]							SLEEP
0x14 ... 0x1F	Reserved									
0x20	NVMKEY	7:0	NVMKEY[7:0]							
		15:8	NVMKEY[15:8]							
		23:16	NVMKEY[23:16]							
		31:24	NVMKEY[31:24]							
0x24 ... 0x2F	Reserved									
0x30	NVMADDR	7:0	NVMADDR[7:0]							
		15:8	NVMADDR[15:8]							
		23:16	NVMADDR[23:16]							
		31:24	NVMADDR[31:24]							
0x34 ... 0x3F	Reserved									
0x40	NVMDATA0	7:0	NVMDATA[7:0]							
		15:8	NVMDATA[15:8]							
		23:16	NVMDATA[23:16]							
		31:24	NVMDATA[31:24]							
0x44 ... 0x4F	Reserved									
0x50	NVMDATA1	7:0	NVMDATA[7:0]							
		15:8	NVMDATA[15:8]							
		23:16	NVMDATA[23:16]							
		31:24	NVMDATA[31:24]							
0x54 ... 0x5F	Reserved									
0x60	NVMDATA2	7:0	NVMDATA[7:0]							
		15:8	NVMDATA[15:8]							
		23:16	NVMDATA[23:16]							
		31:24	NVMDATA[31:24]							
0x64 ... 0x6F	Reserved									
0x70	NVMDATA3	7:0	NVMDATA[7:0]							
		15:8	NVMDATA[15:8]							
		23:16	NVMDATA[23:16]							
		31:24	NVMDATA[31:24]							
0x74 ... 0xBF	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0xC0	NVMSRCADDR	7:0	NVMSRCADDR[7:0]							
		15:8	NVMSRCADDR[15:8]							
		23:16	NVMSRCADDR[23:16]							
		31:24	NVMSRCADDR[31:24]							
0xC4 ... 0xCF	Reserved									
0xD0	NVMPWPLT	7:0	PWPLT[7:0]							
		15:8	PWPLT[15:8]							
		23:16	PWPLT[23:16]							
		31:24	ULOCK							
0xD4 ... 0xDF	Reserved									
0xE0	NVMPWPGTE	7:0	PWPGTE[7:0]							
		15:8	PWPGTE[15:8]							
		23:16	PWPGTE[23:16]							
		31:24	ULOCK							
0xE4 ... 0xEF	Reserved									
0xF0	NVMLBWP	7:0	LBWP[7:0]							
		15:8	LBWP[15:8]							
		23:16	LBWP[23:16]							
		31:24	ULOCK							

24.22.2 Register Description

Flash program, erase, and write protection operations are controlled using the following Non-Volatile Memory (NVM) control registers:

- NVMCON: Programming Control Register
  - This register is the control register for Flash program/erase operations. This register is used to select the operation to be performed, initiate the operation, and provide status of the result when the operation is complete.
- NVMCON2: Programming Control2 Register
  - This register is the control and status register for Flash program/erase operations.
- NVMKEY: Programming Unlock Register
  - This is a write-only register that is used to implement an unlock sequence to help prevent accidental writes/erasures of Flash memory and write permission settings.
- NVMADDR: Flash Address Register
  - This register is used to store the physical target address for row, Quad Double Word and Single Double Word programming as well as page erasing.
- NVMDATAx: Flash Program Data Register (x = 0-3)
  - These registers hold the data to be programmed during Flash Word program operations.
- NVMSRCADDR: Source Data Address Register
  - This register is used to point to the physical address of the data to be programmed when executing a row program operation.
- NVMPWPLT: Flash Program Write Protect Lower Register
  - This register is used to set the program flash pages lower than provided address as a write protected.
- NVMPWPGTE: Flash Program Write Protect Greater Register
  - This register is used to set the program flash pages greater than provided address as a write protected.
- NVMLBWP: Flash Boot Write Protect Register
  - This register is used to set the boot flash partition pages as a write protected.

Following conventions are used in the register description:

- – R = Readable bit
- – W = Writable bit
- – U = Unimplemented bit, read as ‘0’
- – -n = Value at POR
- – ‘1’ = Bit is set
- – ‘0’ = Bit is cleared
- – x = Bit is unknown
- HS = Hardware Set
- HC = Hardware Cleared

24.22.2.1 NVMCON – Programming Control Register

Name: NVMCON
Offset: 0x00
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	WR	WREN	WRERR	LVDERR				HTDPGM
Access	R/HS/HC	R/W	R/HS/HC	R/HS/HC				R/HS/HC
Reset	0	0	0	0				0

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

Bit 15 – WR Write Control Bit(1)

Note: This field can only be modified when WREN = 1, TEMP = 1 and the NVMKEY unlock sequence is satisfied.

Value	Description
1	Initiate a Flash operation. Hardware clears this bit when the operation completes
0	Flash operation complete or inactive

Bit 14 – WREN Write Enable Bit(1)

Value	Description
1	Enables writes to WR
0	Disables writes to WR

Bit 13 – WRERR Write Error Bit(1)

Note: Cleared by setting NVMOP == 0000b and initiating a Flash operation (WR).

Value	Description
1	Program or erase sequence did not complete successfully
0	Program or erase sequence completed normally

Bit 12 – LVDERR Low Voltage Detect Error Bit(1)

The error is only captured for programming/erase operations (when WR = 1).

Note: Cleared by setting NVMOP == 0000b and initiating a Flash operation (WR).

Value	Description
1	Low voltage is detected (possible data corruption if WRERR is set)
0	Normal voltage is detected

Bit 8 – HTDPGM High Temperature Detected during Program/Erase Operation bit

This status is only captured for programming/erase operations (when WR = 1).

Note: Cleared by setting NVMOP == 0000b and initiating a Flash operation (WR).

Value	Description
1	High temperature is detected (possible data corruption, verify operation)
0	High temperature is not detected

Bits 3:0 – NVMOP[3:0] NVM Operation bits
These bits are only writable when WREN = 0.

Value	Description
1111	Reserved
1110	Chip Erase Operation: Erases PFM, BFM (except configuration page) when accessed through SWD interface only.
...	
...	
...	
1000	Reserved
0111	Program erase operation: erase all of program Flash memory (PFM) (all pages must be unprotected)
0110	Upper program Flash memory erase operation: erases only the upper mapped region of program Flash (all pages in that region must be unprotected). It is a single bank Flash in PIC32CX-BZ2; therefore, this NVMOP performs the same as NVMOP = 0111.
0101	Lower program Flash memory erase operation: erases only the lower mapped region of program Flash (all pages in that region must be unprotected). It is a single bank Flash in PIC32CX-BZ2; therefore, this NVMOP performs the same as NVMOP = 0111.
0100	Page erase operation: erases the page selected by NVMADDR if it is not write-protected.
0011	Row program operation: programs the row selected by NVMADDR if it is not write-protected.
0010	Quad Word (128-bit) program operation: programs the 128-bit Flash Word selected by NVMADDR if it is not write-protected.
0001	Word program operation: programs the Word selected by NVMADDR if it is not write-protected(2).
0000	No operation

Notes:

1. These bits are reset by a POR only and are not affected by other Reset sources.
2. This operation results in a No Operation (NOP) when the Dynamic Flash ECC Configuration bits = 00 (ECCCTL[1:0](CFGCON0[29:28])), which enables ECC at all times. For all other ECCCTL[1:0] bit settings, this command will execute but will not write the ECC bits for the Word. It can cause DED (Double-bit Error Detected) errors if dynamic Flash ECC is enabled (ECCCTL[1:0] = 01).

24.22.2.2 NVMCON2 – Programming Control 2 Register

Name: NVMCON2
Offset: 0x10
Reset: 0x011F4000
Property: -

Bit	31	30	29	28	27	26	25	24
	ERS[3:0]							SLEEP
Access	R/W	R/W	R/W	R/W				R/W
Reset	0	0	0	0				1

Bit	23	22	21	20	19	18	17	16
				WS[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
		TEMP	CREAD1	VREAD1			RETRY[1:0]	
Access		R	R/W	R/W			R/W	R/W
Reset		1	0	0			0	0

Bit	7	6	5	4	3	2	1	0
								NVMPREPG
Access								R/W
Reset								0

Bits 31:28 – ERS[3:0] Erase Retry State

These bits are used by software to track the software state of the erase retry procedure in the event of a system Reset (NMCLR) or Brown-out Reset (BOR) event.

Bit 24 – SLEEP Power Down in Sleep mode

Note: This field can only be modified when the NVMKEY unlock sequence is satisfied.

Value	Description
1	Configures Flash for power-down when the system is in Sleep mode
0	Configures Flash for standby when the system is in Sleep mode

Bits 20:16 – WS[4:0] Flash Access Wait State Control for VREAD1 = 1

Notes:

1. When VREAD1 = 1, WS[4:0] only affects the memory containing NVMADDR[31:0].
2. This field can only be modified when the NVMKEY unlock sequence is satisfied.

Value	Description
11111	31 wait states (32 total system clocks)
11110	30 wait states (31 total system clocks)
...	
00010	2 wait states (3 total system clocks)
00001	1 wait state (2 total system clocks)
00000	0 wait state (1 total system clock)

Bit 14 – TEMP Operating Temperature Control bit

Bit 13 – CREAD1 Compare Read of Logic 1 bit

Compare read 1 causes all bits in a Flash Word (including ECC if it exists) to be evaluated during the read. If all bits are '1', the lowest Word in the Flash Word evaluates to 0x0000_0001, all other Words are 0x0001_0000. If any bit is '0', the read evaluates to 0x0000_0000 for all Words in the Flash Word.

Notes:

1. When using erase retry in an ECC Flash system, CREAD1 = 1 must be used.
2. This field can only be modified when the NVMKEY unlock sequence is satisfied.

Value	Description
1	Compare read enabled only if VREAD1 = 1
0	Compare read disabled

Bit 12 – VREAD1 Verify Read of logic 1 Control bit

Notes:

1. When VREAD1 = 1, the Flash wait state control is from WS[4:0] for the memory containing NVMADDR[31:0].
2. Using Page Erase Retry and Verify Read procedure increase the life of the Flash memory.
3. This field can only be modified when NVMCON.WR == 0 and the NVMKEY unlock sequence is satisfied.

Value	Description
1	Selects erase retry procedure with verify read
0	Selects single erase without verify read

Bits 9:8 – RETRY[1:0] Erase Retry Control bit, only used when VREAD1 = 1

Note: This field can only be modified when NVMCON.WR == 0.

Value	Description
11	Erase strength for last retry cycle
10	Erase strength for third retry cycle
01	Erase strength for second retry cycle
00	Erase strength for first retry cycle

Bit 0 – NVMPREPG NVM Pre-Program Control Bit

Note: This field can only be modified when NVMCON.NVMWR= = 0.

Value	Description
1	Program Operations include the Pre-Program step
0	Program Operations exclude the Pre-Program step

24.22.2.3 NVMKEY – Programming Unlock Register

Name: NVMKEY
Offset: 0x20
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	NVMKEY[31:24]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	NVMKEY[23:16]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NVMKEY[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	NVMKEY[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – NVMKEY[31:0] Unlock Register bits
These bits are write-only and read '0' on any read.
Note: This register is used as part of the unlock sequence to prevent inadvertent writes to the program Flash.

24.22.2.4 NVMADDR – Flash Address Register

Name: NVMADDR
Offset: 0x30
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	NVMADDR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	NVMADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NVMADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – NVMADDR[31:0] Flash (Word) Address bits

Table 24-5. Flash (Word) Address Bits

NVMOP	Flash Address Bits
Page Erase	• Address identifies the page to erase • Any address within a 4 Kbytes page boundary will cause the page to be erased
Row program	• Address identifies the row to program • The value of the address must be aligned to a row boundary
Word program	• Address identifies the 32-bit Word to program • NVMADDR[1:0] bits are ignored • Must be aligned to a Word boundary
Quad Word program	• Address identifies the 128-bit Quad Word to program • NVMADDR[3:0] bits are ignored • Must be aligned to a Quad Word boundary

Notes:

1. Hardware prevents writes to this register when NVMCON.WR = 1.
2. For all other NVMOP[3:0] bit settings, the Flash address is ignored. For additional information on these bits, see the *NVMCON* register from Related Links.
3. The bits in this register are reset by a POR only and are not affected by other Reset sources.

Related Links

24.22.2.1. NVMCON

24.22.2.5 NVMDATA0 – Flash Program Data Register 0

Name: NVMDATA0
Offset: 0x40
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	NVMDATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	NVMDATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NVMDATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – NVMDATA[31:0] Flash Programming Data bits
The value in this register is written to Flash when a program operation is commanded.

- Single Word program (32-bit)
  - Writes NVMDATA0 to the target Flash address defined in NVMADDR[31:2].
- Quad Word program (128-bit)
  - Writes NVMDATA3:NVMDATA2:NVMDATA1:NVMDATA0 to the target Flash address defined in NVMADDR[31:4]. NVMDATA0 contains the Least Significant Instruction Word.

Notes:

1. Hardware prevents writes to this register when NVMCON.WR = `1`.
2. The bits in this register are reset on a POR only and are unaffected by other Reset sources.

24.22.2.6 NVMDATA1 – Flash Program Data Register 1

Name: NVMDATA1
Offset: 0x50
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	NVMDATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	NVMDATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NVMDATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – NVMDATA[31:0] Flash Programming Data bits
The value in this register is written to Flash when a program operation is commanded.

- Single Word program (32-bit)
  - Writes NVMDATA0 to the target Flash address defined in NVMADDR[31:2].
- Quad Word program (128-bit)
  - Writes NVMDATA3:NVMDATA2:NVMDATA1:NVMDATA0 to the target Flash address defined in NVMADDR[31:4]. NVMDATA0 contains the Least Significant Instruction Word.

Notes:

1. Hardware prevents writes to this register when NVMCON.WR = `1`.
2. The bits in this register are reset on a POR only and are unaffected by other Reset sources.

24.22.2.7 NVMDATA2 – Flash Program Data Register 2

Name: NVMDATA2
Offset: 0x60
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	NVMDATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	NVMDATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NVMDATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – NVMDATA[31:0] Flash Programming Data bits
The value in this register is written to Flash when a program operation is commanded.

- Single Word program (32-bit)
  - Writes NVMDATA0 to the target Flash address defined in NVMADDR[31:2].
- Quad Word program (128-bit)
  - Writes NVMDATA3:NVMDATA2:NVMDATA1:NVMDATA0 to the target Flash address defined in NVMADDR[31:4]. NVMDATA0 contains the Least Significant Instruction Word.

Notes:

1. Hardware prevents writes to this register when NVMCON.WR = `1`.
2. The bits in this register are reset on a POR only and are unaffected by other Reset sources.

24.22.2.8 NVMDATA3 – Flash Program Data Register 3

Name: NVMDATA3
Offset: 0x70
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	NVMDATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	NVMDATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NVMDATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – NVMDATA[31:0] Flash Programming Data bits
The value in this register is written to Flash when a program operation is commanded.

- Single Word program (32-bit)
  - Writes NVMDATA0 to the target Flash address defined in NVMADDR[31:2].
- Quad Word program (128-bit)
  - Writes NVMDATA3:NVMDATA2:NVMDATA1:NVMDATA0 to the target Flash address defined in NVMADDR[31:4]. NVMDATA0 contains the Least Significant Instruction Word.

Notes:

1. Hardware prevents writes to this register when NVMCON.WR = `1`.
2. The bits in this register are reset on a POR only and are unaffected by other Reset sources.

24.22.2.9 NVMSRCADDR – Source Data Address Register

Name: NVMSRCADDR
Offset: 0xC0
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	NVMSRCADDR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	NVMSRCADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NVMSRCADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – NVMSRCADDR[31:0] Source Data (Word) Address bits
This is the system physical Word address of the data (in DRM) to be programmed into the Flash when NVMCON.NVMOP is set to row programming.
Notes:

1. Hardware prevents writes to this register when NVMCON.WR = `1`.
2. The bits in this register are reset on a POR only and are unaffected by other reset sources.

24.22.2.10 NVMPWPLT – Flash Program Write Protect Lower Register

Name: NVMPWPLT
Offset: 0xD0
Reset: 0x80000000
Property: -

Bit	31	30	29	28	27	26	25	24
	ULOCK							
Access	R/C							
Reset	1							

Bit	23	22	21	20	19	18	17	16
	PWPLT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PWPLT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bit 31 – ULOCK NVMPWPLT Register Unlock bit

Notes:

1. This field can only be modified when the NVMKEY unlock sequence is satisfied.
2. This field can be cleared at the same time as writing to PWPLT[23:0].

Value	Description
1	NVMPWPLT register is not locked and can be modified
0	NVMPWPLT register is locked and cannot be modified

Bits 23:0 – PWPLT[23:0] Flash Program Write Protect Less Than Address

Pages at Flash addresses less than this value are write-protected.

Notes:

1. This field can only be modified when the NVMKEY unlock sequence is satisfied, and ULOCK = 1.
2. This is a byte address force to align to page boundaries.

24.22.2.11 NVMPWPGTE – Flash Program Write Protect Greater Register

Name: NVMPWPGTE
Offset: 0xE0
Reset: 0x80FFFFFF
Property: -

Bit	31	30	29	28	27	26	25	24
	ULOCK							
Access	R/C							
Reset	1							

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

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

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

Bit 31 – ULOCK NVMPWPGTE Register Unlock bit

Notes:

1. This field can only be modified when the NVMKEY unlock sequence is satisfied.
2. This field can be cleared at the same time as writing to PWPGTE[23:0].

Value	Description
1	NVMPWPGTE register is not locked and can be modified
0	NVMPWPGTE register is locked and cannot be modified

Bits 23:0 – PWPGTE[23:0] Flash Program Write Protect Address

Pages at Flash addresses greater than or equal to this value are write-protected.

Notes:

1. This field can only be modified when the NVMKEY unlock sequence is satisfied and ULOCK = 1.
2. This is a byte address forced to align to page boundaries.

24.22.2.12 NVMLBWP – Flash Boot Write Protect Register

Name: NVMLBWP
Offset: 0xF0
Reset: 0x80FFFFFF
Property: -

Bit	31	30	29	28	27	26	25	24
	ULOCK							
Access	R/C							
Reset	1							

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

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

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

Bit 31 – ULOCK Lower Boot Write Protect (LBWPn) Unlock bit

Notes:

1. This field can only be modified when the NVMKEY unlock sequence is satisfied.
2. This field can be cleared at the same time as writing to LBWP[msb:lsb].

Value	Description
1	LBWPn bits are not locked and can be modified
0	LBWPn bits are locked and cannot be modified

Bits 23:0 – LBWP[23:0] Boot Pages Write Protect bits

Notes:

1. This field can only be modified when the NVMKEY unlock sequence is satisfied and ULOCK = 1.
2. The OTP page is always erase-protected and its associated LBWP bit is only for write-protection.

Value	Description
1	Erase and write-protection for upper boot page n is enabled
0	Erase and write-protection for upper boot page n is disabled

25. Integrity Check Monitor (ICM)

25.1 Overview

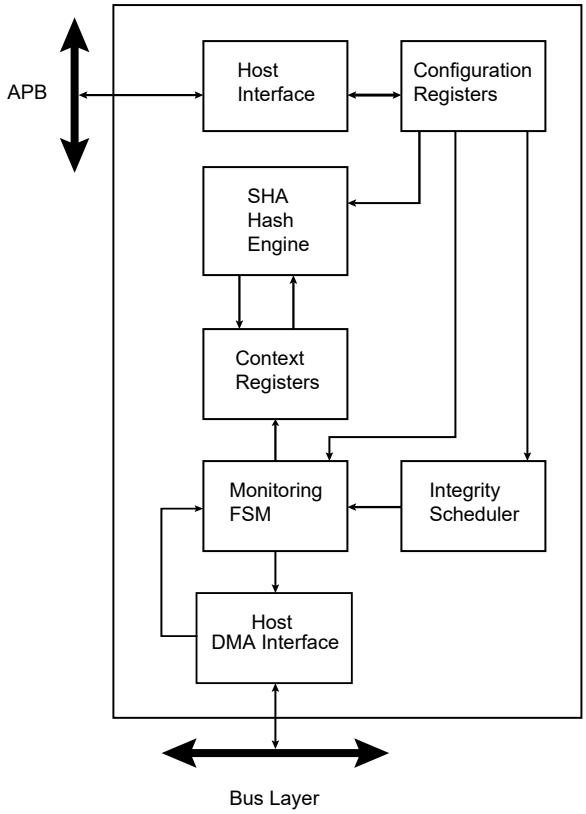
The Integrity Check Monitor (ICM) is a DMA controller that performs hash calculation over multiple memory regions using transfer descriptors located in memory (ICM Descriptor Area). The Hash function is based on the Secure Hash Algorithm (SHA). The ICM controller integrates two modes of operation. The first mode is used to hash a list of memory regions and save the digests to memory (ICM Hash Area). The second mode is an active monitoring of the memory. In this mode, the hash function is evaluated and compared to the digest located at a predefined memory address (ICM Hash Area). If a mismatch occurs, an interrupt is raised.

25.2 Features

- DMA AHB manager interface
- Supports monitoring of up to four non-contiguous memory regions
- Supports block gathering using a linked list
- Supports Secure Hash Algorithm (SHA1, SHA256)
- Compliant with FIPS Publication 180-2
- Configurable processing period:
  - When SHA1 algorithm is processed, the run-time period is either 85 or 209 clock cycles
  - When SHA256 algorithm is processed, the run-time period is either 72 or 194 clock cycles
- Programmable bus burden

25.3 Block Diagram

Figure 25-1. Integrity Check Monitor Block Diagram



25.4 Signal Description

Not applicable.

25.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

25.5.1 Power Management

25.5.2 Clocks

The ICM bus clocks (PB2_CLK) can be enabled and disabled in the CRU module or the PMD2.ICMMD bit. For more details, see *Peripheral Module Disable Register (PMD)* from Related Links.

Related Links

20. Peripheral Module Disable Register (PMD)

25.5.3 DMA

Not applicable.

25.5.4 Events

Not applicable.

25.5.5 Debug Operation

Not applicable.

25.6 Functional Description

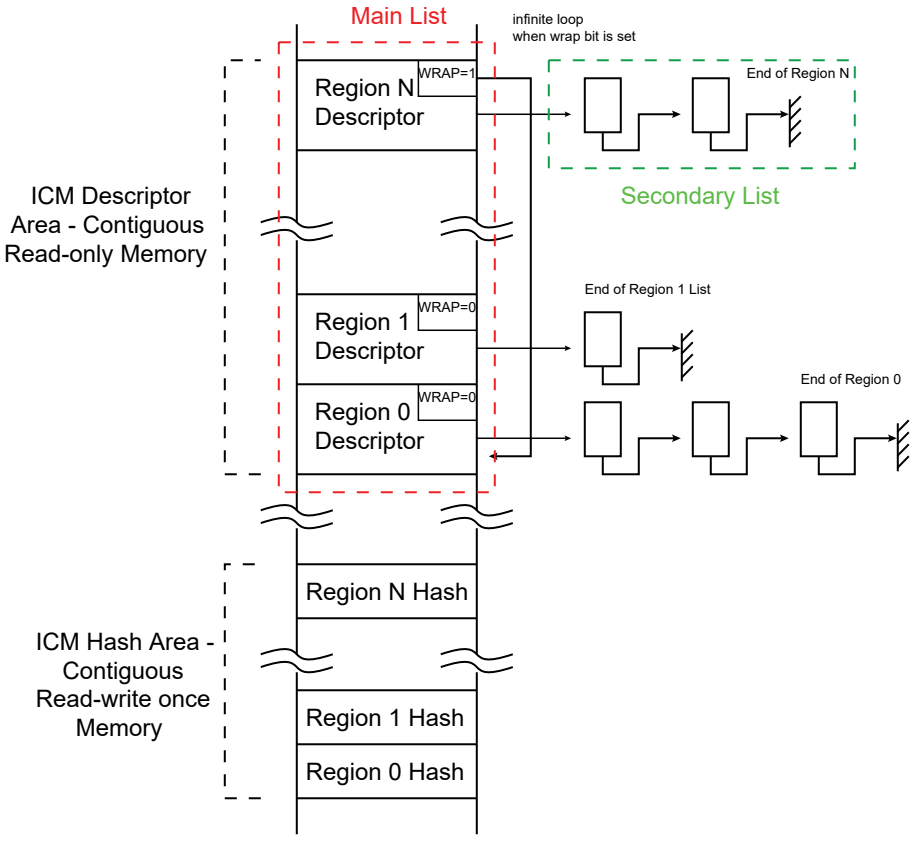
25.6.1 Overview

The Integrity Check Monitor (ICM) is a DMA controller that performs SHA-based memory hashing over memory regions. As shown in the Block Diagram (see *Block Diagram* from Related Links), it integrates a DMA interface, a Monitoring Finite State Machine (FSM), an integrity scheduler, a set of context registers, a SHA engine, an interface for configuration and status registers.

The SHA engine requires a message padded according to FIPS180-4 specification when used as a SHA calculation unit only. Otherwise, if the ICM is used as an integrated check for memory content, the padding is not mandatory. The SHA module produces an N-bit message digest each time a block is read and a processing period ends. N is 160 for SHA1, 256 for SHA256.

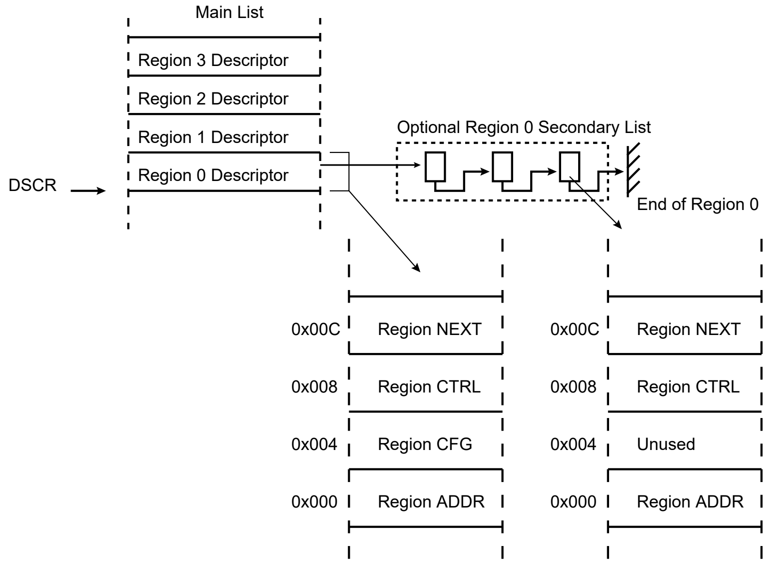
When the ICM module is enabled, it sequentially retrieves a circular list of region descriptors from the memory (Main List described in the following figure). Up to four regions may be monitored. Each region descriptor is composed of four words indicating the layout of the memory region (see *Region Descriptor Structure* from Related Links). It also contains the hashing engine configuration on a per region basis. As soon as the descriptor is loaded from the memory and context registers are updated with the data structure, the hashing operation starts. A programmable number of blocks (see TRSIZE field of the RCTRL structure member) is transferred from the memory to the SHA engine. When the desired number of blocks have transferred, the digest is either moved to memory (Write Back function) or compared with a digest reference located in the system memory (Compare function). If a digest mismatch occurs, an interrupt is triggered if enabled. The ICM module parses through the region descriptor list until the end of the list, marked by an end of list bit set to one. To continuously monitor the list of regions, the WRAP bit must be set to one in the last data structure, and EOM must be cleared.

Figure 25-2. ICM Region Descriptor and Hash Areas



Each region descriptor supports gathering of data through the use of the Secondary List. Unlike the Main List, the Secondary List cannot modify the configuration attributes of the region. When the end of the Secondary List is encountered, the ICM returns to the Main List. Memory integrity monitoring can be considered a background service, and the mandatory bandwidth is very limited. To limit the ICM memory bandwidth, use the BBC field of the CFG register to control the ICM memory load.

Figure 25-3. Region Descriptor



Related Links

25.3. Block Diagram
25.6.3. Region Descriptor Structure

25.6.2 ICM Hash Area

The ICM Hash Area is a contiguous area of system memory that the controller and the processor can access. The physical location is configured in the ICM hash area start address register. This address is a multiple of bytes. If the CDWBN bit of the context register is cleared (i.e., Write Back activated), the ICM controller performs a digest write operation at the following starting location: *(HASH) + (RID<<), where RID is the current region context identifier. If the CDWBN bit of the context register is set (i.e., Digest Comparison activated), the ICM controller performs a digest read operation at the same address.

25.6.2.1 Message Digest Example

Considering the following 512 bits message (example given in FIPS 180-4):

“61626380000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000018”

The message is written to memory in a Little Endian (LE) system architecture.

Memory Address	Address Offset / Byte Lane			
	0x3 / 31:24	0x2 / 23:16	0x1 / 15:8	0x0 / 7:0
0x000	80	63	62	61
0x004–0x038	00	00	00	00
0x03C	18	00	00	00

The digest is stored at the memory location pointed at by the ICM_HASH pointer with a Region Offset.

Memory Address	Address Offset / Byte Lane			
	0x3 / 31:24	0x2 / 23:16	0x1 / 15:8	0x0 / 7:0
0x000	36	3e	99	a9
0x004	6a	81	06	47
0x008	71	25	3e	ba
0x00C	6c	c2	50	78
0x010	9d	d8	d0	9c

Memory Address	Address Offset / Byte Lane			
	0x3 / 31:24	0x2 / 23:16	0x1 / 15:8	0x0 / 7:0
0x000	22	7d	09	23
0x004	22	d8	05	34
0x008	77	a4	42	86
0x00C	b3	55	a2	bd
0x010	e4	bc	ad	2a
0x014	f7	b3	a0	bd
0x018	a7	9d	6c	e3

Memory Address	Address Offset / Byte Lane			
	0x3 / 31:24	0x2 / 23:16	0x1 / 15:8	0x0 / 7:0
0x000	bf	16	78	ba
0x004	ea	cf	01	8f
0x008	de	40	41	41
0x00C	23	22	ae	5d
0x010	a3	61	03	b0
0x014	9c	7a	17	96
0x018	61	ff	10	b4
0x01C	ad	15	00	f2

Considering the following 1024 bits message (example given in FIPS 180-4):

“61626380000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000018”

The message is written to memory in a Little Endian (LE) system architecture.

Memory Address	Address Offset / Byte Lane			
	0x3 / 31:24	0x2 / 23:16	0x1 / 15:8	0x0 / 7:0
0x000	80	63	62	61
0x004–0x078	00	00	00	00

.....continued				
Memory Address	**Address Offset / Byte Lane**			
	0x3 / 31:24	**0x2 / 23:16**	**0x1 / 15:8**	**0x0 / 7:0**
0x07C	18	00	00	00

25.6.3 Region Descriptor Structure

The ICM Region Descriptor Area is a contiguous area of system memory that the controller and the processor can access. When the ICM controller is activated, the controller performs a descriptor fetch operation at the DSCR address. If the Main List contains more than one descriptor (i.e., more than one region is to be moderated), the fetch address is DSCR + RID<<4, where RID is the region identifier.

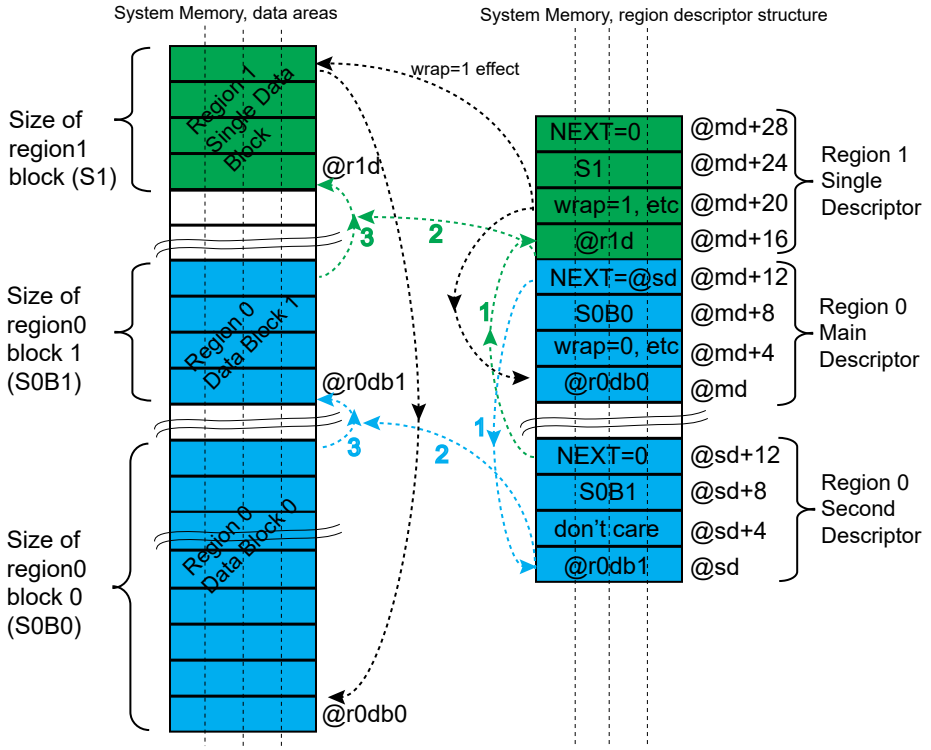
Table 25-1. Region Descriptor Structure (Main List)

Offset	Structure Member	Name
DSCR+0x00+RID*(0x10)	ICM Region Start Address	RADDR
DSCR+0x04+RID*(0x10)	ICM Region Configuration	RCFG
DSCR+0x08+RID*(0x10)	ICM Region Control	RCTRL
DSCR+0x0C+RID*(0x10)	ICM Region Next Address	RNEXT

Example 25-1. ICM Monitoring of 3 Memory Data Blocks (Defined as 2 Regions)

The following figure shows the mandatory ICM settings to monitor three memory data blocks of the system memory (defined as two regions), with one region being not contiguous (two separate areas) and one contiguous memory area. For each said region, the SHA algorithm may be independently selected (different for each region). The wrap allows continuous monitoring.

Figure 25-4. Example – Monitoring of 3 Memory Data Blocks (Defined as 2 Regions)



25.6.3.1 Region Descriptor Structure Overview

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	RADDR	7:0	RADDR[7:0]							
		15:8	RADDR[15:8]							
		23:16	RADDR[23:16]							
		31:24	RADDR[31:24]							
0x04	RCFG	7:0	WCIEN	BEIEN	DMIEN	RHIEN		EOM	WRAP	CDWBN
		15:8		ALGO[2:0]				PROCDLY	SUIEN	ECIEN
		23:16								
		31:24								
0x08	RCTRL	7:0	TRSIZE[7:0]							
		15:8	TRSIZE[15:8]							
		23:16								
		31:24								
0x0C	RNEXT	7:0								
		15:8								
		23:16								
		31:24								

25.6.3.1.1 Region Start Address Structure Member

Name: RADDR
Offset: 0x00
Reset: 0x00000000
Property: Read/Write

Bit	31	30	29	28	27	26	25	24
	RADDR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – RADDR[31:0] Region Start Address
This field indicates the first byte address of the region

25.6.3.1.2 Region Configuration Structure Member

Name: RCFG
Offset: 0x04
Reset: 0x00000000
Property: Read/Write

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
		ALGO[2:0]				PROCDLY	SUIEN	ECIEN
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	1	1

Bit	7	6	5	4	3	2	1	0
	WCIEN	BEIEN	DMIEN	RHIEN		EOM	WRAP	CDWBN
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	1	1	1	1		0	0	0

Bits 14:12 – ALGO[2:0] User SHA Algorithm

Value	Name	Description
0	SHA1	SHA1 algorithm processed
1	SHA256	SHA256 algorithm processed
Other	-	Reserved

Bit 10 – PROCDLY Processing Delay

For a given SHA algorithm, the runtime period has two possible lengths:

Table 25-2. SHA Processing Runtime Periods

Algorithm	SHORTEST [number of cycles]	LONGEST [number of cycles]
SHA1	85	209
SHA256	72	194

Value	Name	Description
0	SHORTEST	SHA processing runtime is the shortest one
1	LONGEST	SHA processing runtime is the longest one

Bit 9 – SUIEN Monitoring Status Updated Condition Interrupt Enable

0: The RSU flag is set when the corresponding descriptor is loaded from memory to ICM.
1: The RSU flag remains cleared even if the condition is met.

Bit 8 – ECIEN End Bit Condition Interrupt Enable

0: The REC flag is set when the descriptor having the EOM bit set is processed.
1: The REC flag remains cleared even if the setting condition is met.

Bit 7 – WCIEN Wrap Condition Interrupt Disable

0: The RWC flag is set when the WRAP

1: The RWC flag remains cleared even if the setting condition is met.

Bit 6 – BEIEN Bus Error Interrupt Disable
0: The flag is set when an error is reported on the system bus by the bus MATRIX.
1: The flag remains cleared even if the setting condition is met.

Bit 5 – DMIEN Digest Mismatch Interrupt Disable
0: The RBE flag is set when the hash value just calculated from the processed region dffers from expected hash value.
1: The RBE flag remains cleared even if the setting condition is met.

Bit 4 – RHIEN Region Hash Completed Interrupt Disable
0: The RHC flag is set when the field NEXT = 0 in a descriptor of the main or second list.
1: The RHC flag remains cleared even if the setting condition is met.

Bit 2 – EOM End of Monitoring
0: The current descriptor does not terminate the monitoring.
1: The current descriptor terminates the Main List. WRAP bit value has no effect.

Bit 1 – WRAP Wrap Command
0: The next region descriptor address loaded is the current region identifier descriptor address incremented by 0x10.
1: The next region descriptor address loaded is DSCR.

Bit 0 – CDWBN Compare Digest or Write Back Digest
0: The digest is written to the Hash area.
1: The digest value is compared to the digest stored in the Hash area.

25.6.3.1.3 Region Control Structure Member

Name: RCTRL
Offset: 0x08
Reset: 0x00000000
Property: R/W

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	TRSIZE[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:0 – TRSIZE[15:0] Transfer Size for the Current Chunk of Data

25.6.3.1.4 Region Next Address Structure Member

Name: RNEXT
Offset: 0x0C
Reset: 0x00000000
Property: Read/Write

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

25.6.4 Using ICM as an SHA Engine

The ICM can be configured to only calculate a SHA1, SHA256 digest value.

25.6.4.1 Settings for Simple SHA Calculation

The start address of the system memory containing the data to hash must be configured in the transfer descriptor of the DMA embedded in the ICM.

The transfer descriptor is a system memory area integer multiple of 4 x 32-bit word and the start address of the descriptor must be configured in DSCR (the start address must be aligned on 64-bytes; six LSB must be cleared). If the data to hash is already padded according to SHA standards, only a single descriptor is required, and the EOM bit of RCFGn must be written to '1'. If the data to hash does not contain a padding area, it is possible to define the padding area in another system memory location, the ICM can be configured to automatically jump from a memory area to another one by writing the descriptor register RNEXT with a value that differs from 0. Writing the RNEXT register with the start address of the padding area forces the ICM to concatenate both areas, thus providing the SHA result from the start address of the hash area configured in HASH.

Whether the system memory is configured as a single or multiple data block area, the bits CDWBN and WRAP must be cleared in the region descriptor structure member RCFGn. The bits WBDIS, EOMDIS, SLBDIS must be cleared in CFG.

Write the bits RHIEN and ECIEN in the Region Configuration Structure Member (RCFGn) to '0':

- The flag RHC[i], 'i' being the region index, is set (if RHIEN is '0') when the hash result is available at address defined in HASH.
- The flag REC[i], 'i' being the region index, is set (if ECIEN is '0') when the hash result is available at the address defined in HASH.

An interrupt is generated if the bit RHC[i] is written to '1' in the IER (if RHC[i] is set in RCTRL of region i) or if the bit REC[i] is written to '1' in the IER (if REC[i] is set in RCTRL of region i).

25.6.4.2 Processing Period

The SHA engine processing period can be configured by writing to the Region Configuration Structure Member register (RCFGn).

The short processing period allows to allocate bandwidth to the SHA module whereas the long processing period allocates more bandwidth on the system bus to other applications.

In SHA mode, the shortest processing period is 85 clock cycles + 2 clock cycles for start command synchronization. The longest period is 209 clock cycles + 2 clock cycles.

In SHA256 mode, the shortest processing period is 72 clock cycles + 2 clock cycles for start command synchronization. The longest period is 194 clock cycles + 2 clock cycles.

25.6.5 ICM Automatic Monitoring Mode

The ASCD bit of the CFG register is used to activate the ICM Automatic Mode. When CFG.ASCD is set, the ICM performs the following actions:

- The ICM controller passes through the Main List once with CDWBN bit in RCFGn at '0' (in other words, Write Back activated) and EOM bit in the RCFGn context register at '0'.
- When RCFGn.WRAP=1, the ICM controller enters active monitoring, with CDWBN bit in context register now set, and EOM bit in context register cleared. Writing to the CDWBN and EOM bits in RCFGn has no effect.

25.6.6 ICM Configuration Parameters

Transfer Type		Main List	RCFG			RNEXT	Comments
			CDWBN	WRAP	EOM	NEXT	
Single Region	Contiguous list of blocks Digest written to memory Monitoring disabled	1 item	0	0	1	0	The Main List contains only one descriptor. The Secondary List is empty for that descriptor. The digest is computed and saved to memory.
	Non-contiguous list of blocks Digest written to memory Monitoring disabled	1 item	0	0	1	Secondary List address of the current region identifier	The Main List contains only one descriptor. The Secondary List describes the layout of the non-contiguous region.
	Contiguous list of blocks Digest comparison enabled Monitoring enabled	1 item	1	1	0	0	When the hash computation is terminated, the digest is compared with the one saved in memory.

.....continued

Transfer Type		Main List	RCFG			RNEXT	Comments
			CDWBN	WRAP	EOM	NEXT	
Multiple Regions	Contiguous list of blocks Digest written to memory Monitoring disabled	More than one item	0	0	1 for the last, 0 otherwise	0	ICM passes through the list once.
	Contiguous list of blocks Digest comparison is enabled Monitoring is enabled	More than one item	1	1 for the last, 0 otherwise	0	0	ICM performs active monitoring of the regions. If a mismatch occurs, an interrupt is raised.
	Non-contiguous list of blocks Digest is written to memory Monitoring is disabled	More than one item	0	0	1	Secondary List address	ICM performs hashing and saves digests to the Hash area.
	Non-contiguous list of blocks Digest comparison is enabled Monitoring is enabled	More than one item	1	1	0	Secondary List address	ICM performs data gathering on a per region basis.

25.6.7 Security Features

When an undefined register access occurs, the URAD bit in the Interrupt Status Register (ISR) is set if unmasked. Its source is then reported in the Undefined Access Status Register (UASR). Only the first undefined register access is available through the UASR.URAT field.

Several kinds of unspecified register accesses can occur:

- Unspecified structure member set to one detected when the descriptor is loaded
- Configuration register (CFG) modified during active monitoring
- Descriptor register (DSCR) modified during active monitoring
- Hash register (HASH) modified during active monitoring
- Write-only register read access

The URAD bit and the URAT field can only be reset by writing a '1' to the CTRL.SWRST bit.

25.7 Register Summary - ICM

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CFG	7:0	BBC[3:0]					SLBDIS	EOMDIS	WBDIS
		15:8	UALGO[2:0]			UIHASH			DUALBUFF	ASCD
		23:16								
		31:24								
0x04	CTRL	7:0	REHASH[3:0]					SWRST	DISABLE	ENABLE
		15:8	RMEN[3:0]				RMDIS[3:0]			
		23:16								
		31:24								
0x08	SR	7:0								ENABLE
		15:8	RMDIS[3:0]				RAWRMDIS[3:0]			
		23:16								
		31:24								
0x0C ... 0x0F	Reserved									
0x10	IER	7:0	RDM[3:0]				RHC[3:0]			
		15:8	RWC[3:0]				RBE[3:0]			
		23:16	RSU[3:0]				REC[3:0]			
		31:24								URAD
0x14	IDR	7:0	RDM[3:0]				RHC[3:0]			
		15:8	RWC[3:0]				RBE[3:0]			
		23:16	RSU[3:0]				REC[3:0]			
		31:24								URAD
0x18	IMR	7:0	RDM[3:0]				RHC[3:0]			
		15:8	RWC[3:0]				RBE[3:0]			
		23:16	RSU[3:0]				REC[3:0]			
		31:24								URAD
0x1C	ISR	7:0	RDM[3:0]				RHC[3:0]			
		15:8	RWC[3:0]				RBE[3:0]			
		23:16	RSU[3:0]				REC[3:0]			
		31:24								URAD
0x20	UASR	7:0						URAT[2:0]		
		15:8								
		23:16								
		31:24								
0x24 ... 0x2F	Reserved									
0x30	DSCR	7:0	DASA[1:0]							
		15:8	DASA[9:2]							
		23:16	DASA[17:10]							
		31:24	DASA[25:18]							
0x34	HASH	7:0								
		15:8								
		23:16								
		31:24								
0x38	UIHVALx0	7:0	VAL[7:0]							
		15:8	VAL[15:8]							
		23:16	VAL[23:16]							
		31:24	VAL[31:24]							
0x3C	UIHVALx1	7:0	VAL[7:0]							
		15:8	VAL[15:8]							
		23:16	VAL[23:16]							
		31:24	VAL[31:24]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x40	UIHVALx2	7:0	VAL[7:0]							
		15:8	VAL[15:8]							
		23:16	VAL[23:16]							
		31:24	VAL[31:24]							
0x44	UIHVALx3	7:0	VAL[7:0]							
		15:8	VAL[15:8]							
		23:16	VAL[23:16]							
		31:24	VAL[31:24]							
0x48	UIHVALx4	7:0	VAL[7:0]							
		15:8	VAL[15:8]							
		23:16	VAL[23:16]							
		31:24	VAL[31:24]							
0x4C	UIHVALx5	7:0	VAL[7:0]							
		15:8	VAL[15:8]							
		23:16	VAL[23:16]							
		31:24	VAL[31:24]							
0x50	UIHVALx6	7:0	VAL[7:0]							
		15:8	VAL[15:8]							
		23:16	VAL[23:16]							
		31:24	VAL[31:24]							
0x54	UIHVALx7	7:0	VAL[7:0]							
		15:8	VAL[15:8]							
		23:16	VAL[23:16]							
		31:24	VAL[31:24]							

25.8 Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to 22.5.7. Register Access Protection.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

25.8.1 Configuration Register

Name: CFG
Offset: 0x00
Reset: 0x0
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	UALGO[2:0]			UIHASH			DUALBUFF	ASCD
Access	-	-	-	-			-	R/W
Reset	0	0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	BBC[3:0]					SLBDIS	EOMDIS	WBDIS
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bits 15:13 – UALGO[2:0] User SHA Algorithm

Value	Name	Description
0	SHA1	SHA1 algorithm processed
1	SHA256	SHA256 algorithm processed
Other	-	Reserved

Bit 12 – UIHASH User Initial Hash Value

Value	Description
0	The secure hash standard provides the initial hash value.
1	The initial hash value is programmable. Field UALGO provides the SHA algorithm. The ALGO field of the RCFGn structure member has no effect.

Bit 9 – DUALBUFF Dual Input Buffer

Value	Description
0	Dual Input buffer mode is disabled.
1	Dual Input buffer mode is enabled (Better performances, higher bandwidth required on system bus).

Bit 8 – ASCD Automatic Switch To Compare Digest

Value	Description
0	Automatic mode is disabled.
1	When this mode is enabled, the ICM controller automatically switches to active monitoring after the first Main List pass. Both CDWBN and WBDIS bits have no effect. A '1' must be written to the End of Monitoring bit in the Region Configuration register (RCFG.EOM) to terminate the monitoring.

Bits 7:4 – BBC[3:0] Bus Burden Control
This field is used to control the burden of the ICM system bus. The number of system clock cycles between the end of the current processing and the next block transfer is set to 2^{BBC}. Up to 32768 cycles can be inserted.

Bit 2 – SLBDIS Secondary List Branching Disable

Value	Description
0	Branching to the Secondary List is permitted.
1	Branching to the Secondary List is forbidden. The NEXT field of the RNEXT structure member has no effect and is always considered as zero.

Bit 1 – EOMDIS End of Monitoring Disable

Value	Description
0	End of Monitoring is permitted.
1	End of Monitoring is forbidden. The EOM bit of the RCFG structure member has no effect.

Bit 0 – WBDIS Write Back Disable

When the Automatic Switch to Compare Digest bit of this register (CFG.ASCD) is written to '1', this bit value has no effect.

Value	Description
0	Write Back Operations are permitted.
1	Write Back Operations are forbidden: Context register CDWBN bit is internally set to '1' and cannot be modified by a linked list element. The CDWBN bit of the RCFG structure member has no effect.

25.8.2 Control Register

Name: CTRL
Offset: 0x04
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	RMEN[3:0]				RMDIS[3:0]			
Access	W	W	W	W	W	W	W	W
Reset								

Bit	7	6	5	4	3	2	1	0
	REHASH[3:0]					SWRST	DISABLE	ENABLE
Access	W	W	W	W		W	W	W
Reset							0	0

Bits 15:12 – RMEN[3:0] Region Monitoring Enable

Value	Description
0	No effect.
1	When bit RMEN[i] is written to '1', the monitoring of region with identifier i is activated.

Bits 11:8 – RMDIS[3:0] Region Monitoring Disable

Value	Description
0	No effect.
1	When REHASH[i] is written to '1', Region i digest is re-computed. This bit is only available when region monitoring is disabled.

Bits 7:4 – REHASH[3:0] Recompute Internal Hash

Value	Description
0	No effect.
1	When REHASH[i] is written to '1', Region i digest is re-computed. This bit is only available when region monitoring is disabled.

Bit 2 – SWRST Software Reset

Value	Description
0	No effect.
1	Resets the ICM controller.

Bit 1 – DISABLE ECM Disable

Value	Description
0	No effect.
1	The ICM controller is disabled. If a region is activated, the region is terminated.

Bit 0 – ENABLE ICM Enable

Value	Description
0	No effect.

Value	Description
1	The ICM controller is activated.

25.8.3 Status Register

Name: SR
Offset: 0x08
Property: Read-Only

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	RMDIS[3:0]				RAWRMDIS[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
								ENABLE
Access								R
Reset								0

Bits 15:12 – RMDIS[3:0] Region Monitoring Disabled Status

Value	Description
0	Region i is being monitored (occurs after integrity check value has been calculated and written to Hash area).
1	Region i is not being monitored.

Bits 11:8 – RAWRMDIS[3:0] Region Monitoring Disabled Raw Status

Value	Description
0	Region i monitoring has been activated by writing a 1 in RMEN[i] of CTRL
1	Region i monitoring has been deactivated by writing a 1 in RMDIS[i] of CTRL

Bit 0 – ENABLE ICM Controller Enable Register

Value	Description
0	ICM controller is disabled.
1	ICM controller is activated.

25.8.4 Interrupt Enable Register

Name: IER
Offset: 0x10
Reset: 0x00000000
Property: Write-Only

Bit	31	30	29	28	27	26	25	24
								URAD
Access								W
Reset								0

Bit	23	22	21	20	19	18	17	16
	RSU[3:0]				REC[3:0]			
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RWC[3:0]				RBE[3:0]			
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RDM[3:0]				RHC[3:0]			
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit 24 – URAD Undefined Register Access Detection Interrupt Enable
0: No effect
1: The Undefined Register Access interrupt is enabled.

Bits 23:20 – RSU[3:0] Region Status Updated Interrupt Enable
0: No effect
1: When RSU[i] is written to ‘1’, the region i Status Updated interrupt is enabled.

Bits 19:16 – REC[3:0] Region End bit Condition Detected Interrupt Enable
0: No effect
1: When REC[i] is written to ‘1’, the region i End bit Condition interrupt is enabled.

Bits 15:12 – RWC[3:0] Region Wrap Condition detected Interrupt Enable
0: No effect
1: When RWC[i] is written to ‘1’, the Region i Wrap Condition interrupt is enabled.

Bits 11:8 – RBE[3:0] Region Bus Error Interrupt Enable

Value	Description
0	No effect.
1	When RBE[i] is written to '1', the Region i Bus Error interrupt is enabled.

Bits 7:4 – RDM[3:0] Region Digest Mismatch Interrupt Enable

Value	Description
0	No effect.
1	When RDM[i] is written to '1', the Region i Digest Mismatch interrupt is enabled.

Bits 3:0 – RHC[3:0] Region Hash Completed Interrupt Enable

Value	Description
0	No effect.

Value	Description
1	When RHC[i] is written to '1', the Region i Hash Completed interrupt is enabled.

25.8.5 Interrupt Disable Register

Name: IDR
Offset: 0x14
Property: Write-Only

Bit	31	30	29	28	27	26	25	24
								URAD
Access								W
Reset								

Bit	23	22	21	20	19	18	17	16
	RSU[3:0]				REC[3:0]			
Access	W	W	W	W	W	W	W	W
Reset								

Bit	15	14	13	12	11	10	9	8
	RWC[3:0]				RBE[3:0]			
Access	W	W	W	W	W	W	W	W
Reset								

Bit	7	6	5	4	3	2	1	0
	RDM[3:0]				RHC[3:0]			
Access	W	W	W	W	W	W	W	W
Reset								

Bit 24 – URAD Undefined Register Access Detection Interrupt Disable

Value	Description
0	No effect.
1	Undefined Register Access Detection interrupt is disabled.

Bits 23:20 – RSU[3:0] Region Status Updated Interrupt Disable

Value	Description
0	No effect.
1	When RSU[i] is written to '1', the region i Status Updated interrupt is disabled.

Bits 19:16 – REC[3:0] Region End bit Condition detected Interrupt Disable

Value	Description
0	No effect.
1	When REC[i] is written to '1', the region i End bit Condition interrupt is disabled.

Bits 15:12 – RWC[3:0] Region Wrap Condition Detected Interrupt Disable

Value	Description
0	No effect.
1	When RWC[i] is written to '1', the Region i Wrap Condition interrupt is disabled.

Bits 11:8 – RBE[3:0] Region Bus Error Interrupt Disable

Value	Description
0	No effect.
1	When RBE[i] is written to '1', the Region i Bus Error interrupt is disabled.

Bits 7:4 – RDM[3:0] Region Digest Mismatch Interrupt Disable

Value	Description
0	No effect.
1	When RDM[i] is written to '1', the Region i Digest Mismatch interrupt is disabled.

Bits 3:0 – RHC[3:0] Region Hash Completed Interrupt Disable

Value	Description
0	No effect.
1	When RHC[i] is written to '1', the Region i Hash Completed interrupt is disabled.

25.8.6 Interrupt Mask Register

Name: IMR
Offset: 0x18
Reset: 0x00000000
Property: Read-Only

Bit	31	30	29	28	27	26	25	24
								URAD
Access								R
Reset								0

Bit	23	22	21	20	19	18	17	16
	RSU[3:0]				REC[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RWC[3:0]				RBE[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RDM[3:0]				RHC[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 24 – URAD Undefined Register Access Detection Interrupt Mask

Value	Description
0	The interrupt is disabled.
1	The interrupt is enabled.

Bits 23:20 – RSU[3:0] Region Status Updated Interrupt Mask

Value	Description
0	When RSU[i] is reading '0', the interrupt is disabled for region i.
1	When RSU[i] is reading '1', the interrupt is enabled for region i.

Bits 19:16 – REC[3:0] Region End bit Condition Detected Interrupt Mask

Value	Description
0	When REC[i] is reading '0', the interrupt is disabled for region i.
1	When REC[i] is reading '1', the interrupt is enabled for region i.

Bits 15:12 – RWC[3:0] Region Wrap Condition Detected Interrupt Mask

Value	Description
0	When RWC[i] is reading '0', the interrupt is disabled for region i.
1	When RWC[i] is reading '1', the interrupt is enabled for region i.

Bits 11:8 – RBE[3:0] Region Bus Error Interrupt Mask

Value	Description
0	When RBE[i] is reading '0', the interrupt is disabled for region i.
1	When RBE[i] is reading '1', the interrupt is enabled for region i.

Bits 7:4 – RDM[3:0] Region Digest Mismatch Interrupt Mask

Value	Description
0	When RDM[i] is reading '0', the interrupt is disabled for region i.

Value	Description
1	When RDM[i] is reading '1', the interrupt is enabled for region i.

Bits 3:0 – RHC[3:0] Region Hash Completed Interrupt Mask

Value	Description
0	When RHC[i] is reading '0', the interrupt is disabled for region i.
1	When RHC[i] is reading '1', the interrupt is enabled for region i.

25.8.7 Interrupt Status Register

Name: ISR
Offset: 0x1C
Reset: 0x0
Property: Read-Only

Bit	31	30	29	28	27	26	25	24
								URAD
Access								R
Reset								0

Bit	23	22	21	20	19	18	17	16
	RSU[3:0]				REC[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RWC[3:0]				RBE[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RDM[3:0]				RHC[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 24 – URAD Undefined Register Access Detection Status
The URAD bit is only reset by the SWRST bit in the CTRL register.
The Undefined Register Access Trace bit field in the Undefined Access Status Register (UASR.URAT) indicates the unspecified access type.

Value	Description
0	No undefined register access has been detected since the last SWRST.
1	At least one undefined register access has been detected since the last SWRST.

Bits 23:20 – RSU[3:0] Region Status Updated Detected
RSU[i] is set when a region status updated condition is detected.

Bits 19:16 – REC[3:0] Region End bit Condition Detected
REC[i] is set when an end bit condition is detected.

Bits 15:12 – RWC[3:0] Region Wrap Condition Detected
RWC[i] is set when a wrap condition is detected.

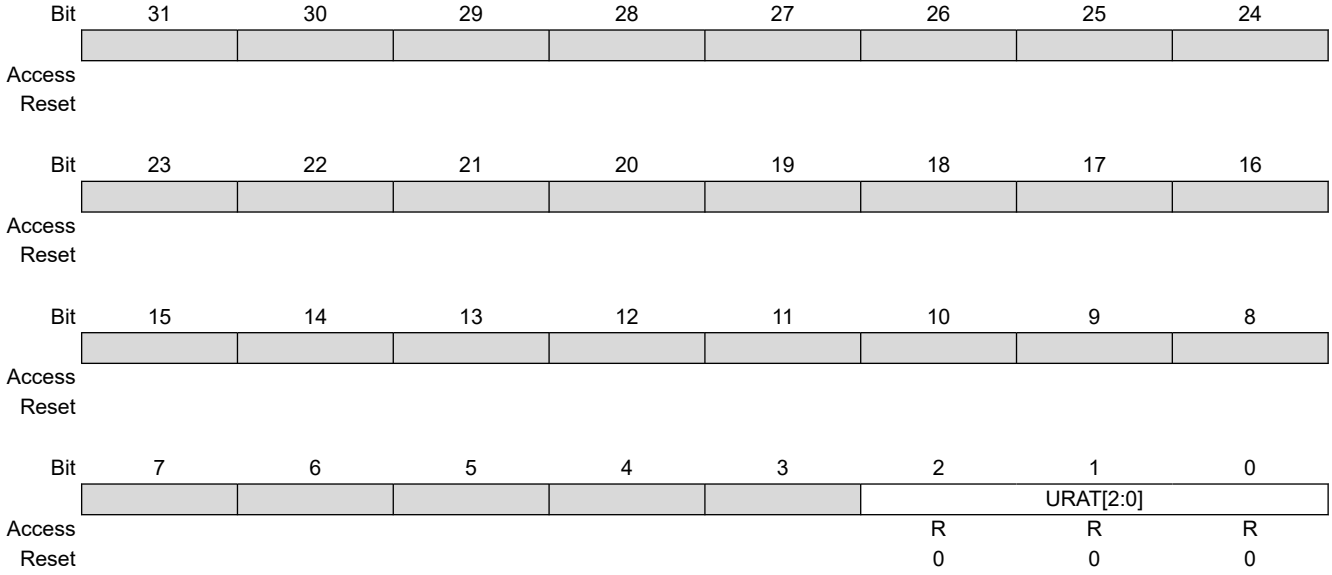
Bits 11:8 – RBE[3:0] Region Bus Error
RBE[i] is set when a bus error is detected while hashing memory region i.

Bits 7:4 – RDM[3:0] Region Digest Mismatch
RDM[i] is set when there is a digest comparison mismatch between the hash value of region i and the reference value located in the Hash Area.

Bits 3:0 – RHC[3:0] Region Hash Completed
RHC[i] is set when the ICM has completed the region with identifier i.

25.8.8 Undefined Access Status Register

Name: UASR
Offset: 0x20
Reset: 0x0
Property: Read-Only



Bits 2:0 – URAT[2:0] Undefined Register Access Trace
Only the first Undefined Register Access Trace is available through the URAT field.
The URAT field is only reset by the Software Reset bit in the Control register (CTRL.SWRST).

Value	Name	Description
0	UNSPEC_STRUCT_MEMBER	Unspecified structure member set to '1' detected when the descriptor is loaded.
1	ICM_CFG_MODIFIED	CFG modified during active monitoring.
2	ICM_DSCR_MODIFIED	DSCR modified during active monitoring.
3	ICM_HASH_MODIFIED	HASH modified during active monitoring
4	READ_ACCESS	Write-only register read access Only the first Undefined Register Access Trace is available through the URAT field. The URAT field is only reset by the SWRST bit in the CTRL register.

25.8.9 Descriptor Area Start Address Register

Name: DSCR
Offset: 0x30
Reset: 0x0
Property: -

Bit	31	30	29	28	27	26	25	24
	DASA[25:18]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DASA[17:10]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DASA[9:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:6 – DASA[25:0] Descriptor Area Start Address
The start address is a multiple of the total size of the data structure (64 bytes).

25.8.10 Hash Area Start Address Register

Name: HASH
Offset: 0x34
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

25.8.11 User Initial Hash Value Register

Name: UIHVALx
Offset: 0x38 + x*0x04 [x=0..7]
Reset: 0
Property: -

Bit	31	30	29	28	27	26	25	24
	VAL[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	VAL[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	VAL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – VAL[31:0] Initial Hash Value
When UIHASH bit of CFG register is set, the Initial Hash Value is user-programmable.
To meet the desired standard, use the following example values.
For UIHVAL0 field:

Example	Comment
0x67452301	SHA1 algorithm
0x6A09E667	SHA256 algorithm

For UIHVAL1 field:

Example	Comment
0xEFCDAB89	SHA1 algorithm
0xBB67AE85	SHA256 algorithm

For UIHVAL2 field:

Example	Comment
0x98BADCFE	SHA1 algorithm
0x3C6EF372	SHA256 algorithm

For UIHVAL3 field:

Example	Comment
0x10325476	SHA1 algorithm
0xA54FF53A	SHA256 algorithm

For UIHVAL4 field:

Example	Comment
0xC3D2E1F0	SHA1 algorithm
0x510E527F	SHA256 algorithm

For UIHVAL5 field:

Example	Comment
0x9B05688C	SHA256 algorithm

For UIHVAL6 field:

Example	Comment
0x1F83D9AB	SHA256 algorithm

For UIHVAL7 field:

Example	Comment
0x5BE0CD19	SHA256 algorithm

Example of Initial Value for SHA-1 Algorithm

Register Address	Address Offset / Byte Lane			
	0x3 / 31:24	0x2 / 23:16	0x1 / 15:8	0x0 / 7:0
0x000 UIHVAL0	01	23	45	67
0x004 UIHVAL1	89	ab	cd	ef
0x008 UIHVAL2	fe	dc	ba	98
0x00C UIHVAL3	76	54	32	10
0x010 UIHVAL4	f0	e1	d2	c3

26. Peripheral Access Controller (PAC)

26.1 Overview

The Peripheral Access Controller provides an interface for the locking and unlocking of peripheral registers within the device. It reports all violations that could happen when accessing a peripheral: write protected access, illegal access, enable protected access, access when clock synchronization or software reset is on-going. These errors are reported in a unique interrupt flag for a peripheral. The PAC module also reports errors occurring at the client bus level, when an access to a non-existing address is detected.

Notes:

1. The modules attached to the PB-PIC bridge and wireless subsystem as well as RTCC, DSCON, PUKCC and ICM are excluded from the PAC. The protection mechanism described in the System Configuration Registers (CFG) protects critical system registers (see *System Configuration Registers (CFG)* from Related Links).
2. Traditional Peripheral Access Controller (PAC) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

Related Links

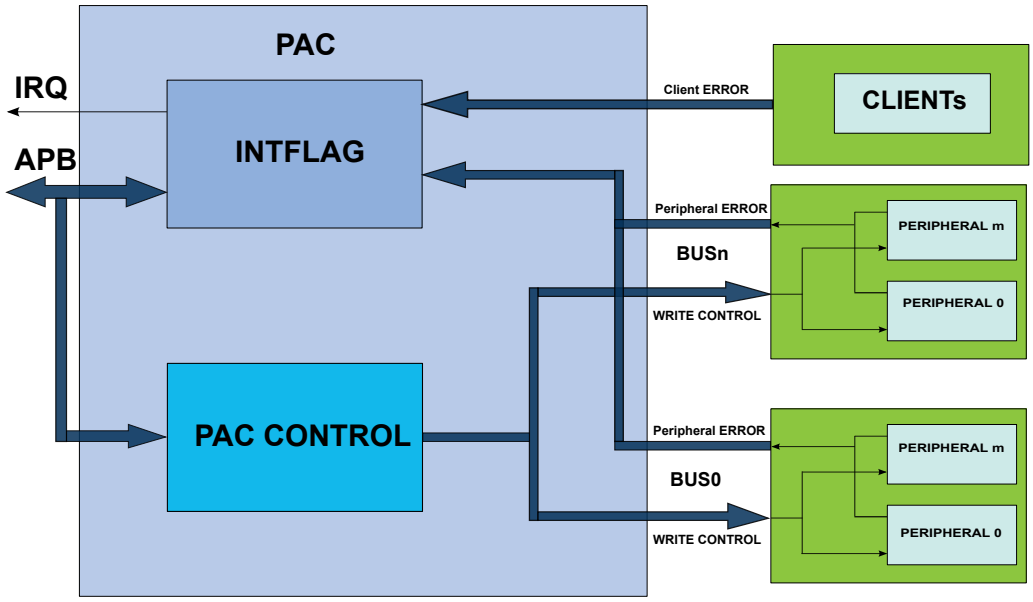
18. System Configuration and Register Locking (CFG)

26.2 Features

- Manages write protection access and reports access errors for the peripheral modules or bridges.

26.3 Block Diagram

Figure 26-1. PAC Block Diagram



26.4 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

26.4.1 IO Lines

Not applicable.

26.4.2 Power Management

The PAC can continue to operate in any Sleep mode where the selected source clock is running. The PAC interrupts can be used to wake up the device from Sleep modes. The events can trigger other operations in the system without exiting sleep modes.

26.4.3 DMA

Not applicable.

26.4.4 Interrupts

The interrupt request line is connected to the Interrupt Controller (NVIC). Using the PAC interrupt requires the Interrupt Controller to be configured first.

Table 26-1. Interrupt Lines

Instances	NVIC Line
PAC	PACERR

26.4.5 Events

The events are connected to the Event System, which may need configuration. See *Event System (EVSYS)* from Related Links.

Related Links

28. Event System (EVSYS)

26.4.6 Debug Operation

When the CPU is halted in Debug mode, write protection of all peripherals is disabled and the PAC continues normal operation.

26.4.7 Register Access Protection

All registers with write access can be write-protected optionally by the Peripheral Access Controller (PAC), except for the following PAC registers:

- Write Control (WRCTRL) register
- AHB Subordinate Bus Interrupt Flag Status and Clear (INTFLAGAHB) register
- Peripheral Interrupt Flag Status and Clear n (INTFLAG A/B/C...) registers

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the “PAC Write Protection” property in each individual register description.

Note: PAC write protection does not apply to accesses through an external debugger.

26.5 Functional Description

26.5.1 Principle of Operation

The Peripheral Access Control module allows the user to set a write protection on peripheral modules and generate an interrupt in case of a peripheral access violation. The peripheral’s protection can be set, cleared or locked at the user discretion. A set of Interrupt Flag and Status registers informs the user on the status of the violation in the peripherals. In addition, client bus errors can be also reported in the cases where reserved area is accessed by the application.

26.5.2 Basic Operation

26.5.2.1 Initialization, Enabling and Resetting

The PAC is always enabled after reset.

Only a hardware reset will reset the PAC module.

26.5.2.2 Operations

The PAC module allows the user to set, clear or lock the write protection status of all peripherals on all Peripheral Bridges, except the peripherals on PB-PIC bus.

If a peripheral register violation occurs, the Peripheral Interrupt Flag n registers (INTFLAGn) are updated to inform the user on the status of the violation in the peripherals connected to the Peripheral Bridge n (n = A,B,C ...). The corresponding Peripheral Write Control Status n register (STATUSn) gives the state of the write protection for all peripherals connected to the corresponding Peripheral Bridge n. See *Peripheral Access Errors* from Related Links.

The PAC module also report the errors occurring at client bus level when an access to reserved area is detected. AHB Subordinate Bus Interrupt Flag register (INTFLAGAHB) informs the user on the status of the violation in the corresponding client. See *AHB Subordinate Bus Errors* from Related Links.

Related Links

26.5.2.3. Peripheral Access Errors
26.5.2.6. AHB Subordinate Bus Errors

26.5.2.3 Peripheral Access Errors

The following events will generate a Peripheral Access Error:

- Protected write: To avoid unexpected writes to a peripheral's registers, each peripheral can be write protected. Only the registers denoted as “PAC Write-Protection” in the module’s datasheet can be protected. If a peripheral is not write protected, write data accesses are performed normally. If a peripheral is write protected and if a write access is attempted, data will not be written and peripheral returns an access error. The corresponding interrupt flag bit in the INTFLAGn register will be set.
- Illegal access: Access to an unimplemented register within the module.
- Synchronized write error: For write-synchronized registers an error will be reported if the register is written while a synchronization is ongoing.

When any of the INTFLAGn registers bit are set, an interrupt will be requested if the PAC interrupt enable bit is set.

26.5.2.4 Write Access Protection Management

Peripheral access control can be enabled or disabled by writing to the WRCTRL register.

The data written to the WRCTRL register is composed of two fields; WRCTRL.PERID and WRCTRL.KEY. The WRCTRL.PERID is an unique identifier corresponding to a peripheral. The WRCTRL.KEY is a key value that defines the operation to be done on the control access bit. These operations can be “clear protection”, “set protection” and “set and lock protection bit”.

The “clear protection” operation will remove the write access protection for the peripheral selected by WRCTRL.PERID. Write accesses are allowed for the registers in this peripheral.

The “set protection” operation will set the write access protection for the peripheral selected by WRCTRL.PERID. Write accesses are not allowed for the registers with write protection property in this peripheral.

The “set and lock protection” operation will set the write access protection for the peripheral selected by WRCTRL.PERID and locks the access rights of the selected peripheral registers. The write access protection will only be cleared by a hardware reset.

The peripheral access control status can be read from the corresponding STATUSn register.

26.5.2.5 Write Access Protection Management Errors

Only word-wise writes to the WRCTRL register will effectively change the access protection. Other type of accesses will have no effect and will cause a PAC write access error. This error is reported in the INTFLAGA.PAC bit.

PAC also offers an additional safety feature for correct program execution with an interrupt generated on double write clear protection or double write set protection. If a peripheral is write protected and a subsequent set protection operation is detected then the PAC returns an error, and similarly for a double clear protection operation.

In addition, an error is generated when writing a “set and lock” protection to a write-protected peripheral or when a write access is done to a locked set protection. This can be used to ensure that the application follows the intended program flow by always following a write protect with an unprotect and conversely. However in applications where a write protected peripheral is used in several contexts, for example, interrupt, care must be taken so that either the interrupt can not happen while the main application or other interrupt levels manipulates the write protection status or when the interrupt handler needs to unprotect the peripheral based on the current protection status by reading the STATUS register.

The errors generated while accessing the PAC module registers (for example, key error, double protect error and so on) will set the INTFLAGA.PAC flag.

26.5.2.6 AHB Subordinate Bus Errors

The PAC module reports errors occurring at the AHB Subordinate bus level. These errors are generated when an access is performed at an address where no subordinate (bridge or peripheral) is mapped. These errors are reported in the corresponding bits of the INTFLAGAHB register.

26.5.2.7 Generating Events

The PAC module can also generate an event when any of the Interrupt Flag registers bit are set. To enable the PAC event generation, the control bit EVCTRL.ERREO must be set a '1'.

26.5.3 DMA Operation

Not applicable.

26.5.4 Interrupts

The PAC has the following interrupt source:

- Error (ERR): Indicates that a peripheral access violation occurred in one of the peripherals controlled by the PAC module, or a bridge error occurred in one of the bridges reported by the PAC
  - This interrupt is a synchronous wake-up source

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear (INTFLAGAHB and INTFLAGn) registers is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a '1' to the corresponding bit in the Interrupt Enable Set (INTENSET) register, and disabled by writing a '1' to the corresponding bit in the Interrupt Enable Clear (INTENCLR) register.

An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled, or the PAC is reset. All interrupt requests from the peripheral are ORed together on system level to generate one combined interrupt request to the NVIC. The user must read the INTFLAGAHB and INTFLAGn registers to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated.

26.5.5 Events

The PAC can generate the following output event:

- Error (ERR): Generated when one of the interrupt flag registers bits is set

Writing a '1' to an Event Output bit in the Event Control Register (EVCTRL.ERREO) enables the corresponding output event. Writing a '0' to this bit disables the corresponding output event.

26.5.6 Sleep Mode Operation

In Sleep mode, the PAC is kept enabled if an available bus host (CPU, DMA) is running. The PAC will continue to catch access errors from the module and generate interrupts or events.

26.5.7 Synchronization

Not applicable.

26.6 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	WRCTRL	7:0	PERID[7:0]							
		15:8	PERID[15:8]							
		23:16	KEY[7:0]							
		31:24								
0x04	EVCTRL	7:0								ERREO
0x05 ... 0x07	Reserved									
0x08	INTENCLR	7:0								ERR
0x09	INTENSET	7:0								ERR
0x0A ... 0x0F	Reserved									
0x10	INTFLAGAHB	7:0	PBBB	PBAB	PFLASH	CFLASH	SRAM3	SRAM2	SRAM1	SRAM0
		15:8						QSPI	PBPICB	PBCB
		23:16								
		31:24								
0x14	INTFLAGA	7:0	TC2	TC1	TC0	SERCOM1	SERCOM0	EIC	FREQM	PAC
		15:8					TCC2	TCC1	TCC0	TC3
		23:16								
		31:24								
0x18	INTFLAGB	7:0				RAMECC	EVSYS	DMAC		DSU
		15:8								
		23:16								
		31:24								
0x1C	INTFLAGC	7:0	AC	CCL						
		15:8								
		23:16								
		31:24								
0x20 ... 0x33	Reserved									
0x34	STATUSA	7:0	TC2	TC1	TC0	SERCOM1	SERCOM0	EIC	FREQM	PAC
		15:8						TCC1		TC3
		23:16								
		31:24								
0x38	STATUSB	7:0				RAMECC	EVSYS	DMAC		
		15:8								
		23:16								
		31:24								
0x3C	STATUSC	7:0	AC	CCL		SERCOM3	SERCOM2			
		15:8								
		23:16								
		31:24								

26.7 Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the "PAC Write-Protection" property in each individual register description. For details, refer to the related links.

26.7.1 Write Control

Name: WRCTRL
Offset: 0x00
Reset: 0x00000000
Property: –

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	KEY[7:0]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PERID[15:8]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PERID[7:0]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – KEY[7:0] Peripheral Access Control Key
These bits define the peripheral access control key:

Value	Name	Description
0x0	OFF	No action
0x1	CLEAR	Clear the peripheral write control
0x2	SET	Set the peripheral write control
0x3	LOCK	Set and lock the peripheral write control until the next hardware reset

Bits 15:0 – PERID[15:0] Peripheral Identifier
The PERID represents the peripheral whose control is changed using the WRCTRL.KEY. The Peripheral Identifier is calculated following formula:
$PERID = 32^{*}\ \mathrm{BridgeNumber} + \mathrm{N}$

Where BridgeNumber represents the Peripheral Bridge Number (0 for Peripheral Bridge A, 1 for Peripheral Bridge B, etc). N represents the peripheral index from the respective Bridge Number:

Table 26-2. PERID Values

Periph. Bridge Name	BridgeNumber	PERID Values
A	0	0+N
B	1	32+N
C	2	64+N
D	3	96+N

Note: GMAC, ICM, SDHC, CAN and PCC peripherals do not support that feature.

26.7.2 Event Control

Name: EVCTRL
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								ERREO
Access								RW
Reset								0

Bit 0 – ERREO Peripheral Access Error Event Output
This bit indicates if the Peripheral Access Error Event Output is enabled or disabled. When enabled, an event will be generated when one of the interrupt flag registers bits (INTFLAGAHB, INTFLAGn) is set:

Value	Description
0	Peripheral Access Error Event Output is disabled.
1	Peripheral Access Error Event Output is enabled.

26.7.3 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x08
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Bit	7	6	5	4	3	2	1	0
								ERR
Access								RW
Reset								0

Bit 0 – ERR Peripheral Access Error Interrupt Disable
This bit indicates that the Peripheral Access Error Interrupt is enabled and an interrupt request will be generated when one of the interrupt flag registers bits (INTFLAGAHB, INTFLAGn) is set:
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Peripheral Access Error interrupt Enable bit and disables the corresponding interrupt request.

Value	Description
0	Peripheral Access Error interrupt is disabled.
1	Peripheral Access Error interrupt is enabled.

26.7.4 Interrupt Enable Set

Name: INTENSET
Offset: 0x09
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
								ERR
Access								RW
Reset								0

Bit 0 – ERR Peripheral Access Error Interrupt Enable
This bit indicates that the Peripheral Access Error Interrupt is enabled and an interrupt request will be generated when one of the interrupt flag registers bits (INTFLAGAHB, INTFLAGn) is set:
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Peripheral Access Error interrupt Enable bit and enables the corresponding interrupt request.

Value	Description
0	Peripheral Access Error interrupt is disabled.
1	Peripheral Access Error interrupt is enabled.

26.7.5 Bridge Interrupt Flag Status

Name: INTFLAGAHB
Offset: 0x10
Reset: 0x00000000
Property: -

These flags are cleared by writing a '1' to the corresponding bit.

These flags are set when an access error is detected by the corresponding AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
						QSPI	PBPICB	PBCB
Access						RW	RW	RW
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
	PBBB	PBAB	PFLASH	CFLASH	SRAM3	SRAM2	SRAM1	SRAM0
Access	RW	RW	RW	RW	RW	U	U	RW
Reset	0	0	0	0	0	0	0	0

Bit 10 – QSPI Interrupt Flag for QSPI
This flag is set when an access error is detected by the QSPI AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' has no effect.
Writing a '1' to this bit will clear the QSPI interrupt flag.

Bit 9 – PBPICB Interrupt Flag for PBPICB (PB-PIC-Bridge)
This flag is set when an access error is detected by the PBPICB AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' has no effect.
Writing a '1' to this bit will clear the PBPICB interrupt flag.

Bit 8 – PBCB Interrupt Flag for PBCB (PB-Bridge-C)
This flag is set when an access error is detected by the PBCB AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' has no effect.
Writing a '1' to this bit will clear the PBCB interrupt flag.

Bit 7 – PBBB Interrupt Flag for PBBB (PB-Bridge-B)
This flag is set when an access error is detected by the PBBB AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' has no effect.
Writing a '1' to this bit will clear the PBBB interrupt flag.

Bit 6 – PBAB Interrupt Flag for HPB1 (PB-Bridge-A)
This flag is set when an access error is detected by the PBAB AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' has no effect.
Writing a '1' to this bit will clear the PBAB interrupt flag.

Bit 5 – PFLASH Interrupt Flag for PFLASH (Peripheral Flash)
This flag is set when an access error is detected by the PFLASH AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' has no effect.
Writing a '1' to this bit will clear the PFLASH interrupt flag.

Bit 4 – CFLASH Interrupt Flag for CFLASH (CPU Flash)
This flag is set when an access error is detected by the CFLASH AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' has no effect.
Writing a '1' to this bit will clear the CFLASH interrupt flag.

Bit 3 – SRAM3 Interrupt Flag for SRAM3
This flag is set when an access error is detected by the SRAM3 AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' has no effect.
Writing a '1' to this bit will clear the SRAM3 interrupt flag.

Bit 2 – SRAM2 Interrupt Flag for SRAM2
This flag is set when an access error is detected by the SRAM2 AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' has no effect.
Writing a '1' to this bit will clear the SRAM2 interrupt flag.

Bit 1 – SRAM1 Interrupt Flag for SRAM1
This flag is set when an access error is detected by the SRAM1 AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' has no effect.
Writing a '1' to this bit will clear the SRAM1 interrupt flag.

Bit 0 – SRAM0 Interrupt Flag for SRAM0
This flag is set when an access error is detected by the SRAM0 AHB Subordinate, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' has no effect.
Writing a '1' to this bit will clear the SRAM0 interrupt flag.

26.7.6 Peripheral Interrupt Flag Status – Bridge A

Name: INTFLAGA
Offset: 0x14
Reset: 0x00000000
Property: –

These flags are set when a Peripheral Access Error occurs while accessing the peripheral associated with the respective INTFLAGx bit, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Writing a '0' to these bits has no effect.

Writing a '1' to these bits will clear the corresponding INTFLAGx interrupt flag.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					TCC2	TCC1	TCC0	TC3
Access					RW	RW	RW	RW
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TC2	TC1	TC0	SERCOM1	SERCOM0	EIC	FREQM	PAC
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit 11 – TCC2 Interrupt Flag for TCC2
This bit is set when a Peripheral Access Error occurs while accessing the TCC2, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 10 – TCC1 Interrupt Flag for TCC1
This bit is set when a Peripheral Access Error occurs while accessing the TCC1, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 9 – TCC0 Interrupt Flag for TCC0
This bit is set when a Peripheral Access Error occurs while accessing the TCC0, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 8 – TC3 Interrupt Flag for TC3
This bit is set when a Peripheral Access Error occurs while accessing the TC3, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 7 – TC2 Interrupt Flag for TC2
This bit is set when a Peripheral Access Error occurs while accessing the TC2, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 6 – TC1 Interrupt Flag for TC1
This bit is set when a Peripheral Access Error occurs while accessing the TC1, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 5 – TC0 Interrupt Flag for TC0
This bit is set when a Peripheral Write Access Error occurs while accessing the TC0, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 4 – SERCOM1 Interrupt Flag for SERCOM1
This bit is set when a Peripheral Access Error occurs while accessing the SERCOM1, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 3 – SERCOM0 Interrupt Flag for SERCOM0
This bit is set when a Peripheral Access Error occurs while accessing the SERCOM0, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 2 – EIC Interrupt Flag for EIC
This bit is set when a Peripheral Access Error occurs while accessing the EIC, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 1 – FREQM Interrupt Flag for FREQM
This bit is set when a Peripheral Access Error occurs while accessing the FREQM, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 0 – PAC Interrupt Flag for PAC
This bit is set when a Peripheral Write Access Error occurs while accessing the PAC, and will generate an interrupt request if SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

26.7.7 Peripheral Interrupt Flag Status – Bridge B

Name: INTFLAGB
Offset: 0x18
Reset: 0x00000000
Property: –

These flags are set when a Peripheral Access Error occurs while accessing the peripheral associated with the respective INTFLAGx bit, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Writing a '0' to these bits has no effect.

Writing a '1' to these bits will clear the corresponding INTFLAGx interrupt flag.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				RAMECC	EVSYS	DMAC		DSU
Access				RW	RW	RW		RW
Reset				0	0	0		0

Bit 4 – RAMECC Interrupt Flag for RAMECC
This flag is set when a Peripheral Access Error occurs while accessing the RAMECC, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the RAMECC interrupt flag.

Bit 3 – EVSYS Interrupt Flag for EVSYS
This flag is set when a Peripheral Access Error occurs while accessing the EVSYS, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the EVSYS interrupt flag.

Bit 2 – DMAC Interrupt Flag for DMAC
This flag is set when a Peripheral Access Error occurs while accessing the DMAC, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the DMAC interrupt flag.

Bit 0 – DSU Interrupt Flag for DSU
This flag is set when a Peripheral Access Error occurs while accessing the DSU, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the DSU interrupt flag.

26.7.8 Peripheral Interrupt Flag Status – Bridge C

Name: INTFLAGC
Offset: 0x1C
Reset: 0x00000000
Property: –

These flags are set when a Peripheral Access Error occurs while accessing the peripheral associated with the respective INTFLAGx bit and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Writing a '0' to these bits has no effect.

Writing a '1' to these bits will clear the corresponding INTFLAGx interrupt flag.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	AC	CCL						
Access	RW	RW						
Reset	0	0						

Bit 7 – AC Interrupt Flag for AC
This flag is set when a Peripheral Access Error occurs while accessing the peripheral associated with the AC and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the AC interrupt flag.

Bit 6 – CCL Interrupt Flag for CCL
This flag is set when a Peripheral Access Error occurs while accessing the peripheral associated with the CCL and will generate an interrupt request if INTENCLR/SET.ERR is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the CCL interrupt flag.

26.7.9 Peripheral Write Protection Status A

Name: STATUSA
Offset: 0x34
Reset: 0x00010000
Property: PAC Write-Protection

Writing to this register has no effect.

Reading STATUS register returns peripheral write protection status:

Value	Description
0	Peripheral is not write protected.
1	Peripheral is write protected.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
						TCC1		TC3
Access						R		R
Reset						0		0

Bit	7	6	5	4	3	2	1	0
	TC2	TC1	TC0	SERCOM1	SERCOM0	EIC	FREQM	PAC
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 10 – TCC1 TCC1 APB Protect Enable

Value	Description
0	TCC1 is not write protected
1	TCC1 is write protected

Bit 8 – TC3 TC3 APB Protect Enable

Value	Description
0	TC3 is not write protected
1	TC3 is write protected

Bit 7 – TC2 TC2 APB Protect Enable

Value	Description
0	TC2 is not write protected
1	TC2 is write protected

Bit 6 – TC1 TC1 APB Protect Enable

Value	Description
0	TC1 is not write protected
1	TC1 is write protected

Bit 5 – TC0 TC0 APB Protect Enable

Value	Description
0	TC0 is not write protected
1	TC0 is write protected

Bit 4 – SERCOM1 SERCOM1 APB Protect Enable

Value	Description
0	SERCOM1 is not write protected
1	SERCOM1 is write protected

Bit 3 – SERCOM0 SERCOM0 APB Protect Enable

Value	Description
0	SERCOM0 is not write protected
1	SERCOM0 is write protected

Bit 2 – EIC EIC APB Protect Enable

Value	Description
0	EIC is not write protected
1	EIC is write protected

Bit 1 – FREQM FREQM APB Protect Enable

Value	Description
0	FREQM is not write protected
1	FREQM is write protected

Bit 0 – PAC PAC APB Protect Enable

Value	Description
0	PAC is not write protected
1	PAC is write protected

26.7.10 Peripheral Write Protection Status – Bridge B

Name: STATUSB
Offset: 0x38
Reset: 0x00000002
Property: PAC Write-Protection

Writing to this register has no effect.

Reading STATUS register returns peripheral write protection status:

Value	Description
0	Peripheral is not write protected.
1	Peripheral is write protected.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				RAMECC	EVSYS	DMAC		
Access				R	R	R		
Reset				0	0	0		

Bit 4 – RAMECC RAMECC APB Protect Enable

Value	Description
0	RAMECC peripheral is not write protected
1	RAMECC peripheral is write protected

Bit 3 – EVSYS EVSYS APB Protect Enable

Value	Description
0	EVSYS peripheral is not write protected
1	EVSYS peripheral is write protected

Bit 2 – DMAC DMAC APB Protect Enable

Value	Description
0	DMAC peripheral is not write protected
1	DMAC peripheral is write protected

26.7.11 Peripheral Write Protection Status – Bridge C

Name: STATUSC
Offset: 0x3C
Reset: 0x00000000
Property: PAC Write-Protection

Writing to this register has no effect.

Reading STATUS register returns peripheral write protection status:

Value	Description
0	Peripheral is not write protected.
1	Peripheral is write protected.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	AC	CCL		SERCOM3	SERCOM2			
Access	R	R		R	R			
Reset	0	0		0	0			

Bit 7 – AC AC APB Protection Enable

Value	Description
0	Peripheral is not write protected
1	Peripheral is write protected

Bit 6 – CCL CCL APB Protection Enable

Value	Description
0	Peripheral is not write protected
1	Peripheral is write protected

Bit 4 – SERCOM3 SERCOM3 APB Protection Enable

Value	Description
0	Peripheral is not write protected
1	Peripheral is write protected

Bit 3 – SERCOM2 SERCOM2 APB Protection Enable

Value	Description
0	Peripheral is not write protected
1	Peripheral is write protected

27. Frequency Meter (FREQM)

27.1 Overview

The Frequency Meter (FREQM) can be used to accurately measure the frequency of a clock by comparing it to a known reference clock.

27.2 Features

- Ratio can be measured with 24-bit accuracy
- Accurately measures the frequency of an input clock with respect to a reference clock
- Reference clock can be selected from the available GCLK_FREQM_REF sources
- Measured clock can be selected from the available GCLK_FREQM_MSR sources

27.3 Block Diagram

Figure 27-1. FREQM Block Diagram

27.4 Signal Description

Not applicable.

27.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

27.5.1 I/O Lines

The REFO lines (REFO[4:1]) can be used as measurement or reference clock sources. This requires the I/O pins to be configured.

27.5.2 Power Management

The FREQM will continue to operate in idle sleep mode where the selected source clock is running. The FREQM's interrupts can be used to wake up the device from idle sleep mode. See *Power Management Unit (PMU)* from Related Links for details on the different sleep modes.

Related Links

15. Power Management Unit (PMU)

27.5.3 Clocks

Two generic clocks are used by the FREQM: Reference Clock (GCLK_FREQM_REF) and Measurement Clock (GCLK_FREQM_MSR).

GCLK_FREQM_REF is required to clock the internal reference timer, which acts as the frequency reference.

GCLK_FREQM_MSR is required to clock a ripple counter for frequency measurement. These clocks must be configured and enabled in the generic clock controller before using the FREQM.

27.5.4 DMA

Not applicable.

27.5.5 Interrupts

The interrupt request line is connected to the interrupt controller. Using FREQM interrupt requires the interrupt controller to be configured first.

27.5.6 Events

Not applicable.

27.5.7 Debug Operation

When the CPU is halted in debug mode the FREQM continues its normal operation. The FREQM cannot be halted when the CPU is halted in debug mode. If the FREQM is configured in a way that requires it to be periodically serviced by the CPU, improper operation or data loss may result during debugging.

27.5.8 Register Access Protection

All registers with write access can be write-protected optionally by the Peripheral Access Controller (PAC), except the following registers:

- Control B register (CTRLB)
- Interrupt Flag Status and Clear register (INTFLAG)
- Status register (STATUS)

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

Write-protection does not apply to accesses through an external debugger.

27.6 Functional Description

27.6.1 Principle of Operation

FREQM counts the number of periods of the measured clock (GCLK_FREQM_MSR) with respect to the reference clock (GCLK_FREQM_REF). The measurement is done for a period of REFNUM/f_{CLK_REF} and stored in the Value register (VALUE.VALUE). REFNUM is the number of Reference clock cycles selected in the Configuration A register (CFGA.REFNUM).

The frequency of the measured clock, f_{CLK_MSR}, is calculated by

$f_{CLK\_MSR} = \left(\frac{VALUE}{REFNUM}\right) f_{CLK\_REF}$. The error can be maximum two measured clock cycles.

27.6.2 Basic Operation

27.6.2.1 Initialization

Before enabling FREQM, the device and peripheral must be configured:

- Write the number of Reference clock cycles for which the measurement is to be done in the Configuration A register (CFGA.REFNUM). This must be a non-zero number.
- Configuration A register (CFGA)

Enable-protection is denoted by the "Enable-Protected" property in the register description.

27.6.2.2 Enabling, Disabling and Resetting

The FREQM is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The peripheral is disabled by writing CTRLA.ENABLE=0.

The FREQM is reset by writing a '1' to the Software Reset bit in the Control A register (CTRLA.SWRST). On software reset, all registers in the FREQM will be reset to their initial state, and the FREQM will be disabled.

Then ENABLE and SWRST bits are write-synchronized.

Related Links

27.6.7. Synchronization

27.6.2.3 Measurement

In the Configuration A register, the Number of Reference Clock Cycles field (CFGA.REFNUM) selects the duration of the measurement. The measurement is given in number of GCLK_FREQM_REF periods.
Note: The REFNUM field must be written before the FREQM is enabled.

After the FREQM is enabled, writing a '1' to the START bit in the Control B register (CTRLB.START) starts the measurement. The BUSY bit in Status register (STATUS.BUSY) is set when the measurement starts, and cleared when the measurement is complete.

There is also an interrupt request for Measurement Done: When the Measurement Done bit in Interrupt Enable Set register (INTENSET.DONE) is '1' and a measurement is finished, the Measurement Done bit in the Interrupt Flag Status and Clear register (INTFLAG.DONE) will be set and an interrupt request is generated.

The result of the measurement can be read from the Value register (VALUE.VALUE). The frequency of the measured clock GCLK_FREQM_MSR is then:

$$f_{\mathrm{CLK\_MSR}} = \left(\frac{\mathrm{VALUE}}{\mathrm{REFNUM}}\right) f_{\mathrm{CLK\_REF}}$$

Notes:

1. In order to make sure the measurement result (VALUE.VALUE[23:0]) is valid, the overflow status (STATUS.OVF) must be checked.
2. Due to asynchronous operations, the VALUE Error measurement can be up to two samples.

If an overflow condition occurred, indicated by the overflow bit in the STATUS register (STATUS.OVF), either the number of reference clock cycles must be reduced (CFGA.REFNUM) or a faster reference clock must be configured. Once the configuration is adjusted, clear the overflow status by writing a '1' to STATUS.OVF. Then, another measurement can be started by writing a '1' to CTRLB.START.

Note: See *CFGA*, *CTRLB*, *STATUS*, *INTENSET*, *INTFLAG*, *VALUE* registers in the *Register Summary - FREQM* from Related Links.

Related Links
27.7. Register Summary - FREQM

27.6.3 DMA Operation

Not applicable.

27.6.4 Interrupts

- DONE: A frequency measurement is done.

The interrupt flag in the Interrupt Flag Status and Clear (27.8.6. INTFLAG) register is set when the interrupt condition occurs. The interrupt can be enabled by writing a '1' to the corresponding bit in the Interrupt Enable Set (27.8.5. INTENSET) register, and disabled by writing a '1' to the corresponding bit in the Interrupt Enable Clear (27.8.4. INTENCLR) register. The status of enabled interrupts can be read from either INTENSET or INTENCLR.

An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled, or the FREQM is reset. See 27.8.6. INTFLAG for details on how to clear interrupt flags. All interrupt requests from the peripheral are ORed together on system level to generate one combined interrupt request to the NVIC. The user must read the 27.8.6. INTFLAG register to determine which interrupt condition is present.

This interrupt is a synchronous wake-up source.

Note that interrupts must be globally enabled for interrupt requests to be generated.

27.6.5 Events

Not applicable.

27.6.6 Sleep Mode Operation

For lowest chip power consumption in sleep modes, FREQM must be disabled before entering a Sleep mode.

27.6.7 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits and registers are write-synchronized:

- Software Reset bit in Control A register (CTRLA.SWRST)
- Enable bit in Control A register (CTRLA.ENABLE)

Required write synchronization is denoted by the "Write-Synchronized" property in the register description.

27.7 Register Summary - FREQM

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0							ENABLE	SWRST
0x01	CTRLB	7:0								START
0x02	CFGA	7:0	REFNUM[7:0]							
		15:8								
0x04 ... 0x07	Reserved									
0x08	INTENCLR	7:0								DONE
0x09	INTENSET	7:0								DONE
0x0A	INTFLAG	7:0								DONE
0x0B	STATUS	7:0							OVF	BUSY
0x0C	SYNCBUSY	7:0							ENABLE	SWRST
		15:8								
		23:16								
		31:24								
0x10	VALUE	7:0	VALUE[7:0]							
		15:8	VALUE[15:8]							
		23:16	VALUE[23:16]							
		31:24								

27.8 Register Description

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

Some registers require synchronization when read and/or written. Synchronization is denoted by the “Read-Synchronized” and/or “Write-Synchronized” property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the “Enable-Protected” property in each individual register description.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the "PAC Write-Protection" property in each individual register description.

27.8.1 Control A

Name: CTRLA
Offset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
							ENABLE	SWRST
Access							R/W	R/W
Reset							0	0

Bit 1 – ENABLE Enable

Due to synchronization there is delay from writing CTRLA.ENABLE until the peripheral is enabled or disabled. The value written to CTRLA.ENABLE will read back immediately and the ENABLE bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the FREQM to their initial state, and the FREQM will be disabled.
Writing a '1' to this bit will always take precedence, meaning that all other writes in the same write-operation will be discarded.
Due to synchronization there is a delay from writing CTRLA.SWRST until the Reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will be cleared when the Reset is complete.

Value	Description
0	There is no ongoing Reset operation.
1	The Reset operation is ongoing.

27.8.2 Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: –

Bit	7	6	5	4	3	2	1	0
								START
Access								W
Reset								0

Bit 0 – START Start Measurement

Value	Description
0	Writing a '0' has no effect.
1	Writing a '1' starts a measurement.

27.8.3 Configuration A

Name: CFGA
Offset: 0x02
Reset: 0x0000
Property: PAC Write-Protection, Enable-protected

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

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

Bits 7:0 – REFNUM[7:0] Number of Reference Clock Cycles
Selects the duration of a measurement in number of CLK_FREQM_REF cycles. This must be a non-zero value, i.e. 0x01 (one cycle) to 0xFF (255 cycles).

27.8.4 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x08
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								DONE
Access								R/W
Reset								0

Bit 0 – DONE Measurement Done Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Measurement Done Interrupt Enable bit, which disables the Measurement Done interrupt.

Value	Description
0	The Measurement Done interrupt is disabled.
1	The Measurement Done interrupt is enabled.

27.8.5 Interrupt Enable Set

Name: INTENSET
Offset: 0x09
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								DONE
Access								R/W
Reset								0

Bit 0 – DONE Measurement Done Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Measurement Done Interrupt Enable bit, which enables the Measurement Done interrupt.

Value	Description
0	The Measurement Done interrupt is disabled.
1	The Measurement Done interrupt is enabled.

27.8.6 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x0A
Reset: 0x00
Property: –

Bit	7	6	5	4	3	2	1	0
								DONE
Access								R/W
Reset								0

Bit 0 – DONE Mesurement Done
This flag is cleared by writing a '1' to it.
This flag is set when the STATUS.BUSY bit has a one-to-zero transition.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the DONE interrupt flag.

27.8.7 Status

Name: STATUS
Offset: 0x0B
Reset: 0x00
Property: –

Bit	7	6	5	4	3	2	1	0
							OVF	BUSY
Access							R/W	R
Reset							0	0

Bit 1 – OVF Sticky Count Value Overflow
This bit is cleared by writing a '1' to it.
This bit is set when an overflow condition occurs to the value counter.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the OVF status.

Bit 0 – BUSY FREQM Status

Value	Description
0	No ongoing frequency measurement.
1	Frequency measurement is ongoing.

27.8.8 Synchronization Busy

Name: SYNCBUSY
Offset: 0x0C
Reset: 0x00000000
Property: –

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							ENABLE	SWRST
Access							R	R
Reset							0	0

Bit 1 – ENABLE Enable
This bit is cleared when the synchronization of CTRLA.ENABLE is complete.
This bit is set when the synchronization of CTRLA.ENABLE is started.

Bit 0 – SWRST Synchronization Busy
This bit is cleared when the synchronization of CTRLA.SWRST is complete.
This bit is set when the synchronization of CTRLA.SWRST is started.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

27.8.9 Value

Name: VALUE
Offset: 0x10
Reset: 0x00000000
Property: –

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VALUE[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	VALUE[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

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

Bits 23:0 – VALUE[23:0] Measurement Value
Result from measurement.

28. Event System (EVSYS)

28.1 Overview

The Event System (EVSYS) allows autonomous, low-latency and configurable communication between peripherals.

Several peripherals can be configured to generate and/or respond to signals known as events. The exact condition to generate an event, or the action taken upon receiving an event, is specific to each peripheral. Peripherals that respond to events are called event users. Peripherals that generate events are called event generators. A peripheral can have one or more event generators and can have one or more event users.

Communication is made without CPU intervention and without consuming system resources such as bus or RAM bandwidth. This reduces the load on the CPU and other system resources, compared to a traditional interrupt-based system.

28.2 Features

- 32 configurable event channels:
  - All channels can be connected to any event generator
  - All channels provide a pure asynchronous path
  - Twelve channels provide a resynchronized or synchronous path
- 69 event generators.
- 52 event users.
- Configurable edge detector.
- Peripherals can be event generators, event users, or both.
- SleepWalking and interrupt for operation in sleep modes.
- Software event generation.
- Each event user can choose which channel to respond to.
- Optional Static or Round-Robin interrupt priority arbitration.

28.3 Block Diagram

Figure 28-1. Event System Block Diagram

28.4 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

28.4.1 I/O Lines

Not applicable.

28.4.2 Power Management

The EVSYS can be used to wake up the CPU from all sleep modes (Deep Sleep/BACKUP and Extreme Deep Sleep/OFF Mode), even if the clock used by the EVSYS channel and the EVSYS bus clock are disabled. See *Power Management Unit (PMU)* from Related Links for details on the different sleep modes.

Although the clock for the EVSYS is stopped, the device still can wake up the EVSYS clock. Some event generators can generate an event when their clocks are stopped. The generic clock for the channel (GCLK_EVSYS_CHANNEL_n) will be restarted if that channel uses a synchronized path or a resynchronized path. It does not need to wake the system from sleep.

Related Links

15. Power Management Unit (PMU)

28.4.3 Clocks

Each EVSYS channel which can be configured as synchronous or resynchronized has a dedicated generic clock (GCLK_EVSYS_CHANNEL_n). These are used for event detection and propagation for each channel. These clocks must be configured and enabled in the generic clock controller before using the EVSYS (see *Clock and Reset (CRU)* from Related Links).

Important: Only EVSYS channel 0 to 11 can be configured as synchronous or resynchronized.

Related Links

13. Clock and Reset Unit (CRU)

28.4.4 DMA

Not applicable.

28.4.5 Interrupts

The interrupt request line is connected to the interrupt controller. Using the EVSYS interrupts requires the interrupt controller to be configured first (see *Nested Vector Interrupt Controller (NVIC)* from Related Links).

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

28.4.6 Events

Not applicable.

28.4.7 Debug Operation

When the CPU is halted in Debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging.

28.4.8 Register Access Protection

Registers with write access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

- Channel Pending Interrupt (INTPEND)
- Channel n Interrupt Flag Status and Clear (CHINTFLAGn)

Note: Optional write protection is indicated by the "PAC Write Protection" property in the register description.

Write protection does not apply for accesses through an external debugger.

28.4.9 Analog Connections

Not applicable.

28.5 Functional Description

28.5.1 Principle of Operation

The Event System consists of channels which route the internal events from peripherals (generators) to other internal peripherals. Each event generator can be selected as source for multiple channels, but a channel cannot be set to use multiple event generators at the same time.

A channel path can be configured in asynchronous, synchronous or resynchronized mode of operation. The mode of operation must be selected based on the requirements of the application.

When using synchronous or resynchronized path, the Event System includes options to transfer events to users when rising, falling or both edges are detected on event generators.

See *Channel Path* from Related Links.

Related Links

28.5.2.6. Channel Path

28.5.2 Basic Operation

28.5.2.1 Initialization

Before enabling event routing within the system, the Event Users Multiplexer and Event Channels must be selected in the Event System (EVSYS), and the two peripherals that generate and use the event must be configured. Follow these steps to configure the event:

1. In the event generator peripheral, enable output of event by writing a '1' to the respective Event Output Enable bit ("EO") in the peripheral's Event Control register, for example, AC.EVCTRL.WINEO0, RTC.EVCTRL.OVFEO.
2. Configure the EVSYS:
   a. Configure the Event User multiplexer by writing the respective EVSYS.USERm register, refer to 28.5.2.3. User Multiplexer Setup.
   b. Configure the Event Channel by writing the respective EVSYS.CHANNELn register, refer to 28.5.2.4. Event System Channel.
3. Configure the action to be executed by the event user peripheral by writing to the Event Action bits (EVACT) in the respective Event control register, for example, TC.EVCTRL.EVACT.
   Note: This step is not applicable for all the peripherals.
4. In the event user peripheral, enable event input by writing a '1' to the respective Event Input Enable bit ("EI") in the peripheral's Event Control register, for example, AC.EVCTRL.IVEI0, ADC.EVCTRL.STARTEI.

28.5.2.2 Enabling, Disabling, and Resetting

The EVSYS is always enabled.

The EVSYS is reset by writing a ‘1’ to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the EVSYS will be reset to their initial state and all ongoing events will be canceled.

Refer to CTRLA.SWRST register for details.

28.5.2.3 User Multiplexer Setup

The user multiplexer defines the channel to be connected to which event user. Each user multiplexer is dedicated to one event user. A user multiplexer receives all event channels output and must be configured to select one of these channels, as shown in Block Diagram section. The channel is selected with the Channel bit group in the User register (USERm.CHANNEL).

The user multiplexer must always be configured before the channel. A list of all available event users is found in the User (USERm) register description.

Related Links

28.3. Block Diagram

28.5.2.4 Event System Channel

An event channel can select one event from a list of event generators. Depending on configuration, the selected event could be synchronized, resynchronized or asynchronously sent to the users. When synchronization or resynchronization is required, the channel includes an internal edge detector, allowing the Event System to generate internal events when rising, falling or both edges are detected on the selected event generator.

An event channel is able to generate internal events for the specific software commands. A channel block diagram is shown in *Block Diagram* section.

Related Links

28.3. Block Diagram

28.5.2.5 Event Generators

Each event channel can receive the events form all event generators. All event generators are listed in the Event Generator bit field in the Channel n register (CHANNELn.EVGEN). For details on event generation, refer to the corresponding module chapter. The channel event generator is selected by the Event Generator bit group in the Channel register (CHANNELn.EVGEN). By default, the channels are not connected to any event generators (ie, CHANNELn.EVGEN = 0)

28.5.2.6 Channel Path

There are different ways to propagate the event from an event generator:

- Asynchronous path
- Resynchronized path

The path is decided by writing to the Path Selection bit group of the Channel register (CHANNELn.PATH).

Asynchronous Path
When using the asynchronous path, the events are propagated from the event generator to the event user without intervention from the Event System. The GCLK for this channel (GCLK_EVSYS_CHANNEL_n) is not mandatory, meaning that an event will be propagated to the user without any clock latency.

When the asynchronous path is selected, the channel cannot generate any interrupts, and the Channel x Status register (CHSTATUSx) is always zero. The edge detection is not required and must be disabled by software. Each peripheral event user has to select which event edge must trigger internal actions. For further details, refer to each peripheral chapter description.

Resynchronized Path
The resynchronized path are used when the event generator and the event channel do not share the same generator for the generic clock. When the resynchronized path is used, resynchronization of the event from the event generator is done in the channel.

When the resynchronized path is used, the channel is able to generate interrupts. The channel status bits in the Channel Status register (CHSTATUS) are also updated and available for use.

28.5.2.7 Edge Detection

When synchronous or resynchronized paths are used, edge detection must be enabled. The event system can execute edge detection in three different ways:

- Generate an event only on the rising edge
- Generate an event only on the falling edge
- Generate an event on rising and falling edges.

Edge detection is selected by writing to the Edge Selection bit group of the Channel register (CHANNELn.EDGSEL).

28.5.2.8 Event Latency

The latency from event generator to event user depends on the channel's configuration:

- Asynchronous Path: The maximum routing latency of an external event is related to the internal signal routing and it is device dependent.
- Resynchronized Path: The maximum routing latency of an external event is three GCLK_EVSYS_CHANNEL_n clock cycles.

The maximum propagation latency of a user event to the peripheral clock core domain is three peripheral clock cycles.

The event generators, event channel and event user clocks ratio must be selected in relation with the internal event latency constraints. Events propagation or event actions in peripherals may be lost if the clock setup violates the internal latencies.

28.5.2.9 The Overrun Channel n Interrupt

The Overrun Channel n Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAGn.OVR) will be set, and the optional interrupt will be generated in the following cases:

- One or more event users on channel n is not ready when there is a new event
- An event occurs when the previous event on channel m has not been handled by all event users connected to that channel

The flag will only be set when using resynchronized paths. In the case of asynchronous path, the INTFLAGn.OVR is always read as zero.

28.5.2.10 The Event Detected Channel n Interrupt

The Event Detected Channel n Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAGn.EVD) is set when an event coming from the event generator configured on channel n is detected.

The flag will only be set when using a resynchronized path. In the case of an asynchronous path, the INTFLAGn.EVD is always zero.

28.5.2.11 Channel Status

The Channel Status register (CHSTATUS) shows the status of the channels when using a synchronous or resynchronized path. There are two different status bits in CHSTATUS for each of the available channels:

- The CHSTATUSn.BUSYCH bit will be set when an event on the corresponding channel n has not been handled by all event users connected to that channel.
- The CHSTATUSn.RDYUSR bit will be set when all event users connected to the corresponding channel are ready to handle incoming events on that channel.

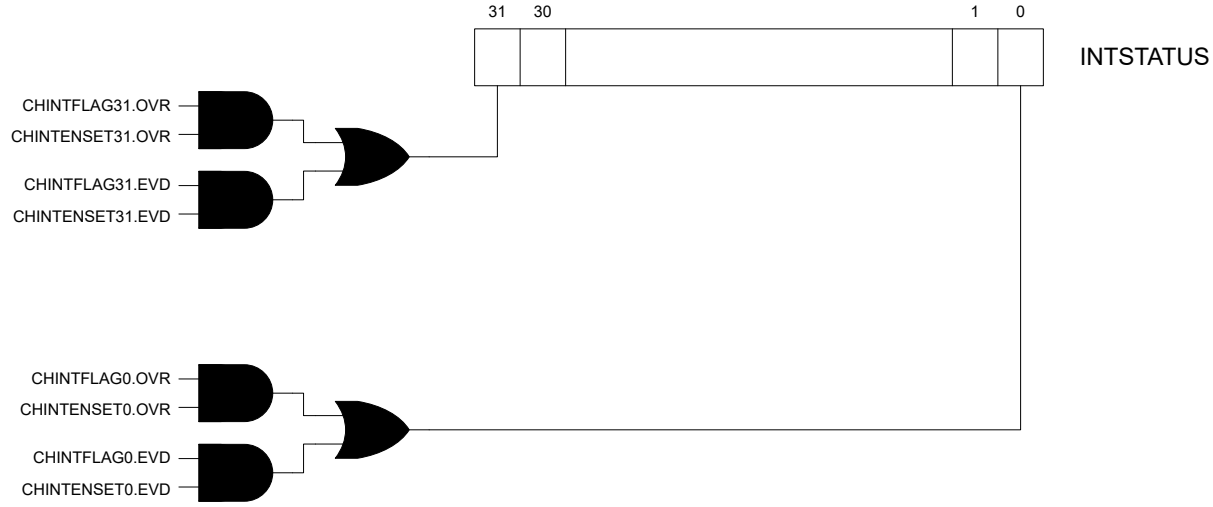
28.5.2.12 Software Event

A software event can be initiated on a channel by writing a '1' to the Software Event bit in the Channel register (SWEVT.CHANNELn). Then the software event can be serviced as any event generator; i.e., when a bit is set to ‘1’, the corresponding event will be generated on the respective channel.

28.5.2.13 Interrupt Status and Interrupts Arbitration

The Interrupt Status register stores all channels with pending interrupts, as shown below.

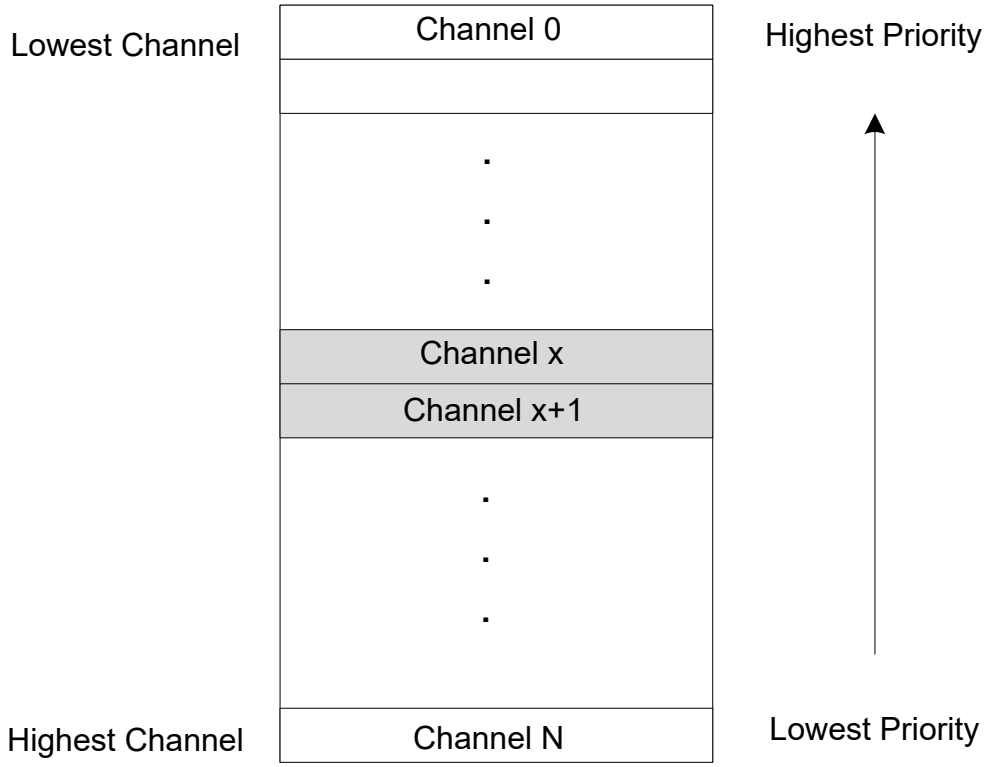
Figure 28-2. Interrupt Status Register



The Event System can arbitrate between all channels with pending interrupts. The arbiter can be configured to prioritize statically or dynamically the incoming events. The priority is evaluated each time a new channel has an interrupt pending, or an interrupt has been cleared. The Channel Pending Interrupt register (INTPEND) will provide the channel number with the highest interrupt priority, and the corresponding channel interrupt flags and status bits.

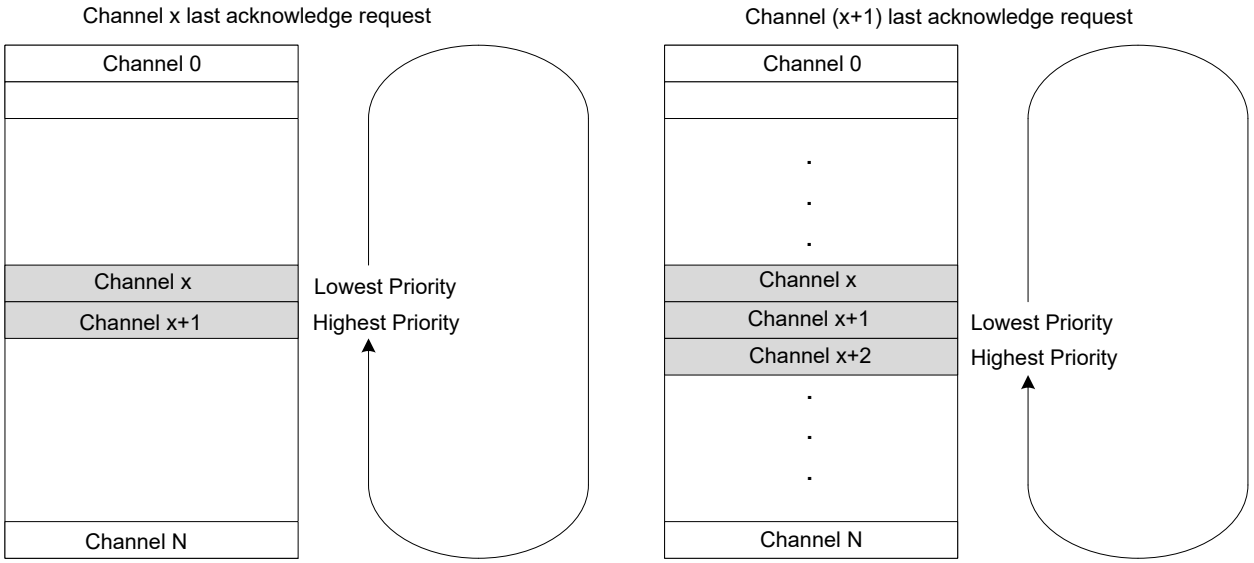
By default, static arbitration is enabled (PRICTRL.RRENx is '0'), the arbiter will prioritize a low channel number over a high channel number as shown below. When using the status scheme, there is a risk of high channel numbers never being granted access by the arbiter. This can be avoided using a dynamic arbitration scheme.

Figure 28-3. Static Priority



The dynamic arbitration scheme available in the Event System is round-robin. Round-robin arbitration is enabled by writing PRICTRL.RREN to one. With the round-robin scheme, the channel number of the last channel being granted access will have the lowest priority the next time the arbiter has to grant access to a channel, as shown below. The channel number of the last channel being granted access, will be stored in the Channel Priority Number bit group in the Priority Control register (PRICTRL.PRI).

Figure 28-4. Round-Robin Scheduling



The Channel Pending Interrupt register (INTPEND) also offers the possibility to indirectly clear the interrupt flags of a specific channel. Writing a flag to one in this register, will clear the corresponding interrupt flag of the channel specified by the INTPEND.ID bits.

28.5.3 Interrupts

The EVSYS has the following interrupt sources for each channel:

- Overrun Channel n interrupt (OVR)
- Event Detected Channel n interrupt (EVD)

These interrupts events are asynchronous wake-up sources.

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the corresponding Channel n Interrupt Flag Status and Clear (CHINTFLAG) register is set when the interrupt condition occurs.

Note: Interrupts must be globally enabled to allow the generation of interrupt requests.

Each interrupt can be individually enabled by writing a '1' to the corresponding bit in the Channel n Interrupt Enable Set (CHINTENSET) register, and disabled by writing a '1' to the corresponding bit in the Channel n Interrupt Enable Clear (CHINTENCLR) register. An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the Event System is reset. All interrupt requests are ORed together on system level to generate one combined interrupt request to the NVIC.

The user must read the Channel Interrupt Status (INTSTATUS) register to identify the channels with pending interrupts, and must read the Channel n Interrupt Flag Status and Clear (CHINTFLAG) register to determine which interrupt condition is present for the corresponding channel. It is also possible to read the Interrupt Pending register (INTPEND), which provides the highest priority channel with pending interrupt and the respective interrupt flags.

28.5.4 Sleep Mode Operation

The Event System can generate interrupts to wake up the device from Idle or Standby mode.

To be able to run in standby, the run in Standby bit in the Channel register (CHANNELn.RUNSTDBY) must be set to '1'. When the Generic Clock On Demand bit in the Channel register (CHANNELn.ONDEMAND) is set to '1' and the event generator is detected, the event channel will request its clock (GCLK_EVSYS_CHANNEL_n). The event latency for a resynchronized channel path will increase by two GCLK_EVSYS_CHANNEL_n clock (that is., up to five GCLK_EVSYS_CHANNEL_n clock cycles).

A channel will behave differently in different sleep modes regarding to CHANNELn.RUNSTDBY and CHANNELn.ONDEMAND:

Table 28-1. Event Channel Sleep Behavior

CHANNELn.PATH	CHANNELn. ONDEMAND	CHANNELn. RUNSTDBY	Sleep Behavior
ASYNC	0	0	Only run in Idle mode if an event must be propagated. Disabled in Standby mode.
SYNC/RESYNC	0	0	N/A. Works only in Active mode.
SYNC/RESYNC	0	1	Run in both Idle and Standby modes.
SYNC/RESYNC	1	0	Only run in Idle mode if an event must be propagated. Disabled in Standby mode. Two GCLK_EVSYS_n latency added in RESYNC path before the event is propagated internally.
SYNC/RESYNC	1	1	Run in both Idle and Standby modes. Two GCLK_EVSYS_n latency added in RESYNC path before the event is propagated internally.

Note: The ONDEMAND and RUNSTDBY bits have no effect for channels when asynchronous path is selected.

28.6 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0								SWRST
0x01 ... 0x03	Reserved									
0x04	SWEVT	7:0	CHANNEL7	CHANNEL6	CHANNEL5	CHANNEL4	CHANNEL3	CHANNEL2	CHANNEL1	CHANNEL0
		15:8	CHANNEL15	CHANNEL14	CHANNEL13	CHANNEL12	CHANNEL11	CHANNEL10	CHANNEL9	CHANNEL8
		23:16	CHANNEL23	CHANNEL22	CHANNEL21	CHANNEL20	CHANNEL19	CHANNEL18	CHANNEL17	CHANNEL16
		31:24	CHANNEL31	CHANNEL30	CHANNEL29	CHANNEL28	CHANNEL27	CHANNEL26	CHANNEL25	CHANNEL24
0x08	PRICTRL	7:0	RREN			PRI[4:0]				
0x09 ... 0x0F	Reserved									
0x10	INTPEND	7:0				ID[4:0]				
		15:8	BUSY	READY					EVD	OVR
0x12 ... 0x13	Reserved									
0x14	INTSTATUS	7:0	CHINT7	CHINT6	CHINT5	CHINT4	CHINT3	CHINT2	CHINT1	CHINT0
		15:8					CHINT11	CHINT10	CHINT9	CHINT8
		23:16								
		31:24								
0x18	BUSYCH	7:0	BUSYCH7	BUSYCH6	BUSYCH5	BUSYCH4	BUSYCH3	BUSYCH2	BUSYCH1	BUSYCH0
		15:8					BUSYCH11	BUSYCH10	BUSYCH9	BUSYCH8
		23:16								
		31:24								
0x1C	READYUSR	7:0	READYUSR7	READYUSR6	READYUSR5	READYUSR4	READYUSR3	READYUSR2	READYUSR1	READYUSR0
		15:8					READYUSR11	READYUSR10	READYUSR9	READYUSR8
		23:16								
		31:24								
0x20	CHANNEL0	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x24	CHINTENCLR0	7:0							EVD	OVR
0x25	CHINTENSET0	7:0							EVD	OVR
0x26	CHINTFLAG0	7:0							EVD	OVR
0x27	CHSTATUSn0	7:0							BUSYCH	RDYUSR
0x28	CHANNEL1	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x2C	CHINTENCLR1	7:0							EVD	OVR
0x2D	CHINTENSET1	7:0							EVD	OVR
0x2E	CHINTFLAG1	7:0							EVD	OVR
0x2F	CHSTATUSn1	7:0							BUSYCH	RDYUSR
0x30	CHANNEL2	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x34	CHINTENCLR2	7:0							EVD	OVR
0x35	CHINTENSET2	7:0							EVD	OVR
0x36	CHINTFLAG2	7:0							EVD	OVR
0x37	CHSTATUSn2	7:0							BUSYCH	RDYUSR

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x38	CHANNEL3	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x3C	CHINTENCLR3	7:0							EVD	OVR
0x3D	CHINTENSET3	7:0							EVD	OVR
0x3E	CHINTFLAG3	7:0							EVD	OVR
0x3F	CHSTATUSn3	7:0							BUSYCH	RDYUSR
0x40	CHANNEL4	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x44	CHINTENCLR4	7:0							EVD	OVR
0x45	CHINTENSET4	7:0							EVD	OVR
0x46	CHINTFLAG4	7:0							EVD	OVR
0x47	CHSTATUSn4	7:0							BUSYCH	RDYUSR
0x48	CHANNEL5	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x4C	CHINTENCLR5	7:0							EVD	OVR
0x4D	CHINTENSET5	7:0							EVD	OVR
0x4E	CHINTFLAG5	7:0							EVD	OVR
0x4F	CHSTATUSn5	7:0							BUSYCH	RDYUSR
0x50	CHANNEL6	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x54	CHINTENCLR6	7:0							EVD	OVR
0x55	CHINTENSET6	7:0							EVD	OVR
0x56	CHINTFLAG6	7:0							EVD	OVR
0x57	CHSTATUSn6	7:0							BUSYCH	RDYUSR
0x58	CHANNEL7	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x5C	CHINTENCLR7	7:0							EVD	OVR
0x5D	CHINTENSET7	7:0							EVD	OVR
0x5E	CHINTFLAG7	7:0							EVD	OVR
0x5F	CHSTATUSn7	7:0							BUSYCH	RDYUSR
0x60	CHANNEL8	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x64	CHINTENCLR8	7:0							EVD	OVR
0x65	CHINTENSET8	7:0							EVD	OVR
0x66	CHINTFLAG8	7:0							EVD	OVR
0x67	CHSTATUSn8	7:0							BUSYCH	RDYUSR
0x68	CHANNEL9	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x6C	CHINTENCLR9	7:0							EVD	OVR
0x6D	CHINTENSET9	7:0							EVD	OVR
0x6E	CHINTFLAG9	7:0							EVD	OVR
0x6F	CHSTATUSn9	7:0							BUSYCH	RDYUSR

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x70	CHANNEL10	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x74	CHINTENCLR10	7:0							EVD	OVR
0x75	CHINTENSET10	7:0							EVD	OVR
0x76	CHINTFLAG10	7:0							EVD	OVR
0x77	CHSTATUSn10	7:0							BUSYCH	RDYUSR
0x78	CHANNEL11	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x7C	CHINTENCLR11	7:0							EVD	OVR
0x7D	CHINTENSET11	7:0							EVD	OVR
0x7E	CHINTFLAG11	7:0							EVD	OVR
0x7F	CHSTATUSn11	7:0							BUSYCH	RDYUSR
0x80	CHANNEL12	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x84 ... 0x87	Reserved									
0x88	CHANNEL13	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x8C ... 0x8F	Reserved									
0x90	CHANNEL14	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x94 ... 0x97	Reserved									
0x98	CHANNEL15	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x9C ... 0x9F	Reserved									
0xA0	CHANNEL16	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xA4 ... 0xA7	Reserved									
0xA8	CHANNEL17	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xAC ... 0xAF	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0xB0	CHANNEL18	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xB4 ... 0xB7	Reserved									
0xB8	CHANNEL19	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xBC ... 0xBF	Reserved									
0xC0	CHANNEL20	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xC4 ... 0xC7	Reserved									
0xC8	CHANNEL21	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xCC ... 0xCF	Reserved									
0xD0	CHANNEL22	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xD4 ... 0xD7	Reserved									
0xD8	CHANNEL23	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xDC ... 0xDF	Reserved									
0xE0	CHANNEL24	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xE4 ... 0xE7	Reserved									
0xE8	CHANNEL25	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xEC ... 0xEF	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0xF0	CHANNEL26	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xF4 ... 0xF7	Reserved									
0xF8	CHANNEL27	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0xFC ... 0xFF	Reserved									
0x0100	CHANNEL28	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x0104 ... 0x0107	Reserved									
0x0108	CHANNEL29	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x010C ... 0x010F	Reserved									
0x0110	CHANNEL30	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x0114 ... 0x0117	Reserved									
0x0118	CHANNEL31	7:0	EVGEN[7:0]							
		15:8	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
		23:16								
		31:24								
0x011C ... 0x011F	Reserved									
0x0120	USER0	7:0	CHANNEL[7:0]							
...										
0x0153	USER51	7:0	CHANNEL[7:0]							

28.7 Register Description

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

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

For more details, see *Register Access Protection* and *Peripheral Access Controller (PAC)* from Related Links.

Related Links

28.4.8. Register Access Protection

26. Peripheral Access Controller (PAC)

28.7.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								SWRST
Access								W
Reset								0

Bit 0 – SWRST Software Reset
Writing '0' to this bit has no effect.
Writing '1' to this bit resets all registers in the EVSYS to their initial state.
Note: Before applying a Software Reset it is recommended to disable the event generators.

28.7.2 Software Event

Name: SWEVT
Offset: 0x04
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	CHANNEL31	CHANNEL30	CHANNEL29	CHANNEL28	CHANNEL27	CHANNEL26	CHANNEL25	CHANNEL24
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CHANNEL23	CHANNEL22	CHANNEL21	CHANNEL20	CHANNEL19	CHANNEL18	CHANNEL17	CHANNEL16
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CHANNEL15	CHANNEL14	CHANNEL13	CHANNEL12	CHANNEL11	CHANNEL10	CHANNEL9	CHANNEL8
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CHANNEL7	CHANNEL6	CHANNEL5	CHANNEL4	CHANNEL3	CHANNEL2	CHANNEL1	CHANNEL0
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 – CHANNELx
Channel x Software Selection [x=0..7]

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will trigger a software event for channel x.
These bits always return '0' when read.

28.7.3 Priority Control

Name: PRICTRL
Offset: 0x08
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
	RREN			PRI[4:0]				
Access	RW			RW	RW	RW	RW	RW
Reset	0			0	0	0	0	0

Bit 7 – RREN Round-Robin Scheduling Enable
For details on scheduling schemes, refer to Interrupt Status and Interrupts Arbitration

Value	Description
0	Static scheduling scheme for channels with level priority
1	Round-robin scheduling scheme for channels with level priority

Bits 4:0 – PRI[4:0] Channel Priority Number
When round-robin arbitration is enabled (PRICTRL.RREN=1) for priority level, this register holds the channel number of the last EVSYS channel being granted access as the active channel with priority level. The value of this bit group is updated each time the INTPEND or any of CHINTFLAG registers are written.
When static arbitration is enabled (PRICTRL.RREN=0) for priority level, and the value of this bit group is nonzero, it will not affect the static priority scheme.
This bit group is not reset when round-robin scheduling gets disabled (PRICTRL.RREN written to zero).

28.7.4 Channel Pending Interrupt

Name: INTPEND
Offset: 0x10
Reset: 0x4000

An interrupt that handles several channels must consult the INTPEND register to find out which channel number has priority (ignoring/filtering each channel that has its own interrupt line). An interrupt dedicated to only one channel must not use the INTPEND register.

Bit	15	14	13	12	11	10	9	8
	BUSY	READY					EVD	OVR
Access	R	R					RW	RW
Reset	0	1					0	0

Bit	7	6	5	4	3	2	1	0
				ID[4:0]				
Access				RW	RW	RW	RW	RW
Reset				0	0	0	0	0

Bit 15 – BUSY Busy
This bit is read '1' when the event on a channel selected by Channel ID field (ID) has not been handled by all the event users connected to this channel.

Bit 14 – READY Ready
This bit is read '1' when all event users connected to the channel selected by Channel ID field (ID) are ready to handle incoming events on this channel.

Bit 9 – EVD Channel Event Detected
This flag is set on the next CLK_EVSYS_APB cycle when an event is being propagated through the channel, and an interrupt request will be generated if CHINTENCLR/SET.EVD is '1'.
When the event channel path is asynchronous, the EVD bit will not be set.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear it. It will also clear the corresponding flag in the Channel n Interrupt Flag Status and Clear register (CHINTFLAGn) of this peripheral, where n is determined by the Channel ID bit field (ID) in this register.

Bit 8 – OVR Channel Overrun
This flag is set on the next CLK_EVSYS cycle after an overrun channel condition occurs, and an interrupt request will be generated if CHINTENCLR/SET.OVRx is '1'.
There are two possible overrun channel conditions:

- One or more of the event users on channel selected by Channel ID field (ID) are not ready when a new event occurs
- An event happens when the previous event on channel selected by Channel ID field (ID) has not yet been handled by all event users

When the event channel path is asynchronous, the OVR interrupt flag will not be set.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear it. It will also clear the corresponding flag in the Channel n Interrupt Flag Status and Clear register (CHINTFLAGn) of this peripheral, where n is determined by the Channel ID bit field (ID) in this register.

Bits 4:0 – ID[4:0] Channel ID
These bits store the channel number of the highest priority.
When the bits are written, indirect access to the corresponding Channel Interrupt Flag register is enabled.

28.7.5 Interrupt Status

Name: INTSTATUS
Offset: 0x14
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					CHINT11	CHINT10	CHINT9	CHINT8
Access					R	R	R	R
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CHINT7	CHINT6	CHINT5	CHINT4	CHINT3	CHINT2	CHINT1	CHINT0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 – CHINTx Channel x Pending Interrupt
This bit is set when Channel x has a pending interrupt.
This bit is cleared when the corresponding Channel x interrupts are disabled, or the source interrupt sources are cleared.

28.7.6 Busy Channels

Name: BUSYCH
Offset: 0x18
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					BUSYCH11	BUSYCH10	BUSYCH9	BUSYCH8
Access					R	R	R	R
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BUSYCH7	BUSYCH6	BUSYCH5	BUSYCH4	BUSYCH3	BUSYCH2	BUSYCH1	BUSYCH0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 – BUSYCHx Busy Channel x
This bit is set if an event occurs on channel x has not been handled by all event users connected to channel x.
This bit is cleared when channel x is idle.
When the event channel x path is asynchronous, this bit is always read '0'.

28.7.7 Ready Users

Name: READYUSR
Offset: 0x1C
Reset: 111111111111

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					READYUSR11	READYUSR10	READYUSR9	READYUSR8
Access					R	R	R	R
Reset					1	1	1	1

Bit	7	6	5	4	3	2	1	0
	READYUSR7	READYUSR6	READYUSR5	READYUSR4	READYUSR3	READYUSR2	READYUSR1	READYUSR0
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 – READYUSRn Ready User for Channel n
This bit is set when all event users connected to channel n are ready to handle incoming events on channel n.
This bit is cleared when at least one of the event users connected to the channel is not ready.
When the event channel n path is asynchronous, this bit is always read zero.

28.7.8 Channel n Control

Name: CHANNEL
Offset: 0x20 + n*0x08 [n=0..31]
Reset: 0x00008000
Property: PAC Write-Protection, Mix-Secure

This register allows the user to configure channel n. To write to this register, do a single, 32-bit write of all the configuration data.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
Access	RW	RW			RW	RW	RW	RW
Reset	1	0			0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EVGEN[7:0]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit 15 – ONDEMAND Generic Clock On Demand

Value	Description
0	Generic clock for a channel is always on, if the channel is configured and generic clock source is enabled.
1	Generic clock is requested on demand while an event is handled

Bit 14 – RUNSTDBY Run in Standby

This bit is used to define the behavior during standby sleep mode.

Value	Description
0	The channel is disabled in standby sleep mode.
1	The channel is not stopped in standby sleep mode and depends on the CHANNEL.ONDEMAND bit.

Bits 11:10 – EDGSEL[1:0] Edge Detection Selection

These bits set the type of edge detection to be used on the channel.
These bits must be written to zero when using the asynchronous path.

Value	Name	Description
0x0	NO_EVT_OUTPUT	No event output when using the resynchronized path
0x1	RISING_EDGE	Event detection only on the rising edge of the signal from the event generator
0x2	FALLING_EDGE	Event detection only on the falling edge of the signal from the event generator
0x3	BOTH_EDGES	Event detection on rising and falling edges of the signal from the event generator

Bits 9:8 – PATH[1:0] Path Selection

These bits are used to choose which path will be used by the selected channel.
Note: The path choice can be limited by the channel source (see *USERm* from Related Links).

Important: Only EVSYS channel 0 to 3 can be configured as synchronous or resynchronized.

Value	Name	Description
0x1	RESYNCHRONIZED	Resynchronized path
0x2	ASYNCHRONOUS	Asynchronous path
Other	-	Reserved

Bits 7:0 – EVGEN[7:0] Event Generator Selection

These bits are used to choose the event generator to connect to the selected channel.

Table 28-2. Event Generator Selection

Value	Name	Description
0x00 - 0x07	RTC_PERx	RTC period x=0..7
0x08 - 0x0B	RTC_CMPx	RTC comparison x=0..3
0x0C	RTC_TAMPER	RTC tamper detection
0x0D	RTC_OVF	RTC Overflow
0x0E - 0x11	EIC_EXTINTx	EIC external interrupt x=0..3
0x12 - 0x15	DMAC_CHx	DMA channel x=0..3
0x16	PAC_ACCERR	PAC Acc. error
0x17	TCC0_OVF	TCC0 Overflow
0x18	TCC0_TRG	TCC0 Trigger Event
0x19	TCC0_CNT	TCC0 Counter
0x1A-0x1F	TCC0_MCx	TCC0 Match/Compare x=0..5
0x20	TCC1_OVF	TCC1 Overflow
0x21	TCC1_TRG	TCC1 Trigger Event
0x22	TCC1_CNT	TCC1 Counter
0x23 - 0x28	TCC1_MCx	TCC1 Match/Compare x=0..5
0x29	TCC2_OVF	TCC2 Overflow
0x2A	TCC2_TRG	TCC2 Trigger Event
0x2B	TCC2_CNT	TCC2 Counter
0x2C - 0x2D	TCC2_MCx	TCC2 Match/Compare x=0..1
0x2E	TC0_OVF	TC0 Overflow
0x2F-0x30	TC0_MCx	TC0 Match/Compare x=0..1
0x31	TC1_OVF	TC1 Overflow
0x32 - 0x33	TC1_MCx	TC1 Match/Compare x=0..1
0x34	TC2_OVF	TC2 Overflow
0x35 - 0x36	TC2_MCx	TC2 Match/Compare x=0..1
0x37	TC3_OVF	TC3 Overflow
0x38 - 0x39	TC3_MCx	TC3 Match/Compare x=0..1
0x3A	ADC_RESRDY	ADC End-Of-Scan Ready Interrupt
0x3B - 0x3C	Not used	—
0x3D - 0x3E	AC_COMPx	AC Comparator, x=0..1
0x3F	AC_WIN_0	AC0 Window
0x40	TRNG_READY	TRNG ready
0x41 - 0x42	CCL_LUTOUTx	CCL LUTOUT x-0..1
0x43	ZB_TX_TS_ACTIVE	Zigbee Transmit Packet Active time
0x44	ZB_RX_TS_ACTIVE	Zigbee Receive Packet Active time

Related Links

28.7.13. USERm

28.7.9 Channel n Interrupt Enable Clear

Name: CHINTENCLR
Offset: 0x24 + n*0x08 [n=0..11]
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
							EVD	OVR
Access							RW	RW
Reset							0	0

Bit 1 – EVD Channel Event Detected Interrupt Disable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Event Detected Channel Interrupt Enable bit, which disables the Event Detected Channel interrupt.

Value	Description
0	The Event Detected Channel interrupt is disabled.
1	The Event Detected Channel interrupt is enabled.

Bit 0 – OVR Channel Overrun Interrupt Disable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Overrun Channel Interrupt Enable bit, which disables the Overrun Channel interrupt.

Value	Description
0	The Overrun Channel interrupt is disabled.
1	The Overrun Channel interrupt is enabled.

28.7.10 Channel n Interrupt Enable Set

Name: CHINTENSET
Offset: 0x25 + n*0x08 [n=0..11]
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
							EVD	OVR
Access							RW	RW
Reset							0	0

Bit 1 – EVD Channel Event Detected Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Event Detected Channel Interrupt Enable bit, which enables the Event Detected Channel interrupt.

Value	Description
0	The Event Detected Channel interrupt is disabled.
1	The Event Detected Channel interrupt is enabled.

Bit 0 – OVR Channel Overrun Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Overrun Channel Interrupt Enable bit, which enables the Overrun Channel interrupt.

Value	Description
0	The Overrun Channel interrupt is disabled.
1	The Overrun Channel interrupt is enabled.

28.7.11 Channel n Interrupt Flag Status and Clear

Name: CHINTFLAG
Offset: 0x26 + n*0x08 [n=0..11]
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
							EVD	OVR
Access							RW	RW
Reset							0	0

Bit 1 – EVD Channel Event Detected

This flag is set on the next CLK_EVSYS_APB cycle when an event is being propagated through the channel, and an interrupt request will be generated if CHINTENCLR/SET.EVD is '1'.
When the event channel path is asynchronous, the EVD interrupt flag will not be set.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Event Detected Channel interrupt flag.

Bit 0 – OVR Channel Overrun

This flag is set on the next CLK_EVSYS cycle after an overrun channel condition occurs, and an interrupt request will be generated if CHINTENCLR/SET.OVR is '1'.
There are two possible overrun channel conditions:

- One or more of the event users on the channel are not ready when a new event occurs.
- An event happens when the previous event on channel has not yet been handled by all event users.

When the event channel path is asynchronous, the OVR interrupt flag will not be set.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Overrun Channel interrupt flag.

28.7.12 Channel n Status

Name: CHSTATUSn
Offset: 0x27 + n*0x08 [n=0..11]
Reset: 0x01

Bit	7	6	5	4	3	2	1	0
							BUSYCH	RDYUSR
Access							R	R
Reset							0	0

Bit 1 – BUSYCH Busy Channel
This bit is cleared when channel is idle.
This bit is set if an event on channel has not been handled by all event users connected to channel.
When the event channel path is asynchronous, this bit is always read '0'.

Bit 0 – RDYUSR Ready User
This bit is cleared when at least one of the event users connected to the channel is not ready.
This bit is set when all event users connected to channel are ready to handle incoming events on the channel.
When the event channel path is asynchronous, this bit is always read zero.

28.7.13 Event User m

Name: USERm
Offset: 0x0120 + m*0x01 [m=0..51]
Reset: 0x0
Property: PAC Write-Protection

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

Bits 7:0 – CHANNEL[7:0] Channel Event Selection
These bits select channel n to connect to the event user m.
Note: A value x of this bit field selects channel n = x-1.

USER*m*	UserMultiplexer	Description	PathType(1)
m = 0	RTC_TAMPER	RTCTamper	A, S, R
m = 1..8	DMAC_CH0..7	Channel0..7	S, R
m = 9	CM4_TRACE_START	CM4trace start	A, S, R
m = 10	CM4_TRACE_STOP	CM4trace stop	A, S, R
m = 11	CM4_TRACE_TRIG	CM4trace trigger	A, S, R
m = 12..13	TCC0EV0..1	TCC0 EVx	A, S, R
m = 14..19	TCC0MC0..5	TCC0 MCx	A, S, R
m = 20..21	TCC1EV0..1	TCC1 EVx	A, S, R
m = 22..27	TCC1MC0..5	TCC1 MCx	A, S, R
m = 28..29	TCC2EV0..1	TCC2 EVx	A, S, R
m = 30..31	TCC2MC0..1	TCC2 MCx	A, S, R
m = 32	TC0 EVU	TC0 EVU	A, S, R
m = 33	TC1 EVU	TC1 EVU	A, S, R
m = 34	TC2 EVU	TC2 EVU	A, S, R
m = 35	TC3 EVU	TC3 EVU	A, S, R
m = 36..47	ADC_TRIGGER5..16	ADC_TRIGGERx	A
m = 48..49	AC_SOC0..1	AC_SOCx	A, S, R
m = 50..51	CCL_LUTIN0..1	CCL_LUTINx	A, S, R

1) A = Asynchronous path, S = Synchronous path, R = Resynchronized path

Value	Description
11	12 bits (default)
10	10 bits
01	8 bits
00	6 bits

29. Serial Communication Interface (SERCOM)

29.1 Overview

There are instances of the Serial Communication interface (SERCOM) peripheral.

A SERCOM can be configured to support a number of modes: I^2C, SPI and USART. When an instance of SERCOM is configured and enabled, all of the resources of that SERCOM instance will be dedicated to the selected mode.

The SERCOM serial engine consists of a transmitter and receiver, baud-rate generator and address matching functionality. It can use the internal generic clock or an external clock. Using an external clock allows the SERCOM to be operated in all Sleep modes.

Note: Traditional Serial Communication Interface documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

Note: SERCOM3 (4th instance of SERCOM) is only supported using Peripheral Pin Select (PPS).

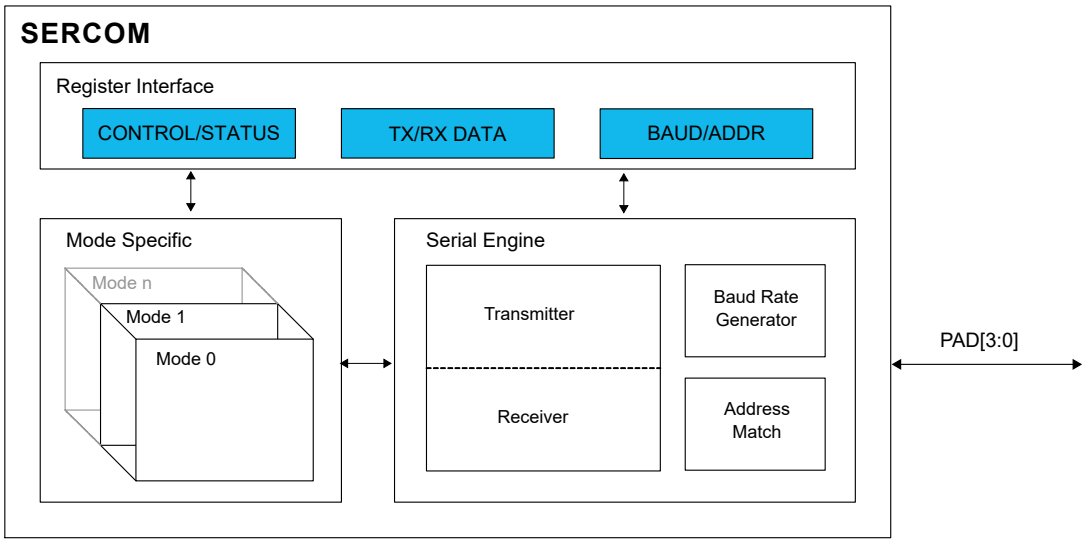
29.2 Features

- Interface for Configuring into one of the following (selected by CTRLA.MODE[2:0]):
  - Inter-Integrated Circuit (I^2C) two-wire serial interface
  - System Management Bus (SMBus™) compatible
  - Serial Peripheral Interface (SPI)
  - Universal Synchronous/Asynchronous Receiver/Transmitter (USART)
- Single Transmit Buffer and Double Receive Buffer
- Baud-rate Generator
- Address Match/mask Logic
- Operational in all Sleep modes with an External Clock Source
- Can be used with DMA

See the Related Links for full feature lists of the interface configurations.

29.3 Block Diagram

Figure 29-1. SERCOM Block Diagram



29.4 Signal Description

See the respective SERCOM mode chapters for details.

29.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

29.5.1 I/O Lines

Using the SERCOM I/O lines requires the I/O pins to be configured using the System Configuration registers or PPS registers.

The SERCOM has four internal pads, PAD[3:0], and the signals from I^2C, SPI and USART are routed through these SERCOM pads through a multiplexer. The configuration of the multiplexer is available from the different SERCOM modes. Refer to the mode specific chapters for additional information.

29.5.2 Power Management

The SERCOM can operate in any Sleep mode provided the selected clock source is running. SERCOM interrupts can be configured to wake the device from sleep modes.

29.5.3 Clocks

The SERCOM uses two generic clocks: GCLK_SERCOMx_CORE and GCLK_SERCOMx_SLOWGCLK_SERCOMx_SLOW. The core clock (GCLK_SERCOMx_CORE) is required to clock the SERCOM while working as a host. The slow clock (GCLK_SERCOMx_SLOW) is only required for certain functions. See specific mode chapters for details.

These clocks must be configured and enabled in the Clock and Reset Unit (CRU) registers before using the SERCOM.

The generic clocks are asynchronous to the bus clock (PBx_CLK). Therefore, writing to certain registers will require synchronization between the clock domains.

29.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). The DMAC must be configured before the SERCOM DMA requests are used.

29.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller (NVIC). The NVIC must be configured before the SERCOM interrupts are used.

29.5.6 Events

Not applicable.

29.5.7 Debug Operation

When the CPU is halted in Debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

29.5.8 Register Access Protection

Registers with write-access can be write-protected optionally by the Peripheral Access Controller (PAC).

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

PAC write protection does not apply to accesses through an external debugger.

29.5.9 Analog Connections

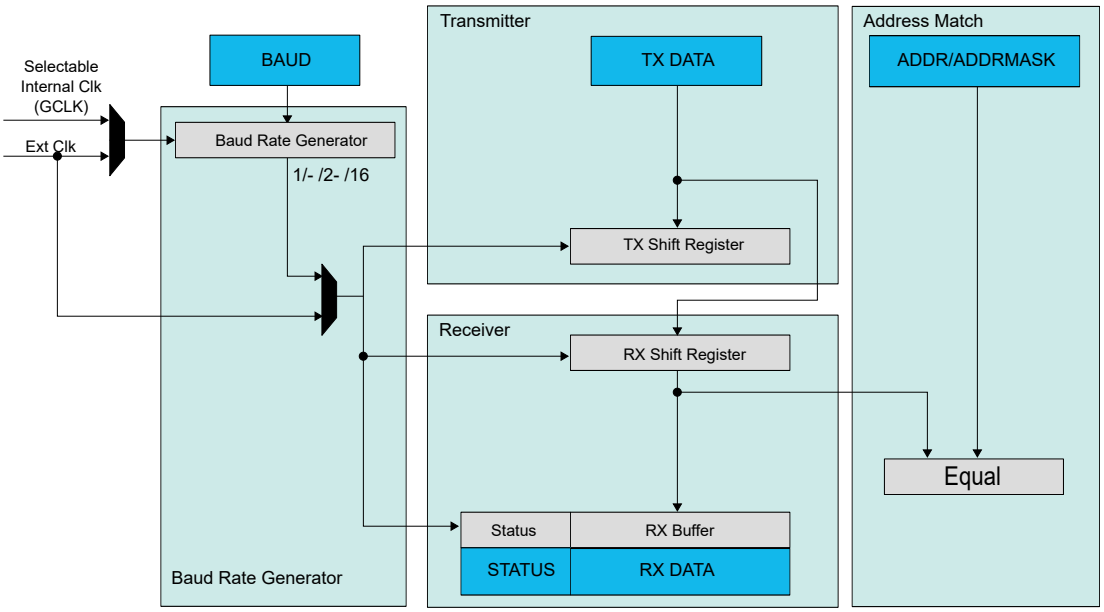
Not applicable.

29.6 Functional Description

29.6.1 Principle of Operation

The basic structure of the SERCOM serial engine is shown in *SERCOM Serial Engine*. Labels in capital letters are synchronous to the system clock and accessible by the CPU; labels in lowercase letters can be configured to run on the GCLK_SERCOMx_CORE clock or an external clock.

Figure 29-2. SERCOM Serial Engine



The transmitter consists of a single write buffer and a Shift register.

The receiver consists of a one-level (I^2C), or two-level (USART, SPI) receive buffer and a Shift register.

The baud-rate generator is capable of running on the GCLK_SERCOMx_CORE clock or an external clock.

Address matching logic is included for SPI and I^2C operation.

29.6.2 Basic Operation

29.6.2.1 Initialization

The SERCOM must be configured to the desired mode by writing the Operating Mode bits in the Control A register (CTRLA.MODE) as shown in the table below.

Table 29-1. SERCOM Modes

CTRLA.MODE	Description
0x0	USART with external clock
0x1	USART with internal clock
0x2	SPI in client operation
0x3	SPI in host operation
0x4	I^2C client operation

.....continued	
CTRLA.MODE	**Description**
0x5	I^2C host operation
0x6-0x7	Reserved

For further initialization information, see the respective SERCOM mode chapters:

29.6.2.2 Enabling, Disabling, and Resetting

This peripheral is enabled by writing '1' to the Enable bit in the Control A register (CTRLA.ENABLE), and disabled by writing '0' to it.

Writing '1' to the Software Reset bit in the Control A register (CTRLA.SWRST) will reset all registers of this peripheral to their initial states, except the DBGCTRL register, and the peripheral is disabled.

Refer to the CTRLA register description for details.

Related Links

30.8.1. CTRLA

29.6.2.3 Clock Generation – Baud-Rate Generator

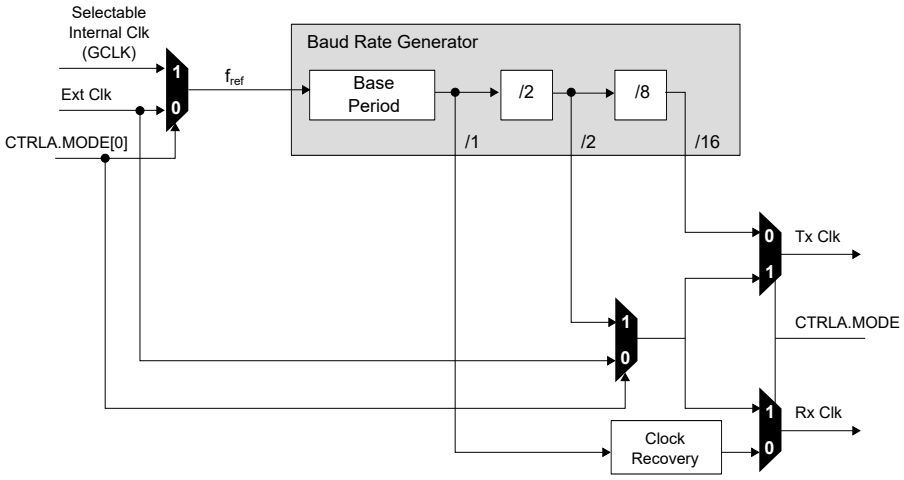
The baud-rate generator, as shown in the following figure, generates internal clocks for asynchronous and synchronous communication. The output frequency (f_{BAUD}) is determined by the Baud register (BAUD) setting and the baud reference frequency (f_{ref}). The baud reference clock is the serial engine clock, and it can be internal or external.

For asynchronous communication, the /16 (divide-by-16) output is used when transmitting, whereas the /1 (divide-by-1) output is used while receiving.

For synchronous communication, the /2 (divide-by-2) output is used.

This functionality is automatically configured, depending on the selected operating mode.

Figure 29-3. Baud Rate Generator



The following table contains equations for the baud rate (in bits per second) and the BAUD register value for each operating mode.

For asynchronous operation, there are two modes:

- *Arithmetic mode*: the BAUD register value is 16 bits (0 to 65,535)
- *Fractional mode*: the BAUD register value is 13 bits, while the fractional adjustment is 3 bits. In this mode the BAUD setting must be greater than or equal to 1.

For synchronous operation, the BAUD register value is 8 bits (0 to 255).

Table 29-2. Baud Rate Equations

Operating Mode	Condition	Baud Rate (Bits Per Second)	BAUD Register Value Calculation
Asynchronous Arithmetic	$f_{BAUD} \leq \frac{f_{ref}}{16}$	$f_{BAUD} = \frac{f_{ref}}{16}\left(1 - \frac{BAUD}{65536}\right)$	$BAUD = 65536 \cdot \left(1 - S \cdot \frac{f_{BAUD}}{f_{ref}}\right)$
Asynchronous Fractional	$f_{BAUD} \leq \frac{f_{ref}}{S}$	$f_{BAUD} = \frac{f_{ref}}{S \cdot \left(BAUD + \frac{FP}{8}\right)}$	$BAUD = \frac{f_{ref}}{S \cdot f_{BAUD}} - \frac{FP}{8}$
Synchronous	$f_{BAUD} \leq \frac{f_{ref}}{2}$	$f_{BAUD} = \frac{f_{ref}}{2 \cdot (BAUD + 1)}$	$BAUD = \frac{f_{ref}}{2 \cdot f_{BAUD}} - 1$

S - Number of samples per bit, which can be 16, 8, or 3.

The Asynchronous Fractional option is used for auto-baud detection.

The baud rate error is represented by the following formula:

$$\text{Error} = 1 - \left(\frac{\text{ExpectedBaudRate}}{\text{ActualBaudRate}}\right)$$

29.6.3 Additional Features

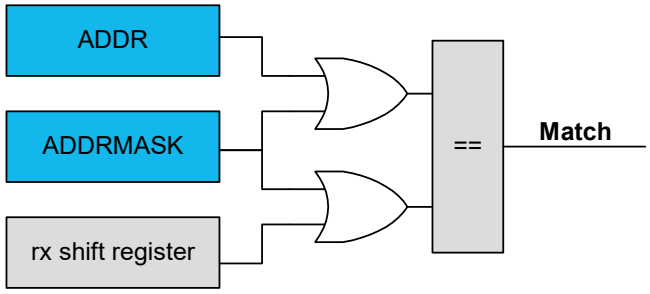
29.6.3.1 Address Match and Mask

The SERCOM address match and mask feature is capable of matching either one address, two unique addresses, or a range of addresses with a mask, based on the mode selected. The match uses seven or eight bits, depending on the mode.

29.6.3.1.1 Address With Mask

An address written to the Address bits in the Address register (ADDR.ADDR), and a mask written to the Address Mask bits in the Address register (ADDR.ADDRMASK) will yield an address match. All bits that are masked are not included in the match. Note that writing the ADDR.ADDRMASK to 'all zeros' will match a single unique address, while writing ADDR.ADDRMASK to 'all ones' will result in all addresses being accepted.

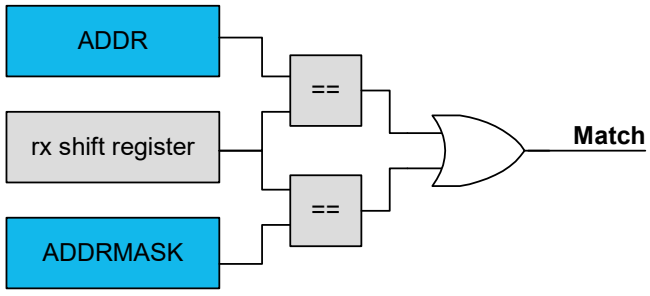
Figure 29-4. Address With Mask



29.6.3.1.2 Two Unique Addresses

The two addresses written to ADDR and ADDRMASK will cause a match.

Figure 29-5. Two Unique Addresses



29.6.3.1.3 Address Range

The range of addresses between and including ADDR.ADDR and ADDR.ADDRMASK will cause a match. ADDR.ADDR and ADDR.ADDRMASK can be set to any two addresses, with ADDR.ADDR acting as the upper limit and ADDR.ADDRMASK acting as the lower limit.

Figure 29-6. Address Range

29.6.4 DMA Operation

The available DMA interrupts and their depend on the operation mode of the SERCOM peripheral. Refer to the Functional Description sections of the respective SERCOM mode.

29.6.5 Interrupts

Interrupt sources are mode specific. See the respective SERCOM mode chapters for details.

Each interrupt source has its own Interrupt flag.

The Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) will be set when the Interrupt condition is met.

Each interrupt can be individually enabled by writing '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing '1' to the corresponding bit in the Interrupt Enable Clear register (INTENCLR).

An interrupt request is generated when the Interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until either the Interrupt flag is cleared, the interrupt is disabled, or the SERCOM is reset. For details on clearing Interrupt flags, refer to the INTFLAG register description.

The value of INTFLAG indicates which Interrupt condition occurred. The user must read the INTFLAG register to determine which Interrupt condition is present.

Note: Interrupts must be globally enabled for interrupt requests to be generated.

Related Links

30.8.8. INTFLAG

29.6.6 Events

Not applicable.

29.6.7 Sleep Mode Operation

The peripheral can operate in any Sleep mode where the selected serial clock is running. This clock can be external or generated by the internal baud-rate generator.

The SERCOM interrupts can be used to wake-up the device from Sleep modes. Refer to the different SERCOM mode chapters for details.

29.6.8 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

Required write synchronization is denoted by the "Write-Synchronized" property in the register description.

Required read synchronization is denoted by the "Read-Synchronized" property in the register description.

30. SERCOM Synchronous and Asynchronous Receiver and Transmitter (SERCOM USART)

30.1 Overview

The Universal Synchronous and Asynchronous Receiver and Transmitter (USART) is one of the available modes in the Serial Communication Interface (SERCOM).

The USART uses the SERCOM transmitter and receiver (see *USART Block Diagram* in the *Block Diagram* section from Related Links). Labels in uppercase letters are synchronous to PBx_CLK and accessible for CPU. Labels in lowercase letters can be programmed to run on the internal generic clock or an external clock.

The transmitter consists of a single write buffer, a Shift register, and control logic for different frame formats. The write buffer supports data transmission without any delay between frames. The receiver consists of a two-level receive buffer and a Shift register. Status information of the received data is available for error checking. Data and clock recovery units ensure robust synchronization and noise filtering during asynchronous data reception.

Note: Traditional Universal Synchronous and Asynchronous Receiver and Transmitter (USART) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Commander” and “Responder”, respectively.

Related Links

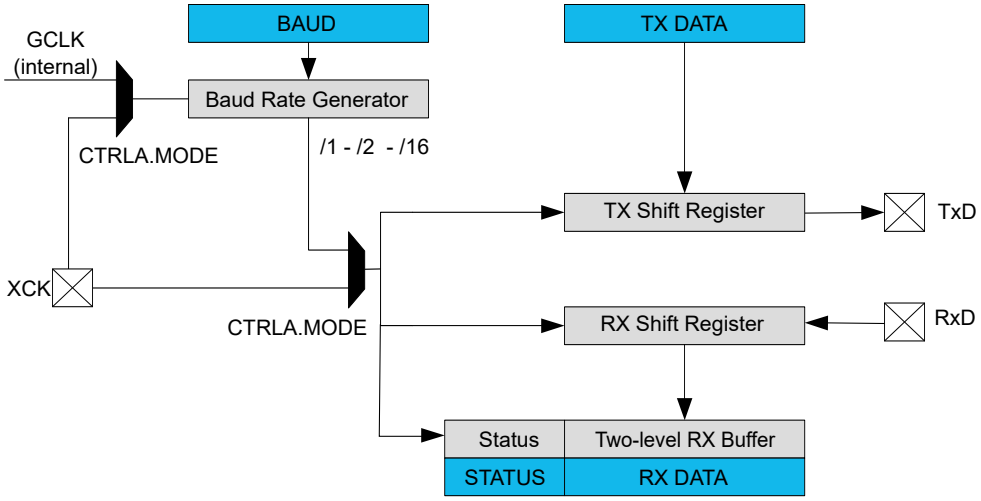
30.3. Block Diagram

30.2 USART Features

- Full-duplex Operation
- Asynchronous (with Clock Reconstruction) or Synchronous Operation
- Internal or External Clock source for Asynchronous and Synchronous Operation
- Baud-rate Generator
- Supports Serial Frames with 5, 6, 7, 8 or 9 Data bits and 1 or 2 Stop bits
- Odd or Even Parity Generation and Parity Check
- Selectable LSB- or MSB-first Data Transfer
- Buffer Overflow and Frame Error Detection
- Noise Filtering, Including False Start bit Detection and Digital Low-pass Filter
- Collision Detection
- Can Operate in all Sleep modes
- Operation at Speeds up to Half the System Clock for Internally Generated Clocks
- Operation at Speeds up to the System Clock for Externally Generated Clocks
- RTS and CTS Flow Control
- IrDA Modulation and Demodulation up to 115.2 kbps
- LIN Commander Support
- LIN Responder Support
  - Auto-baud and break character detection
- Start-of-frame detection
- Can work with DMA

30.3 Block Diagram

Figure 30-1. USART Block Diagram



30.4 Signal Description

Table 30-1. SERCOM USART Signals

Signal Name	Type	Description
PAD[3:0]	Digital I/O	General SERCOM pins

One signal can be mapped to one of several pins.

30.5 Product Dependencies

To use this peripheral, other parts of the system must be configured correctly, as described below.

30.5.1 I/O Lines

Using the USART's I/O lines requires the I/O pins to be configured using the System Configuration registers or PPS registers.

When the SERCOM is used in USART mode, the SERCOM controls the direction and value of the I/O pins according to the table below. If the receiver or transmitter is disabled, these pins can be used for other purposes.

Table 30-2. USART Pin Configuration

Pin	Pin Configuration
TxD	Output
RxD	Input
XCK	Output or input

The combined configuration of PORT and the Transmit Data Pinout and Receive Data Pinout bit fields in the Control A register (CTRLA.TXPO and CTRLA.RXPO, respectively) will define the physical position of the USART signals in the above table.

30.5.2 Power Management

This peripheral can continue to operate in any Sleep mode where its source clock is running. The interrupts can wake-up the device from Sleep modes.

30.5.3 Clocks

A generic clock (GCLK_SERCOMx_CORE) is required to clock the SERCOMx_CORE. This clock must be configured and enabled in the CRU registers before using the SERCOMx_CORE. See *Clock and Reset (CRU)* and *Peripheral Module Disable Register (PMD)* from Related Links.

This generic clock is asynchronous to the bus clock (PBx_CLK). Therefore, writing to certain registers will require synchronization to the clock domains.

Related Links

20. Peripheral Module Disable Register (PMD)
13. Clock and Reset Unit (CRU)

30.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). To use DMA requests with this peripheral, the DMAC must be configured first (see *Direct Memory Access Controller (DMAC)* from Related Links).

Related Links

22. Direct Memory Access Controller (DMAC)

30.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. In order to use interrupt requests of this peripheral, the NVIC must be configured first. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)
30.8.8. INTFLAG

30.5.6 Events

Not applicable.

30.5.7 Debug Operation

When the CPU is halted in Debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

Related Links

30.8.12. DBGCTRL

30.5.8 Register Access Protection

Registers with write access can be write-protected optionally by the Peripheral Access Controller (PAC).

PAC write protection is not available for the following registers:

- Interrupt Flag Clear and Status register (INTFLAG)
- Status register (STATUS)
- Data register (DATA)

Optional PAC write protection is denoted by the "PAC Write-Protection" property in each individual register description.

Write-protection does not apply to accesses through an external debugger.

30.5.9 Analog Connections

Not applicable.

30.6 Functional Description

30.6.1 Principle of Operation

The USART uses the following lines for data transfer:

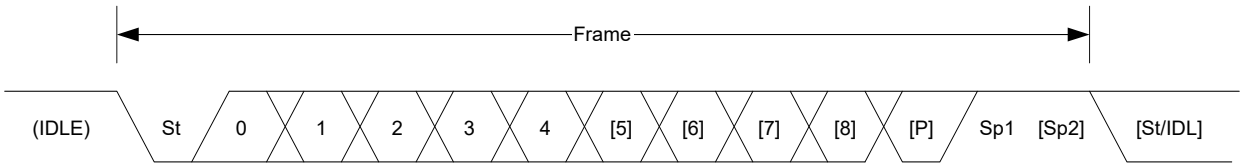
- RxD for receiving
- TxD for transmitting
- XCK for the transmission clock in synchronous operation

USART data transfer is frame based. A serial frame consists of:

- 1 start bit
- From 5 to 9 data bits (MSB or LSB first)
- No, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the Start bit followed by one character of Data bits. If enabled, the parity bit is inserted after the Data bits and before the first Stop bit. After the stop bit(s) of a frame, either the next frame can follow immediately, or the communication line can return to the Idle (high) state. The figure below illustrates the possible frame formats. Values inside brackets ([x]) denote optional bits.

Figure 30-2. Frame Formats



St Start bit. Signal is always low.

n, [n] Data bits. 0 to [5..9]

[P] Parity bit. Either odd or even.

Sp, [Sp] Stop bit. Signal is always high.

IDLE No frame is transferred on the communication line. Signal is always high in this state.

30.6.2 Basic Operation

30.6.2.1 Initialization

The following registers are enable-protected, meaning they can only be written when the USART is disabled (CTRL.ENABLE=0):

- Control A register (CTRLA), except the Enable (ENABLE) and Software Reset (SWRST) bits.
- Control B register (CTRLB), except the Receiver Enable (RXEN) and Transmitter Enable (TXEN) bits.
- Baud register (BAUD)

When the USART is enabled or is being enabled (CTRLA.ENABLE=1), any writing attempt to these registers will be discarded. If the peripheral is being disabled, writing to these registers will be executed after disabling is completed. Enable-protection is denoted by the "Enable-Protection" property in the register description.

Before the USART is enabled, it must be configured by these steps:

1. Select either external (0x0) or internal clock (0x1) by writing the Operating Mode value in the CTRLA register (CTRLA.MODE).
2. Select either Asynchronous (0) or Synchronous (1) Communication mode by writing the Communication Mode bit in the CTRLA register (CTRLA.CMODE).

3. Select pin for receive data by writing the Receive Data Pinout value in the CTRLA register (CTRLA.RXPO).
4. Select pads for the transmitter and external clock by writing the Transmit Data Pinout bit in the CTRLA register (CTRLA.TXPO).
5. Configure the Character Size field in the CTRLB register (CTRLB.CHSIZE) for character size.
6. Set the Data Order bit in the CTRLA register (CTRLA.DORD) to determine MSB- or LSB-first data transmission.
7. To use parity mode:
   a. Enable Parity mode by writing 0x1 to the Frame Format field in the CTRLA register (CTRLA.FORM).
   b. Configure the Parity Mode bit in the CTRLB register (CTRLB.PMODE) for even or odd parity.
8. Configure the number of stop bits in the Stop Bit Mode bit in the CTRLB register (CTRLB.SBMODE).
9. When using an internal clock, write the Baud register (BAUD) to generate the desired baud rate.
10. Enable the transmitter and receiver by writing '1' to the Receiver Enable and Transmitter Enable bits in the CTRLB register (CTRLB.RXEN and CTRLB.TXEN).

30.6.2.2 Enabling, Disabling, and Resetting

This peripheral is enabled by writing '1' to the Enable bit in the Control A register (CTRLA.ENABLE), and disabled by writing '0' to it.

Writing '1' to the Software Reset bit in the Control A register (CTRLA.SWRST) will reset all registers of this peripheral to their initial states, except the DBGCTRL register, and the peripheral is disabled.

Refer to the CTRLA register description for details.

Related Links

30.8.1. CTRLA

30.6.2.3 Clock Generation and Selection

For both Synchronous and Asynchronous modes, the clock used for shifting and sampling data can be generated internally by the SERCOM baud-rate generator or supplied externally through the XCK line.

The Synchronous mode is selected by writing a '1' to the Communication Mode bit in the Control A register (CTRLA.CMODE), the Asynchronous mode is selected by writing '0' to CTRLA.CMODE.

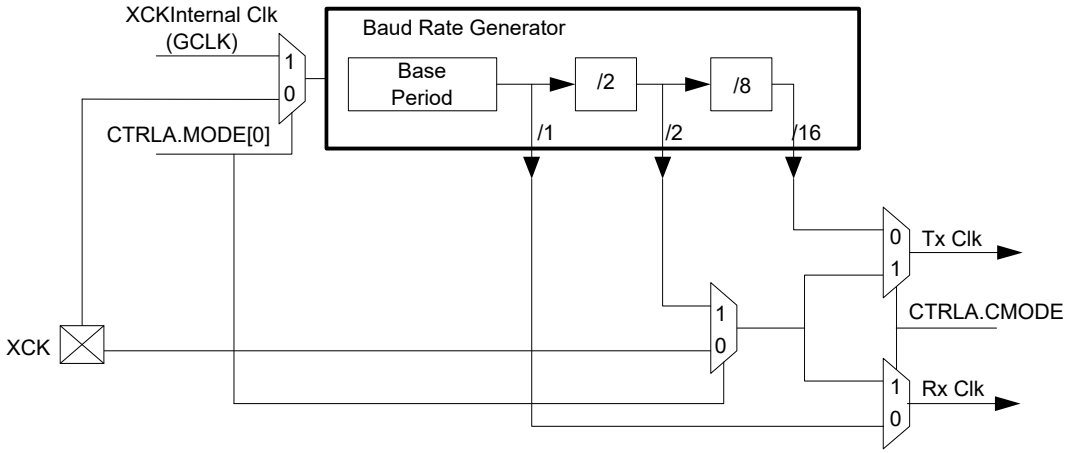
The internal clock source is selected by writing '1' to the Operation Mode bit field in the Control A register (CTRLA.MODE), the external clock source is selected by writing '0' to CTRLA.MODE.

The SERCOM baud-rate generator is configured as in the following figure.

In Asynchronous mode (CTRLA.CMODE=0), the 16-bit Baud register value is used.

In Synchronous mode (CTRLA.CMODE=1), the eight LSBs of the Baud register are used. For more details on configuring the baud rate (see *Clock Generation – Baud-Rate Generator* from Related Links).

Figure 30-3. Clock Generation



Related Links

29.6.2.3. Clock Generation – Baud-Rate Generator

30.6.2.3.1 Synchronous Clock Operation

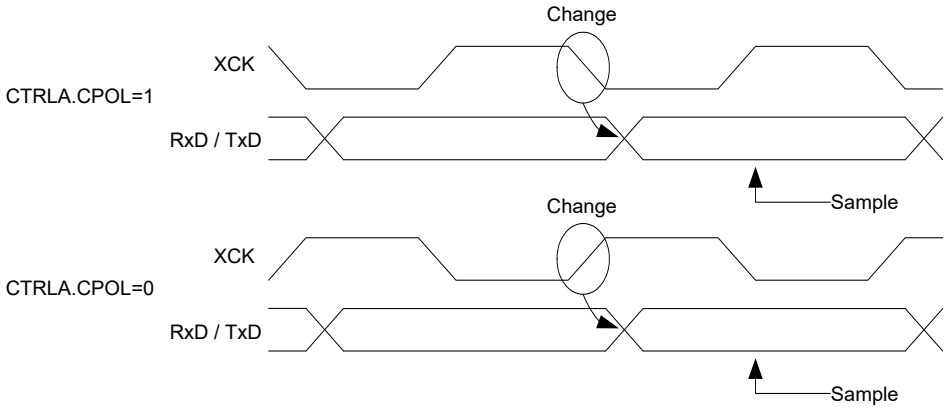
In Synchronous mode, the CTRLA.MODE bit field determines whether the transmission clock line (XCK) serves either as input or output. The dependency between clock edges, data sampling, and data change is the same for internal and external clocks. Data input on the RxD pin is sampled at the opposite XCK clock edge when data is driven on the TxD pin.

The Clock Polarity bit in the Control A register (CTRLA.CPOL) selects which XCK clock edge is used for RxD sampling, and which is used for TxD change:

When CTRLA.CPOL is '0', the data will be changed on the rising edge of XCK, and sampled on the falling edge of XCK.

When CTRLA.CPOL is '1', the data will be changed on the falling edge of XCK, and sampled on the rising edge of XCK.

Figure 30-4. Synchronous Mode XCK Timing



When the clock is provided through XCK (CTRLA.MODE=0x0), the Shift registers operate directly on the XCK clock. This means that XCK is not synchronized with the system clock and, therefore, can operate at frequencies up to the system frequency.

30.6.2.4 Data Register

The USART Transmit Data register (TxDATA) and USART Receive Data register (RxDATA) share the same I/O address, referred to as the Data register (DATA). Writing the DATA register will update the TxDATA register. Reading the DATA register will return the contents of the RxDATA register.

30.6.2.5 Data Transmission

Data transmission is initiated by writing the data to be sent into the DATA register. Then, the data in TxDATA will be moved to the Shift register when the Shift register is empty and ready to send a new frame. After the Shift register is loaded with data, the data frame will be transmitted.

When the entire data frame including Stop bit(s) has been transmitted and no new data was written to DATA, the Transmit Complete Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.TXC) will be set, and the optional interrupt will be generated.

The Data Register Empty flag in the Interrupt Flag Status and Clear register (INTFLAG.DRE) indicates that the register is empty and ready for new data. The DATA register must be written to when INTFLAG.DRE is set.

Disabling the Transmitter

The transmitter is disabled by writing '0' to the Transmitter Enable bit in the CTRLB register (CTRLB.TXEN).

Disabling the transmitter will complete only after any ongoing and pending transmissions are completed, in other words, there is no data in the transmit shift register and TxDATA to transmit.

30.6.2.5.1 Disabling the Transmitter

The transmitter is disabled by writing '0' to the Transmitter Enable bit in the CTRLB register (CTRLB.TXEN).

Disabling the transmitter will complete only after any ongoing and pending transmissions are completed, i.e., there is no data in the Transmit Shift register and TxDATA to transmit.

30.6.2.6 Data Reception

The receiver accepts data when a valid Start bit is detected. Each bit following the Start bit will be sampled according to the baud rate or XCK clock, and shifted into the receive Shift register until the first Stop bit of a frame is received. The second Stop bit will be ignored by the receiver.

When the first Stop bit is received and a complete serial frame is present in the Receive Shift register, the contents of the Shift register will be moved into the two-level receive buffer. Then, the Receive Complete Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.RXC) will be set, and the optional interrupt will be generated.

The received data can be read from the DATA register when the Receive Complete Interrupt flag is set.

Disabling the Receiver

Writing '0' to the Receiver Enable bit in the CTRLB register (CTRLB.RXEN) will disable the receiver, flush the two-level receive buffer, and data from ongoing receptions will be lost.

Error Bits

The USART receiver has three error bits in the Status (STATUS) register: Frame Error (FERR), Buffer Overflow (BUFOVF), and Parity Error (PERR). Once an error happens, the corresponding error bit will be set until it is cleared by writing ‘1’ to it. These bits are also cleared automatically when the receiver is disabled.

There are two methods for buffer overflow notification, selected by the Immediate Buffer Overflow Notification bit in the Control A register (CTRLA.IBON):

When CTRLA.IBON=1, STATUS.BUFOVF is raised immediately upon buffer overflow. Software can then empty the receive FIFO by reading RxDATA, until the receiver complete interrupt flag (INTFLAG.RXC) is cleared.

When CTRLA.IBON=0, the buffer overflow condition is attending data through the receive FIFO. After the received data is read, STATUS.BUFOVF will be set along with INTFLAG.RXC.

Asynchronous Data Reception

The USART includes a clock recovery and data recovery unit for handling asynchronous data reception.

The clock recovery logic can synchronize the incoming asynchronous serial frames at the RxD pin to the internally generated baud-rate clock.

The data recovery logic samples and applies a low-pass filter to each incoming bit, thereby improving the noise immunity of the receiver.

Asynchronous Operational Range

The operational range of the asynchronous reception depends on the accuracy of the internal baud-rate clock, the rate of the incoming frames, and the frame size (in number of bits). In addition, the operational range of the receiver is depending on the difference between the received bit rate and the internally generated baud rate. If the baud rate of an external transmitter is too high or too low compared to the internally generated baud rate, the receiver will not be able to synchronize the frames to the start bit.

There are two possible sources for a mismatch in baud rate: First, the reference clock will always have some minor instability. Second, the baud-rate generator cannot always do an exact division of the reference clock frequency to get the baud rate desired. In this case, the BAUD register value must be set to give the lowest possible error, see *Clock Generation – Baud-Rate Generator* from Related Links.

Recommended maximum receiver baud-rate errors for various character sizes are shown in the table below.

Table 30-3. Asynchronous Receiver Error for 16-fold Oversampling

D (Data bits+Parity)	R_{SLOW} [%]	R_{FAST} [%]	Max. total error [%]	Recommended max. Rx error [%]
5	94.12	107.69	+5.88/-7.69	±2.5
6	94.92	106.67	+5.08/-6.67	±2.0

.....continued				
D (Data bits+Parity)	R_{SLOW} [%]	R_{FAST} [%]	Max. total error [%]	Recommended max. Rx error [%]
7	95.52	105.88	+4.48/-5.88	±2.0
8	96.00	105.26	+4.00/-5.26	±2.0
9	96.39	104.76	+3.61/-4.76	±1.5
10	96.70	104.35	+3.30/-4.35	±1.5

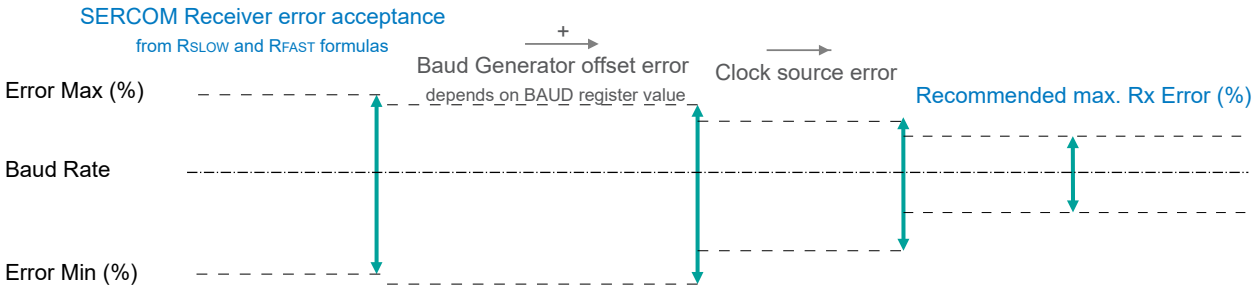
The following equations calculate the ratio of the incoming data rate and internal receiver baud rate:

$$R_{\mathrm{SLOW}} = \frac{16(D+1)}{16(D+1)+6} \quad , \qquad R_{\mathrm{FAST}} = \frac{16(D+2)}{16(D+1)+8}$$

- R_{SLOW} is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate
- R_{FAST} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate
- D is the sum of character size and parity size (D = 5 to 10 bits)

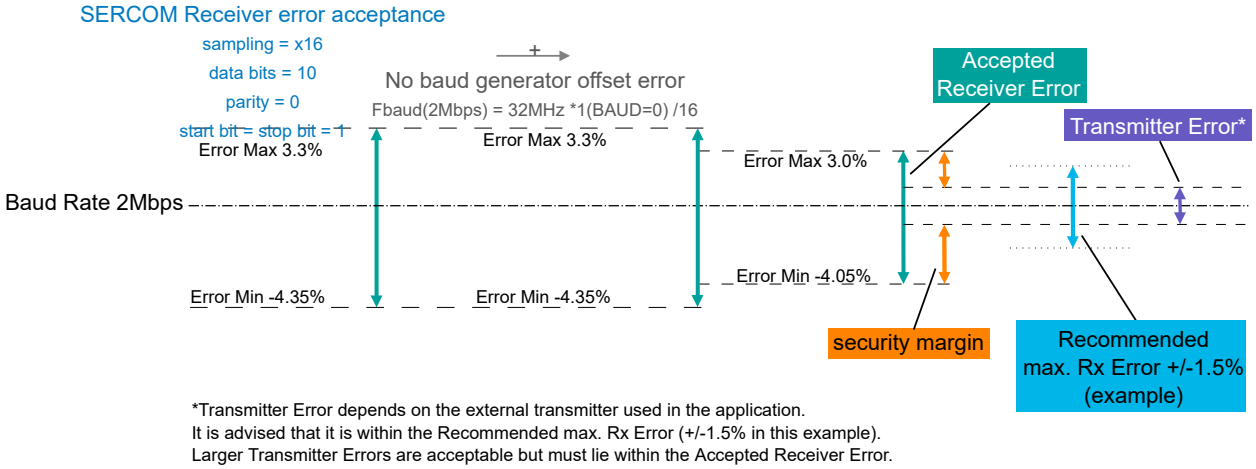
The recommended maximum Rx Error assumes that the receiver and transmitter equally divide the maximum total error. Its connection to the SERCOM Receiver error acceptance is depicted in this figure:

Figure 30-5. USART Rx Error Calculation



The recommendation values in the table above accommodate errors of the clock source and the baud generator. The following figure gives an example for a baud rate of 3Mbps:

Figure 30-6. USART Rx Error Calculation Example



Related Links

29.6.2.3. Clock Generation – Baud-Rate Generator

30.6.2.6.1 Disabling the Receiver

Writing '0' to the Receiver Enable bit in the CTRLB register (CTRLB.RXEN) will disable the receiver, flush the two-level receive buffer, and data from ongoing receptions will be lost.

30.6.2.6.2 Error Bits

The USART receiver has three error bits in the Status (STATUS) register: Frame Error (FERR), Buffer Overflow (BUFOVF), and Parity Error (PERR). Once an error happens, the corresponding error bit will be set until it is cleared by writing '1' to it. These bits are also cleared automatically when the receiver is disabled.

There are two methods for buffer overflow notification, selected by the Immediate Buffer Overflow Notification bit in the Control A register (CTRLA.IBON):

When CTRLA.IBON=1, STATUS.BUFOVF is raised immediately upon buffer overflow. Software can then empty the receive FIFO by reading RxDATA, until the Receiver Complete Interrupt flag (INTFLAG.RXC) is cleared.

When CTRLA.IBON=0, the Buffer Overflow condition is attending data through the receive FIFO, which will then set the INTFLAG.ERROR bit. After the received data is read, STATUS.BUFOVF (and INTFLAG.ERROR) will be set along with INTFLAG.RXC.

30.6.2.6.3 Asynchronous Data Reception

The USART includes a clock recovery and data recovery unit for handling asynchronous data reception.

The clock recovery logic can synchronize the incoming asynchronous serial frames at the RxD pin to the internally generated baud-rate clock.

The data recovery logic samples and applies a low-pass filter to each incoming bit, thereby improving the noise immunity of the receiver.

30.6.2.6.4 Asynchronous Operational Range

The operational range of the asynchronous reception depends on the accuracy of the internal baud-rate clock, the rate of the incoming frames, and the frame size (in number of bits). In addition, the operational range of the receiver is depending on the difference between the received bit rate and the internally generated baud rate. If the baud rate of an external transmitter is too high or too low compared to the internally generated baud rate, the receiver will not be able to synchronize the frames to the start bit.

There are two possible sources for a mismatch in baud rate: First, the reference clock will always have some minor instability. Second, the baud-rate generator cannot always do an exact division of the reference clock frequency to

get the baud rate desired. In this case, the BAUD register value must be set to give the lowest possible error (see *Clock Generation – Baud-Rate Generator* from Related Links).

Recommended maximum receiver baud-rate errors for various character sizes are shown in the following table.

Table 30-4. Asynchronous Receiver Error for 16-fold Oversampling

D (Data bits+Parity)	R_{SLOW} [%]	R_{FAST} [%]	Max. total error [%]	Recommended max. Rx error [%]
5	94.12	107.69	+5.88/-7.69	±2.5
6	94.92	106.67	+5.08/-6.67	±2.0
7	95.52	105.88	+4.48/-5.88	±2.0
8	96.00	105.26	+4.00/-5.26	±2.0
9	96.39	104.76	+3.61/-4.76	±1.5
10	96.70	104.35	+3.30/-4.35	±1.5

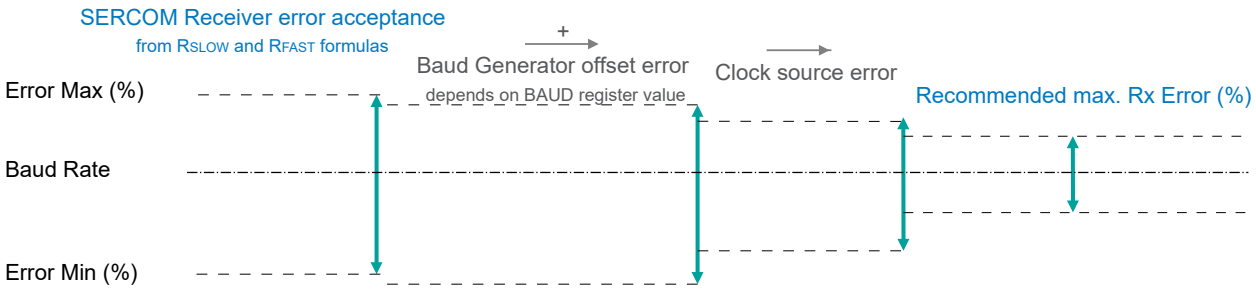
The following equations calculate the ratio of the incoming data rate and internal receiver baud rate:

$$R_{\text{SLOW}} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \quad , \qquad R_{\text{FAST}} = \frac{(D+2)S}{(D+1)S+S_M}$$

- R_{SLOW} is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate
- R_{FAST} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate
- D is the sum of character size and parity size (D = 5 to 10 bits)
- S is the number of samples per bit (S = 16, 8 or 3)
- S_F is the first sample number used for majority voting (S_F = 7, 3 or 2) when CTRLA.SAMPA=0.
- S_M is the middle sample number used for majority voting (S_M = 8, 4 or 2) when CTRLA.SAMPA=0.

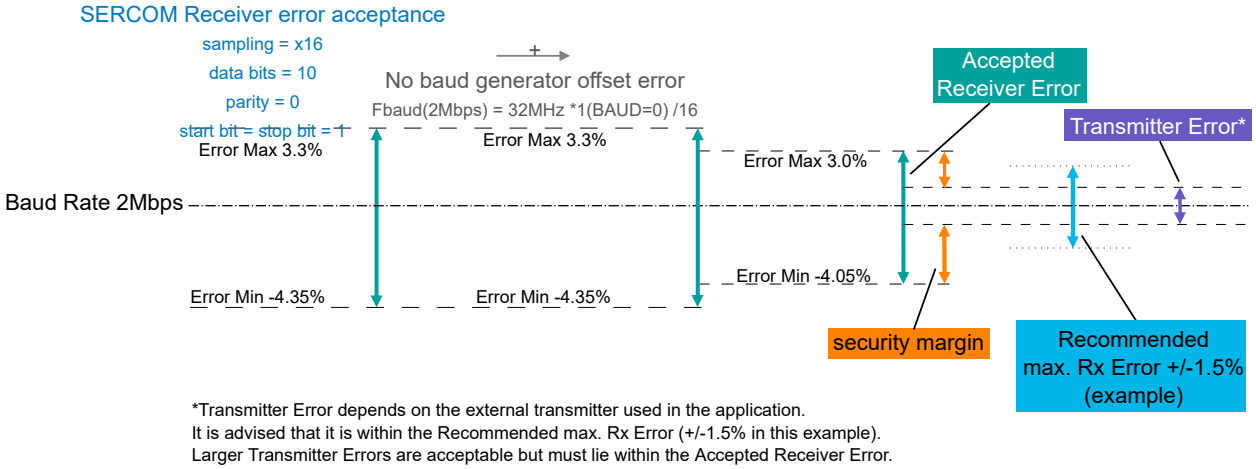
The recommended maximum Rx Error assumes that the receiver and transmitter equally divide the maximum total error. Its connection to the SERCOM Receiver error acceptance is depicted in this figure:

Figure 30-7. USART Rx Error Calculation



The recommendation values in the table above accommodate errors of the clock source and the baud generator. The following figure gives an example for a baud rate of 3 Mbps:

Figure 30-8. USART Rx Error Calculation Example



*Transmitter Error depends on the external transmitter used in the application.
It is advised that it is within the Recommended max. Rx Error (+/-1.5% in this example).
Larger Transmitter Errors are acceptable but must lie within the Accepted Receiver Error.

Related Links

29.6.2.3. Clock Generation – Baud-Rate Generator

30.6.3 Additional Features

30.6.3.1 Parity

Even or odd parity can be selected for error checking by writing 0x1 to the Frame Format bit field in the Control A register (CTRLA.FORM).

If *even parity* is selected (CTRLB.PMODE=0), the Parity bit of an outgoing frame is '1' if the data contains an odd number of bits that are '1', making the total number of '1' even.

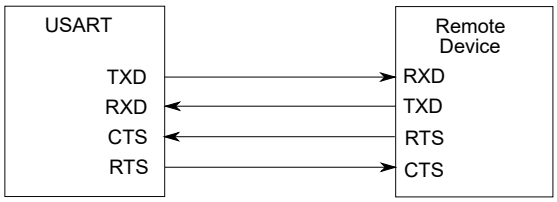
If *odd parity* is selected (CTRLB.PMODE=1), the Parity bit of an outgoing frame is '1' if the data contains an even number of bits that are '0', making the total number of '1' odd.

When parity checking is enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the Parity bit of the corresponding frame. If a parity error is detected, the Parity Error bit in the Status register (STATUS.PERR) is set.

30.6.3.2 Hardware Handshaking

The USART features an out-of-band hardware handshaking flow control mechanism, implemented by connecting the RTS and CTS pins with the remote device, as shown in the figure below.

Figure 30-9. Connection with a Remote Device for Hardware Handshaking



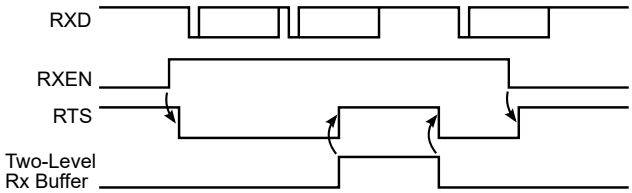
Hardware handshaking is only available in the following configuration:

- USART with internal clock (CTRLA.MODE=1),
- Asynchronous mode (CTRLA.CMODE=0), and
- Flow control pinout (CTRLA.TXPO=2).

When the receiver is disabled or the receive FIFO is full, the receiver will drive the RTS pin high. This notifies the remote device to stop transfer after the ongoing transmission. Enabling and disabling the receiver by writing to

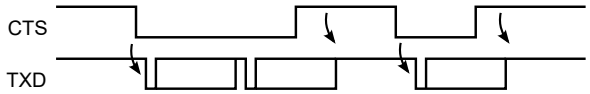
CTRLB.RXEN will set/clear the RTS pin after a synchronization delay. When the receive FIFO goes full, RTS will be set immediately and the frame being received will be stored in the Shift register until the receive FIFO is no longer full.

Figure 30-10. Receiver Behavior when Operating with Hardware Handshaking



The current CTS Status is in the STATUS register (STATUS.CTS). Character transmission will start only if STATUS.CTS=0. When CTS is set, the transmitter will complete the ongoing transmission and stop transmitting.

Figure 30-11. Transmitter Behavior when Operating with Hardware Handshaking



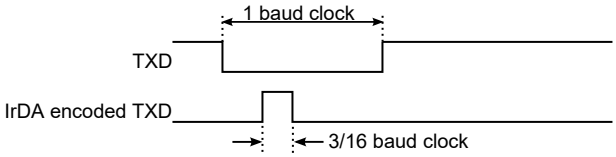
30.6.3.3 IrDA Modulation and Demodulation

Transmission and reception can be encoded IrDA compliant up to 115.2 kb/s. IrDA modulation and demodulation work in the following configuration:

- IrDA encoding enabled (CTRLB.ENC=1)
- Asynchronous mode (CTRLA.CMODE=0)
- 16x sample rate (CTRLA.SAMPR[0]=0)

During transmission, each low bit is transmitted as a high pulse. The pulse width is 3/16 of the baud rate period, as illustrated in the following figure.

Figure 30-12. IrDA Transmit Encoding



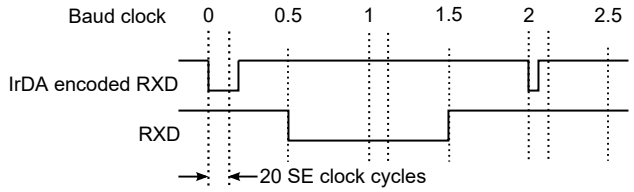
The reception decoder has two main functions:

- To synchronize the incoming data to the IrDA baud rate counter. Synchronization is performed at the start of each zero pulse.
- To decode incoming Rx data. If a pulse width meets the minimum length set by configuration (RXPL.RXPL), it is accepted. When the baud rate counter reaches its middle value (1/2 bit length), it is transferred to the receiver.

Note: The polarity of the transmitter and receiver are opposite: During transmission, a '0' bit is transmitted as a '1' pulse. During reception, an accepted '0' pulse is received as a '0' bit.

> **Example:** The following figure illustrates reception where RXPL.RXPL is set to 19. This indicates that the pulse width must be at least 20 SE clock cycles. When using BAUD=0xE666 or 160 SE cycles per bit, this corresponds to 2/16 baud clock as minimum pulse width required. In this case the first bit is accepted as a '0', the second bit is a '1', and the third bit is also a '1'. A low pulse is rejected since it does not meet the minimum requirement of 2/16 baud clock.

Figure 30-13. IrDA Receive Decoding



30.6.3.4 Break Character Detection and Auto-Baud

Break character detection and auto-baud are available in this configuration:

- Auto-baud frame format (CTRLA.FORM = 0x04 or 0x05),
- Asynchronous mode (CTRLA.CMODE = 0),
- and 16x sample rate using fractional baud rate generation (CTRLA.SAMPR = 1).

The USART uses a break detection threshold of greater than 11 nominal bit times at the configured baud rate. At any time, if more than 11 consecutive dominant bits are detected on the bus, the USART detects a Break Field. When a break field has been detected, the Receive Break Interrupt Flag (INTFLAG.RXBRK) is set and the USART expects the sync field character to be 0x55. This field is used to update the actual baud rate in order to stay synchronized. If the received sync character is not 0x55, then the Inconsistent Sync Field error flag (STATUS.ISF) is set along with the Error Interrupt Flag (INTFLAG.ERROR), and the baud rate is unchanged.

After a break field is detected and the Start bit of the sync field is detected, a counter is started. The counter is then incremented for the next 8 bit times of the sync field. At the end of these 8 bit times, the counter is stopped. At this moment, the 13 Most Significant bits of the counter (value divided by 8) give the new clock divider (BAUD.BAUD), and the 3 Least Significant bits of this value (the remainder) give the new Fractional Part (BAUD.FP).

When the sync field has been received, the clock divider (BAUD.BAUD) and the Fractional Part (BAUD.FP) are updated after a synchronization delay. After the break and sync fields are received, multiple characters of data can be received.

30.6.3.5 LIN Commander

LIN commander is available with the following configuration:

- LIN commander format (CTRLA.FORM = 0x02)
- Asynchronous mode (CTRLA.CMODE = 0)
- 16x sample rate using fractional baud rate generation (CTRLA.SAMPR = 1)

LIN frames start with a header transmitted by the commander. The header consists of the break, sync, and identifier fields. After the commander transmits the header, the addressed responder will respond with 1-8 bytes of data plus checksum.

Figure 30-14. LIN Frame Format

Using the LIN command field (CTRLB.LINCMD), the complete header can be automatically transmitted, or software can control transmission of the various header components.

When CTRLB.LINCMD=0x1, software controls transmission of the LIN header. In this case, software uses the following sequence.

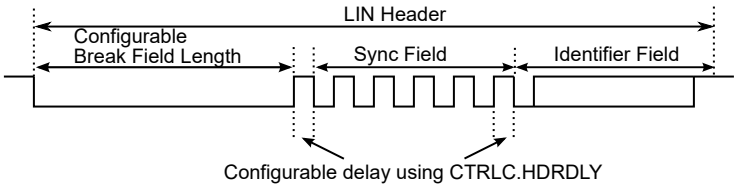
- CTRLB.LINCMD is written to 0x1.
- DATA register written to 0x00. This triggers transmission of the break field by hardware. Note that writing the DATA register with any other value will also result in the transmission of the break field by hardware.
- DATA register written to 0x55. The 0x55 value (sync) is transmitted.
- DATA register written to the identifier. The identifier is transmitted.

When CTRLB.LINCMD=0x2, hardware controls transmission of the LIN header. In this case, software uses the following sequence.

- CTRLB.LINCMD is written to 0x2.
- DATA register written to the identifier. This triggers transmission of the complete header by hardware. First the break field is transmitted. Next, the sync field is transmitted, and finally the identifier is transmitted.

In LIN commander mode, the length of the break field is programmable using the break length field (CTRLC.BRKLEN). When the LIN header command is used (CTRLB.LINCMD=0x2), the delay between the break and sync fields, in addition to the delay between the sync and ID fields are configurable using the header delay field (CTRLC.HDRDLY). When manual transmission is used (CTRLB.LINCMD=0x1), software controls the delay between break and sync.

Figure 30-15. LIN Header Generation



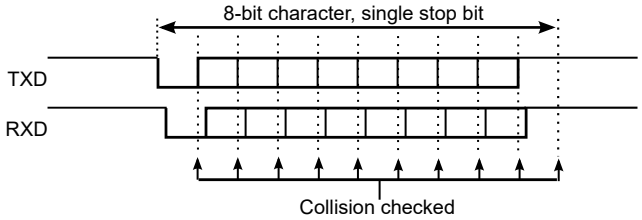
After header transmission is complete, the responder responds with 1-8 data bytes plus checksum.

30.6.3.6 Collision Detection

When the receiver and transmitter are connected either through pin configuration or externally, transmit collision can be detected after selecting the Collision Detection Enable bit in the CTRLB register (CTRLB.COLDEN=1). To detect collision, the receiver and transmitter must be enabled (CTRLB.RXEN=1 and CTRLB.TXEN=1).

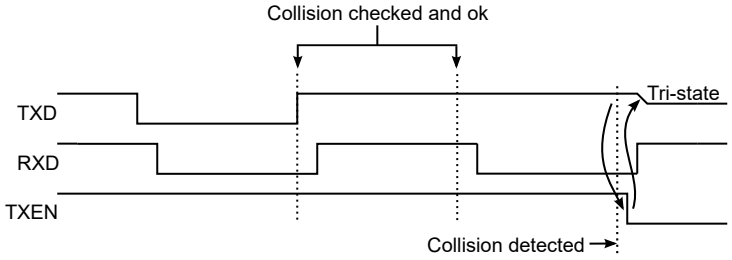
Collision detection is performed for each bit transmitted by comparing the received value with the transmit value, as shown in the figure below. While the transmitter is idle (no transmission in progress), characters can be received on RxD without triggering a collision.

Figure 30-16. Collision Checking



The next figure shows the conditions for a collision detection. In this case, the Start bit and the first Data bit are received with the same value as transmitted. The second received Data bit is found to be different than the transmitted bit at the detection point, which indicates a collision.

Figure 30-17. Collision Detected



When a collision is detected, the USART follows this sequence:

1. Abort the current transfer.
2. Flush the transmit buffer.
3. Disable transmitter (CTRLB.TXEN=0)
   - This is done after a synchronization delay. The CTRLB Synchronization Busy bit (SYNCBUSY.CTRLB) will be set until this is complete.
   - After disabling, the TxD pin will be tri-stated.
4. Set the Collision Detected bit (STATUS.COLL) along with the Error Interrupt Flag (INTFLAG.ERROR).

5. Set the Transmit Complete Interrupt Flag (INTFLAG.TXC), since the transmit buffer no longer contains data.

After a collision, software must manually enable the transmitter again before continuing, after assuring that the CTRLB Synchronization Busy bit (SYNCBUSY.CTRLB) is not set.

30.6.3.7 Loop-Back Mode

For Loop-Back mode, configure the Receive Data Pinout (CTRLA.RXPO) and Transmit Data Pinout (CTRLA.TXPO) to use the same data pins for transmit and receive. The loop-back is through the pad, so the signal is also available externally.

30.6.3.8 Start-of-Frame Detection

The USART start-of-frame detector can wake up the CPU when it detects a Start bit. In Standby Sleep mode, the internal fast start-up oscillator must be selected as the GCLK_SERCOMx_CORE source.

When a 1-to-0 transition is detected on RxD, the 8 MHz Internal Oscillator is powered up and the USART clock is enabled. After start-up, the rest of the data frame can be received, provided that the baud rate is slow enough in relation to the fast start-up internal oscillator start-up time. See *Electrical Characteristics* from Related Links for details. The start-up time of this oscillator varies with supply voltage and temperature.

The USART start-of-frame detection works both in Asynchronous and Synchronous modes. It is enabled by writing '1' to the Start of Frame Detection Enable bit in the Control B register (CTRLB.SFDE).

If the Receive Start Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.RXS) is set, the Receive Start interrupt is generated immediately when a start is detected.

When using start-of-frame detection without the Receive Start interrupt, start detection will force the 8 MHz internal oscillator and USART clock active while the frame is being received. In this case, the CPU will not wake up until the receive complete interrupt is generated.

Related Links

43. Electrical Characteristics

30.6.3.9 Sample Adjustment

In Asynchronous mode (CTRLA.CMODE=0), three samples in the middle are used to determine the value based on majority voting. The three samples used for voting can be selected using the Sample Adjustment bit field in Control A register (CTRLA.SAMPA). When CTRLA.SAMPA=0, samples 7-8-9 are used for 16x oversampling, and samples 3-4-5 are used for 8x oversampling.

30.6.4 DMA, Interrupts and Events

30.6.4.1 DMA Operation

The USART generates the following DMA requests:

30.6.4.2 Interrupts

The USART has the following interrupt sources. These are asynchronous interrupts, and can wake-up the device from any Sleep mode:

- Data Register Empty (DRE)
- Receive Complete (RXC)
- Transmit Complete (TXC)
- Receive Start (RXS)
- Clear to Send Input Change (CTSIC)
- Received Break (RXBRK)
- Error (ERROR)

Each interrupt source has its own Interrupt flag. The Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) will be set when the Interrupt condition is met. Each interrupt can be individually enabled by writing '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing '1' to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). The status of enabled interrupts can be read from either INTENSET or INTENCLR.

An interrupt request is generated when the Interrupt flag is set and if the corresponding interrupt is enabled. The interrupt request remains active until either the Interrupt flag is cleared, the interrupt is disabled, or the USART is reset. For details on clearing Interrupt flags, see *INTFLAG* from Related Links.

The value of INTFLAG indicates which interrupt is executed. Note that interrupts must be globally enabled for interrupt requests. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

30.8.8. INTFLAG

30.6.4.3 Events

Not applicable.

30.6.5 Sleep Mode Operation

The behavior in Sleep mode is depending on the clock source and the Run In Standby bit in the Control A register (CTRLA.RUNSTDBY):

- Internal clocking, CTRLA.RUNSTDBY=1: GCLK_SERCOMx_CORE can be enabled in all Sleep modes. Any interrupt can wake-up the device.
- External clocking, CTRLA.RUNSTDBY=1: The Receive Complete interrupt(s) can wake-up the device.
- Internal clocking, CTRLA.RUNSTDBY=0: Internal clock will be disabled, after any ongoing transfer was completed. The Receive Complete interrupt(s) can wake-up the device.
- External clocking, CTRLA.RUNSTDBY=0: External clock will be disconnected, after any ongoing transfer was completed. All reception will be dropped.

30.6.6 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset bit in the CTRLA register (CTRLA.SWRST)
- Enable bit in the CTRLA register (CTRLA.ENABLE)
- Receiver Enable bit in the CTRLB register (CTRLB.RXEN)
- Transmitter Enable bit in the Control B register (CTRLB.TXEN)

Note: CTRLB.RXEN is write-synchronized somewhat differently. See also 30.8.2. CTRLB for details.

Required write synchronization is denoted by the "Write-Synchronized" property in the register description.

30.7 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
		15:8	SAMPR[2:0]							IBON
		23:16	SAMPA[1:0]		RXPO[1:0]				TXPO[1:0]	
		31:24		DORD	CPOL	CMODE	FORM[3:0]			
0x04	CTRLB	7:0		SBMODE				CHSIZE[2:0]		
		15:8			PMODE			ENC		COLDEN
		23:16							RXEN	TXEN
		31:24								
0x08	CTRLC	7:0								
		15:8								
		23:16								
		31:24								
0x0C	BAUD	7:0	BAUD[7:0]							
		15:8	BAUD[15:8]							
0x0E	RXPL	7:0	RXPL[7:0]							
0x0F ... 0x13	Reserved									
0x14	INTENCLR	7:0	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
0x15	Reserved									
0x16	INTENSET	7:0	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
0x19	Reserved									
0x1A	STATUS	7:0		TXE	COLL	ISF	CTS	BUFOVF	FERR	PERR
		15:8								
0x1C	SYNCBUSY	7:0						CTRLB	ENABLE	SWRST
		15:8								
		23:16								
		31:24								
0x20 ... 0x27	Reserved									
0x28	DATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							
0x2C ... 0x2F	Reserved									
0x30	DBGCTRL	7:0								DBGSTOP

30.8 Register Description

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

Some registers require synchronization when read and/or written. Synchronization is denoted by the “Read-Synchronized” and/or “Write-Synchronized” property in each individual register description.

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the “Enable-Protected” property in each individual register description.

30.8.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
		DORD	CPOL	CMODE	FORM[3:0]			
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SAMPA[1:0]		RXPO[1:0]				TXPO[1:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	15	14	13	12	11	10	9	8
	SAMPR[2:0]							IBON
Access	R/W	R/W	R/W					R/W
Reset	0	0	0					0

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit 30 – DORD Data Order
This bit selects the data order when a character is shifted out from the Data register.
This bit is not synchronized.

Value	Description
0	MSB is transmitted first.
1	LSB is transmitted first.

Bit 29 – CPOL Clock Polarity
This bit selects the relationship between data output change and data input sampling in synchronous mode.
This bit is not synchronized.

CPOL	TxD Change	RxD Sample
0x0	Rising XCK edge	Falling XCK edge
0x1	Falling XCK edge	Rising XCK edge

Bit 28 – CMODE Communication Mode
This bit selects asynchronous or synchronous communication.
This bit is not synchronized.

Value	Description
0	Asynchronous communication.
1	Synchronous communication.

Bits 27:24 – FORM[3:0] Frame Format
These bits define the frame format.
These bits are not synchronized.

FORM[3:0]	Description
0x0	USART frame
0x1	USART frame with parity

.....continued	
FORM[3:0]	**Description**
0x4	Auto-baud (LIN Responder) - break detection and auto-baud.
0x5	Auto-baud - break detection and auto-baud with parity

Bits 23:22 – SAMPA[1:0] Sample Adjustment
These bits define the sample adjustment.
These bits are not synchronized.

SAMPA[1:0]	16x Over-sampling (CTRLA.SAMPR=0 or 1)	8x Over-sampling (CTRLA.SAMPR=2 or 3)
0x0	7-8-9	3-4-5
0x1	9-10-11	4-5-6
0x2	11-12-13	5-6-7
0x3	13-14-15	6-7-8

Bits 21:20 – RXPO[1:0] Receive Data Pinout
These bits define the receive data (RxD) pin configuration.
These bits are not synchronized.

RXPO[1:0]	Name	Description
0x0	PAD[0]	SERCOM PAD[0] is used for data reception
0x1	PAD[1]	SERCOM PAD[1] is used for data reception
0x2	PAD[2]	SERCOM PAD[2] is used for data reception
0x3	PAD[3]	SERCOM PAD[3] is used for data reception

Bits 17:16 – TXPO[1:0] Transmit Data Pinout
These bits define the transmit data (TxD) and XCK pin configurations.
This bit is not synchronized.

Bits 15:13 – SAMPR[2:0] Sample Rate
These bits select the sample rate.
These bits are not synchronized.

SAMPR[2:0]	Description
0x0	16x over-sampling using arithmetic baud rate generation.
0x1	16x over-sampling using fractional baud rate generation.
0x2	8x over-sampling using arithmetic baud rate generation.
0x3	8x over-sampling using fractional baud rate generation.
0x4	3x over-sampling using arithmetic baud rate generation.
0x5-0x7	Reserved

Bit 8 – IBON Immediate Buffer Overflow Notification
This bit controls when the buffer overflow status bit (STATUS.BUFOVF) is asserted when a buffer overflow occurs.
This bit is not synchronized.

Value	Description
0	STATUS.BUFOVF is asserted when it occurs in the data stream.
1	STATUS.BUFOVF is asserted immediately upon buffer overflow.

Bit 7 – RUNSTDBY Run In Standby
This bit defines the functionality in standby sleep mode.
This bit is not synchronized.

RUNSTDBY	External Clock	Internal Clock
0x0	External clock is disconnected when ongoing transfer is finished. All reception is dropped.	Generic clock is disabled when ongoing transfer is finished. The device will not wake up on Transfer Complete interrupt unless the appropriate ONDEMAND bits are set in the clocking chain.

.....continued		
RUNSTDBY	**External Clock**	**Internal Clock**
0x1	Wake on Receive Complete interrupt.	Generic clock is enabled in all sleep modes. Any interrupt can wake up the device.

Bits 4:2 – MODE[2:0] Operating Mode

These bits select the USART serial communication interface of the SERCOM.
These bits are not synchronized.

Value	Description
0x0	USART with external clock
0x1	USART with internal clock

Bit 1 – ENABLE Enable

Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the Enable Synchronization Busy bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE is cleared when the operation is complete.
This bit is not enable-protected.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled or being enabled.

Bit 0 – SWRST Software Reset

Writing '0' to this bit has no effect.
Writing '1' to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.
Writing '1' to CTRLA.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded. Any register write access during the ongoing reset will result in an APB error. Reading any register will return the reset value of the register.
Due to synchronization, there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the reset is complete.
This bit is not enable-protected.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

30.8.2 Control B

Name: CTRLB
Offset: 0x04
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
							RXEN	TXEN
Access							R/W	R/W
Reset							0	0

Bit	15	14	13	12	11	10	9	8
			PMODE			ENC		COLDEN
Access			R/W			R/W		R/W
Reset			0			0		0

Bit	7	6	5	4	3	2	1	0
		SBMODE				CHSIZE[2:0]		
Access		R/W				R/W	R/W	R/W
Reset		0				0	0	0

Bit 17 – RXEN Receiver Enable

Writing '0' to this bit will disable the USART receiver. Disabling the receiver will flush the receive buffer and clear the FERR, PERR and BUFOVF bits in the STATUS register.
Writing '1' to CTRLB.RXEN when the USART is disabled will set CTRLB.RXEN immediately. When the USART is enabled, CTRLB.RXEN will be cleared, and SYNCBUSY.CTRLB will be set and remain set until the receiver is enabled. When the receiver is enabled, CTRLB.RXEN will read back as '1'.
Writing '1' to CTRLB.RXEN when the USART is enabled will set SYNCBUSY.CTRLB, which will remain set until the receiver is enabled, and CTRLB.RXEN will read back as '1'.
This bit is not enable-protected.

Value	Description
0	The receiver is disabled or being enabled.
1	The receiver is enabled or will be enabled when the USART is enabled.

Bit 16 – TXEN Transmitter Enable

Writing '0' to this bit will disable the USART transmitter. Disabling the transmitter will not become effective until ongoing and pending transmissions are completed.
Writing '1' to CTRLB.TXEN when the USART is disabled will set CTRLB.TXEN immediately. When the USART is enabled, CTRLB.TXEN will be cleared, and SYNCBUSY.CTRLB will be set and remain set until the transmitter is enabled. When the transmitter is enabled, CTRLB.TXEN will read back as '1'.
Writing '1' to CTRLB.TXEN when the USART is enabled will set SYNCBUSY.CTRLB, which will remain set until the transmitter is enabled, and CTRLB.TXEN will read back as '1'.
This bit is not enable-protected.

Value	Description
0	The transmitter is disabled or being enabled.
1	The transmitter is enabled or will be enabled when the USART is enabled.

Bit 13 – PMODE Parity Mode

This bit selects the type of parity used when parity is enabled (CTRLA.FORM is '1'). The transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The receiver will generate a parity value for the incoming data and parity bit, compare it to the parity mode and, if a mismatch is detected, STATUS.PERR will be set.

This bit is not synchronized.

Value	Description
0	Even parity.
1	Odd parity.

Bit 10 – ENC Encoding Format

This bit selects the data encoding format.

This bit is not synchronized.

Value	Description
0	Data is not encoded.
1	Data is IrDA encoded.

Bit 8 – COLDEN Collision Detection Enable

This bit enables collision detection.

This bit is not synchronized.

Value	Description
0	Collision detection is not enabled.
1	Collision detection is enabled.

Bit 6 – SBMODE Stop Bit Mode

This bit selects the number of stop bits transmitted.

This bit is not synchronized.

Value	Description
0	One stop bit.
1	Two stop bits.

Bits 2:0 – CHSIZE[2:0] Character Size

These bits select the number of bits in a character.

These bits are not synchronized.

CHSIZE[2:0]	Description
0x0	8 bits
0x1	9 bits
0x2-0x4	Reserved
0x5	5 bits
0x6	6 bits
0x7	7 bits

30.8.3 Control C

Name: CTRLC
Offset: 0x08
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

30.8.4 Baud

Name: BAUD
Offset: 0x0C
Reset: 0x0000
Property: Enable-Protected, PAC Write-Protection

Bit	15	14	13	12	11	10	9	8
	BAUD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:0 – BAUD[15:0] Baud Value

Arithmetic Baud Rate Generation (`CTRLA.SAMPR[0]=0`):
These bits control the clock generation, as described in the SERCOM Baud Rate section.
If Fractional Baud Rate Generation (`CTRLA.SAMPR[0]=1 or =3`) bit positions 15 to 13 are replaced by FP[2:0] Fractional Part:

- **Bits 15:13 - FP[2:0]: Fractional Part**

  These bits control the clock generation, as described in the *SERCOM Clock Generation – Baud-Rate Generator* section.

- **Bits 12:0 - BAUD[12:0]: Baud Value**

  These bits control the clock generation, as described in the *SERCOM Clock Generation – Baud-Rate Generator* section.

30.8.5 Receive Pulse Length Register

Name: RXPL
Offset: 0x0E
Reset: 0x00
Property: Enable-Protected, PAC Write-Protection

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

Bits 7:0 – RXPL[7:0] Receive Pulse Length
When the encoding format is set to IrDA (CTRLB.ENC=1), these bits control the minimum pulse length that is required for a pulse to be accepted by the IrDA receiver with regards to the serial engine clock period SE_{per}.

$PULSE \geq \left(\mathrm{RXPL} + 1\right) \cdot SE_{per}$

30.8.6 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x14
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Bit	7	6	5	4	3	2	1	0
	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
Access	R/W		R/W	R/W		R/W	R/W	R/W
Reset	0		0	0		0	0	0

Bit 7 – ERROR Error Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

Bit 5 – RXBRK Receive Break Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Receive Break Interrupt Enable bit, which disables the Receive Break interrupt.

Value	Description
0	Receive Break interrupt is disabled.
1	Receive Break interrupt is enabled.

Bit 4 – CTSIC Clear to Send Input Change Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Clear To Send Input Change Interrupt Enable bit, which disables the Clear To Send Input Change interrupt.

Value	Description
0	Clear To Send Input Change interrupt is disabled.
1	Clear To Send Input Change interrupt is enabled.

Bit 2 – RXC Receive Complete Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Receive Complete Interrupt Enable bit, which disables the Receive Complete interrupt.

Value	Description
0	Receive Complete interrupt is disabled.
1	Receive Complete interrupt is enabled.

Bit 1 – TXC Transmit Complete Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Transmit Complete Interrupt Enable bit, which disables the Receive Complete interrupt.

Value	Description
0	Transmit Complete interrupt is disabled.
1	Transmit Complete interrupt is enabled.

Bit 0 – DRE Data Register Empty Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Data Register Empty Interrupt Enable bit, which disables the Data Register Empty interrupt.

Value	Description
0	Data Register Empty interrupt is disabled.
1	Data Register Empty interrupt is enabled.

30.8.7 Interrupt Enable Set

Name: INTENSET
Offset: 0x16
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
Access	R/W		R/W	R/W		R/W	R/W	R/W
Reset	0		0	0		0	0	0

Bit 7 – ERROR Error Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

Bit 5 – RXBRK Receive Break Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Receive Break Interrupt Enable bit, which enables the Receive Break interrupt.

Value	Description
0	Receive Break interrupt is disabled.
1	Receive Break interrupt is enabled.

Bit 4 – CTSIC Clear to Send Input Change Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Clear To Send Input Change Interrupt Enable bit, which enables the Clear To Send Input Change interrupt.

Value	Description
0	Clear To Send Input Change interrupt is disabled.
1	Clear To Send Input Change interrupt is enabled.

Bit 2 – RXC Receive Complete Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Receive Complete Interrupt Enable bit, which enables the Receive Complete interrupt.

Value	Description
0	Receive Complete interrupt is disabled.
1	Receive Complete interrupt is enabled.

Bit 1 – TXC Transmit Complete Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Transmit Complete Interrupt Enable bit, which enables the Transmit Complete interrupt.

Value	Description
0	Transmit Complete interrupt is disabled.
1	Transmit Complete interrupt is enabled.

Bit 0 – DRE Data Register Empty Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Data Register Empty Interrupt Enable bit, which enables the Data Register Empty interrupt.

Value	Description
0	Data Register Empty interrupt is disabled.
1	Data Register Empty interrupt is enabled.

30.8.8 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x18
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ERROR		RXBRK	CTSIC		RXC	TXC	DRE
Access	R/W		R/W	R/W		R	R/W	R
Reset	0		0	0		0	0	0

Bit 7 – ERROR Error
This flag is cleared by writing '1' to it.
This bit is set when any error is detected. Errors that will set this flag have corresponding status flags in the STATUS register. Errors that will set this flag are COLL, ISF, BUFOVF, FERR, and PERR.Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the flag.

Bit 5 – RXBRK Receive Break
This flag is cleared by writing '1' to it.
This flag is set when auto-baud is enabled (CTRLA.FORM) and a break character is received.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the flag.

Bit 4 – CTSIC Clear to Send Input Change
This flag is cleared by writing a '1' to it.
This flag is set when a change is detected on the CTS pin.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the flag.

Bit 2 – RXC Receive Complete
This flag is cleared by reading the Data register (DATA) or by disabling the receiver.
This flag is set when there are unread data in DATA.
Writing '0' to this bit has no effect.
Writing '1' to this bit has no effect.

Bit 1 – TXC Transmit Complete
This flag is cleared by writing '1' to it or by writing new data to DATA.
This flag is set when the entire frame in the Transmit Shift register has been shifted out and there are no new data in DATA.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the flag.

Bit 0 – DRE Data Register Empty
This flag is cleared by writing new data to DATA.
This flag is set when DATA is empty and ready to be written.
Writing '0' to this bit has no effect.
Writing '1' to this bit has no effect.

30.8.9 Status

Name: STATUS
Offset: 0x1A
Reset: 0x0000
Property: -

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		TXE	COLL	ISF	CTS	BUFOVF	FERR	PERR
Access		R/W	R/W	R/W	R	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 6 – TXE Transmitter Empty
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear it.

Bit 5 – COLL Collision Detected
This bit is cleared by writing '1' to the bit or by disabling the receiver.
This bit is set when collision detection is enabled (CTRLB.COLDEN) and a collision is detected.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear it.

Bit 4 – ISF Inconsistent Sync Field
This bit is cleared by writing '1' to the bit or by disabling the receiver.
This bit is set when the frame format is set to auto-baud (CTRLA.FORM) and a sync field not equal to 0x55 is received.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear it.

Bit 3 – CTS Clear to Send
This bit indicates the current level of the CTS pin when flow control is enabled (CTRLA.TXPO).
Writing '0' to this bit has no effect.
Writing '1' to this bit has no effect.

Bit 2 – BUFOVF Buffer Overflow
Reading this bit before reading the Data register will indicate the error status of the next character to be read.
This bit is cleared by writing '1' to the bit or by disabling the receiver.
This bit is set when a buffer overflow condition is detected. A buffer overflow occurs when the receive buffer is full, there is a new character waiting in the receive shift register and a new start bit is detected.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear it.

Bit 1 – FERR Frame Error
Reading this bit before reading the Data register will indicate the error status of the next character to be read.
This bit is cleared by writing '1' to the bit or by disabling the receiver.
This bit is set if the received character had a frame error, i.e., when the first stop bit is zero.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear it.

Bit 0 – PERR Parity Error

Reading this bit before reading the Data register will indicate the error status of the next character to be read.
This bit is cleared by writing '1' to the bit or by disabling the receiver.
This bit is set if parity checking is enabled (CTRLA.FORM is 0x1, 0x5) and a parity error is detected.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear it.

30.8.10 Synchronization Busy

Name: SYNCBUSY
Offset: 0x1C
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						CTRLB	ENABLE	SWRST
Access						R	R	R
Reset						0	0	0

Bit 2 – CTRLB CTRLB Synchronization Busy

Writing to the CTRLB register when the SERCOM is enabled requires synchronization. When writing to CTRLB the SYNCBUSY.CTRLB bit will be set until synchronization is complete. If CTRLB is written while SYNCBUSY.CTRLB is asserted, an APB error will be generated.

Value	Description
0	CTRLB synchronization is not busy.
1	CTRLB synchronization is busy.

Bit 1 – ENABLE SERCOM Enable Synchronization Busy

Enabling and disabling the SERCOM (CTRLA.ENABLE) requires synchronization. When written, the SYNCBUSY.ENABLE bit will be set until synchronization is complete.

Value	Description
0	Enable synchronization is not busy.
1	Enable synchronization is busy.

Bit 0 – SWRST Software Reset Synchronization Busy

Resetting the SERCOM (CTRLA.SWRST) requires synchronization. When written, the SYNCBUSY.SWRST bit will be set until synchronization is complete.

Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

30.8.11 Data

Name: DATA
Offset: 0x28
Reset: 0x0000
Property: -

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – DATA[31:0] Data

Reading these bits will return the contents of the Receive Data register. The register must be read only when the Receive Complete Interrupt Flag bit in the Interrupt Flag Status and Clear register (INTFLAG.RXC) is set. The status bits in STATUS must be read before reading the DATA value in order to get any corresponding error.
Writing these bits will write the Transmit Data register. This register must be written only when the Data Register Empty Interrupt Flag bit in the Interrupt Flag Status and Clear register (INTFLAG.DRE) is set.
Reads and writes are 32-bit or CTLB.CHSIZE based on the CTRLC.DATA32B setting.

30.8.12 Debug Control

Name: DBGCTRL
Offset: 0x30
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								DBGSTOP
Access								R/W
Reset								0

Bit 0 – DBGSTOP Debug Stop Mode

This bit controls the baud-rate generator functionality when the CPU is halted by an external debugger.

Value	Description
0	The baud-rate generator continues normal operation when the CPU is halted by an external debugger.
1	The baud-rate generator is halted when the CPU is halted by an external debugger.

31. SERCOM Serial Peripheral Interface (SERCOM SPI)

31.1 Overview

The Serial Peripheral Interface (SPI) is one of the available modes in the Serial Communication Interface (SERCOM).

The SPI uses the SERCOM transmitter and receiver configured as shown in the Block Diagram (see *Full-Duplex SPI Host Client Interconnection* in the *Block Diagram* from Related Links). Each side, host and client, depicts a separate SPI containing a Shift register, a transmit buffer and a two-level receive buffer. In addition, the SPI host uses the SERCOM baud-rate generator, while the SPI Client can use the SERCOM address match logic. Labels in capital letters are synchronous to PBx_CLK and accessible by the CPU, while labels in lowercase letters are synchronous to the SCK clock.

Note: Traditional Serial Peripheral Interface (SPI) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

Related Links

31.3. Block Diagram

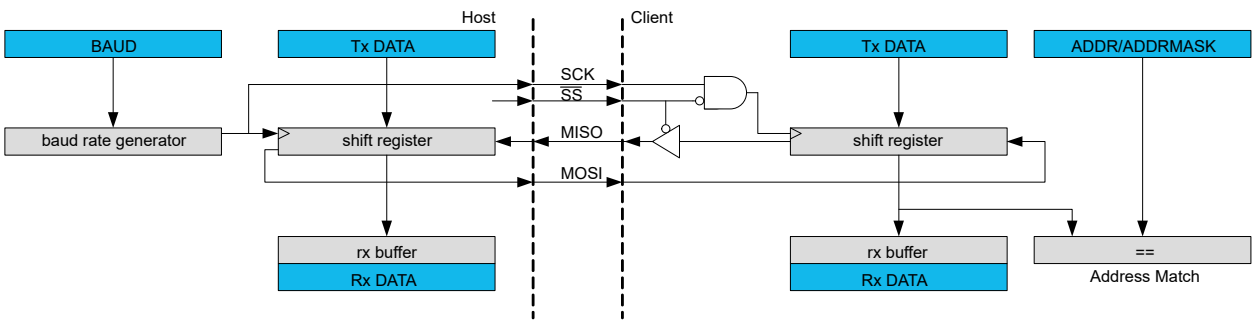
31.2 Features

SERCOM SPI includes the following features:

- Full-duplex, four-wire interface (MISO, MOSI, SCK, $\overline{SS}$)
- One-level transmit buffer, two-level receive buffer
- Supports all four SPI modes of operation
- Single data direction operation allows alternate function on MISO or MOSI pin
- Selectable LSB- or MSB-first data transfer
- Can be used with DMA
- Host operation:
  - Serial clock speed up to half the system clock
  - 8-bit clock generator
  - Hardware controlled $\overline{SS}$
- Client operation:
  - Serial clock speed up to half the system clock
  - Optional 8-bit address match operation
  - Operation in all sleep modes
  - Wake on $\overline{SS}$ transition

31.3 Block Diagram

Figure 31-1. Full-Duplex SPI Host Client Interconnection



31.4 Signal Description

Table 31-1. SERCOM SPI Signals

Signal Name	Type	Description
PAD[3:0]	Digital I/O	General SERCOM pins

One signal can be mapped to one of several pins.

31.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

31.5.1 I/O Lines

In order to use the SERCOM's I/O lines, the I/O pins must be configured using the System Configuration registers or PPS registers.

When the SERCOM is configured for SPI operation, the SERCOM controls the direction and value of the I/O pins according to the following table. Both PORT Control bits PINCFGn.PULLEN and PINCFGn.DRVSTR are still effective. If the receiver is disabled, the data input pin can be used for other purposes. In Host mode, the Client Select line ($\overline{SS}$) is hardware controlled when the Host Client Select Enable bit in the Control B register (CTRLB.MSSEN) is '1'.

Table 31-2. SPI Pin Configuration

Pin	Host SPI	Client SPI
MOSI	Output	Input
MISO	Input	Output
SCK	Output	Input

The combined configuration of PORT, the Data In Pinout and the Data Out Pinout bit groups in the Control A register (CTRLA.DIPO and CTRLA.DOPO) define the physical position of the SPI signals in the table above.

31.5.2 Power Management

This peripheral can continue to operate in any Sleep mode where its source clock is running. The interrupts can wake-up the device from Sleep modes.

31.5.3 Clocks

A generic clock (GCLK_SERCOMx_CORE) is required to clock the SPI. This clock must be configured and enabled in the Clock and Reset Unit (CRU) and Configuration (CFG.CFGPCLKGEN1) registers before using the SPI.

This generic clock is asynchronous to the bus clock (PBx_CLK). Therefore, writes to certain registers will require synchronization to the clock domains.

31.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). To use DMA requests with this peripheral, the DMAC must be configured first (see *Direct Memory Access Controller (DMAC)* from Related Links).

Related Links

22. Direct Memory Access Controller (DMAC)

31.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. In order to use interrupt requests of this peripheral, the Interrupt Controller (NVIC) must be configured first. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

31.5.6 Events

Not applicable.

31.5.7 Debug Operation

When the CPU is halted in Debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

Related Links

31.8.11. DBGCTRL

31.5.8 Register Access Protection

Registers with write access can be write-protected optionally by the Peripheral Access Controller (PAC).

PAC write protection is not available for the following registers:

- Interrupt Flag Clear and Status register (INTFLAG)
- Status register (STATUS)
- Data register (DATA)

Optional PAC write protection is denoted by the "PAC Write-Protection" property in each individual register description.

Write-protection does not apply to accesses through an external debugger.

31.5.9 Analog Connections

Not applicable.

31.6 Functional Description

31.6.1 Principle of Operation

The SPI is a high-speed synchronous data transfer interface. It allows high-speed communication between the device and peripheral devices.

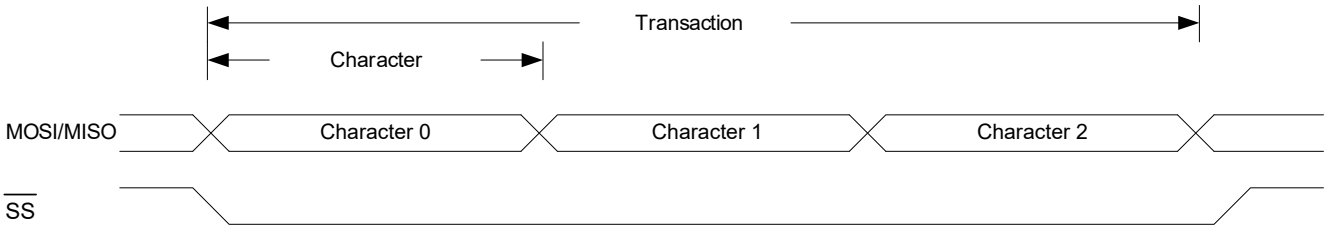
The SPI can operate as Host or Client. As Host, the SPI initiates and controls all data transactions. The SPI is single buffered for transmitting and double buffered for receiving.

When transmitting data, the Data register can be loaded with the next character to be transmitted during the current transmission.

When receiving, the data is transferred to the two-level receive buffer, and the receiver is ready for a new character.

The SPI transaction format is shown in SPI Transaction Format. Each transaction can contain one or more characters. The character size is configurable, and can be either 8 or 9 bits.

Figure 31-2. SPI Transaction Format



The SPI Host must pull the SPI select line ($\overline{SS}$) of the desired Client low to initiate a transaction. The Host and Client prepare data to send via their respective Shift registers, and the Host generates the serial clock on the SCK line.

Data are always shifted from Host to Client on the Host Output Client Input line (MOSI); data is shifted from Client to Host on the Host Input Client Output line (MISO).

Each time character is shifted out from the Host, a character will be shifted out from the Client simultaneously. To signal the end of a transaction, the Host will pull the $\overline{SS}$ line high

31.6.2 Basic Operation

31.6.2.1 Initialization

The following registers are enable-protected, meaning that they can only be written when the SPI is disabled (CTRL.ENABLE=0):

- Control A register (CTRLA), except Enable (CTRLA.ENABLE) and Software Reset (CTRLA.SWRST)
- Control B register (CTRLB), except Receiver Enable (CTRLB.RXEN)
- Baud register (BAUD)
- Address register (ADDR)

When the SPI is enabled or is being enabled (CTRLA.ENABLE=1), any writing to these registers will be discarded.

When the SPI is being disabled, writing to these registers will be completed after the disabling.

Enable-protection is denoted by the Enable-Protection property in the register description.

Initialize the SPI by following these steps:

1. Select SPI mode in host/client operation in the Operating Mode bit group in the CTRLA register (CTRLA.MODE= 0x2 or 0x3).
2. Select Transfer mode for the Clock Polarity bit and the Clock Phase bit in the CTRLA register (CTRLA.CPOL and CTRLA.CPHA) if desired.
3. Select the Frame Format value in the CTRLA register (CTRLA.FORM).
4. Configure the Data In Pinout field in the Control A register (CTRLA.DIPO) for SERCOM pads of the receiver.
5. Configure the Data Out Pinout bit group in the Control A register (CTRLA.DOPO) for SERCOM pads of the transmitter.
6. Select the Character Size value in the CTRLB register (CTRLB.CHSIZE).
7. Write the Data Order bit in the CTRLA register (CTRLA.DORD) for data direction.
8. If the SPI is used in Host mode:
    a. Select the desired baud rate by writing to the Baud register (BAUD).
    b. If Hardware $\overline{SS}$ control is required, write '1' to the Host SPI Select Enable bit in CTRLB register (CTRLB.MSSEN).
9. Enable the receiver by writing the Receiver Enable bit in the CTRLB register (CTRLB.RXEN=1).

31.6.2.2 Enabling, Disabling, and Resetting

This peripheral is enabled by writing '1' to the Enable bit in the Control A register (CTRLA.ENABLE), and disabled by writing '0' to it.

Writing ‘1’ to the Software Reset bit in the Control A register (CTRLA.SWRST) will reset all registers of this peripheral to their initial states, except the DBGCTRL register, and the peripheral is disabled.

Refer to the CTRLA register description for details.

Related Links

31.8.1. CTRLA

31.6.2.3 Clock Generation

In the SPI host operation (CTRLA.MODE = `0x3`), the serial clock (SCK) is generated internally by the SERCOM Baud Rate Generator (BRG).

In the SPI mode, the BRG is set to Synchronous mode. The 8-bit Baud register (BAUD) value is used for generating SCK and clocking the Shift register (see *Clock Generation – Baud-Rate Generator* from Related Links).

In the SPI client operation (CTRLA.MODE = `0x2`), the clock is provided by an external host on the SCK pin. This clock is used to clock the SPI Shift register.

Related Links

29.6.2.3. Clock Generation – Baud-Rate Generator

31.6.2.4 Data Register

The SPI Transmit Data register (TxDATA) and SPI Receive Data register (RxDATA) share the same I/O address, referred to as the SPI Data register (DATA). Writing DATA register will update the Transmit Data register. Reading the DATA register will return the contents of the Receive Data register.

31.6.2.5 SPI Transfer Modes

There are four combinations of SCK phase and polarity to transfer serial data. The SPI Data Transfer modes are shown in SPI Transfer Modes (Table) and SPI Transfer Modes (Figure).

SCK phase is configured by the Clock Phase bit in the CTRLA register (CTRLA.CPHA). SCK polarity is programmed by the Clock Polarity bit in the CTRLA register (CTRLA.CPOL). Data bits are shifted out and latched in on opposite edges of the SCK signal. This ensures sufficient time for the data signals to stabilize.

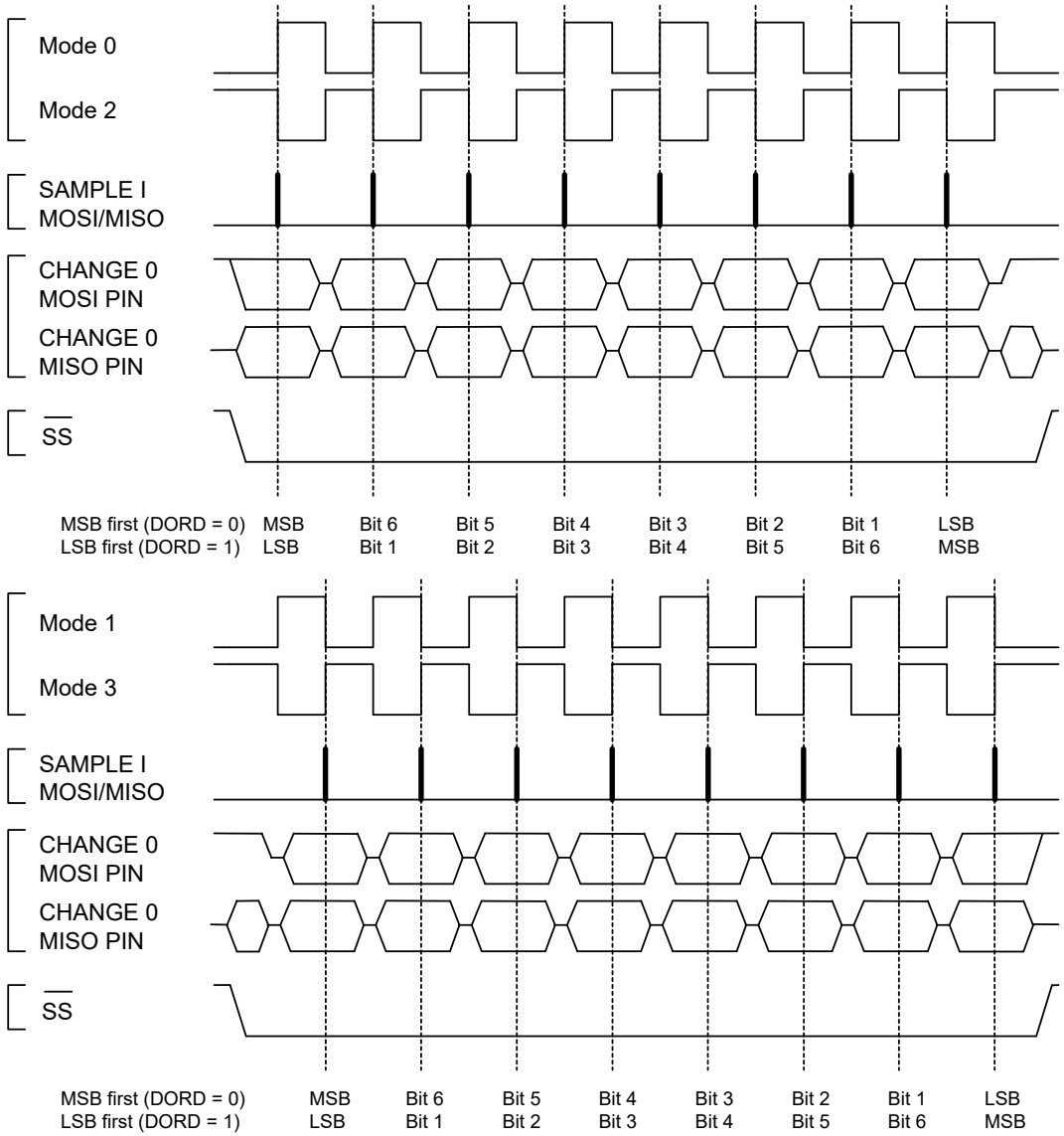
Table 31-3. SPI Transfer Modes

Mode	CPOL	CPHA	Leading Edge	Trailing Edge
0	0	0	Rising, sample	Falling, setup
1	0	1	Rising, setup	Falling, sample
2	1	0	Falling, sample	Rising, setup
3	1	1	Falling, setup	Rising, sample

Note:
Leading edge is the first clock edge in a clock cycle.

Trailing edge is the second clock edge in a clock cycle.

Figure 31-3. SPI Transfer Modes



31.6.2.6 Transferring Data

31.6.2.6.1 Host

In Host mode (CTRLA.MODE=0x3), when Host SPI Select Enable (CTRLB.MSSEN) is ‘1’, hardware will control the $\overline{SS}$ line.

When Host SPI Select Enable (CTRLB.MSSEN) is '0', the $\overline{SS}$ line must be configured as an output. $\overline{SS}$ can be assigned to any general purpose I/O pin. When the SPI is ready for a data transaction, software must pull the $\overline{SS}$ line low.

When writing a character to the Data register (DATA), the character will be transferred to the Shift register. Once the content of TxDATA has been transferred to the Shift register, the Data Register Empty flag in the Interrupt Flag Status and Clear register (INTFLAG.DRE) will be set. And a new character can be written to DATA.

Each time one character is shifted out from the Host, another character will be shifted in from the Client simultaneously. If the receiver is enabled (CTRLA.RXEN=1), the contents of the Shift register will be transferred to the two-level receive buffer. The transfer takes place in the same clock cycle as the last data bit is shifted in. And the Receive Complete Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.RXC) will be set. The received data can be retrieved by reading DATA.

When the last character has been transmitted and there is no valid data in DATA, the Transmit Complete Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.TXC) will be set. When the transaction is finished, the Host must pull the $\overline{SS}$ line high to notify the Client. If Host SPI Select Enable (CTRLB.MSSEN) is set to '0', the software must pull the $\overline{SS}$ line high.

31.6.2.6.2 Client

In Client mode (CTRLA.MODE = 0x2), the SPI interface will remain inactive with the MISO line tri-stated as long as the $\overline{SS}$ pin is pulled high. Software may update the contents of DATA at any time as long as the Data Register Empty flag in the Interrupt Status and Clear register (INTFLAG.DRE) is set.

When $\overline{SS}$ is pulled low and SCK is running, the client will sample and shift out data according to the Transaction mode set. Once the content of TxDATA is loaded into the Shift register, INTFLAG.DRE will be set and new data can be written to DATA.

Similar to the host, the client will receive one character for each character transmitted. A character will be transferred into the two-level receive buffer within the same clock cycle its last data bit is received. The received character can be retrieved from DATA when the Receive Complete interrupt flag (INTFLAG.RXC) is set.

When the host pulls the $\overline{SS}$ line high, the transaction is done and the Transmit Complete Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.TXC) will be set.

After DATA is written it takes up to three SCK clock cycles until the content of DATA is ready to be loaded into the Shift register on the next character boundary. As a consequence, the first character transferred in a SPI transaction will not be the content of DATA. This can be avoided by using the preloading feature (see *Preloading of the Client Shift Register* from Related Links).

When transmitting several characters in one SPI transaction, the data has to be written into DATA register with at least three SCK clock cycles left in the current character transmission. If this criteria is not met, the previously received character will be transmitted.

Once the DATA register is empty, it takes three CLK_SERCOM_APB cycles for INTFLAG.DRE to be set.

Related Links

31.6.3.2. Preloading of the Client Shift Register

31.6.2.7 Receiver Error Bit

The SPI receiver has one error bit: the Buffer Overflow bit (BUFOVF), which can be read from the Status register (STATUS). Once an error happens, the bit will stay set until it is cleared by writing '1' to it. The bit is also automatically cleared when the receiver is disabled.

There are two methods for buffer overflow notification, selected by the immediate Buffer Overflow Notification bit in the Control A register (CTRLA.IBON):

If CTRLA.IBON=1, STATUS.BUFOVF is raised immediately upon buffer overflow. Software can then empty the receive FIFO by reading RxDATA until the receiver complete Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.RXC) goes low.

If CTRLA.IBON=0, the Buffer Overflow condition travels with data through the receive FIFO. After the received data is read, STATUS.BUFOVF and INTFLAG.ERROR will be set along with INTFLAG.RXC, and RxDATA will be zero.

31.6.3 Additional Features

31.6.3.1 Address Recognition

When the SPI is configured for client operation (CTRLA.MODE=0x2) with address recognition (CTRLA.FORM is 0x2), the SERCOM address recognition logic is enabled: the first character in a transaction is checked for an address match.

If there is a match, the Receive Complete Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.RXC) is set, the MISO output is enabled, and the transaction is processed. If the device is in Sleep mode, an address match can wake-up the device in order to process the transaction.

If there is no match, the complete transaction is ignored.

If a 9-bit frame format is selected, only the lower 8 bits of the Shift register are checked against the Address register (ADDR).

Preload must be disabled (CTRLB.PLOADEN=0) in order to use this mode.

Related Links

29.6.3.1. Address Match and Mask

31.6.3.2 Preloading of the Client Shift Register

When starting a transaction, the client will first transmit the contents of the shift register before loading new data from DATA. The first character sent can be either the reset value of the shift register (if this is the first transmission since the last reset) or the last character in the previous transmission.

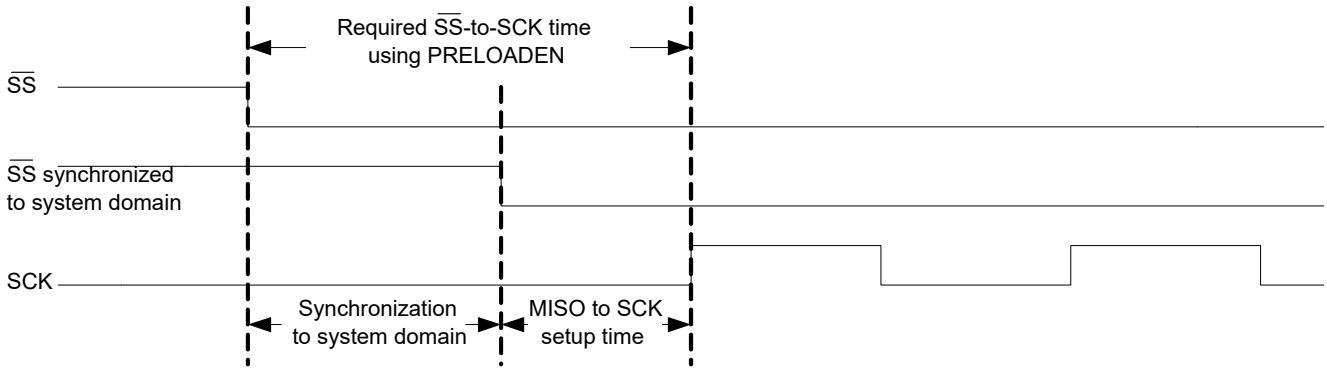
Preloading can be used to preload data into the shift register while $\overline{SS}$ is high: this eliminates sending a dummy character when starting a transaction. If the shift register is not preloaded, the current contents of the shift register will be shifted out.

Only one data character will be preloaded into the shift register while the synchronized $\overline{SS}$ signal is high. If the next character is written to DATA before $\overline{SS}$ is pulled low, the second character will be stored in DATA until transfer begins.

For proper preloading, sufficient time must elapse between $\overline{SS}$ going low and the first SCK sampling edge, as shown in the following figure. For timing details, see *Electrical Characteristics* from Related Links.

Preloading is enabled by writing '1' to the Client Data Preload Enable bit in the CTRLB register (CTRLB.PLOADEN).

Figure 31-4. Timing Using Preloading



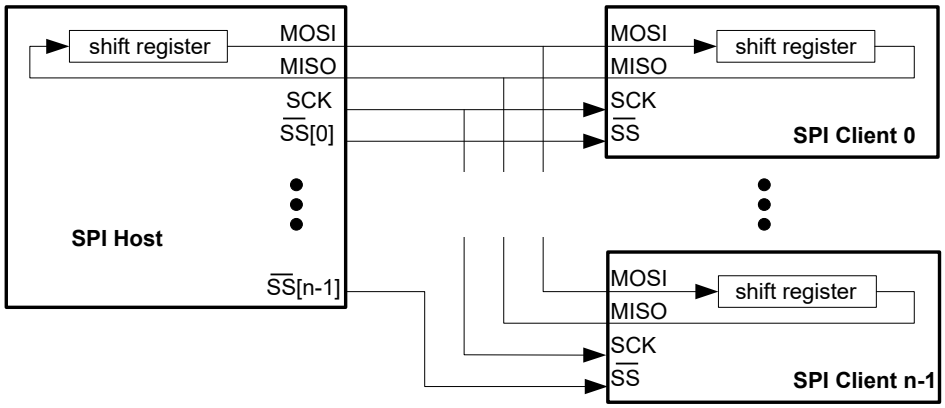
Related Links

43. Electrical Characteristics

31.6.3.3 Host with Several Clients

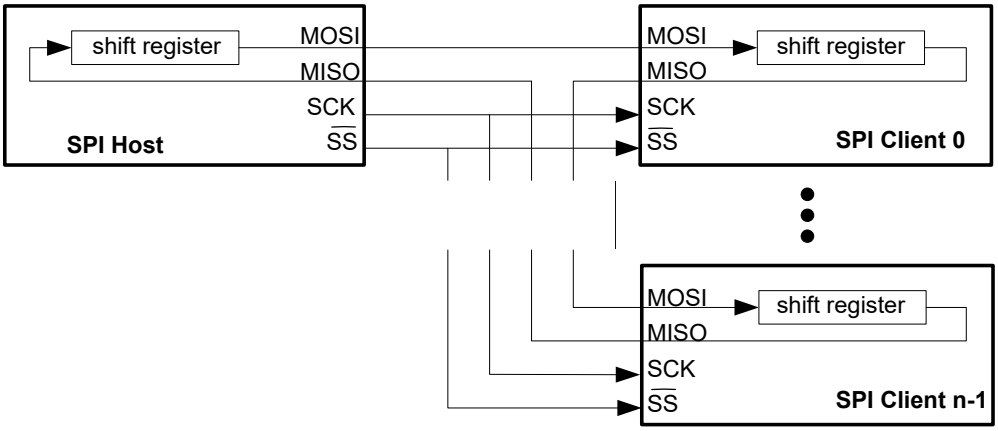
If the bus consists of several SPI clients, a SPI host can use general purpose I/O pins to control the $\overline{SS}$ line to each of the clients on the bus, as shown in the following figure. In this configuration, the single selected SPI client will drive the tri-state MISO line.

Figure 31-5. Multiple Clients in Parallel



Another configuration is multiple clients in series, as shown in the following figure. In this configuration, all n attached clients are connected in series. A common $\overline{SS}$ line is provided to all clients, enabling them simultaneously. The host must shift n characters for a complete transaction. The $\overline{SS}$ line is controlled by a normal GPIO.

Figure 31-6. Multiple Clients in Series



31.6.3.4 Loop-Back Mode

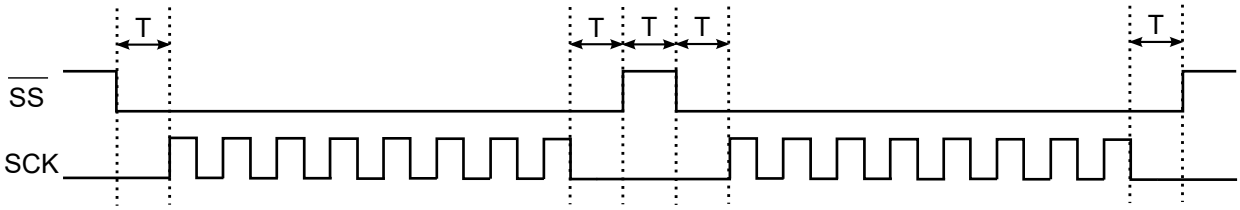
For Loop-back mode, configure the Data In Pinout (CTRLA.DIPO) and Data Out Pinout (CTRLA.DOPO) to use the same data pins for transmit and receive. The loop-back is through the pad, so the signal is also available externally.

31.6.3.5 Hardware Controlled $\overline{SS}$

In Host mode, a single $\overline{SS}$ chip select can be controlled by hardware by writing the Host SPI Select Enable (CTRLB.MSSEN) bit to '1'. In this mode, the $\overline{SS}$ pin is driven low for a minimum of one baud cycle before transmission begins, and stays low for a minimum of one baud cycle after transmission completes. If back-to-back frames are transmitted, the $\overline{SS}$ pin will always be driven high for a minimum of one baud cycle between frames.

In Hardware Controlled $\overline{SS}$, the time T is between one and two baud cycles depending on the SPI Transfer mode.

Figure 31-7. Hardware Controlled $\overline{SS}$



T = 1 to 2 baud cycles

When CTRLB.MSSEN=0, the $\overline{SS}$ pin(s) is/are controlled by user software and normal GPIO.

31.6.3.6 SPI Select Low Detection

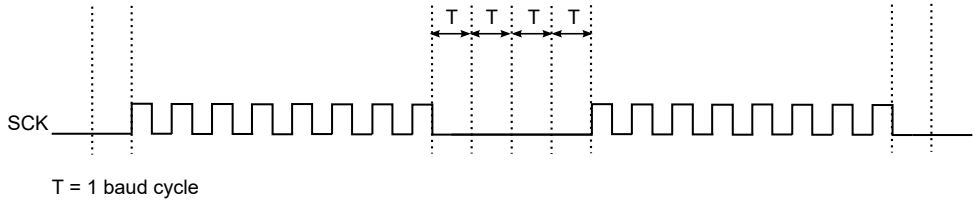
In Client mode, the SPI can wake the CPU when the SPI Select ($\overline{SS}$) goes low. When the SPI Select Low Detect is enabled (CTRLB.SSDE=1), a high-to-low transition will set the SPI Select Low Interrupt flag (INTFLAG.SSL) and the device will wake-up if applicable.

31.6.3.7 Host Inter-Character Spacing

When configured as host, inter-character spacing can be increased by writing a non-zero value to the Inter-Character Spacing bit field in the Control C register (CTRLC.ICSPACE). When non-zero, CTRLC.ICSPACE represents the minimum number of baud cycles that the SCK clock line does not toggle and the next character is stalled.

The figure gives an example for CTRLC.ICSPACE=4; In this case, the SCK is inactive for 4 baud cycles.

Figure 31-8. Four Cycle Inter-Character Spacing Example



31.6.3.8 32-bit Extension

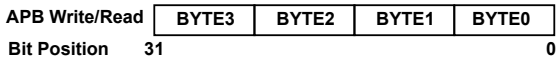
For better system bus utilization, 32-bit data receive and transmit can be enabled by writing to the Data 32-bit bit field in the Control C register (CTRLC.DATA32B=1). When enabled, write and read transaction to/from the DATA register are 32 bit in size.

If frames are not multiples of 4 Bytes, the Length Counter (LENGTH.LEN) and Length Enable (LENGTH.LENEN) must be configured before data transfer begins. LENGTH.LEN must be enabled only when CTRLC.DATA32B is enabled.

The following figure shows the order of transmit and receive when using 32-bit mode. Bytes are transmitted or received and stored in order from 0 to 3.

Only 8-bit character size is supported.

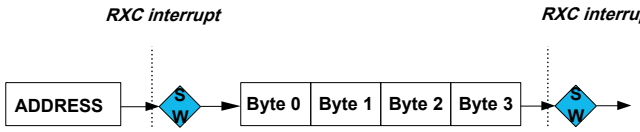
Figure 31-9. 32-bit Extension Byte Ordering



32-bit Extension Client Operation

The following figure shows a transaction with 32-bit Extension enabled (CTRLC.DATA32B=1). When address recognition is enabled (CTRLA.FORM=0x2) and there is an address match, the address is loaded into the FIFO as Byte zero and data begins with Byte 1. INTFLAGS.RXC will then be raised for every 4 Bytes transferred. For transmit, there is a 32-bit holding buffer in the core domain. Once DATA has been registered in the core domain, INTFLAG.DRE will be raised, so that the next 32 bits can be written to the DATA register.

Figure 31-10. 32-bit Extension Client Operation



When utilizing the length counter, the LENGTH register must be written before the frame begins. If the frame length while $\overline{SS}$ is low is not a multiple of LENGTH.LEN Bytes, the Length Error Status bit (STATUS.LENERR) is raised. If LENGTH.LEN is not a multiple of 4 Bytes, the final INTFLAG.RXC interrupt will be raised when the last Byte is received.

The length count is based on the received Bytes, or the number of clocks if the receiver is not enabled. If pre-loading is disabled and DATA is written to for transmit before SCK starts, transmitted data will be delayed by one Byte, but the length counter will still increment for the first (empty) Byte transmission. When the counter reaches LENGTH.LEN, the internal length counter, Rx Byte counter, and Tx Byte counter are reset. If multiple lengths are to be transmitted, INTFLAG.TXC must go high before writing DATA for subsequent lengths.

If there is a Length Error (STATUS.LENERR), the remaining Bytes in the length will be transmitted at the beginning of the next frame. If this is not desired, the SERCOM must be disabled and re-enabled in order to flush the Tx and Rx pipelines.

Writing the LENGTH register while a frame is in progress will produce unpredictable results. If LENGTH.LENEN is not configured and a frame is not a multiple of 4 Bytes (while $\overline{SS}$ is low), the remainder will be transmitted in the next frame.

32-bit Extension Host Operation

When using the SPI configured as Host, the Length and the Length Enable bit fields (LENGTH.LEN and LENGTH.LENEN) must be written before the frame begins. When LENGTH.LENEN is written to '1', the value of LENGTH.LEN determines the number of data bytes in the transaction from 1 to 255.

For receive data, INTFLAG.RXC is raised every 4 Bytes received. If LENGTH.LEN is not a multiple of 4 Bytes, the final INTFLAG.RXC is set when the final byte is received.

For transmit, there is a holding buffer for the 32-bit data in the core domain. Once DATA has been registered in the core domain, INTFLAG.DRE will be raised so that the next 32 bits can be written to the DATA register.

If multiple lengths are to be transmitted, INTFLAG.TXC must go high before writing DATA for subsequent lengths.

31.6.4 DMA, Interrupts, and Events

Table 31-4. Module Request for SERCOM SPI

Condition	Request		
	DMA	Interrupt	Event
Data Register Empty (DRE)	Yes (request cleared when data is written)	Yes	NA
Receive Complete (RXC)	Yes (request cleared when data is read)	Yes	
Transmit Complete (TXC)	NA	Yes	
Client Select low (SSL)	NA	Yes	
Error (ERROR)	NA	Yes	

31.6.4.1 DMA Operation

The SPI generates the following DMA requests:

31.6.4.2 Interrupts

The SPI has the following interrupt sources. These are asynchronous interrupts, and can wake-up the device from any Sleep mode:

- Data Register Empty (DRE)
- Receive Complete (RXC)
- Transmit Complete (TXC)
- SPI Select Low (SSL)
- Error (ERROR)

Each interrupt source has its own Interrupt flag. The Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) will be set when the Interrupt condition is met. Each interrupt can be individually enabled by writing '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing '1' to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). The status of enabled interrupts can be read from either INTENSET or INTENCLR.

An interrupt request is generated when the Interrupt flag is set and if the corresponding interrupt is enabled. The interrupt request remains active until either the Interrupt flag is cleared, the interrupt is disabled, or the SPI is reset. For details on clearing Interrupt flags, see INTFLAG register description.

The value of INTFLAG indicates which interrupt is executed. Note that interrupts must be globally enabled for interrupt requests. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

31.6.4.3 Events

Not applicable.

31.6.5 Sleep Mode Operation

The behavior in sleep mode is depending on the Host/Client configuration and the Run In Standby bit in the Control A register (CTRLA.RUNSTDBY):

- **Host operation, CTRLA.RUNSTDBY=1:** The peripheral clock GCLK_SERCOM_CORE will continue to run in idle sleep mode and in standby sleep mode. Any interrupt can wake up the device.
- **Host operation, CTRLA.RUNSTDBY=0:** GLK_SERCOMx_CORE will be disabled after the ongoing transaction is finished. Any interrupt can wake up the device.
- **Client operation, CTRLA.RUNSTDBY=1:** The Receive Complete interrupt can wake up the device
- **Client operation, CTRLA.RUNSTDBY=0:** All reception will be dropped, including the ongoing transaction

31.6.6 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset bit in the CTRLA register (CTRLA.SWRST)
- Enable bit in the CTRLA register (CTRLA.ENABLE)
- Receiver Enable bit in the CTRLB register (CTRLB.RXEN)

Note: CTRLB.RXEN is write-synchronized somewhat differently. See *CTRLB* register from Related Links.

Required write synchronization is denoted by the "Write-Synchronized" property in the register description.

Related Links

31.8.2. CTRLB

31.7 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
		15:8								IBON
		23:16			DIPO[1:0]				DOPO[1:0]	
		31:24		DORD	CPOL	CPHA	FORM[3:0]			
0x04	CTRLB	7:0		PLOADEN				CHSIZE[2:0]		
		15:8	AMODE[1:0]		MSSEN				SSDE	
		23:16							RXEN	
		31:24								
0x08 ... 0x0B	Reserved									
0x0C	BAUD	7:0	BAUD[7:0]							
0x0D ... 0x13	Reserved									
0x14	INTENCLR	7:0	ERROR				SSL	RXC	TXC	DRE
0x15	Reserved									
0x16	INTENSET	7:0	ERROR				SSL	RXC	TXC	DRE
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR				SSL	RXC	TXC	DRE
0x19	Reserved									
0x1A	STATUS	7:0						BUFOVF		
		15:8								
0x1C	SYNCBUSY	7:0						CTRLB	ENABLE	SWRST
		15:8								
		23:16								
		31:24								
0x20 ... 0x23	Reserved									
0x24	ADDR	7:0	ADDR[7:0]							
		15:8								
		23:16	ADDRMASK[7:0]							
		31:24								
0x28	DATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							
0x2C ... 0x2F	Reserved									
0x30	DBGCTRL	7:0								DBGSTOP

31.8 Register Description

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

Some registers require synchronization when read and/or written. Synchronization is denoted by the “Read-Synchronized” and/or “Write-Synchronized” property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the “Enable-Protected” property in each individual register description.

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

See *Peripheral Access Controller (PAC)* from Related Links.

Related Links

26. Peripheral Access Controller (PAC)

31.8.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
		DORD	CPOL	CPHA	FORM[3:0]			
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
			DIPO[1:0]				DOPO[1:0]	
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bit	15	14	13	12	11	10	9	8
								IBON
Access								R/W
Reset								0

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit 30 – DORD Data Order
This bit selects the data order when a character is shifted out from the shift register.
This bit is not synchronized.

Value	Description
0	MSB is transferred first.
1	LSB is transferred first.

Bit 29 – CPOL Clock Polarity
In combination with the Clock Phase bit (CPHA), this bit determines the SPI transfer mode.
This bit is not synchronized.

Value	Description
0	SCK is low when idle. The leading edge of a clock cycle is a rising edge, while the trailing edge is a falling edge.
1	SCK is high when idle. The leading edge of a clock cycle is a falling edge, while the trailing edge is a rising edge.

Bit 28 – CPHA Clock Phase
In combination with the Clock Polarity bit (CPOL), this bit determines the SPI transfer mode.
This bit is not synchronized.

Mode	CPOL	CPHA	Leading Edge	Trailing Edge
0x0	0	0	Rising, sample	Falling, change
0x1	0	1	Rising, change	Falling, sample
0x2	1	0	Falling, sample	Rising, change
0x3	1	1	Falling, change	Rising, sample

Value	Description
0	The data is sampled on a leading SCK edge and changed on a trailing SCK edge.
1	The data is sampled on a trailing SCK edge and changed on a leading SCK edge.

Bits 27:24 – FORM[3:0] Frame Format

This bit field selects the various frame formats supported by the SPI in client mode. When the 'SPI frame with address' format is selected, the first byte received is checked against the ADDR register.

FORM[3:0]	Name	Description
0x0	SPI	SPI frame
0x1	—	Reserved
0x2	SPI_ADDR	SPI frame with address
0x3-0xF	—	Reserved

Bits 21:20 – DIPO[1:0] Data In Pinout

These bits define the data in (DI) pad configurations.
In host operation, DI is MISO.
In client operation, DI is MOSI.
These bits are not synchronized.

DIPO[1:0]	Name	Description
0x0	PAD[0]	SERCOM PAD[0] is used as data input
0x1	PAD[1]	SERCOM PAD[1] is used as data input
0x2	PAD[2]	SERCOM PAD[2] is used as data input
0x3	PAD[3]	SERCOM PAD[3] is used as data input

Bits 17:16 – DOPO[1:0] Data Out Pinout

This bit defines the available pad configurations for data out (DO), the serial clock (SCK) and the SPI Select ($\overline{SS}$). In Client operation, the SPI Select line ($\overline{SS}$) is controlled by DOPO. In host operation, the SPI Select line ($\overline{SS}$) is either controlled by DOPO when CTRLB.MSSEN=1, or by a GPIO driven by the application when CTRLB.MSSEN=0.
In host operation, DO is MOSI.
In client operation, DO is MISO.
These bits are not synchronized.

DOPO	DO	SCK	Client $\overline{SS}$	Host $\overline{SS}$ (MSSEN = 1)	Host $\overline{SS}$ (MSSEN = 0)
0x0	PAD[0]	PAD[1]	PAD[2]	PAD[2]	Any GPIO configured by the application
0x2	PAD[3]	PAD[1]	PAD[2]	PAD[2]	Any GPIO configured by the application

Bit 8 – IBON Immediate Buffer Overflow Notification

This bit controls when the Buffer Overflow Status bit (STATUS.BUFOVF) is set when a buffer overflow occurs.
This bit is not synchronized.

Value	Description
0	STATUS.BUFOVF is set when it occurs in the data stream.
1	STATUS.BUFOVF is set immediately upon buffer overflow.

Bit 7 – RUNSTDBY Run In Standby

This bit defines the functionality in Standby mode.
These bits are not synchronized.

RUNSTDBY	Client	Host
0x0	Disabled. All reception is dropped, including the ongoing transaction.	Generic clock is disabled when ongoing transaction is finished. All interrupts can wake up the device.
0x1	Ongoing transaction continues, wake on Receive Complete interrupt.	Generic clock is enabled while in sleep modes. All interrupts can wake up the device.

Bits 4:2 – MODE[2:0] Operating Mode

These bits must be written to 0x2 or 0x3 to select the SPI serial communication interface of the SERCOM.
0x2: SPI client operation
0x3: SPI host operation
These bits are not synchronized.

Bit 1 – ENABLE Enable

Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled.
The value written to CTRL.ENABLE will read back immediately and the Synchronization Enable Busy bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE is cleared when the operation is complete.
This bit is not enable-protected.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled or being enabled.

Bit 0 – SWRST Software Reset

Writing '0' to this bit has no effect.
Writing '1' to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.
Writing '1' to CTRL.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded. Any register write access during the ongoing reset will result in an APB error. Reading any register will return the reset value of the register.
Due to synchronization, there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY. SWRST will both be cleared when the reset is complete.
This bit is not enable-protected.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

31.8.2 Control B

Name: CTRLB
Offset: 0x04
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
							RXEN	
Access							R/W	
Reset							0	

Bit	15	14	13	12	11	10	9	8
	AMODE[1:0]		MSSEN				SSDE	
Access	R/W	R/W	R/W				R/W	
Reset	0	0	0				0	

Bit	7	6	5	4	3	2	1	0
		PLOADEN				CHSIZE[2:0]		
Access		R/W				R/W	R/W	R/W
Reset		0				0	0	0

Bit 17 – RXEN Receiver Enable

Writing '0' to this bit will disable the SPI receiver immediately. The receive buffer will be flushed, data from ongoing receptions will be lost and STATUS.BUFOVF will be cleared.
Writing '1' to CTRLB.RXEN when the SPI is disabled will set CTRLB.RXEN immediately. When the SPI is enabled, CTRLB.RXEN will be cleared, SYNCBUSY.CTRLB will be set and remain set until the receiver is enabled. When the receiver is enabled CTRLB.RXEN will read back as '1'.
Writing '1' to CTRLB.RXEN when the SPI is enabled will set SYNCBUSY.CTRLB, which will remain set until the receiver is enabled, and CTRLB.RXEN will read back as '1'.
This bit is not enable-protected.

Value	Description
0	The receiver is disabled.
1	The receiver is enabled or it will be enabled when SPI is enabled.

Bits 15:14 – AMODE[1:0] Address Mode

These bits set the Client Addressing mode when the frame format (CTRLA.FORM) with address is used. They are unused in Host mode.
These bits are not synchronized.

AMODE[1:0]	Name	Description
0x0	MASK	ADDRMASK is used as a mask to the ADDR register
0x1	2_ADDRS	The client responds to the two unique addresses in ADDR and ADDRMASK
0x2	RANGE	The client responds to the range of addresses between and including ADDR and ADDRMASK. ADDR is the upper limit
0x3	—	Reserved

Bit 13 – MSSEN Host SPI Select Enable

This bit enables hardware SPI Select ($\overline{SS}$) control.
This bit is not synchronized.

Value	Description
0	Hardware $\overline{SS}$ control is disabled.
1	Hardware $\overline{SS}$ control is enabled.

Bit 9 – SSDE SPI Select Low Detect Enable

This bit enables wake-up when the SPI Select ($\overline{SS}$) pin transitions from high-to-low.
This bit is not synchronized.

Value	Description
0	$\overline{SS}$ low detector is disabled.
1	$\overline{SS}$ low detector is enabled.

Bit 6 – PLOADEN Client Data Preload Enable

Setting this bit will enable preloading of the Client Shift register when there is no transfer in progress. If the $\overline{SS}$ line is high when DATA is written, it will be transferred immediately to the Shift register.
This bit is not synchronized.

Bits 2:0 – CHSIZE[2:0] Character Size

These bits are not synchronized.

CHSIZE[2:0]	Name	Description
0x0	8BIT	8 bits
0x1	9BIT	9 bits
0x2-0x7	—	Reserved

31.8.3 Baud Rate

Name: BAUD
Offset: 0x0C
Reset: 0x00
Property: PAC Write-Protection, Enable-Protected

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

Bits 7:0 – BAUD[7:0] Baud Register
These bits control the clock generation, as described in the *SERCOM Clock Generation – Baud-Rate Generator*.

31.8.4 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x14
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Bit	7	6	5	4	3	2	1	0
	ERROR				SSL	RXC	TXC	DRE
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0

Bit 7 – ERROR Error Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

Bit 3 – SSL Client Select Low Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Client Select Low Interrupt Enable bit, which disables the Client Select Low interrupt.

Value	Description
0	Client Select Low interrupt is disabled.
1	Client Select Low interrupt is enabled.

Bit 2 – RXC Receive Complete Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Receive Complete Interrupt Enable bit, which disables the Receive Complete interrupt.

Value	Description
0	Receive Complete interrupt is disabled.
1	Receive Complete interrupt is enabled.

Bit 1 – TXC Transmit Complete Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Transmit Complete Interrupt Enable bit, which disable the Transmit Complete interrupt.

Value	Description
0	Transmit Complete interrupt is disabled.
1	Transmit Complete interrupt is enabled.

Bit 0 – DRE Data Register Empty Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Data Register Empty Interrupt Enable bit, which disables the Data Register Empty interrupt.

Value	Description
0	Data Register Empty interrupt is disabled.
1	Data Register Empty interrupt is enabled.

31.8.5 Interrupt Enable Set

Name: INTENSET
Offset: 0x16
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
	ERROR				SSL	RXC	TXC	DRE
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0

Bit 7 – ERROR Error Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

Bit 3 – SSL Client Select Low Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Client Select Low Interrupt Enable bit, which enables the Client Select Low interrupt.

Value	Description
0	Client Select Low interrupt is disabled.
1	Client Select Low interrupt is enabled.

Bit 2 – RXC Receive Complete Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Receive Complete Interrupt Enable bit, which enables the Receive Complete interrupt.

Value	Description
0	Receive Complete interrupt is disabled.
1	Receive Complete interrupt is enabled.

Bit 1 – TXC Transmit Complete Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Transmit Complete Interrupt Enable bit, which enables the Transmit Complete interrupt.

Value	Description
0	Transmit Complete interrupt is disabled.
1	Transmit Complete interrupt is enabled.

Bit 0 – DRE Data Register Empty Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Data Register Empty Interrupt Enable bit, which enables the Data Register Empty interrupt.

Value	Description
0	Data Register Empty interrupt is disabled.
1	Data Register Empty interrupt is enabled.

31.8.6 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x18
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ERROR				SSL	RXC	TXC	DRE
Access	R/W				R/W	R	R/W	R
Reset	0				0	0	0	0

Bit 7 – ERROR Error
This flag is cleared by writing '1' to it.
This bit is set when any error is detected. Errors that will set this flag have corresponding Status flags in the STATUS register. The BUFOVF error will set this Interrupt flag.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the flag.

Bit 3 – SSL SPI Select Low
This flag is cleared by writing '1' to it.
This bit is set when a high to low transition is detected on the $\overline{SS}$ pin in Client mode and SPI Select Low Detect (CTRLB.SSDE) is enabled.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the flag.

Bit 2 – RXC Receive Complete
This flag is cleared by reading the Data (DATA) register or by disabling the receiver.
This flag is set when there are unread data in the receive buffer. If address matching is enabled, the first data received in a transaction will be an address.
Writing '0' to this bit has no effect.
Writing '1' to this bit has no effect.

Bit 1 – TXC Transmit Complete
This flag is cleared by writing '1' to it or by writing new data to DATA.
In Host mode, this flag is set when the data have been shifted out and there are no new data in DATA.
In Client mode, this flag is set when the $\overline{SS}$ pin is pulled high. If address matching is enabled, this flag is only set if the transaction was initiated with an address match.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the flag.

Bit 0 – DRE Data Register Empty
This flag is cleared by writing new data to DATA.
This flag is set when DATA is empty and ready for new data to transmit.
Writing '0' to this bit has no effect.
Writing '1' to this bit has no effect.

31.8.7 Status

Name: STATUS
Offset: 0x1A
Reset: 0x0000
Property: –

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						BUFOVF		
Access						R/W		
Reset						0		

Bit 2 – BUFOVF Buffer Overflow

Reading this bit before reading DATA will indicate the error status of the next character to be read.
This bit is cleared by writing '1' to the bit or by disabling the receiver.
This bit is set when a Buffer Overflow condition is detected. See *CTRLA* from Related Links for overflow handling.
When set, the corresponding RxDATA will be zero.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear it.

Value	Description
0	No Buffer Overflow has occurred.
1	A Buffer Overflow has occurred.

Related Links

31.8.1. CTRLA

31.8.8 Synchronization Busy

Name: SYNCBUSY
Offset: 0x1C
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						CTRLB	ENABLE	SWRST
Access						R	R	R
Reset						0	0	0

Bit 2 – CTRLB CTRLB Synchronization Busy

Writing to the CTRLB when the SERCOM is enabled requires synchronization. Ongoing synchronization is indicated by SYNCBUSY.CTRLB=1 until synchronization is complete. If CTRLB is written while SYNCBUSY.CTRLB=1, an APB error will be generated.

Value	Description
0	CTRLB synchronization is not busy.
1	CTRLB synchronization is busy.

Bit 1 – ENABLE SERCOM Enable Synchronization Busy

Enabling and disabling the SERCOM (CTRLA.ENABLE) requires synchronization. Ongoing synchronization is indicated by SYNCBUSY.ENABLE=1 until synchronization is complete.

Value	Description
0	Enable synchronization is not busy.
1	Enable synchronization is busy.

Bit 0 – SWRST Software Reset Synchronization Busy

Resetting the SERCOM (CTRLA.SWRST) requires synchronization. Ongoing synchronization is indicated by SYNCBUSY.SWRST=1 until synchronization is complete.

Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

31.8.9 Address

Name: ADDR
Offset: 0x24
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	ADDRMASK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

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

Bits 23:16 – ADDRMASK[7:0] Address Mask
These bits hold the address mask when the transaction format with address is used (CTRLA.FORM, CTRLB.AMODE).

Bits 7:0 – ADDR[7:0] Address
These bits hold the address when the transaction format with address is used (CTRLA.FORM, CTRLB.AMODE).

31.8.10 Data

Name: DATA
Offset: 0x28
Reset: 0x0000
Property: –

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – DATA[31:0] Data

Reading these bits will return the contents of the receive data buffer. The register must be read only when the Receive Complete Interrupt Flag bit in the Interrupt Flag Status and Clear register (INTFLAG.RXC) is set.
Writing these bits will write the transmit data buffer. This register must be written only when the Data Register Empty Interrupt Flag bit in the Interrupt Flag Status and Clear register (INTFLAG.DRE) is set.
Reads and writes are 32-bit or CTLB.CHSIZE based on the CTRLC.DATA32B setting.

31.8.11 Debug Control

Name: DBGCTRL
Offset: 0x30
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								DBGSTOP
Access								R/W
Reset								0

Bit 0 – DBGSTOP Debug Stop Mode
This bit controls the functionality when the CPU is halted by an external debugger.

Value	Description
0	The baud-rate generator continues normal operation when the CPU is halted by an external debugger.
1	The baud-rate generator is halted when the CPU is halted by an external debugger.

32. SERCOM Inter-Integrated Circuit (SERCOM I²C)

32.1 Overview

The Inter-Integrated Circuit (I²C) interface is one of the available modes in the serial communication interface (SERCOM).

The I²C interface uses the SERCOM transmitter and receiver configured as shown in Figure 32-1. Labels in capital letters are registers accessible by the CPU, while lowercase labels are internal to the SERCOM.

A SERCOM instance can be configured to be either an I²C host or an I²C client. Both host and client have an interface containing a shift register, a transmit buffer and a receive buffer. In addition, the I²C host uses the SERCOM baud-rate generator, while the I²C client uses the SERCOM address match logic.

Note: Traditional Inter-Integrated Circuit (I²C) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

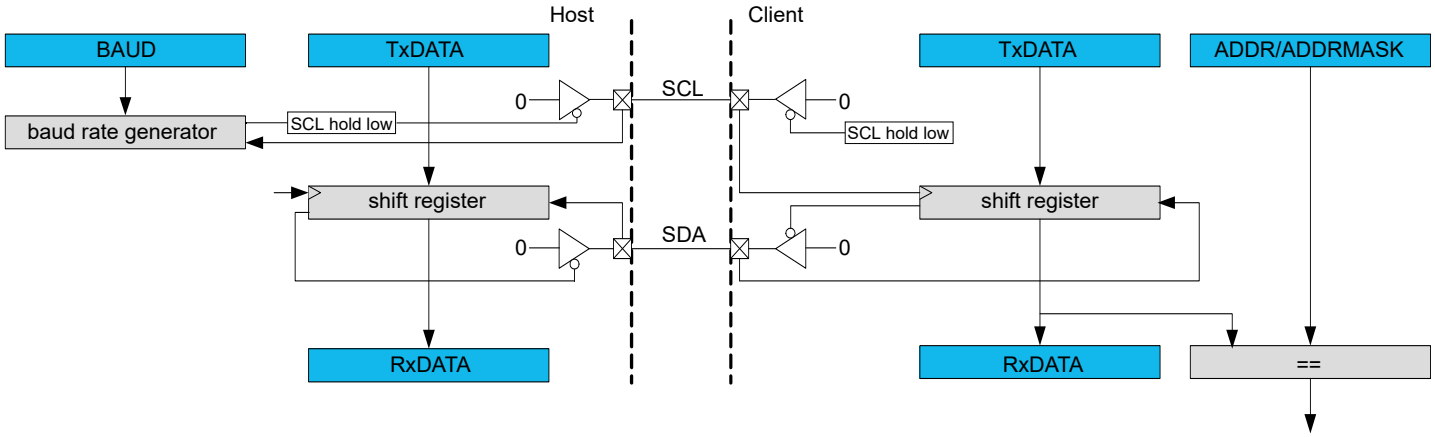
32.2 Features

SERCOM I²C includes the following features:

- Host or Client Operation
- Can be used with DMA
- Philips I²C Compatible
- SMBus Compatible
- PMBus™ Compatible
- Support of 100 kHz and 400 kHz, 1 MHz I²C mode
- 4-Wire Operation Supported
- Physical interface includes:
  - Slew-rate limited outputs
  - Filtered inputs
- Client Operation:
  - Operation in all Sleep modes
  - Wake-up on address match
  - 7-bit Address match in hardware for:
  - –
    - Unique address and/or 7-bit general call address
    - Address range
    - Two unique addresses can be used with DMA

32.3 Block Diagram

Figure 32-1. I²C Single-Host Single-Client Interconnection



32.4 Signal Description

Signal Name	Type	Description
PAD[0]	Digital I/O	SDA
PAD[1]	Digital I/O	SCL
PAD[2]	Digital I/O	SDA_OUT (4-wire operation)
PAD[3]	Digital I/O	SCL_OUT (4-wire operation)

One signal can be mapped on several pins.

Not all the pins are I²C pins. Refer to Table 7-5. SERCOM Pins Supporting I²C for additional information .

Related Links

32.6.3.3. 4-Wire Mode

32.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

32.5.1 I/O Lines

In order to use the SERCOM's I/O lines, the I/O pins must be configured using the System Configuration registers only. I2C does not operate through PPS. See DEVCFG1 configuration bits SCOMn_HSEN in Configuration Bits Fuses and also CFGCON1 SCOMn_HSEN in CFGCON1(L) register. See *CFGCON1(L)* register from Related Links.

When the SERCOM is used in I²C mode, the SERCOM controls the direction and value of the I/O pins. In I²C mode pull-up resistors are disabled. External pull-up resistors are required for proper function.

Related Links

18.4.2. CFGCON1(L)

32.5.2 Power Management

This peripheral can continue to operate in any Sleep mode where its source clock is running. The interrupts can wake-up the device from Sleep modes.

32.5.3 Clocks

Two generic clocks are used by SERCOM, GCLK_SERCOMx_CORE and GCLK_SERCOMx_SLOW. The core clock (GCLK_SERCOMx_CORE) can clock the I^2C when working as a host. The slow clock (GCLK_SERCOMx_SLOW) is required only for certain functions, e.g., SMBus timing. These two clocks must be configured and enabled in the CRU registers before using the I^2C.

These generic clocks are asynchronous to the bus clock (PBx_CLK). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains.

32.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). To use DMA requests with this peripheral, the DMAC must be configured first (see *Direct Memory Access Controller (DMAC)* from Related Links).

Related Links

22. Direct Memory Access Controller (DMAC)

32.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. In order to use interrupt requests of this peripheral, the Interrupt Controller (NVIC) must be configured first. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

32.5.6 Events

Not applicable.

32.5.7 Debug Operation

When the CPU is halted in Debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

Related Links

32.10.11. DBGCTRL

32.5.8 Register Access Protection

Registers with write access can be write-protected optionally by the Peripheral Access Controller (PAC).

PAC write protection is not available for the following registers:

- Interrupt Flag Clear and Status register (INTFLAG)
- Status register (STATUS)
- Data register (DATA)
- Address register (ADDR) in Host mode

Optional PAC write protection is denoted by the "PAC Write-Protection" property in each individual register description.

Write-protection does not apply to accesses through an external debugger.

32.5.9 Analog Connections

Not applicable.

32.6 Functional Description

32.6.1 Principle of Operation

The I^2C interface uses two physical lines for communication:

- Serial Data Line (SDA) for data transfer
- Serial Clock Line (SCL) for the bus clock

A transaction starts with the I^2C host sending the Start condition, followed by a 7-bit address and a direction bit (read or write to/from the client).

The addressed I^2C client will then Acknowledge (ACK) the address, and data packet transactions can begin. Every 9-bit data packet consists of 8 data bits followed by a one-bit reply indicating whether the data was acknowledged or not.

If a data packet is Not Acknowledged (NACK), whether by the I^2C client or host, the I^2C host takes action by either terminating the transaction by sending the Stop condition, or by sending a repeated start to transfer more data.

The figure below illustrates the possible transaction formats and Transaction Diagram Symbols explains the transaction symbols. These symbols will be used in the following descriptions.

Figure 32-2. Transaction Diagram Symbols

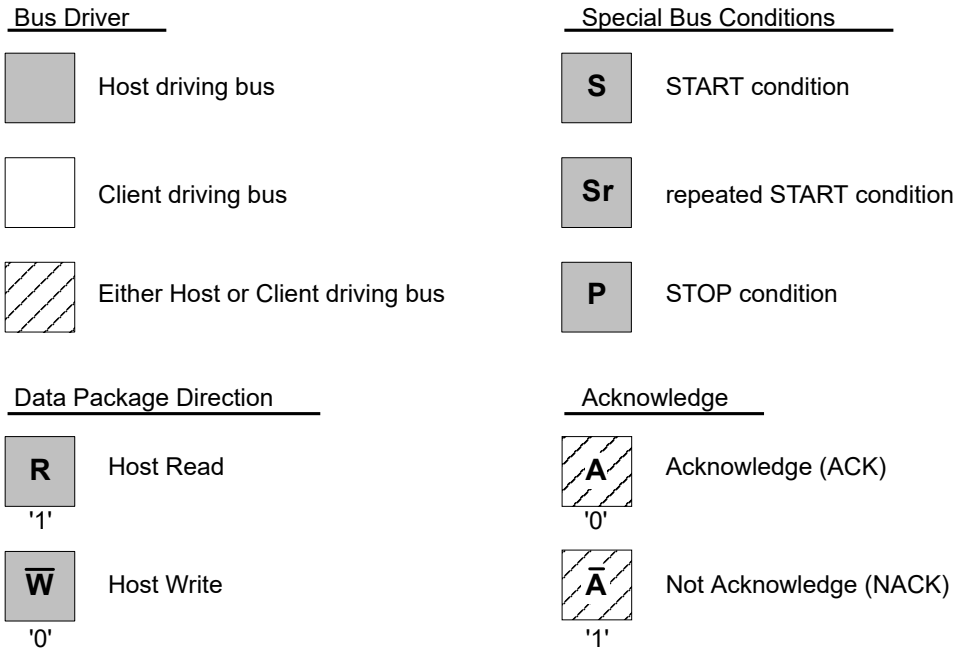
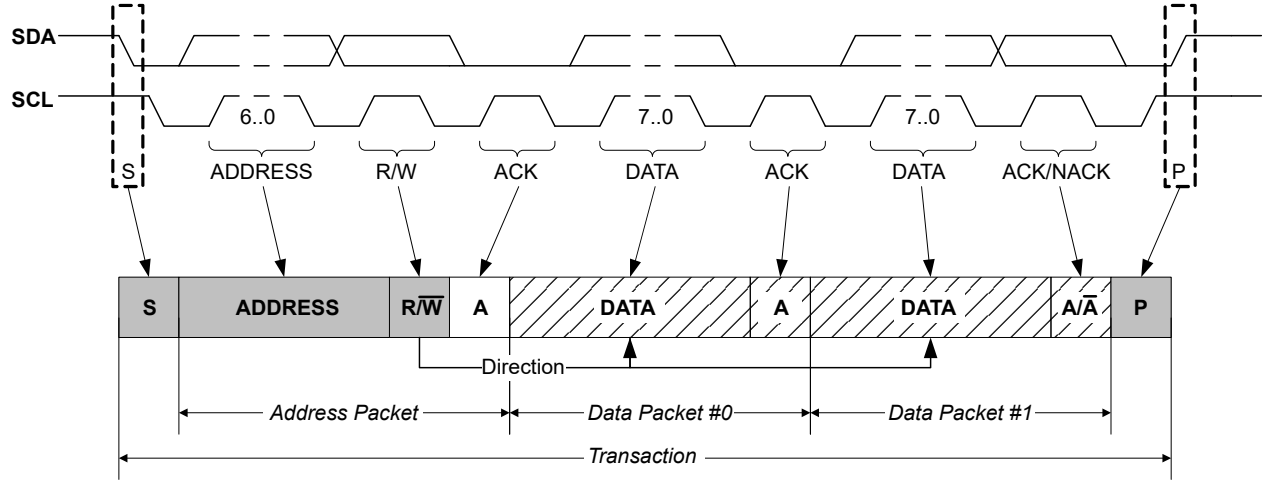


Figure 32-3. Basic I^2C Transaction Diagram



32.6.2 Basic Operation

32.6.2.1 Initialization

The following registers are enable-protected, meaning they can be written only when the I²C interface is disabled (CTRLA.ENABLE is '0'):

- Control A register (CTRLA), except Enable (CTRLA.ENABLE) and Software Reset (CTRLA.SWRST) bits
- Control B register (CTRLB), except Acknowledge Action (CTRLB.ACKACT) and Command (CTRLB.CMD) bits
- Baud register (BAUD)
- Address register (ADDR) in client operation.

When the I²C is enabled or is being enabled (CTRLA.ENABLE=1), writing to these registers will be discarded. If the I²C is being disabled, writing to these registers will be completed after the disabling.

Enable-protection is denoted by the "Enable-Protection" property in the register description.

Before the I²C is enabled it must be configured as outlined by the following steps:

1. Select I²C Host or Client mode by writing 0x4 (Client mode) or 0x5 (Host mode) to the Operating Mode bits in the CTRLA register (CTRLA.MODE).
2. If desired, select the SDA Hold Time value in the CTRLA register (CTRLA.SDAHOLD).
3. If desired, enable smart operation by setting the Smart Mode Enable bit in the CTRLB register (CTRLB.SMEN).
4. If desired, enable SCL low time-out by setting the SCL Low Time-Out bit in the Control A register (CTRLA.LOWTOUT).
5. In Host mode:
   a. Select the inactive bus time-out in the Inactive Time-Out bit group in the CTRLA register (CTRLA.INACTOUT).
   b. Write the Baud Rate register (BAUD) to generate the desired baud rate.

   In Client mode:
   a. Configure the address match configuration by writing the Address Mode value in the CTRLB register (CTRLB.AMODE).
   b. Set the Address and Address Mask value in the Address register (ADDR.ADDR and ADDR.ADDRMASK) according to the address configuration.

32.6.2.2 Enabling, Disabling, and Resetting

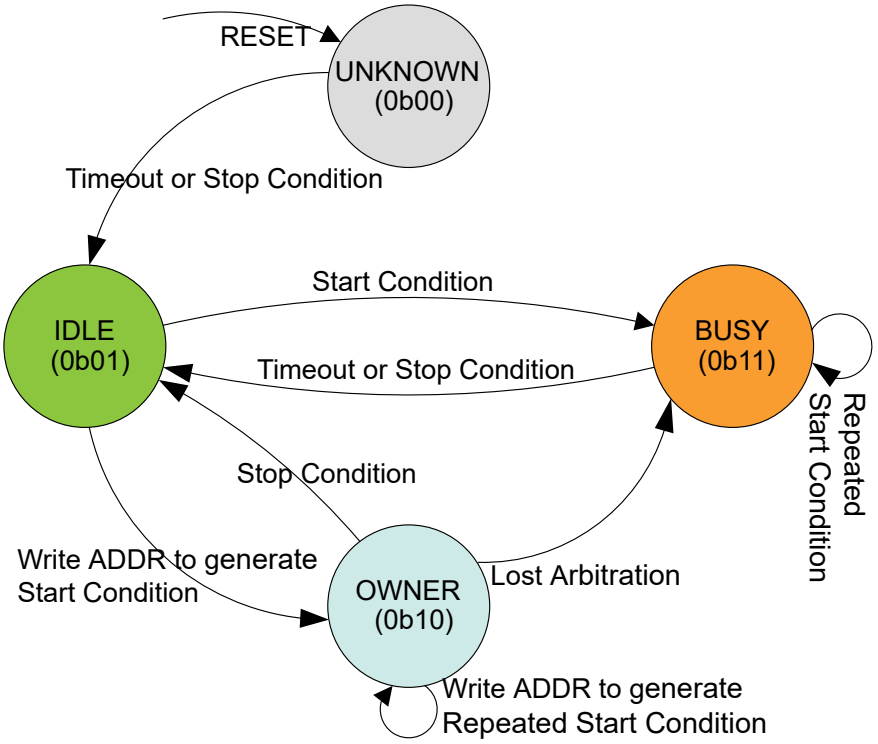
This peripheral is enabled by writing '1' to the Enable bit in the Control A register (CTRLA.ENABLE), and disabled by writing '0' to it.

Writing '1' to the Software Reset bit in the Control A register (CTRLA.SWRST) will reset all registers of this peripheral to their initial states, except the DBGCTRL register, and the peripheral is disabled.

32.6.2.3 I²C Bus State Logic

The Bus state logic includes several logic blocks that continuously monitor the activity on the I²C bus lines in all Sleep modes with running GCLK_SERCOM_x clocks. The start and stop detectors and the bit counter are all essential in the process of determining the current Bus state. The Bus state is determined according to Bus State Diagram. Software can get the current Bus state by reading the Host Bus State bits in the Status register (STATUS.BUSSTATE). The value of STATUS.BUSSTATE in the figure is shown in binary.

Figure 32-4. Bus State Diagram



The Bus state machine is active when the I²C host is enabled.

After the I²C host has been enabled, the Bus state is UNKNOWN (0b00). From the UNKNOWN state, the bus will transition to IDLE (0b01) by either:

- Forcing by writing 0b01 to STATUS.BUSSTATE
- A Stop condition is detected on the bus
- If the inactive bus time-out is configured for SMBus compatibility (CTRLA.INACTOUT) and a time-out occurs.

Note: Once a known Bus state is established, the Bus state logic will not re-enter the UNKNOWN state.

When the bus is IDLE it is ready for a new transaction. If a Start condition is issued on the bus by another I²C host in a multi-host setup, the bus becomes BUSY (0b11). The bus will re-enter IDLE either when a Stop condition is detected, or when a time-out occurs (inactive bus time-out needs to be configured).

If a Start condition is generated internally by writing the Address bit group in the Address register (ADDR.ADDR) while IDLE, the OWNER state (0b10) is entered. If the complete transaction was performed without interference, i.e., arbitration was not lost, the I²C host can issue a Stop condition, which will change the Bus state back to IDLE.

However, if a packet collision is detected while in OWNER state, the arbitration is assumed lost and the Bus state becomes BUSY until a Stop condition is detected. A repeated Start condition will change the Bus state only if arbitration is lost while issuing a repeated start.

Note: Violating the protocol may cause the I²C to hang. If this happens it is possible to recover from this state by a software Reset (CTRLA.SWRST='1').

32.6.2.4 I²C Host Operation

The I²C host is byte-oriented and interrupt based. The number of interrupts generated is kept at a minimum by automatic handling of most incidents. The software driver complexity and code size are reduced by auto-triggering of operations, and a Special Smart mode, which can be enabled by the Smart Mode Enable bit in the Control A register (CTRLA.SMEN).

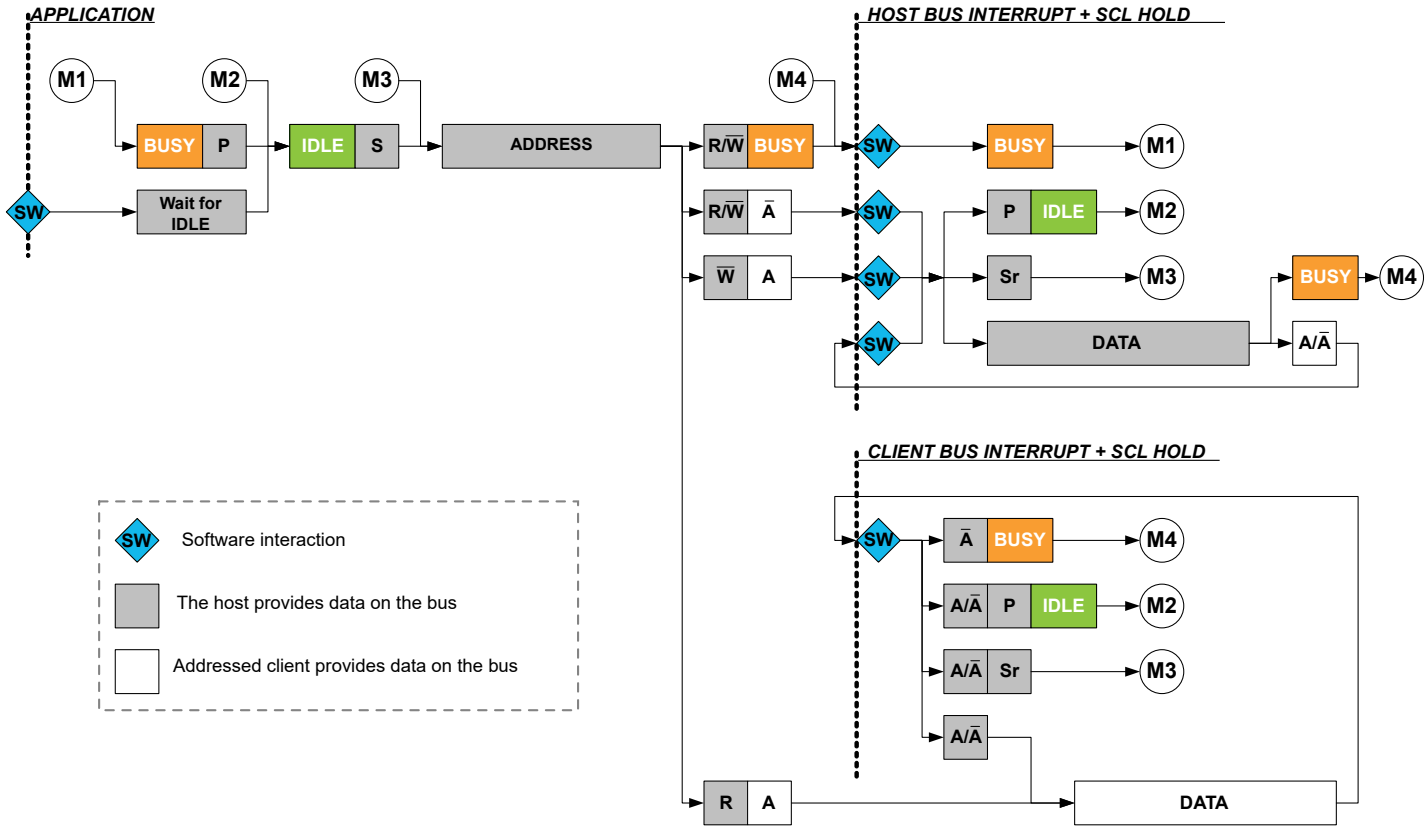
The I²C host has two interrupt strategies.

When SCL Stretch Mode (CTRLA.SCLSM) is '0', SCL is stretched before or after the Acknowledge bit . In this mode the I²C host operates according to *I²C Host Behavioral Diagram (SCLSM=0)* as shown in the following figure.

The circles labeled "M*n*" (M1, M2..) indicate the nodes the bus logic can jump to, based on software or hardware interaction.

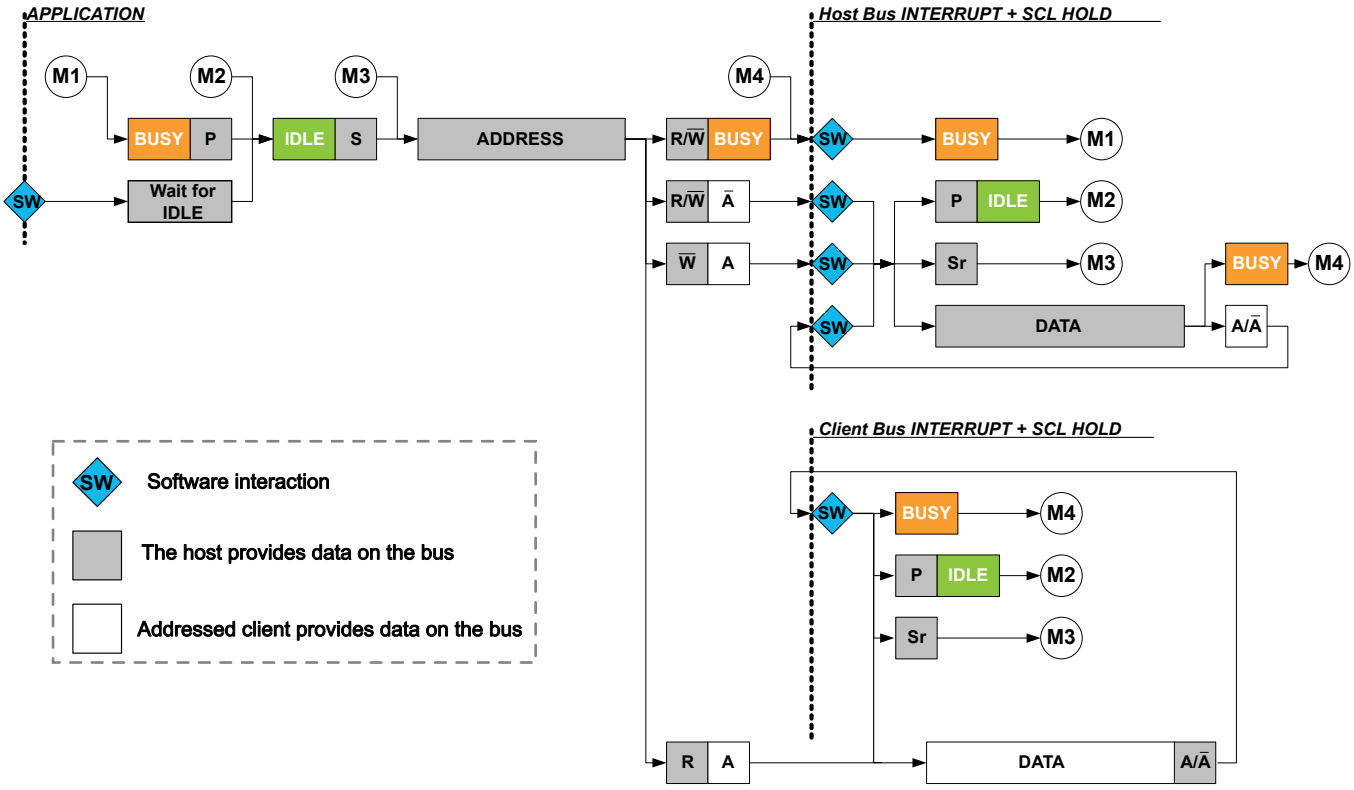
This diagram is used as reference for the description of the I^2C host operation throughout the document.

Figure 32-5. I^2C Host Behavioral Diagram (SCLSM=0)



In the second strategy (CTRLA.SCLSM=1), interrupts only occur after the ACK bit, as in *Host Behavioral Diagram (SCLSM=1)* shown in the following figure. This strategy can be used when it is not necessary to check DATA before acknowledging.

Figure 32-6. I²C Host Behavioral Diagram (SCLSM=1)



32.6.2.4.1 Host Clock Generation

The SERCOM peripheral supports several I²C bidirectional modes:

- Standard mode (*Sm*) up to 100 kHz
- Fast mode (*Fm*) up to 400 kHz
- Fast mode Plus (*Fm+*) up to 1 MHz

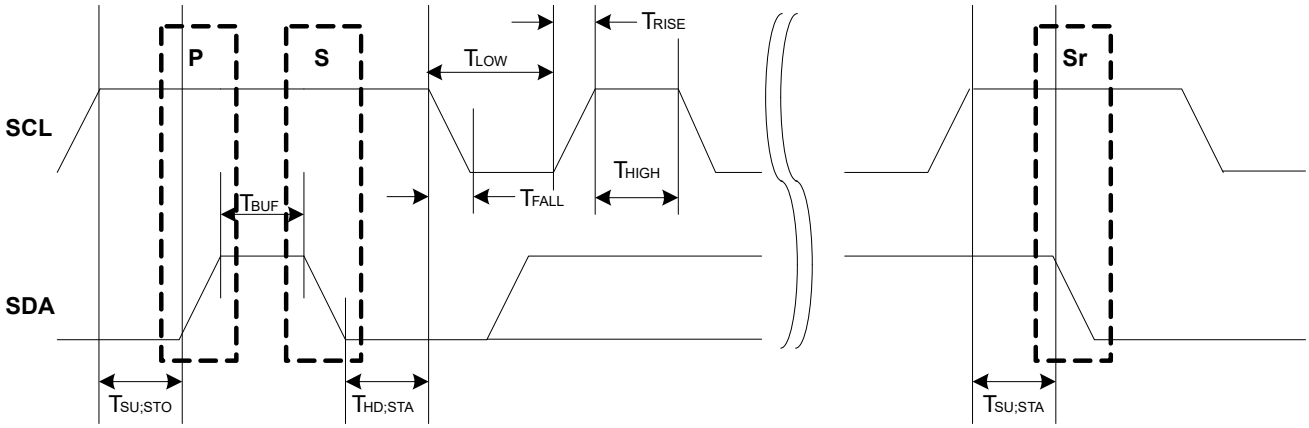
The Host clock configuration for *Sm, Fm* and *Fm+* are described in *Clock Generation (Standard-Mode, Fast-Mode and Fast-Mode Plus)* as follows.

Clock Generation (Standard-Mode, Fast-Mode, and Fast-Mode Plus)

In I²C *Sm, Fm*, and *Fm+* mode, the Host clock (SCL) frequency is determined as described in this section:

The low (T_{LOW}) and high (T_{HIGH}) times are determined by the Baud Rate register (BAUD), while the rise (T_{RISE}) and fall (T_{FALL}) times are determined by the bus topology. Because of the wired-AND logic of the bus, T_{FALL} will be considered as part of T_{LOW}. Likewise, T_{RISE} will be in a state between T_{LOW} and T_{HIGH} until a high state has been detected.

Figure 32-7. SCL Timing



The following parameters are timed using the SCL low time period T_{LOW}. This comes from the Host Baud Rate Low bit group in the Baud Rate register (BAUD.BAUDLOW). When BAUD.BAUDLOW=0, or the Host Baud Rate bit group in the Baud Rate register (BAUD.BAUD) determines it.

- T_{LOW} – Low period of SCL clock
- $T_{SU;STO}$ – Set-up time for stop condition
- T_{BUF} – Bus free time between stop and start conditions
- $T_{HD;STA}$ – Hold time (repeated) start condition
- $T_{SU;STA}$ – Set-up time for repeated start condition
- T_{HIGH} is timed using the SCL high time count from BAUD.BAUD
- T_{RISE} is determined by the bus impedance; for internal pull-ups.
- T_{FALL} is determined by the open-drain current limit and bus impedance; can typically be regarded as zero.

The SCL frequency is given by:

$$f_{\mathrm{SCL}} = \frac{1}{T_{\mathrm{LOW}} + T_{\mathrm{HIGH}} + T_{\mathrm{RISE}}}$$

When BAUD.BAUDLOW is zero, the BAUD.BAUD value is used to time both SCL high and SCL low. In this case the following formula will give the SCL frequency:

$$f_{\mathrm{SCL}} = \frac{f_{\mathrm{GCLK}}}{10 + 2BAUD + f_{\mathrm{GCLK}} \cdot T_{\mathrm{RISE}}}$$

When BAUD.BAUDLOW is non-zero, the following formula determines the SCL frequency:

$$f_{\mathrm{SCL}} = \frac{f_{\mathrm{GCLK}}}{10 + BAUD + BAUDLOW + f_{\mathrm{GCLK}} \cdot T_{\mathrm{RISE}}}$$

The following formulas can determine the SCL T_{LOW} and T_{HIGH} times:

$$T_{\mathrm{LOW}} = \frac{BAUDLOW + 5}{f_{\mathrm{GCLK}}}$$

$$T_{\mathrm{HIGH}} = \frac{BAUD + 5}{f_{\mathrm{GCLK}}}$$

Note: The I^2C standard *Fm+* (Fast-mode plus) requires a nominal high to low SCL ratio of 1:2, and BAUD must be set accordingly. At a minimum, BAUD.BAUD and/or BAUD.BAUDLOW must be non-zero.

Start-up Timing: The minimum time between SDA transition and SCL rising edge is 6 APB cycles when the DATA register is written in smart mode. If a greater start-up time is required due to long rise times, the time between DATA write and IF clear must be controlled by software.
Note: When timing is controlled by user, the Smart Mode cannot be enabled.

32.6.2.4.2 Transmitting Address Packets

The I²C host starts a bus transaction by writing the I²C client address to ADDR.ADDR and the direction bit, as described in Principle of Operation, see *Principle of Operation* from Related Links. If the bus is busy, the I²C host will wait until the bus becomes idle before continuing the operation. When the bus is idle, the I²C host will issue a start condition on the bus. The I²C host will then transmit an address packet using the address written to ADDR.ADDR. After the address packet has been transmitted by the I²C host, one of four cases will arise according to arbitration and transfer direction.

Case 1: Arbitration lost or bus error during address packet transmission

If arbitration was lost during transmission of the address packet, the Host on Bus bit in the Interrupt Flag Status and Clear register (INTFLAG.MB) and the Arbitration Lost bit in the Status register (STATUS.ARBLOST) are both set. Serial data output to SDA is disabled, and the SCL is released, which disables clock stretching. In effect the I²C host is no longer allowed to execute any operation on the bus until the bus is idle again. A bus error will behave similarly to the Arbitration Lost condition. In this case, the MB Interrupt flag and Host Bus Error bit in the Status register (STATUS.BUSERR) are both set in addition to STATUS.ARBLOST.

The Host Received Not Acknowledge bit in the Status register (STATUS.RXNACK) will always contain the last successfully received acknowledge or not acknowledge indication.

In this case, software will typically inform the application code of the condition and then clear the Interrupt flag before exiting the interrupt routine. No other flags have to be cleared at this moment, because all flags will be cleared automatically the next time the ADDR.ADDR register is written.

Case 2: Address packet transmit complete – No ACK received

If there is no I²C client device responding to the address packet, then the INTFLAG.MB Interrupt flag and STATUS.RXNACK will be set. The clock hold is active at this point, preventing further activity on the bus.

The missing ACK response can indicate that the I²C client is busy with other tasks or sleeping. Therefore, it is not able to respond. In this event, the next step can be either issuing a Stop condition (recommended) or resending the address packet by a repeated Start condition. When using SMBus logic, the client must ACK the address. If there is no response, it means that the client is not available on the bus.

Case 3: Address packet transmit complete – Write packet, Host on Bus set

If the I²C host receives an acknowledge response from the I²C client, INTFLAG.MB will be set and STATUS.RXNACK will be cleared. The clock hold is active at this point, preventing further activity on the bus.

In this case, the software implementation becomes highly protocol dependent. Three possible actions can enable the I²C operation to continue:

- Initiate a data transmit operation by writing the data byte to be transmitted into DATA.DATA.
- Transmit a new address packet by writing ADDR.ADDR. A repeated Start condition will automatically be inserted before the address packet.
- Issue a Stop condition, consequently terminating the transaction.

Case 4: Address packet transmit complete – Read packet, Client on Bus set

If the I²C host receives an ACK from the I²C client, the I²C host proceeds to receive the next byte of data from the I²C client. When the first data byte is received, the Client on Bus bit in the Interrupt Flag register (INTFLAG.SB) will be set and STATUS.RXNACK will be cleared. The clock hold is active at this point, preventing further activity on the bus.

In this case, the software implementation becomes highly protocol dependent. Three possible actions can enable the I²C operation to continue:

- Let the I²C host continue to read data by acknowledging the data received. ACK can be sent by software, or automatically in Smart mode.
- Transmit a new address packet.
- Terminate the transaction by issuing a Stop condition.

Note: An ACK or NACK will be automatically transmitted if Smart mode is enabled. The Acknowledge Action bit in the Control B register (CTRLB.ACKACT) determines whether ACK or NACK must be sent.

Related Links

32.6.1. Principle of Operation

32.6.2.4.3 Transmitting Data Packets

When an address packet with direction Host Write (see Figure 32-3) was transmitted successfully , INTFLAG.MB will be set. The I²C host will start transmitting data via the I²C bus by writing to DATA.DATA, and monitor continuously for packet collisions.

If a collision is detected, the I²C host will lose arbitration and STATUS.ARBLOST will be set. If the transmit was successful, the I²C host will receive an ACK bit from the I²C client, and STATUS.RXNACK will be cleared. INTFLAG.MB will be set in both cases, regardless of arbitration outcome.

It is recommended to read STATUS.ARBLOST and handle the arbitration lost condition in the beginning of the I²C Host on Bus interrupt. This can be done as there is no difference between handling address and data packet arbitration.

STATUS.RXNACK must be checked for each data packet transmitted before the next data packet transmission can commence. The I²C host is not allowed to continue transmitting data packets if a NACK is received from the I²C client.

32.6.2.4.4 Receiving Data Packets (SCLSM=0)

When INTFLAG.SB is set, the I²C host will already have received one data packet. The I²C host must respond by sending either an ACK or NACK. Sending a NACK may be unsuccessful when arbitration is lost during the transmission. In this case, a lost arbitration will prevent setting INTFLAG.SB. Instead, INTFLAG.MB will indicate a change in arbitration. Handling of lost arbitration is the same as for data bit transmission.

32.6.2.4.5 Receiving Data Packets (SCLSM=1)

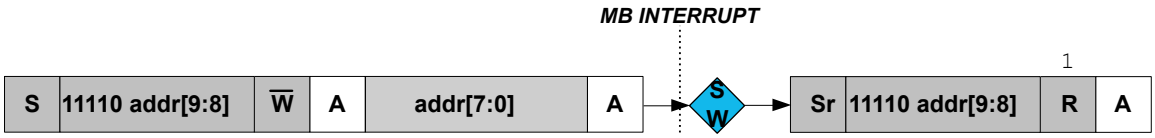
When INTFLAG.SB is set, the I²C host will already have received one data packet and transmitted an ACK or NACK, depending on CTRLB.ACKACT. At this point, CTRLB.ACKACT must be set to the correct value for the next ACK bit, and the transaction can continue by reading DATA and issuing a command if not in the Smart mode.

32.6.2.4.6 10-Bit Addressing

When 10-bit addressing is enabled by the Ten Bit Addressing Enable bit in the Address register (ADDR.TENBITEN=1) and the Address bit field ADDR.ADDR is written, the two address bytes will be transmitted, see 10-bit Address Transmission for a Read Transaction. The addressed client acknowledges the two address bytes, and the transaction continues. Regardless of whether the transaction is a read or write, the host must start by sending the 10-bit address with the direction bit (ADDR.ADDR[0]) being zero.

If the host receives a NACK after the first byte, the Write Interrupt flag will be raised and the STATUS.RXNACK bit will be set. If the first byte is acknowledged by one or more clients, then the host will proceed to transmit the second address byte and the host will first see the Write Interrupt flag after the second byte is transmitted. If the transaction direction is read-from-client, the 10-bit address transmission must be followed by a repeated start and the first 7 bits of the address with the read/write bit equal to '1'.

Figure 32-8. 10-bit Address Transmission for a Read Transaction



This implies the following procedure for a 10-bit read operation:

1. Write the 10-bit address to ADDR.ADDR[10:1]. ADDR.TENBITEN must be '1', the direction bit (ADDR.ADDR[0]) must be '0' (can be written simultaneously with ADDR).
2. Once the Host on Bus interrupt is asserted, Write ADDR[7:0] register to '11110 `address[9:8]` 1'. ADDR.TENBITEN must be cleared (can be written simultaneously with ADDR).
3. Proceed to transmit data.

32.6.2.5 I²C Client Operation

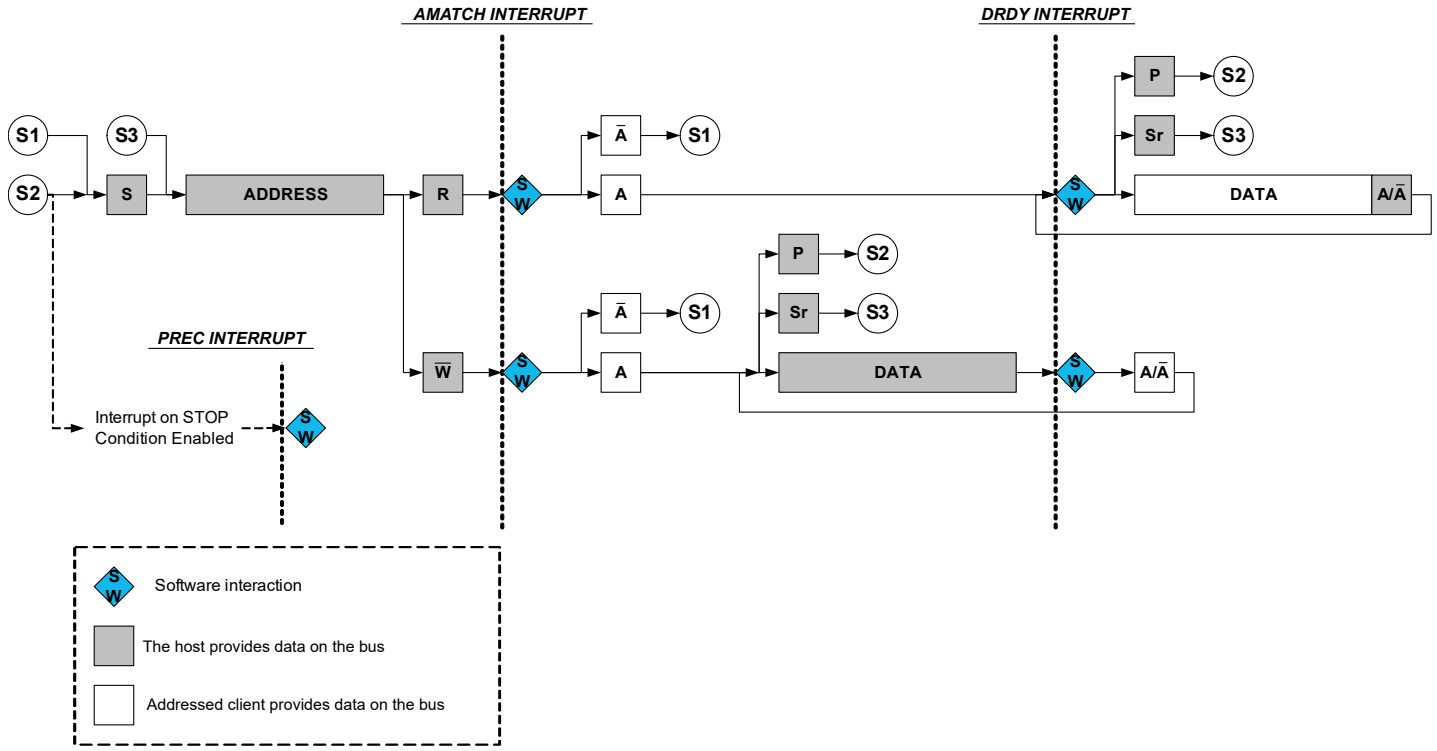
The I²C client is byte-oriented and interrupt-based. The number of interrupts generated is kept at a minimum by automatic handling of most events. The software driver complexity and code size are reduced by auto-triggering of operations, and a special smart mode, which can be enabled by the Smart Mode Enable bit in the Control A register (CTRLA.SMEN).

The I²C client has two interrupt strategies.

When SCL Stretch Mode bit (CTRLA.SCLSM) is '0', SCL is stretched before or after the acknowledge bit. In this mode, the I²C client operates according to *I²C Client Behavioral Diagram (SCLSM=0)* as shown in the following figure. The circles labelled "S*n*" (S1, S2..) indicate the nodes the bus logic can jump to, based on software or hardware interaction.

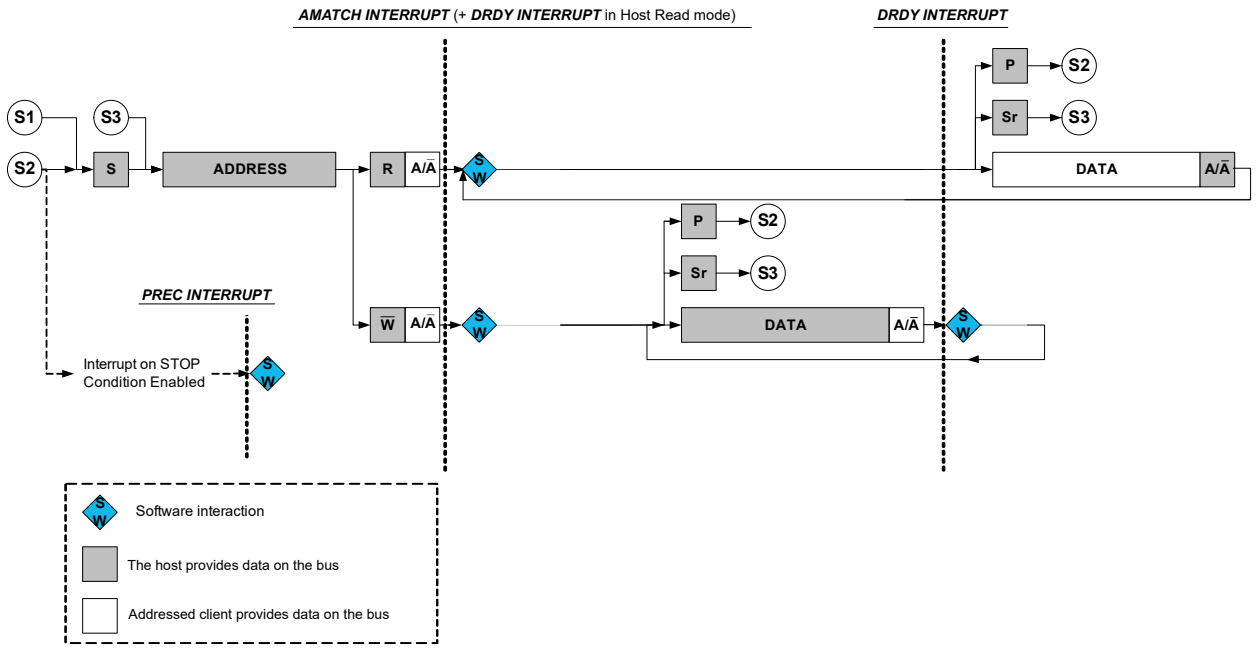
This diagram is used as reference for the description of the I²C client operation throughout the document.

Figure 32-9. I²C Client Behavioral Diagram (SCLSM=0)



In the second strategy (CTRLA.SCLSM=1), interrupts only occur after the ACK bit is sent as shown in the following figure *I²C Client Behavioral Diagram (SCLSM=1)*. This strategy can be used when it is not necessary to check DATA before acknowledging. For host reads, an address and data interrupt will be issued simultaneously after the address acknowledge. However, for host writes, the first data interrupt will be seen after the first data byte has been received by the client and the acknowledge bit has been sent to the host.

Figure 32-10. I²C Client Behavioral Diagram (SCLSM=1)



32.6.2.5.1 Receiving Address Packets (SCLSM=0)

When CTRLA.SCLSM=0, the I²C client stretches the SCL line according to Figure 32-9. When the I²C client is properly configured, it will wait for a Start condition.

When a Start condition is detected, the successive address packet will be received and checked by the address match logic. If the received address is not a match, the packet will be rejected, and the I²C client will wait for a new Start condition. If the received address is a match, the Address Match bit in the Interrupt Flag register (INTFLAG.AMATCH) will be set.

SCL will be stretched until the I²C client clears INTFLAG.AMATCH. As the I²C client holds the clock by forcing SCL low, the software has unlimited time to respond.

The direction of a transaction is determined by reading the Read/Write Direction bit in the Status register (STATUS.DIR). This bit will be updated only when a valid address packet is received.

If the Transmit Collision bit in the Status register (STATUS.COLL) is set, this indicates that the last packet addressed to the I²C client had a packet collision. A collision causes the SDA and SCL lines to be released without any notification to software. Therefore, the next AMATCH interrupt is the first indication of the previous packet's collision. Collisions are intended to follow the SMBus Address Resolution Protocol (ARP).

After the address packet has been received from the I²C host, one of two cases will arise based on transfer direction.

Case 1: Address packet accepted – Read flag set

The STATUS.DIR bit is '1', indicating an I²C host read operation. The SCL line is forced low, stretching the bus clock. If an ACK is sent, I²C client hardware will set the Data Ready bit in the Interrupt Flag register (INTFLAG.DRDY), indicating data are needed for transmit. If a NACK is sent, the I²C client will wait for a new Start condition and address match.

Typically, software will immediately acknowledge the address packet by sending an ACK/NACK bit. The I²C client Command bit field in the Control B register (CTRLB.CMD) can be written to '0x3' for both read and write operations as the command execution is dependent on the STATUS.DIR bit. Writing '1' to INTFLAG.AMATCH will also cause an ACK/NACK to be sent corresponding to the CTRLB.ACKACT bit.

Case 2: Address packet accepted – Write flag set

The STATUS.DIR bit is cleared, indicating an I²C host write operation. The SCL line is forced low, stretching the bus clock. If an ACK is sent, the I²C client will wait for data to be received. Data, repeated start or stop can be received.

If a NACK is sent, the I²C client will wait for a new Start condition and address match. Typically, software will immediately acknowledge the address packet by sending an ACK/NACK. The I²C client command CTRLB.CMD = 3 can be used for both read and write operation as the command execution is dependent on STATUS.DIR.

Writing '1' to INTFLAG.AMATCH will also cause an ACK/NACK to be sent corresponding to the CTRLB.ACKACT bit.

32.6.2.5.2 Receiving Address Packets (SCLSM=1)

When SCLSM=1, the I²C client will stretch the SCL line only after an ACK (see Figure 32-10). When the I²C client is properly configured, it will wait for a Start condition to be detected.

When a Start condition is detected, the successive address packet will be received and checked by the address match logic.

If the received address is not a match, the packet will be rejected and the I²C client will wait for a new Start condition.

If the address matches, the acknowledge action as configured by the Acknowledge Action bit Control B register (CTRLB.ACKACT) will be sent and the Address Match bit in the Interrupt Flag register (INTFLAG.AMATCH) is set. SCL will be stretched until the I²C client clears INTFLAG.AMATCH. As the I²C client holds the clock by forcing SCL low, the software is given unlimited time to respond to the address.

The direction of a transaction is determined by reading the Read/Write Direction bit in the Status register (STATUS.DIR). This bit will be updated only when a valid address packet is received.

If the Transmit Collision bit in the Status register (STATUS.COLL) is set, the last packet addressed to the I²C client had a packet collision. A collision causes the SDA and SCL lines to be released without any notification to software. The next AMATCH interrupt is, therefore, the first indication of the previous packet's collision. Collisions are intended to follow the SMBus Address Resolution Protocol (*ARP*).

After the address packet has been received from the I²C host, INTFLAG.AMATCH can be set to '1' to clear it.

32.6.2.5.3 Receiving and Transmitting Data Packets

After the I²C client has received an address packet, it will respond according to the direction either by waiting for the data packet to be received or by starting to send a data packet by writing to DATA.DATA. When a data packet is received or sent, INTFLAG.DRDY will be set. After receiving data, the I²C client will send an acknowledge according to CTRLB.ACKACT.

Case 1: Data received

INTFLAG.DRDY is set, and SCL is held low, pending for SW interaction.

Case 2: Data sent

When a byte transmission is successfully completed, the INTFLAG.DRDY Interrupt flag is set. If NACK is received, indicated by STATUS.RXNACK=1, the I²C client must expect a stop or a repeated start to be received. The I²C client must release the data line to allow the I²C host to generate a stop or repeated start. Upon detecting a Stop condition, the Stop Received bit in the Interrupt Flag register (INTFLAG.PREC) will be set and the I²C client will return to IDLE state.

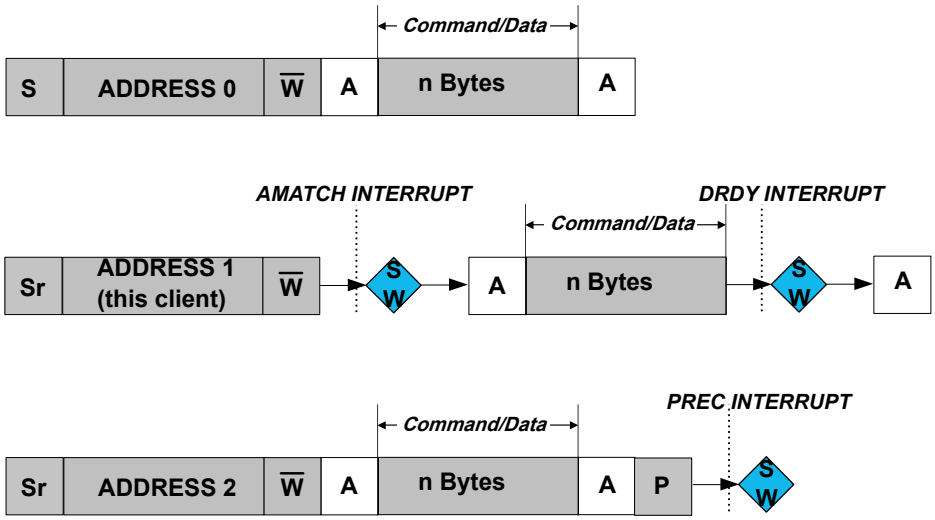
32.6.2.5.4 PMBus Group Command

When the PMBus Group Command bit in the CTRLB register is set (CTRLB.GCMD=1) and 7-bit addressing is used, INTFLAG.PREC will be set if the client has been addressed since the last STOP condition. When CTRLB.GCMD=0, a STOP condition without address match will not set INTFLAG.PREC.

The group command protocol is used to send commands to more than one device. The commands are sent in one continuous transmission with a single STOP condition at the end. When the STOP condition is detected by the clients addressed during the group command, they all begin executing the command they received.

The following figure shows an example where this client, bearing ADDRESS 1, is addressed after a repeated START condition. There can be multiple clients addressed before and after this client. Eventually, at the end of the group command, a single STOP is generated by the host. At this point a STOP interrupt is asserted.

Figure 32-11. PMBus Group Command Example



32.6.3 Additional Features

32.6.3.1 SMBus

The I^{2}C includes three hardware SCL low time-outs which allow a time-out to occur for SMBus SCL low time-out, host extend time-out, and client extend time-out. This allows for SMBus functionality These time-outs are driven by the GCLK_SERCOM_SLOW clock. The GCLK_SERCOM_SLOW clock is used to accurately time the time-out and must be configured to use a 32.768 kHz oscillator. The I^{2}C interface also allows for a SMBus compatible SDA hold time.

- **$T_{TIMEOUT}$:** SCL low time of 25..35ms – Measured for a single SCL low period. It is enabled by CTRLA.LOWTOUTEN
- **$T_{LOW:SEXT}$:** Cumulative clock low extend time of 25 ms – Measured as the cumulative SCL low extend time by a client device in a single message from the initial START to the STOP. It is enabled by CTRLA.SEXTTOEN.
- **$T_{LOW:MEXT}$:** Cumulative clock low extend time of 10 ms – Measured as the cumulative SCL low extend time by the host device within a single byte from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is enabled by CTRLA.MEXTTOEN.

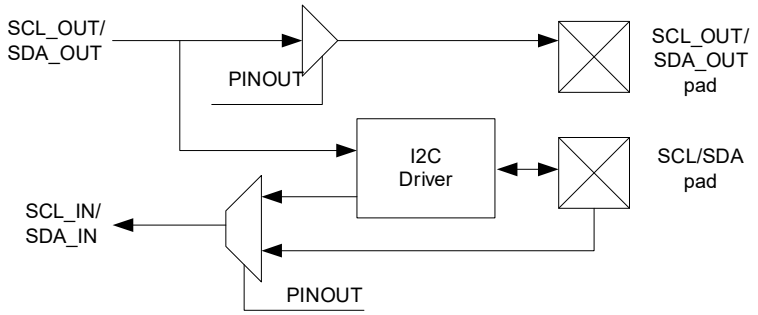
32.6.3.2 Smart Mode

The I^{2}C interface has a Smart mode that simplifies application code and minimizes the user interaction needed to adhere to the I^{2}C protocol. The Smart mode accomplishes this by automatically issuing an ACK or NACK (based on the content of CTRLB.ACKACT) as soon as DATA.DATA is read.

32.6.3.3 4-Wire Mode

Writing a '1' to the Pin Usage bit in the Control A register (CTRLA.PINOUT) will enable 4-Wire mode operation. In this mode, the internal I^{2}C tri-state drivers are bypassed, and an external I^{2}C compliant tri-state driver is needed when connecting to an I^{2}C bus.

Figure 32-12. I²C Pad Interface



32.6.3.4 Quick Command

Setting the Quick Command Enable bit in the Control B register (CTRLB.QCEN) enables quick command. When quick command is enabled, the corresponding Interrupt flag (INTFLAG.SB or INTFLAG.MB) is set immediately after the client acknowledges the address. At this point, the software can either issue a Stop command or a repeated start by writing CTRLB.CMD or ADDR.ADDR.

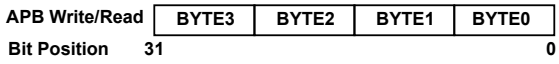
32.6.3.5 32-bit Extension

For better system bus utilization, 32-bit data receive and transmit can be enabled by writing to the Data 32-bit bit field in the Control C register (CTRLC.DATA32B=1). When enabled, write and read transaction to/from the DATA register are 32 bit in size.

If frames are not multiples of 4 Bytes, the Length Counter (LENGTH.LEN) and Length Enable (LENGTH.LENEN) must be configured before data transfer begins. LENGTH.LEN must be enabled only when CTRLC.DATA32B is enabled.

The following figure shows the order of transmit and receive when using 32-bit mode. Bytes are transmitted or received and stored in order from 0 to 3.

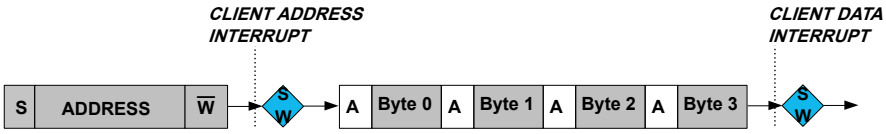
Figure 32-13. 32-bit Extension Byte Ordering



32-bit Extension Client Operation

The following figure shows a transaction with 32-bit Extension enabled (CTRLC.DATA32B=1). In client operation, the Address Match interrupt in the Interrupt Flag Status and Clear register (INTFLAG.AMATCH) is set after the address is received and available in the DATA register. The Data Ready interrupt (INTFLAG.DRDY) will then be raised for every 4 Bytes transferred.

Figure 32-14. 32-bit Extension Client Operation



The LENGTH register can be written before the frame begins, or when the AMATCH interrupt is set. If the frame size is not LENGTH.LEN Bytes, the Length Error status bit (STATUS.LENERR) is raised. If LENGTH.LEN is not a multiple of 4 Bytes, the final INTFLAG.DRDY interrupt is raised when the last Byte is received for host reads. For host writes, the last data byte will be automatically NACKed. On address recognition, the internal length counter is reset in preparation for the incoming frame.

High Speed transactions start with a Full Speed Host Code. When a Host Code is detected, no data is received and the next expected operation is a repeated start. For this reason, the length is not counted after a Host Code is received. In this case, no Length Error (STATUS.LENERR) is registered, regardless of the LENGTH.LENEN setting.

When SCL clock stretch mode is selected (CTRLA.SCLSM=1) and the transaction is a host write, the selected Acknowledge Action (CTRLB.ACKACT) will only be used to ACK/NACK each 4th byte. All other bytes are ACKed. This allows the user to write CTRLB.ACKACT=1 in the final interrupt, so that the last byte in a 32-bit word will be NACKed.

Writing to the LENGTH register while a frame is in progress will produce unpredictable results. If LENGTH.LENEN is not set and a frame is not a multiple of 4 Bytes, the remainder will be lost.

32-bit Extension Host Operation

When using the I^2C configured as Host, the Address register must be written with the desired address (ADDR.ADDR), and optionally, the transaction Length and transaction Length Enable bits (ADDR.LEN and ADDR.LENEN) can be written. When ADDR.LENEN is written to '1' along with ADDR.ADDR, ADDR.LEN determines the number of data bytes in the transaction from 0 to 255. Then, the ADDR.LEN bytes are transferred, followed by an automatically generated NACK (for host reads) and a STOP.

The INTFLAG.SB or INTFLAG.MB are raised for every 4 Bytes transferred. If the transaction is a host read and ADDR.LEN is not a multiple of 4 Bytes, the final INTFLAG.SB is set when the last byte is received.

When SCL clock stretch mode is enabled (CTRLA.SCLSM=1) and the transaction is a host read, the selected Acknowledge Action (CTRLB.ACKACT) will only be used to ACK/NACK each 4th Byte. All other bytes are ACKed. This allows the user to set CTRLB.ACKACT=1 in the final interrupt, so that the last byte in a 32-bit word will be NACKed.

If a NACK is received by the client for a host write transaction before ADDR.LEN bytes, a STOP will be automatically generated, and the length error (STATUS.LENERR) is raised along with the INTFLAG.ERROR interrupt.

32.6.4 DMA, Interrupts and Events

Each interrupt source has its own Interrupt flag. The Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) will be set when the Interrupt condition is meet. Each interrupt can be individually enabled by writing '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing '1' to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). An interrupt request is generated when the Interrupt flag is set and the corresponding interrupt is enabled. The interrupt request is active until the Interrupt flag is cleared, the interrupt is disabled or the I^2C is reset. See the INTFLAG (Client) or INTFLAG (Host) register for details on how to clear Interrupt flags.

Table 32-1. Module Request for SERCOM I^2C Client

Condition	Request		
	DMA	Interrupt	Event
Data needed for transmit (TX) (Client Transmit mode)	Yes (request cleared when data is written)	—	NA
Data received (RX) (Client Receive mode)	Yes (request cleared when data is read)	—	
Data Ready (DRDY)	—	Yes	
Address Match (AMATCH)	—	Yes	
Stop received (PREC)	—	Yes	
Error (ERROR)	—	Yes	

Table 32-2. Module Request for SERCOM I²C Host

Condition	Request		
	DMA	Interrupt	Event
Data needed for transmit (TX) (Host Transmit mode)	Yes (request cleared when data is written)	—	NA
Data needed for transmit (RX) (Host Transmit mode)	Yes (request cleared when data is read)	—	
Host on Bus (MB)	—	Yes	
Stop received (SB)	—	Yes	
Error (ERROR)	—	Yes	

32.6.4.1 DMA Operation

Smart mode must be enabled for DMA operation in the Control B register by writing CTRLB.SMEN=1.

32.6.4.1.1 Client DMA

When using the I²C client with DMA, an address match will cause the address Interrupt flag (INTFLAG.ADDRMATCH) to be raised. After the interrupt has been serviced, data transfer will be performed through DMA.

The I²C client generates the following requests:

32.6.4.1.2 Host DMA

When using the I²C host with DMA, the ADDR register must be written with the desired address (ADDR.ADDR), transaction length (ADDR.LEN), and transaction length enable (ADDR.LENEN). When ADDR.LENEN is written to 1 along with ADDR.ADDR, ADDR.LEN determines the number of data bytes in the transaction from 0 to 255. DMA is then used to transfer ADDR.LEN bytes followed by an automatically generated NACK (for host reads) and a STOP.

If a NACK is received by the client for a host write transaction before ADDR.LEN bytes, a STOP will be automatically generated and the length error (STATUS.LENERR) will be raised along with the INTFLAG.ERROR interrupt.

The I²C host generates the following requests:

32.6.4.2 Interrupts

The I²C client has the following interrupt sources. These are asynchronous interrupts. They can wake-up the device from any Sleep mode:

- Error (ERROR)
- Data Ready (DRDY)
- Address Match (AMATCH)
- Stop Received (PREC)

The I²C host has the following interrupt sources. These are asynchronous interrupts. They can wake-up the device from any Sleep mode:

- Error (ERROR)
- Client on Bus (SB)
- Host on Bus (MB)

Each interrupt source has its own Interrupt flag. The Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) will be set when the Interrupt condition is met. Each interrupt can be individually enabled by writing '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing '1' to the corresponding bit in the Interrupt Enable Clear register (INTENCLR).

The status of enabled interrupts can be read from either INTENSET or INTENCLR. An interrupt request is generated when the Interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the

Interrupt flag is cleared, the interrupt is disabled or the I²C is reset. For details on how to clear Interrupt flags, see *INTFLAG* register from Related Links.

The value of INTFLAG indicates which interrupt is executed. Note that interrupts must be globally enabled for interrupt requests. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

32.10.6. INTFLAG

32.6.4.3 Events

Not applicable.

32.6.5 Sleep Mode Operation

I²C Host Operation

The generic clock (GCLK_SERCOMx_CORE) will continue to run in idle sleep mode. If the Run In Standby bit in the Control A register (CTRLA.RUNSTDBY) is '1', the GLK_SERCOMx_CORE will also run in Standby Sleep mode. Any interrupt can wake-up the device.

If CTRLA.RUNSTDBY=0, the GLK_SERCOMx_CORE will be disabled after any ongoing transaction is finished. Any interrupt can wake-up the device.

I²C Client Operation

Writing CTRLA.RUNSTDBY=1 will allow the Address Match interrupt to wake-up the device.

When CTRLA.RUNSTDBY=0, all receptions will be dropped.

32.7 Register Summary - I2C Client

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
		15:8								
		23:16	SEXTTOEN		SDAHOLD[1:0]					PINOUT
		31:24		LOWTOUT			SCLSM		SPEED[1:0]	
0x04	CTRLB	7:0								
		15:8	AMODE[1:0]					AACKEN	GCMD	SMEN
		23:16						ACKACT	CMD[1:0]	
		31:24								
0x08	CTRLC	7:0								
		15:8								
		23:16								
		31:24								DATA32B
0x0C ... 0x13	Reserved									
0x14	INTENCLR	7:0	ERROR					DRDY	AMATCH	PREC
0x15	Reserved									
0x16	INTENSET	7:0	ERROR					DRDY	AMATCH	PREC
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR					DRDY	AMATCH	PREC
0x19	Reserved									
0x1A	STATUS	7:0	CLKHOLD	LOWTOUT		SR	DIR	RXNACK	COLL	BUSERR
		15:8					LENERR		SEXTTOUT	
0x1C	SYNCBUSY	7:0							ENABLE	SWRST
		15:8								
		23:16								
		31:24								
0x20 ... 0x23	Reserved									
0x24	ADDR	7:0	ADDR[6:0]							GENCEN
		15:8						ADDR[9:7]		
		23:16	ADDRMASK[6:0]							
		31:24						ADDRMASK[9:7]		
0x28	DATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							

32.8 Register Description - I2C Client

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the “PAC Write-Protection” property in each individual register description.

Some registers are synchronized when read and/or written. Synchronization is denoted by the “Write-Synchronized” or the “Read-Synchronized” property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the “Enable-Protected” property in each individual register description.

32.8.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
		LOWTOUT			SCLSM		SPEED[1:0]	
Access		R/W			R/W		R/W	R/W
Reset		0			0		0	0

Bit	23	22	21	20	19	18	17	16
	SEXTTOEN		SDAHOLD[1:0]					PINOUT
Access	R/W		R/W	R/W				R/W
Reset	0		0	0				0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit 30 – LOWTOUT SCL Low Time-Out
This bit enables the SCL low time-out. If SCL is held low for 25 ms-35 ms, the client will release its clock hold, if enabled, and reset the internal state machine. Any interrupt flags set at the time of time-out will remain set.
This bit is not synchronized.

Value	Description
0	Time-out disabled.
1	Time-out enabled.

Bit 27 – SCLSM SCL Clock Stretch Mode
This bit controls when SCL will be stretched for software interaction.
This bit is not synchronized.

Value	Description
0	SCL stretch according to Figure 32-9
1	SCL stretch only after ACK bit according to Figure 32-10

Bits 25:24 – SPEED[1:0] Transfer Speed
These bits define bus speed.
These bits are not synchronized.

Value	Description
0x0	Standard-mode (Sm) up to 100 kHz and Fast-mode (Fm) up to 400 kHz
0x1	Fast-mode Plus (Fm+) up to 1 MHz
0x2	Reserved
0x3	Reserved

Bit 23 – SEXTTOEN Client SCL Low Extend Time-Out
This bit enables the client SCL low extend time-out. If SCL is cumulatively held low for greater than 25 ms from the initial START to a STOP, the client will release its clock hold if enabled and reset the internal state machine. Any interrupt flags set at the time of time-out will remain set. If the address was recognized, PREC will be set when a STOP is received.

This bit is not synchronized.

Value	Description
0	Time-out disabled
1	Time-out enabled

Bits 21:20 – SDAHOLD[1:0] SDA Hold Time

These bits define the SDA hold time with respect to the negative edge of SCL.
These bits are not synchronized.

Value	Name	Description
0x0	DIS	Disabled
0x1	75	50-100ns hold time
0x2	450	300-600ns hold time
0x3	600	400-800ns hold time

Bit 16 – PINOUT Pin Usage

This bit sets the pin usage to either two- or four-wire operation:
This bit is not synchronized.

Value	Description
0	4-wire operation disabled
1	4-wire operation enabled

Bit 7 – RUNSTDBY Run in Standby

This bit defines the functionality in standby sleep mode.
This bit is not synchronized.

Value	Description
0	Disabled – All reception is dropped.
1	Wake on address match, if enabled.

Bits 4:2 – MODE[2:0] Operating Mode

These bits must be written to 0x04 to select the I^2C client serial communication interface of the SERCOM.
These bits are not synchronized.

Bit 1 – ENABLE Enable

Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately and the Enable Synchronization Busy bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.
This bit is not enable-protected.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled.

Bit 0 – SWRST Software Reset

Writing '0' to this bit has no effect.
Writing '1' to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.
Writing '1' to CTRLA.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded. Any register write access during the ongoing reset will result in an APB error. Reading any register will return the reset value of the register.
Due to synchronization, there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the reset is complete.
This bit is not enable-protected.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

32.8.2 Control B

Name: CTRLB
Offset: 0x04
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						ACKACT	CMD[1:0]	
Access						R/W	W	W
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
	AMODE[1:0]					AACKEN	GCMD	SMEN
Access	R/W	R/W				R/W	R/W	R/W
Reset	0	0				0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 18 – ACKACT Acknowledge Action

This bit defines the client's acknowledge behavior after an address or data byte is received from the host.
The acknowledge action is executed when a command is written to the CMD bits. If smart mode is enabled (CTRLB.SMEN=1), the acknowledge action is performed when the DATA register is read.
ACKACT shall not be updated more than once between each peripheral interrupts request.
This bit is not enable-protected.

Value	Description
0	Send ACK
1	Send NACK

Bits 17:16 – CMD[1:0] Command

This bit field triggers the client operation as the below. The CMD bits are strobe bits, and always read as zero.
The operation is dependent on the client interrupt flags, INTFLAG.DRDY and INTFLAG.AMATCH, in addition to STATUS.DIR.
All interrupt flags (INTFLAG.DRDY, INTFLAG.AMATCH and INTFLAG.PREC) are automatically cleared when a command is given.
This bit is not enable-protected.

Table 32-3. Command Description

CMD[1:0]	DIR	Action
0x0	X	(No action)
0x1	X	(Reserved)
0x2	Used to complete a transaction in response to a data interrupt (DRDY)	
	0 (Host write)	Execute acknowledge action succeeded by waiting for any start (S/Sr) condition
	1 (Host read)	Wait for any start (S/Sr) condition

.....continued		
CMD[1:0]	**DIR**	**Action**
0x3	Used in response to an address interrupt (AMATCH)	
	0 (Host write)	Execute acknowledge action succeeded by reception of next byte
	1 (Host read)	Execute acknowledge action succeeded by client data interrupt
	Used in response to a data interrupt (DRDY)	
	0 (Host write)	Execute acknowledge action succeeded by reception of next byte
	1 (Host read)	Execute a byte read operation followed by ACK/NACK reception

Bits 15:14 – AMODE[1:0] Address Mode

These bits set the addressing mode.

Value	Name	Description
0x0	MASK	The client responds to the address written in ADDR.ADDR masked by the value in ADDR.ADDRMASK.
0x1	2_ADDRS	The client responds to the two unique addresses in ADDR.ADDR and ADDR.ADDRMASK.
0x2	RANGE	The client responds to the range of addresses between and including ADDR.ADDR and ADDR.ADDRMASK. ADDR.ADDR is the upper limit.
0x3	—	Reserved.

Bit 10 – AACKEN Automatic Acknowledge Enable

This bit enables the address to be automatically acknowledged if there is an address match.

Value	Description
0	Automatic acknowledge is disabled.
1	Automatic acknowledge is enabled.

Bit 9 – GCMD PMBus Group Command

This bit enables PMBus group command support. When enabled, the Stop Received interrupt flag (INTFLAG.PREC) will be set when a STOP condition is detected if the client has been addressed since the last STOP condition on the bus.

Value	Description
0	Group command is disabled.
1	Group command is enabled.

Bit 8 – SMEN Smart Mode Enable

When smart mode is enabled, data is acknowledged automatically when DATA.DATA is read.

Value	Description
0	Smart mode is disabled.
1	Smart mode is enabled.

32.8.3 Control C

Name: CTRLC
Offset: 0x08
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
								DATA32B
Access								R/W
Reset								0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 24 – DATA32B Data 32 Bit

This bit enables 32-bit data writes and reads to/from the DATA register.

Value	Description
0	Data transaction to/from DATA are 8-bit in size
1	Data transaction to/from DATA are 32-bit in size

32.8.4 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x14
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Bit	7	6	5	4	3	2	1	0
	ERROR					DRDY	AMATCH	PREC
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

Bit 7 – ERROR Error Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

Bit 2 – DRDY Data Ready Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Data Ready bit, which disables the Data Ready interrupt.

Value	Description
0	The Data Ready interrupt is disabled.
1	The Data Ready interrupt is enabled.

Bit 1 – AMATCH Address Match Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Address Match Interrupt Enable bit, which disables the Address Match interrupt.

Value	Description
0	The Address Match interrupt is disabled.
1	The Address Match interrupt is enabled.

Bit 0 – PREC Stop Received Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Stop Received Interrupt Enable bit, which disables the Stop Received interrupt.

Value	Description
0	The Stop Received interrupt is disabled.
1	The Stop Received interrupt is enabled.

32.8.5 Interrupt Enable Set

Name: INTENSET
Offset: 0x16
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
	ERROR					DRDY	AMATCH	PREC
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

Bit 7 – ERROR Error Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

Bit 2 – DRDY Data Ready Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Data Ready bit, which enables the Data Ready interrupt.

Value	Description
0	The Data Ready interrupt is disabled.
1	The Data Ready interrupt is enabled.

Bit 1 – AMATCH Address Match Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Address Match Interrupt Enable bit, which enables the Address Match interrupt.

Value	Description
0	The Address Match interrupt is disabled.
1	The Address Match interrupt is enabled.

Bit 0 – PREC Stop Received Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Stop Received Interrupt Enable bit, which enables the Stop Received interrupt.

Value	Description
0	The Stop Received interrupt is disabled.
1	The Stop Received interrupt is enabled.

32.8.6 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x18
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ERROR					DRDY	AMATCH	PREC
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

Bit 7 – ERROR Error
This bit is set when any error is detected. Errors that will set this flag have corresponding status flags in the STATUS register. The corresponding bits in STATUS are SEXTTOUT, LOWTOUT, COLL and BUSERR.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the flag.

Bit 2 – DRDY Data Ready
This flag is set when a I^2C client byte transmission is successfully completed.
The flag is cleared by hardware when either:
- Writing to the DATA register.
- Reading the DATA register with Smart mode enabled.
- Writing a valid command to the CMD register.

Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Data Ready Interrupt flag.

Bit 1 – AMATCH Address Match
This flag is set when the I^2C client address match logic detects that a valid address has been received.
The flag is cleared by hardware when CTRL.CMD is written.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Address Match Interrupt flag. When cleared, an ACK/NACK will be sent according to CTRLB.ACKACT.

Bit 0 – PREC Stop Received
This flag is set when a Stop condition is detected for a transaction being processed. A Stop condition detected between a bus host and another client will not set this flag, unless the PMBus Group Command is enabled in the Control B register (CTRLB.GCMD=1).
This flag is cleared by hardware after a command is issued on the next address match.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Stop Received Interrupt flag.

32.8.7 Status

Name: STATUS
Offset: 0x1A
Reset: 0x0000
Property: -

Bit	15	14	13	12	11	10	9	8
					LENERR		SEXTTOUT	
Access					R/W		R/W	
Reset					0		0	

Bit	7	6	5	4	3	2	1	0
	CLKHOLD	LOWTOUT		SR	DIR	RXNACK	COLL	BUSERR
Access	R	R/W		R	R	R	R/W	R/W
Reset	0	0		0	0	0	0	0

Bit 11 – LENERR Transaction Length Error
This bit is set when the length counter is enabled (LENGTH.LENEN) and a STOP or repeated START is received before or after the length in LENGTH.LEN is reached.
This bit is cleared automatically if responding to a new start condition with ACK or NACK (write 3 to CTRLB.CMD) or when INTFLAG.AMATCH is cleared.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the status.

Value	Description
0	No length error has occurred.
1	Length error has occurred.

Bit 9 – SEXTTOUT Client SCL Low Extend Time-Out
This bit is set if a client SCL low extend time-out occurs.
This bit is cleared automatically if responding to a new start condition with ACK or NACK (write 3 to CTRLB.CMD) or when INTFLAG.AMATCH is cleared.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the status.

Value	Description
0	No SCL low extend time-out has occurred.
1	SCL low extend time-out has occurred.

Bit 7 – CLKHOLD Clock Hold
The client Clock Hold bit (STATUS.CLKHOLD) is set when the client is holding the SCL line low, stretching the I2C clock. Software must consider this bit a read-only status flag that is set when INTFLAG.DRDY or INTFLAG.AMATCH is set.
This bit is automatically cleared when the corresponding interrupt is also cleared.

Bit 6 – LOWTOUT SCL Low Time-out
This bit is set if an SCL low time-out occurs.
This bit is cleared automatically if responding to a new start condition with ACK or NACK (write 3 to CTRLB.CMD) or when INTFLAG.AMATCH is cleared.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the status.

Value	Description
0	No SCL low time-out has occurred.
1	SCL low time-out has occurred.

Bit 4 – SR Repeated Start
When INTFLAG.AMATCH is raised due to an address match, SR indicates a repeated start or start condition.

This flag is only valid while the INTFLAG.AMATCH flag is one.

Value	Description
0	Start condition on last address match
1	Repeated start condition on last address match

Bit 3 – DIR Read / Write Direction

The Read/Write Direction (STATUS.DIR) bit stores the direction of the last address packet received from a host .

Value	Description
0	Host write operation is in progress.
1	Host read operation is in progress.

Bit 2 – RXNACK Received Not Acknowledge

This bit indicates whether the last data packet sent was acknowledged or not.

Value	Description
0	Host responded with ACK.
1	Host responded with NACK.

Bit 1 – COLL Transmit Collision

If set, the I2C client was not able to transmit a high data or NACK bit, the I2C client will immediately release the SDA and SCL lines and wait for the next packet addressed to it.
This flag is intended for the SMBus address resolution protocol (ARP). A detected collision in non-ARP situations indicates that there has been a protocol violation, and must be treated as a bus error.
Note: This status will not trigger any interrupt, and must be checked by software to verify that the data were sent correctly. This bit is cleared automatically if responding to an address match with an ACK or a NACK (writing 0x3 to CTRLB.CMD), or INTFLAG.AMATCH is cleared.

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the status.

Value	Description
0	No collision detected on last data byte sent.
1	Collision detected on last data byte sent.

Bit 0 – BUSERR Bus Error

The Bus Error bit (STATUS.BUSERR) indicates that an illegal bus condition has occurred on the bus, regardless of bus ownership. An illegal bus condition is detected if a protocol violating start, repeated start or stop is detected on the I2C bus lines. A start condition directly followed by a stop condition is one example of a protocol violation. If a time-out occurs during a frame, this is also considered a protocol violation, and will set STATUS.BUSERR.
This bit is cleared automatically if responding to an address match with an ACK or a NACK (writing 0x3 to CTRLB.CMD) or INTFLAG.AMATCH is cleared.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the status.

Value	Description
0	No bus error detected.
1	Bus error detected.

32.8.8 Synchronization Busy

Name: SYNCBUSY
Offset: 0x1C
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							ENABLE	SWRST
Access							R	R
Reset							0	0

Bit 1 – ENABLE SERCOM Enable Synchronization Busy
Enabling and disabling the SERCOM (CTRLA.ENABLE) requires synchronization. Ongoing synchronization is indicated by SYNCBUSY.ENABLE = 1 until synchronization is complete.

Value	Description
0	Enable synchronization is not busy.
1	Enable synchronization is busy.

Bit 0 – SWRST Software Reset Synchronization Busy
Resetting the SERCOM (CTRLA.SWRST) requires synchronization. Ongoing synchronization is indicated by SYNCBUSY.SWRST = 1 until synchronization is complete.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

32.8.9 Address

Name: ADDR
Offset: 0x24
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
						ADDRMASK[9:7]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	23	22	21	20	19	18	17	16
	ADDRMASK[6:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	

Bit	15	14	13	12	11	10	9	8
						ADDR[9:7]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[6:0]							GENCEN
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 26:17 – ADDRMASK[9:0] Address Mask
These bits act as a second address match register, an address mask register or the lower limit of an address range, depending on the CTRLB.AMODE setting.

Bits 10:1 – ADDR[9:0] Address
These bits contain the I^2C client address used by the client address match logic to determine if a host has addressed the client.
When using 7-bit addressing, the client address is represented by ADDR[6:0].
When the address match logic detects a match, INTFLAG.AMATCH is set and STATUS.DIR is updated to indicate whether it is a read or a write transaction.

Bit 0 – GENCEN General Call Address Enable
A general call address is an address consisting of all-zeroes, including the direction bit (host write).

Value	Description
0	General call address recognition disabled.
1	General call address recognition enabled.

32.8.10 Data

Name: DATA
Offset: 0x28
Reset: 0x00000000
Property: Read/Write

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – DATA[31:0] Data

The client data register I/O location (DATA.DATA) provides access to the host transmit and receive data buffers. Reading valid data or writing data to be transmitted can be successfully done only when SCL is held low by the client (STATUS.CLKHOLD is set). An exception occurs when reading the last data byte after the stop condition has been received.

Accessing DATA.DATA auto-triggers I^2C bus operations. The operation performed depends on the state of CTRLB.ACKACT, CTRLB.SMEN and the type of access (read/write).

When CTRLC.DATA32B=1, read and write transactions from/to the DATA register are 32 bit in size. Otherwise, reads and writes are 8 bit.

32.9 Register Summary - I2C Host

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
		15:8								
		23:16	SEXTTOEN	MEXTTOEN	SDAHOLD[1:0]					PINOUT
		31:24		LOWTOUT	INACTOUT[1:0]		SCLSM		SPEED[1:0]	
0x04	CTRLB	7:0								
		15:8							QCEN	SMEN
		23:16						ACKACT	CMD[1:0]	
		31:24								
0x08 ... 0x0B	Reserved									
0x0C	BAUD	7:0	BAUD[7:0]							
		15:8	BAUDLOW[7:0]							
		23:16								
		31:24								
0x10 ... 0x13	Reserved									
0x14	INTENCLR	7:0	ERROR						SB	MB
0x15	Reserved									
0x16	INTENSET	7:0	ERROR						SB	MB
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR						SB	MB
0x19	Reserved									
0x1A	STATUS	7:0	CLKHOLD	LOWTOUT	BUSSTATE[1:0]			RXNACK	ARBLOST	BUSERR
		15:8						LENERR	SEXTTOUT	MEXTTOUT
0x1C	SYNCBUSY	7:0						SYSOP	ENABLE	SWRST
		15:8								
		23:16								
		31:24								
0x20 ... 0x23	Reserved									
0x24	ADDR	7:0	ADDR[7:0]							
		15:8	TENBITEN		LENEN			ADDR[10:8]		
		23:16	LEN[7:0]							
		31:24								
0x28	DATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							
0x2C ... 0x2F	Reserved									
0x30	DBGCTRL	7:0								DBGSTOP

32.10 Register Description – I²C Host

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the “PAC Write-Protection” property in each individual register description.

Some registers are synchronized when read and/or written. Synchronization is denoted by the “Write-Synchronized” or the “Read-Synchronized” property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the “Enable-Protected” property in each individual register description.

32.10.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
		LOWTOUT	INACTOUT[1:0]		SCLSM		SPEED[1:0]	
Access		R/W	R/W	R/W	R/W		R/W	R/W
Reset		0	0	0	0		0	0

Bit	23	22	21	20	19	18	17	16
	SEXTTOEN	MEXTTOEN	SDAHOLD[1:0]					PINOUT
Access	R/W	R/W	R/W	R/W				R/W
Reset	0	0	0	0				0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit 30 – LOWTOUT SCL Low Time-Out

This bit enables the SCL low time-out. If SCL is held low for 25ms-35ms, the host will release its clock hold, if enabled, and complete the current transaction. A stop condition will automatically be transmitted. INTFLAG.SB or INTFLAG.MB will be set as normal, but the clock hold will be released. The STATUS.LOWTOUT and STATUS.BUSERR status bits will be set.
This bit is not synchronized.

Value	Description
0	Time-out disabled.
1	Time-out enabled.

Bits 29:28 – INACTOUT[1:0] Inactive Time-Out

If the inactive bus time-out is enabled and the bus is inactive for longer than the time-out setting, the bus state logic will be set to idle. An inactive bus arise when either an I^2C host or client is holding the SCL low.
Enabling this option is necessary for SMBus compatibility, but can also be used in a non-SMBus set-up.
Calculated time-out periods are based on a 100kHz baud rate.
These bits are not synchronized.

Value	Name	Description
0x0	DIS	Disabled
0x1	55US	5-6 SCL cycle time-out (50-60µs)
0x2	105US	10-11 SCL cycle time-out (100-110µs)
0x3	205US	20-21 SCL cycle time-out (200-210µs)

Bit 27 – SCLSM SCL Clock Stretch Mode

This bit controls when SCL will be stretched for software interaction.
This bit is not synchronized.

Value	Description
0	SCL stretch according to Figure 32-5
1	SCL stretch only after ACK bit, Figure 32-6

Bits 25:24 – SPEED[1:0] Transfer Speed

These bits define bus speed.
These bits are not synchronized.

Value	Description
0x0	Standard-mode (Sm) up to 100 kHz and Fast-mode (Fm) up to 400 kHz
0x1	Fast-mode Plus (Fm+) up to 1 MHz
0x2	Reserved
0x3	Reserved

Bit 23 – SEXTTOEN Client SCL Low Extend Time-Out

This bit enables the client SCL low extend time-out. If SCL is cumulatively held low for greater than 25ms from the initial START to a STOP, the host will release its clock hold if enabled, and complete the current transaction. A STOP will automatically be transmitted.
SB or MB will be set as normal, but CLKHOLD will be release. The MEXTTOUT and BUSERR status bits will be set.
This bit is not synchronized.

Value	Description
0	Time-out disabled
1	Time-out enabled

Bit 22 – MEXTTOEN Host SCL Low Extend Time-Out

This bit enables the host SCL low extend time-out. If SCL is cumulatively held low for greater than 10ms from START-to-ACK, ACK-to-ACK, or ACK-to-STOP the host will release its clock hold if enabled, and complete the current transaction. A STOP will automatically be transmitted.
SB or MB will be set as normal, but CLKHOLD will be released. The MEXTTOUT and BUSERR status bits will be set.
This bit is not synchronized.

Value	Description
0	Time-out disabled
1	Time-out enabled

Bits 21:20 – SDAHOLD[1:0] SDA Hold Time

These bits define the SDA hold time with respect to the negative edge of SCL.
These bits are not synchronized.

Value	Name	Description
0x0	DIS	Disabled
0x1	75NS	50-100ns hold time
0x2	450NS	300-600ns hold time
0x3	600NS	400-800ns hold time

Bit 16 – PINOUT Pin Usage

This bit set the pin usage to either two- or four-wire operation:
This bit is not synchronized.

Value	Description
0	4-wire operation disabled.
1	4-wire operation enabled.

Bit 7 – RUNSTDBY Run in Standby

This bit defines the functionality in standby sleep mode.
This bit is not synchronized.

Value	Description
0	GCLK_SERCOMx_CORE is disabled and the I^2C host will not operate in standby sleep mode.
1	GCLK_SERCOMx_CORE is enabled in all sleep modes.

Bits 4:2 – MODE[2:0] Operating Mode

These bits must be written to 0x5 to select the I^2C host serial communication interface of the SERCOM.
These bits are not synchronized.

Bit 1 – ENABLE Enable

Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled.
The value written to CTRL.ENABLE will read back immediately and the Synchronization Enable Busy bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.
This bit is not enable-protected.

Value	Description
0	The peripheral is disabled or being disabled.
1	The peripheral is enabled.

Bit 0 – SWRST Software Reset

Writing '0' to this bit has no effect.
Writing '1' to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.
Writing '1' to CTRLA.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded. Any register write access during the ongoing reset will result in an APB error. Reading any register will return the reset value of the register.
Due to synchronization there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the reset is complete.
This bit is not enable-protected.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

32.10.2 Control B

Name: CTRLB
Offset: 0x04
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						ACKACT	CMD[1:0]	
Access						R/W	W	W
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
							QCEN	SMEN
Access							R/W	R/W
Reset							0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 18 – ACKACT Acknowledge Action

This bit defines the I^2C Host's acknowledge behavior after a data byte is received from the I^2C Client. The acknowledge action is executed when a command is written to CTRLB.CMD, or if Smart mode is enabled (CTRLB.SMEN is written to one), when DATA.DATA is read.
This bit is not enable-protected.
This bit is not write-synchronized.

Value	Description
0	Send ACK.
1	Send NACK.

Bits 17:16 – CMD[1:0] Command

Writing these bits triggers a Host operation as described below. The CMD bits are strobe bits, and always read as zero. The acknowledge action is only valid in Host Read mode. In Host Write mode, a command will only result in a repeated Start or Stop condition. The CTRLB.ACKACT bit and the CMD bits can be written at the same time, and then the acknowledge action will be updated before the command is triggered.
Commands can only be issued when either the Client on Bus Interrupt flag (INTFLAG.SB) or Host on Bus Interrupt flag (INTFLAG.MB) is '1'.
If CMD 0x1 is issued, a repeated start will be issued followed by the transmission of the current address in ADDR.ADDR. If another address is desired, ADDR.ADDR must be written instead of the CMD bits. This will trigger a repeated start followed by transmission of the new address.
Issuing a command will set the System Operation bit in the Synchronization Busy register (SYNCBUSY.SYSOP).

Table 32-4. Command Description

CMD[1:0]	Direction	Action
0x0	X	(No action)
0x1	X	Execute acknowledge action succeeded by repeated Start
0x2	0 (Write)	No operation
	1 (Read)	Execute acknowledge action succeeded by a byte read operation
0x3	X	Execute acknowledge action succeeded by issuing a Stop condition

These bits are not enable-protected.

Bit 9 – QCEN Quick Command Enable

This bit is not write-synchronized.

Value	Description
0	Quick Command is disabled.
1	Quick Command is enabled.

Bit 8 – SMEN Smart Mode Enable

When Smart mode is enabled, acknowledge action is sent when DATA.DATA is read.

This bit is not write-synchronized.

Value	Description
0	Smart mode is disabled.
1	Smart mode is enabled.

32.10.3 Baud Rate

Name: BAUD
Offset: 0x0C
Reset: 0x0000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BAUDLOW[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:8 – BAUDLOW[7:0] Host Baud Rate Low

If this bit field is non-zero, the SCL low time will be described by the value written.
For more details on how to calculate the frequency, see *Clock Generation – Baud-Rate Generator* from Related Links.

Bits 7:0 – BAUD[7:0] Host Baud Rate

This bit field is used to derive the SCL high time if BAUD.BAUDLOW is non-zero. If BAUD.BAUDLOW is zero, BAUD will be used to generate both high and low periods of the SCL.
For more details on how to calculate the frequency, see *Clock Generation – Baud-Rate Generator* from Related Links.

Related Links

29.6.2.3. Clock Generation – Baud-Rate Generator

32.10.4 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x14
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Bit	7	6	5	4	3	2	1	0
	ERROR						SB	MB
Access	R/W						R/W	R/W
Reset	0						0	0

Bit 7 – ERROR Error Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

Bit 1 – SB Client on Bus Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Client on Bus Interrupt Enable bit, which disables the Client on Bus interrupt.

Value	Description
0	The Client on Bus interrupt is disabled.
1	The Client on Bus interrupt is enabled.

Bit 0 – MB Host on Bus Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the Host on Bus Interrupt Enable bit, which disables the Host on Bus interrupt.

Value	Description
0	The Host on Bus interrupt is disabled.
1	The Host on Bus interrupt is enabled.

32.10.5 Interrupt Enable Set

Name: INTENSET
Offset: 0x16
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
	ERROR						SB	MB
Access	R/W						R/W	R/W
Reset	0						0	0

Bit 7 – ERROR Error Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	Error interrupt is disabled.
1	Error interrupt is enabled.

Bit 1 – SB Client on Bus Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Client on Bus Interrupt Enable bit, which enables the Client on Bus interrupt.

Value	Description
0	The Client on Bus interrupt is disabled.
1	The Client on Bus interrupt is enabled.

Bit 0 – MB Host on Bus Interrupt Enable
Writing '0' to this bit has no effect.
Writing '1' to this bit will set the Host on Bus Interrupt Enable bit, which enables the Host on Bus interrupt.

Value	Description
0	The Host on Bus interrupt is disabled.
1	The Host on Bus interrupt is enabled.

32.10.6 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x18
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ERROR						SB	MB
Access	R/W						R/W	R/W
Reset	0						0	0

Bit 7 – ERROR Error

This flag is cleared by writing '1' to it.
This bit is set when any error is detected. Errors that will set this flag have corresponding status bits in the STATUS register. These status bits are LENERR, SEXTTOUT, MEXTTOUT, LOWTOUT, ARBLOST, and BUSERR.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear the flag.

Bit 1 – SB Client on Bus

The Client on Bus flag (SB) is set when a byte is successfully received in Host Read mode, for example, no arbitration lost or bus error occurred during the operation. When this flag is set, the host forces the SCL line low, stretching the I^2C clock period. The SCL line will be released and SB will be cleared on one of the following actions:

- Writing to ADDR.ADDR
- Writing to DATA.DATA
- Reading DATA.DATA when Smart mode is enabled (CTRLB.SMEN)
- Writing a valid command to CTRLB.CMD

Writing '1' to this bit location will clear the SB flag. The transaction will not continue or be terminated until one of the above actions is performed.
Writing '0' to this bit has no effect.

Bit 0 – MB Host on Bus

This flag is set when a byte is transmitted in Host Write mode. The flag is set regardless of the occurrence of a bus error or an Arbitration Lost condition. MB is also set when arbitration is lost during sending of NACK in Host Read mode, or when issuing a Start condition if the bus state is unknown. When this flag is set and arbitration is not lost, the host forces the SCL line low, stretching the I^2C clock period. The SCL line will be released and MB will be cleared on one of the following actions:

- Writing to ADDR.ADDR
- Writing to DATA.DATA
- Reading DATA.DATA when Smart mode is enabled (CTRLB.SMEN)
- Writing a valid command to CTRLB.CMD

Writing '1' to this bit location will clear the MB flag. The transaction will not continue or be terminated until one of the above actions is performed.
Writing '0' to this bit has no effect.

32.10.7 Status

Name: STATUS
Offset: 0x1A
Reset: 0x0000
Property: Write-Synchronized

Bit	15	14	13	12	11	10	9	8
						LENERR	SEXTTOUT	MEXTTOUT
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
	CLKHOLD	LOWTOUT	BUSSTATE[1:0]			RXNACK	ARBLOST	BUSERR
Access	R	R/W	R/W	R/W		R	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit 10 – LENERR Transaction Length Error
This bit is set when automatic length is used for a DMA transaction and the client sends a NACK before ADDR.LEN bytes have been written by the host.
Writing '1' to this bit location will clear STATUS.LENERR. This flag is automatically cleared when writing to the ADDR register.
Writing '0' to this bit has no effect.
This bit is not write-synchronized.

Bit 9 – SEXTTOUT Client SCL Low Extend Time-Out
This bit is set if a client SCL low extend time-out occurs.
This bit is automatically cleared when writing to the ADDR register.
Writing '1' to this bit location will clear SEXTTOUT. Normal use of the I^2C interface does not require the SEXTTOUT flag to be cleared by this method.
Writing '0' to this bit has no effect.
This bit is not write-synchronized.

Bit 8 – MEXTTOUT Host SCL Low Extend Time-Out
This bit is set if a Host SCL low time-out occurs.
Writing '1' to this bit location will clear STATUS.MEXTTOUT. This flag is automatically cleared when writing to the ADDR register.
Writing '0' to this bit has no effect.
This bit is not write-synchronized.

Bit 7 – CLKHOLD Clock Hold
This bit is set when the host is holding the SCL line low, stretching the I^2C clock. Software must consider this bit when INTFLAG.SB or INTFLAG.MB is set.
This bit is cleared when the corresponding Interrupt flag is cleared and the next operation is given.
Writing '0' to this bit has no effect.
Writing '1' to this bit has no effect.
This bit is not write-synchronized.

Bit 6 – LOWTOUT SCL Low Time-Out
This bit is set if an SCL low time-out occurs.
Writing '1' to this bit location will clear this bit. This flag is automatically cleared when writing to the ADDR register.
Writing '0' to this bit has no effect.
This bit is not write-synchronized.

Bits 5:4 – BUSSTATE[1:0] Bus State
These bits indicate the current I^2C Bus state.

When in UNKNOWN state, writing 0x1 to BUSSTATE forces the bus state into the IDLE state. The bus state cannot be forced into any other state.
Writing BUSSTATE to idle will set SYNCBUSY.SYSOP.

Value	Name	Description
0x0	UNKNOWN	The Bus state is unknown to the I^2C host and will wait for a Stop condition to be detected or wait to be forced into an Idle state by software
0x1	IDLE	The Bus state is waiting for a transaction to be initialized
0x2	OWNER	The I^2C host is the current owner of the bus
0x3	BUSY	Some other I^2C host owns the bus

Bit 2 – RXNACK Received Not Acknowledge

This bit indicates whether the last address or data packet sent was acknowledged or not.
Writing '0' to this bit has no effect.
Writing '1' to this bit has no effect.
This bit is not write-synchronized.

Value	Description
0	Client responded with ACK.
1	Client responded with NACK.

Bit 1 – ARBLOST Arbitration Lost

This bit is set if arbitration is lost while transmitting a high data bit or a NACK bit, or while issuing a Start or Repeated Start condition on the bus. The Host on Bus Interrupt flag (INTFLAG.MB) will be set when STATUS.ARBLOST is set.
Writing the ADDR.ADDR register will automatically clear STATUS.ARBLOST.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear it.
This bit is not write-synchronized.

Bit 0 – BUSERR Bus Error

This bit indicates that an illegal Bus condition has occurred on the bus, regardless of bus ownership. An illegal Bus condition is detected if a protocol violating start, repeated start or stop is detected on the I^2C bus lines. A Start condition directly followed by a Stop condition is one example of a protocol violation. If a time-out occurs during a frame, this is also considered a protocol violation, and will set BUSERR.
If the I^2C host is the bus owner at the time a bus error occurs, STATUS.ARBLOST and INTFLAG.MB will be set in addition to BUSERR.
Writing the ADDR.ADDR register will automatically clear the BUSERR flag.
Writing '0' to this bit has no effect.
Writing '1' to this bit will clear it.
This bit is not write-synchronized.

32.10.8 Synchronization Busy

Name: SYNCBUSY
Offset: 0x1C
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						SYSOP	ENABLE	SWRST
Access						R	R	R
Reset						0	0	0

Bit 2 – SYSOP System Operation Synchronization Busy

Value	Description
0	System operation synchronization is not busy.
1	System operation synchronization is busy.

Bit 1 – ENABLE SERCOM Enable Synchronization Busy

Enabling and disabling the SERCOM (CTRLA.ENABLE) requires synchronization. When written, the SYNCBUSY.ENABLE bit will be set until synchronization is complete.

Value	Description
0	Enable synchronization is not busy.
1	Enable synchronization is busy.

Bit 0 – SWRST Software Reset Synchronization Busy

Resetting the SERCOM (CTRLA.SWRST) requires synchronization. When written, the SYNCBUSY.SWRST bit will be set until synchronization is complete.

Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

Value	Description
0	SWRST synchronization is not busy.
1	SWRST synchronization is busy.

32.10.9 Address

Name: ADDR
Offset: 0x24
Reset: 0x0000
Property: Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	LEN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TENBITEN		LENEN			ADDR[10:8]		
Access	R/W		R/W			R/W	R/W	R/W
Reset	0		0			0	0	0

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

Bits 23:16 – LEN[7:0] Transaction Length
These bits define the transaction length of a DMA transaction from 0 to 255 bytes. The Transfer Length Enable (LENEN) bit must be written to '1' in order to use DMA.

Bit 15 – TENBITEN Ten Bit Addressing Enable
This bit enables 10-bit addressing. This bit can be written simultaneously with ADDR to indicate a 10-bit or 7-bit address transmission.

Value	Description
0	10-bit addressing disabled.
1	10-bit addressing enabled.

Bit 13 – LENEN Transfer Length Enable

Value	Description
0	Automatic transfer length disabled.
1	Automatic transfer length enabled.

Bits 10:0 – ADDR[10:0] Address
When ADDR is written, the consecutive operation will depend on the bus state:
UNKNOWN: INTFLAG.MB and STATUS.BUSERR are set, and the operation is terminated.
BUSY: The I^2C host will await further operation until the bus becomes IDLE.
IDLE: The I^2C host will issue a start condition followed by the address written in ADDR. If the address is acknowledged, SCL is forced and held low, and STATUS.CLKHOLD and INTFLAG.MB are set.
OWNER: A repeated start sequence will be performed. If the previous transaction was a read, the acknowledge action is sent before the repeated start bus condition is issued on the bus. Writing ADDR to issue a repeated start is performed while INTFLAG.MB or INTFLAG.SB is set.
STATUS.BUSERR, STATUS.ARBLOST, INTFLAG.MB and INTFLAG.SB will be cleared when ADDR is written.
The ADDR register can be read at any time without interfering with ongoing bus activity, as a read access does not trigger the host logic to perform any bus protocol related operations.
The I^2C host control logic uses bit 0 of ADDR as the bus protocol's read/write flag (R/W); 0 for write and 1 for read.

32.10.10 Data

Name: DATA
Offset: 0x28
Reset: 0x00000000
Property: Read/Write

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – DATA[31:0] Data

The host data register I/O location (DATA) provides access to the host transmit and receive data buffers. Reading valid data or writing data to be transmitted can be successfully done only when SCL is held low by the host (STATUS.CLKHOLD is set). An exception is reading the last data byte after the stop condition has been sent. Accessing DATA.DATA auto-triggers I^2C bus operations. The operation performed depends on the state of CTRLB.ACKACT, CTRLB.SMEN and the type of access (read/write).

When CTRLC.DATA32B=1, read and write transactions from/to the DATA register are 32 bit in size. Otherwise, reads and writes are 8 bit.

32.10.11 Debug Control

Name: DBGCTRL
Offset: 0x30
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								DBGSTOP
Access								R/W
Reset								0

Bit 0 – DBGSTOP Debug Stop Mode
This bit controls functionality when the CPU is halted by an external debugger.

Value	Description
0	The baud-rate generator continues normal operation when the CPU is halted by an external debugger.
1	The baud-rate generator is halted when the CPU is halted by an external debugger.

33. Quad Serial Peripheral Interface (QSPI)

33.1 Overview

The Quad SPI Interface (QSPI) circuit is a synchronous serial data link that provides communication with external devices in Host mode.

The QSPI can be used in “SPI mode” to interface serial peripherals, such as ADCs, DACs, LCD controllers and sensors, or in “Serial Memory Mode” to interface serial Flash memories.

The QSPI allows the system to execute code directly from a serial Flash memory (XIP) without code shadowing to SRAM. The serial Flash memory mapping is seen in the system as other memories (ROM, SRAM, DRAM, embedded Flash memories, etc.,).

With the support of the quad-SPI protocol, the QSPI allows the system to use high performance serial Flash memories which are small and inexpensive, in place of larger and more expensive parallel Flash memories.

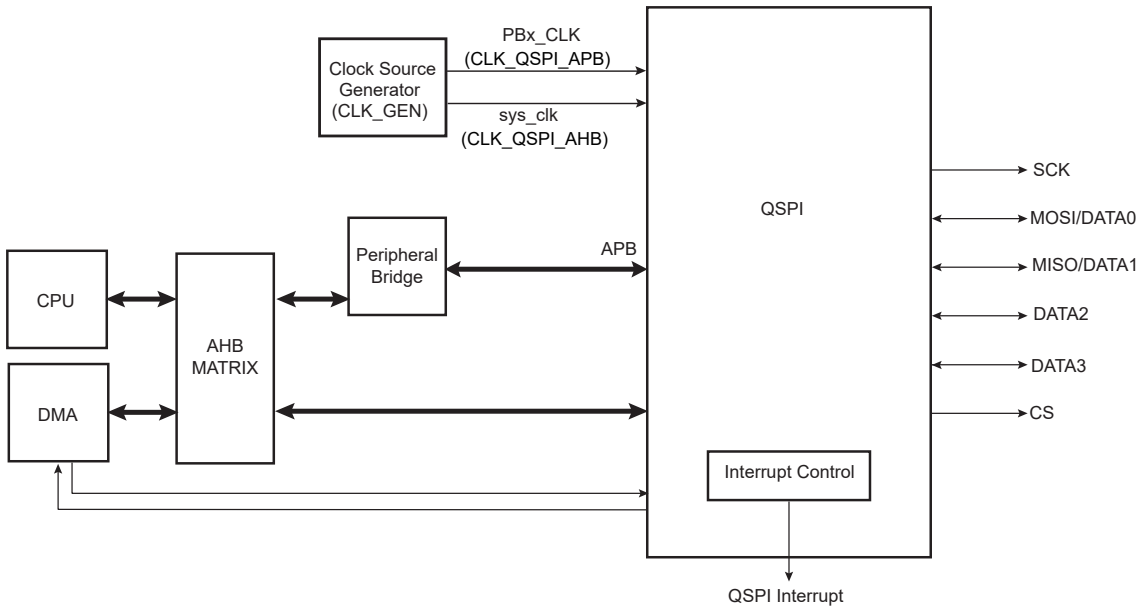
Note: Traditional Quad SPI Interface (QSPI) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client” respectively.

33.2 Features

- Host SPI Interface:
  - Programmable clock phase and clock polarity
  - Programmable transfer delays between consecutive transfers, between clock and data, between deactivation and activation of chip select (CS)
- SPI Mode:
  - To use serial peripherals, such as ADCs, DACs, LCD controllers, and sensors
  - 8-bit, 16-bit, or 32-bit programmable data length
- Serial Memory Mode:
  - To use serial Flash memories operating in single-bit SPI, Dual SPI and Quad SPI
  - Supports “execute in place” (XIP). The system can execute code directly from a Serial Flash memory
  - Flexible instruction register, to be compatible with all serial Flash memories
  - 32-bit Address mode (default is 24-bit address) to support serial Flash memories larger than 128 Mbit
  - Continuous Read mode
  - Scrambling/Unscrambling “On-the-Fly”
  - Double data rate support
- Connection to DMA Channel Capabilities Optimizes Data Transfers
  - One channel for the receiver and one channel for the transmitter
- Register Write Protection

33.3 Block Diagram

Figure 33-1. QSPI Block Diagram



33.4 Signal Description

Table 33-1. Quad-SPI Signals

Signal	Description	Type
SCK	Serial Clock	Output
CS	Chip Select	Output
MOSI(DATA0)	Data Output (Data Input Output 0)	Output (Input/Output)
MISO(DATA1)	Data Input (Data Input Output 1)	Input (Input/Output)
DATA2	Data Input Output 2	Input/Output
DATA3	Data Input Output 3	Input/Output

Notes:

1. MOSI and MISO are used for single-bit SPI operation.
2. DATA0-DATA1 are used for Dual SPI operation.
3. DATA0-DATA3 are used for Quad SPI operation.

See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links for details on the pin mapping for the QSPI peripheral.

Related Links

6. I/O Ports and Peripheral Pin Select (PPS)

33.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

33.5.1 I/O Lines

Using the QSPI I/O lines requires the I/O pins to be configured.

33.5.2 Power Management

The QSPI will continue to operate in any Sleep mode where the selected source clock is running. The QSPI interrupts can be used to wake up the device from sleep modes. See *Power Management Unit (PMU)* from Related Links for details on the different sleep modes.

Related Links

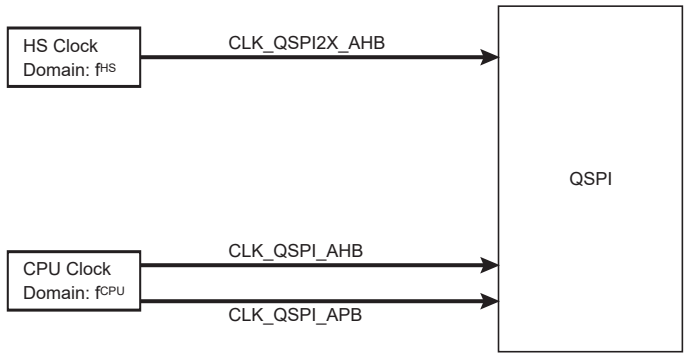
15. Power Management Unit (PMU)

33.5.3 Clocks

An AHB clock (CLK_QSPI_AHB) is required to clock the QSPI. This clock can be enabled and disabled in the CRU.

A FAST clock (CLK_QSPI2X_AHB) is required to clock the QSPI. This clock can be enabled and disabled in the CFGCON1 register, bit 29 (CFGCON1.QSPIDDRM). When using QSPI DDR mode, the System Clock (SYS_CLK) must be <= 48 MHz.

Figure 33-2. QSPI Clock Organization



Important: The CLK_QSPI2x_AHB must be 2 times faster to CLK_QSPI_AHB when the QSPI is operated in DDR mode. In SDR, the CLK_QSPI2x_AHB is not used.

CLK_QSPI_APB, CLK_QSPI_AHB and CLK_QSPI2X_AHB, respectively, are all synchronous but can be divided by a prescaler and may run even when the module clock is turned off.

33.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). Using the QSPI DMA requests requires the DMA Controller to be configured first.

Note: DMAC write access must be 32-bit aligned. If a single byte is to be written in a 32-bit word, the rest of the word must be filled with 'ones'.

33.5.5 Interrupts

The interrupt request lines are connected to the interrupt controller. Using the QSPI interrupts requires the interrupt controller to be configured first. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

33.5.6 Events

Not applicable.

33.5.7 Debug Operation

When the CPU is halted in debug mode the QSPI continues normal operation. If the QSPI is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

33.5.8 Register Access Protection

All registers with write-access are optionally write-protected by the peripheral access controller (PAC), except the following registers:

- Control A (CTRLA) register
- Transmit Data (TXDATA) register
- Interrupt Flag Status and Clear (INTFLAG) register
- Scrambling Key (SCRAMBKEY) register

PAC write-protection is denoted by the 'PAC Write-Protection' property in the register description.

Write-protection does not apply to accesses through an external debugger.

33.6 Functional Description

33.6.1 Principle of Operation

The QSPI is a high-speed synchronous data transfer interface. It allows high-speed communication between the device and peripheral or serial memory devices.

The QSPI operates as a host. It initiates and controls all data transactions.

When transmitting, the TXDATA register can be loaded with the next character to be transmitted during the current transmission.

When receiving, the data is transferred to the RXDATA register, and the receiver is ready for a new character.

33.6.2 Basic Operation

33.6.2.1 Initialization

After Power-On Reset, this peripheral is enabled .

33.6.2.2 Enabling, Disabling and Resetting

The peripheral is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE).

The peripheral is disabled by writing a '0' to CTRLA.ENABLE.

The peripheral is reset by writing a '1' to the Software Reset bit (CTRLA.SWRST).

33.6.3 Transfer Data Rate

By default, the QSPI module is enabled in single data rate mode. In this operating mode, the CLK_QSPI2X_AHB clock is not used and must be disabled.

The dual data rate operating mode is enabled by writing a '1' to the Double Data Rate Enable bit in the CFGCON1 register (CFGCON1.QSPIDDRM). This operating mode requires the CLK_QSPI2X_AHB clock and must be enabled before writing the DDREN bit.

33.6.4 Serial Clock Baud Rate

The QSPI Baud rate clock is generated by dividing the module clock (CLK_QSPI_AHB) by a value between 1 and 255.

This allows a maximum operating baud rate at up to Host Clock and a minimum operating baud rate of CLK_QSPI_AHB divided by 255.

33.6.5 Serial Clock Phase and Polarity

Four combinations of polarity and phase are available for data transfers. Writing the Clock Polarity bit in the QSPI Baud register (BAUD.CPOL) selects the polarity. The Clock Phase bit in the BAUD register programs the clock phase

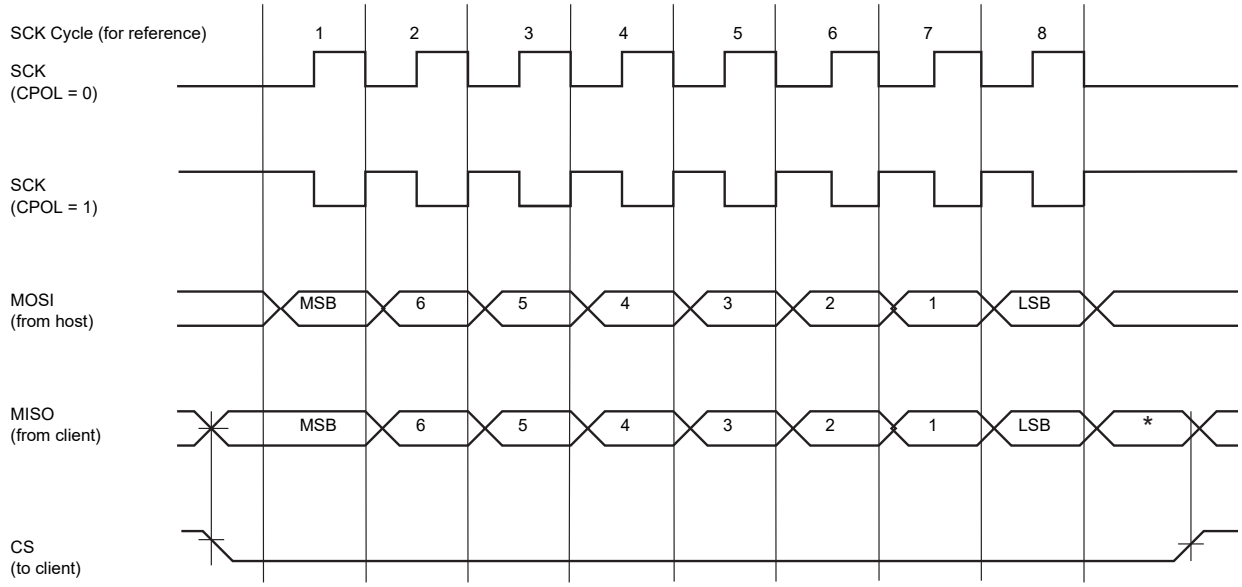
(BAUD.CPHA). These two parameters determine the edges of the clock signal on which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations

Note: The polarity/phase combinations are incompatible. Thus, the interfaced client must use the same parameter values to communicate.

Table 33-2. SPI Transfer Mode

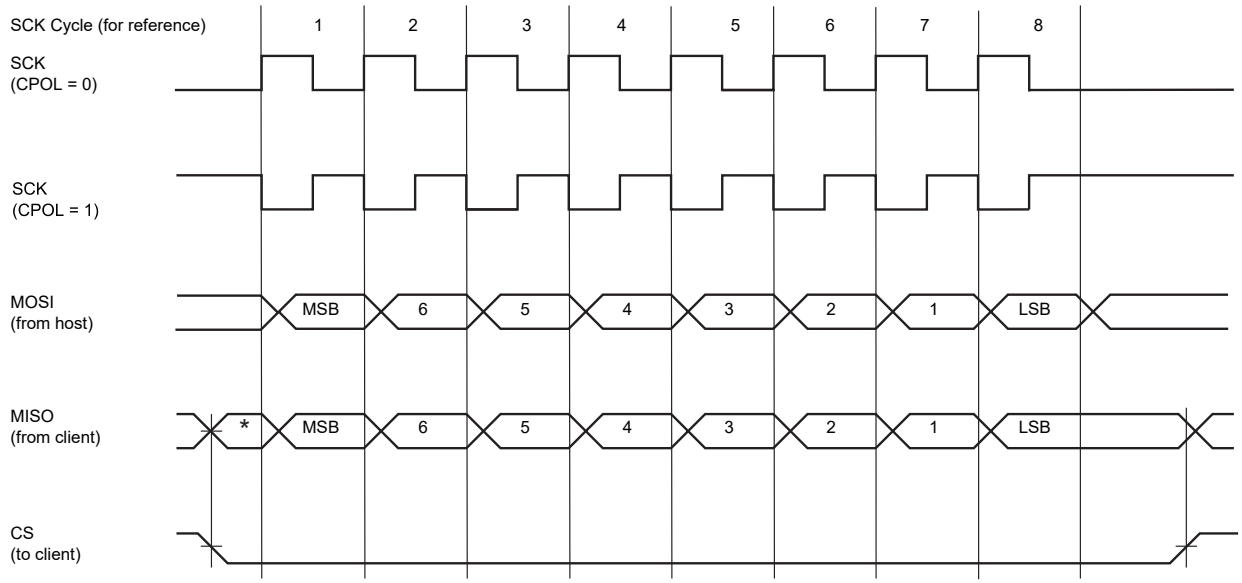
Clock Mode	BAUD.CPOL	BAUD.CPHA	Shift SCK Edge	Capture SCK Edge	SCK Inactive Level
0	0	0	Falling	Rising	Low
1	0	1	Rising	Falling	Low
2	1	0	Rising	Falling	High
3	1	1	Falling	Rising	High

Figure 33-3. QSPI Transfer Modes (BAUD.CPHA = 0, 8-bit transfer)



* Not defined, but normally MSB of previous character received

Figure 33-4. QSPI Transfer Modes (BAUD.CPHA = 1, 8-bit transfer)



* Not defined, but normally LSB of previous character received

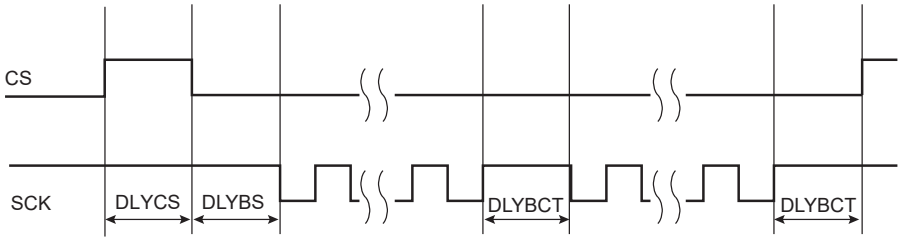
33.6.6 Transfer Delays

The QSPI supports several consecutive transfers while the chip select is active. Three delays can be programmed to modify the transfer waveforms:

- The delay between the inactivation and the activation of CS is programmed by writing the Minimum Inactive CS Delay bit field in the Control B register (CTRLB.DLYCS), allowing to tune the minimum time of CS at high level.
- The delay between consecutive transfers is programmed by writing the Delay Between Consecutive Transfers bit field in the Control B register (CTRLB.DLYBCT), allowing to insert a delay between two consecutive transfers. In Serial Memory mode, this delay is not programmable and DLYBCT settings are ignored.
- The delay before SCK is programmed by writing the Delay Before SCK bit field in the BAUD register (BAUD.DLYBS), allowing to delay the start of SPCK after the chip select has been asserted.

These delays allow the QSPI to be adapted to the interfaced peripherals and their speed and bus release time.

Figure 33-5. Programmable Delay



33.6.7 QSPI SPI Mode

In this mode, the QSPI acts as a regular SPI Host.

To activate this mode, the MODE bit in the Control B register must be cleared (CTRLB.MODE=0).

33.6.7.1 SPI Mode Operations

The QSPI in standard SPI mode operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the client connected to the SPI bus. The QSPI drives the chip select line to the client (CS) and the serial clock signal (SCK).

The QSPI features a single internal shift register and two holding registers: the Transmit Data Register (TXDATA) and the Receive Data Register (RXDATA). The holding registers maintain the data flow at a constant rate.

After enabling the QSPI, a data transfer begins when the processor writes to the TXDATA. The written data is immediately transferred into the internal shift register and transfer on the SPI bus starts. While the data in the internal shift register is shifted on the MOSI line, the MISO line is sampled and shifted into the internal shift register. Receiving data cannot occur without transmitting data.

If new data is written in TXDATA during the transfer, it stays in TXDATA until the current transfer is completed. Then, the received data is transferred from the internal shift register to the RXDATA, the data in TXDATA is loaded into the internal shift register, and a new transfer starts.

The transfer of data written in TXDATA in the internal shift register is indicated by the Transmit Data Register Empty (DRE) bit in the Interrupt Flag Status and Clear register (INTFLAG.DRE). When new data is written in TXDATA, this bit is cleared. The DRE bit is used to trigger the Transmit DMA channel.

The end of transfer is indicated by the Transmission Complete flag (INTFLAG.TXC). If the transfer delay for the last transfer was configured to be greater than 0 (CTRLB.DLYBCT), TXC is set after the completion of the delay. The module clock (CLK_QSPI_AHB) can be switched off at this time.

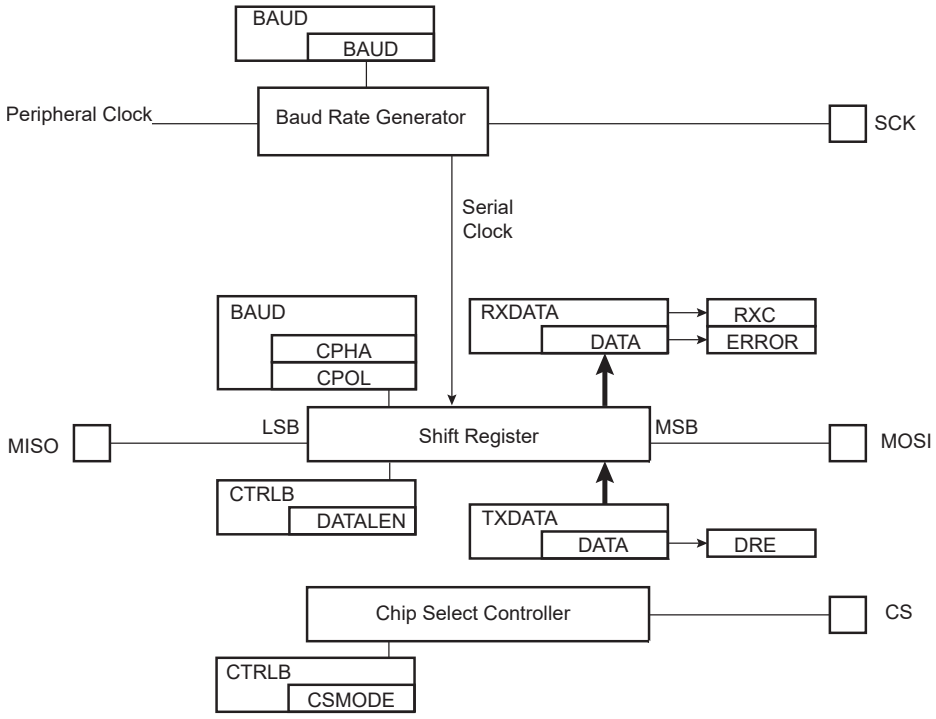
Ongoing transfer of received data from the internal shift register into RXDATA is indicated by the Receive Data Register Full flag (INTFLAG.RXC). When the received data is read, the RXC bit is cleared.

If the RXDATA has not been read before new data is received, the Overrun Error flag in INTFLAG register (INTFLAG.ERROR) is set. As long as this flag is set, data is loaded in RXDATA.

The SPI Mode Block Diagram shows a flow chart describing how transfers are handled.

33.6.7.2 SPI Mode Block Diagram

Figure 33-6. SPI Mode Block Diagram



33.6.7.3 SPI Mode Flow Diagram

Figure 33-7. SPI Mode Flow Diagram

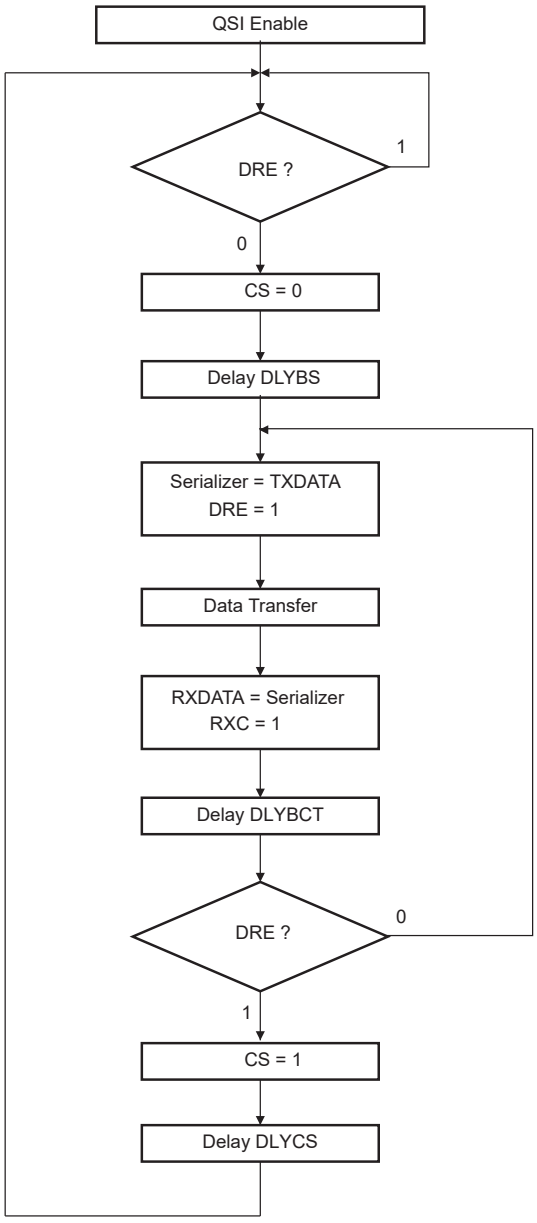
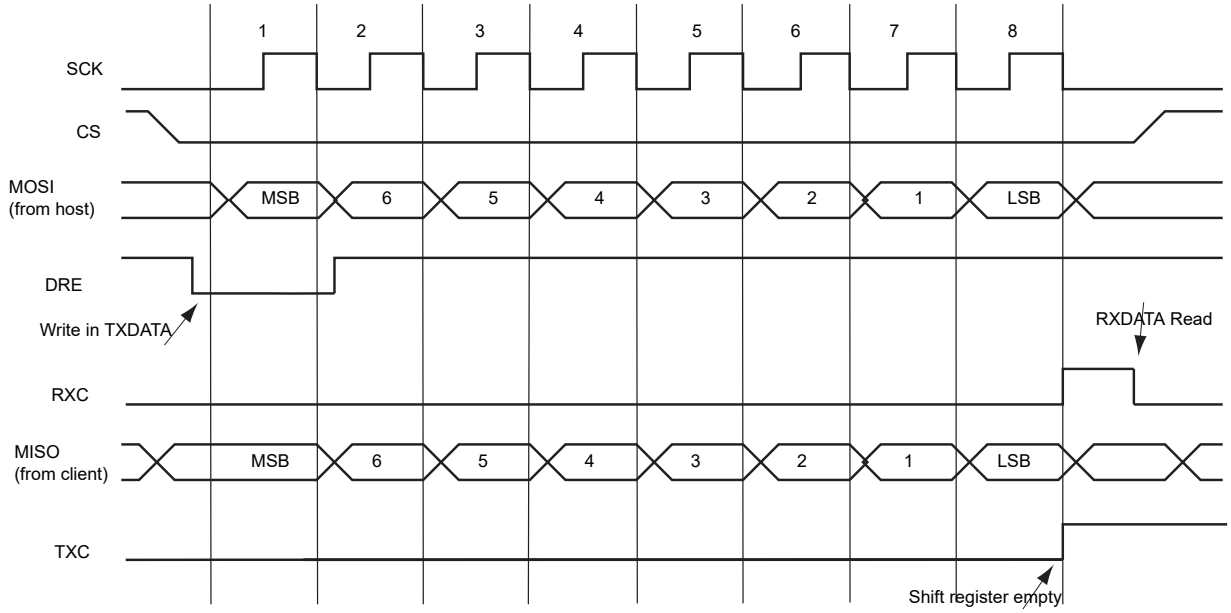


Figure 33-8. Interrupt Flags Behaviour



33.6.7.4 Peripheral Deselection with DMA

When the Direct Memory Access Controller is used, the Chip Select line will remain low during the whole transfer because the Transmit Data Register Empty flag in the Interrupt Flag Status and Clear register (INTFLAG.DRE) is managed by the DMA itself. The reloading of the TXDATA by the DMA is done as soon as the INTFLAG.DRE flag is set. In this case, setting the Chip Select mode bit field in the Control B register (CTRLB.CSMODE) to 0x1 is not mandatory.

However, it may happen that when other DMA channels connected to other peripherals are in use as well, the QSPI DMA could be delayed by another DMA transfer with a higher priority on the bus. Having DMA buffers in slower memories, like Flash memory or SDRAM (compared to fast internal SRAM), may lengthen the reload time of the TXDATA by the DMA as well. This means that TXDATA might not be reloaded in time to keep the Chip Select line low. In this case, the Chip Select line may toggle between data transfer and some SPI Client devices, and the communication might get lost. Writing CTRLB.CSMODE=0x1 can prevent this loss.

When CTRLB.CSMODE=0x0, the CS does not rise in all cases between two transfers on the same peripheral. During a transfer on a Chip Select, the INTFLAG.DRE flag is raised as soon as the content of the TXDATA is transferred into the internal shifter. When this flag is detected, the TXDATA can be reloaded. If this reload occurs before the end of the current transfer and if the next transfer is performed on the same Chip Select as the current transfer, the Chip Select is not de-asserted between the two transfers. This may lead to difficulties for interfacing with some serial peripherals requiring the Chip Select to be de-asserted after each transfer. To facilitate interfacing with such devices, it is recommended to write CTRLB.CSMODE to 0x2.

33.6.7.5 Peripheral Deselection without DMA

During multiple data transfers on a Chip Select without the DMA, the TXDATA is loaded by the processor, and the Transmit Data Register Empty flag in the Interrupt Flag Status and Clear register (INTFLAG.DRE) rises as soon as the content of the RXDATA is transferred into the internal shift register. When this flag is detected high, the TXDATA can be reloaded. If this reload-by-processor occurs before the end of the current transfer and if the next transfer is performed on the same Chip Select as the current transfer, the Chip Select is not de-asserted between the two transfers.

Depending on the application software handling the flags or servicing other interrupts or other tasks, the processor may not reload the TXDATA in time to keep the Chip Select active (low). A null Delay Between Consecutive Transfer bit field value in the CTRLB register (CTRLB.DLYBCT) will give even less time for the processor to reload the TXDATA. With some SPI client peripherals, requiring the Chip Select line to remain active (low) during a full set of transfers might lead to communication errors.

To facilitate interfacing with such devices, the Chip Select Mode bit field in the CTRLB register (CTRLB.CSMODE) can be written to 0x1. This allows the Chip Select lines to remain in their current state (low = active) until the end of transfer is indicated by the Last Transfer bit in the CTRLA register (CTRLA.LASTXFER). Even if the TXDATA is not reloaded, the Chip Select will remain active. To have the Chip Select line rise at the end of the last data transfer, the LASTXFER bit in the CTRLA must be set before writing the last data to transmit into the TXDATA.

33.6.8 QSPI Serial Memory Mode

In this mode the QSPI acts as a serial Flash memory controller. The QSPI can be used to read data from the serial Flash memory allowing the CPU to execute code from it (XIP execute in place). The QSPI can also be used to control the serial Flash memory (Program, Erase, Lock, and so on) by sending specific commands. In this mode, the QSPI is compatible with single-bit SPI, Dual-SPI and Quad-SPI protocols.

To activate this mode, the MODE bit in Control B register must be set to one (CTRLB.MODE = 1).

In serial memory mode, data cannot be transferred by the TXDATA and the RXDATA, but by writing or reading the QSPI memory space (0x0400 0000 – 0x0500 0000).

Important: QSPI memory space region can be cached to improve data transfer speed. However, external Flash devices which have command/status registers mapped in the QSPI memory space region must be managed carefully by applying any one of the following configurations:

- Data cache must be disabled.
- If data cache is required, then cache line must be invalidated before reading the status register.

33.6.8.1 Instruction Frame

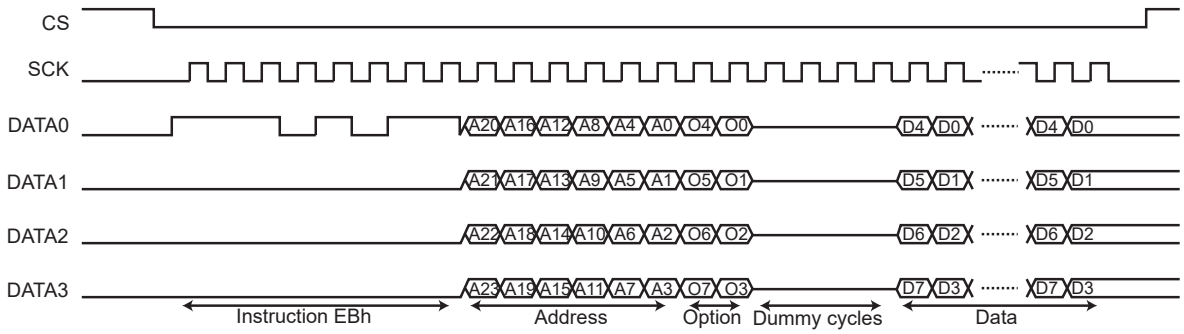
In order to control serial Flash memories, the QSPI is able to sent instructions by the SPI bus (ex: READ, PROGRAM, ERASE, LOCK, etc.). Because instruction set implemented in serial Flash memories is memory vendor dependent, the QSPI includes a complete instruction registers, which makes it very flexible and compatible with all serial Flash memories.

An instruction frame includes:

- **An instruction code (size: 8 bits):** The instruction can be optional in some cases
- **An address (size: 24 bits or 32 bits):** The address is optional but is required by instructions such as READ, PROGRAM, ERASE, LOCK. By default the address is 24 bits long, but it can be 32 bits long to support serial Flash memories larger than 128 Mbit (16 Mbyte).
- **An option code (size: 1/2/4/8 bits):** The option code is optional but is useful for activate the “XIP mode” or the “Continuous Read Mode” for READ instructions, in some serial Flash memory devices. These modes allow to improve the data read latency.
- **Dummy cycles:** Dummy cycles are optional but required by some READ instructions
- **Data bytes are optional:** Data bytes are present for data transfer instructions such as READ or PROGRAM

The instruction code, the address/option and the data can be sent with Single-bit SPI, Dual SPI or Quad SPI protocols.

Figure 33-9. Instruction Frame



33.6.8.2 Instruction Frame Sending

To send an instruction frame, the user must first configure the address to send by writing the field ADDR in the Instruction Address Register (INSTRADDR.ADDR). This step is required if the instruction frame includes an address and no data. When data is present, the address of the instruction is defined by the address of the data accesses in the QSPI memory space, and not by the INSTRADDR register.

If the instruction frame includes the instruction code and/or the option code, the user must configure the instruction code and/or the option code to send by writing the fields INST and OPTCODE bit fields in the Instruction Control Register (INSTRCTRL.OPTCODE, INSTRCTRL.INSTR).

Then, the user must write the Instruction Frame Register (INSTRFRAME) to configure the instruction frame depending on which instruction must be sent. If the instruction frame does not include data, writing in this register triggers the send of the instruction frame in the QSPI. If the instruction frame includes data, the send of the instruction frame is triggered by the first data access in the QSPI memory space.

The instruction frame is configured by the following bits and fields of INSTRFRAME:

- WIDTH field is used to configure which data lanes are used to send the instruction code, the address, the option code and to transfer the data. It is possible to use two unidirectional data lanes (MISO-MOSI Single-bit SPI), two bidirectional data lanes (DATA0 - DATA1 Dual SPI) or four bidirectional data lanes (DATA0 - DATA3).

Table 33-3. WIDTH Encoding

INSTRFRAME	Instruction	Address/Option	Data
0	Single-bit SPI	Single-bit SPI	Single-bit SPI
1	Single-bit SPI	Single-bit SPI	Dual SPI
2	Single-bit SPI	Single-bit SPI	Quad SPI
3	Single-bit SPI	Dual SPI	Dual SPI
4	Single-bit SPI	Quad SPI	Quad SPI
5	Dual SPI	Dual SPI	Dual SPI
6	Quad SPI	Quad SPI	Quad SPI
7	Reserved		

- INSTREN bit enables sending an instruction code
- ADDREN bit enables sending of an address after the instruction code
- OPTCODEEN bit enables sending of an option code after the address
- DATAEN bit enables the transfer of data (READ or PROGRAM instruction)
- OPTCODELEN field configures the option code length (0 -> 1-bit / 1 -> 2-bit / 2 -> 4-bit / 3 -> 8-bit). The value written in OPTCODELEN must be consistent with value written in the field WIDTH. For example: OPTCODELEN = 0 (1-bit option code) is not coherent with WIDTH = 6 (option code sent with QuadSPI protocol, thus the minimum length of the option code is 4-bit).
- ADDRLEN bit configures the address length (0 -> 24 bits / 1-> 32 bits)
- TFRTYPE field defines which type of data transfer must be performed
- DUMMYLEN field configures the number of dummy cycles when reading data from the serial Flash memory. Between the address/option and the data, with some instructions, dummy cycles are inserted by the serial Flash memory.

If data transfer is enabled, the user can access the serial memory by reading or writing the QSPI memory space following these rules:

- Reading from the serial memory, but not memory data (for example reading the JEDEC-ID or the STATUS), requires TFRTYPE to be written to 0x0
- Reading from the serial memory, and particularly memory data, requires TFRTYPE to be written to '1'
- Writing to the serial memory, but not memory data (for example writing the configuration or STATUS), requires TFRTYPE to be written to 0x2
- Writing to the serial memory, and particularly memory data, requires TFRTYPE to be written to 0x3

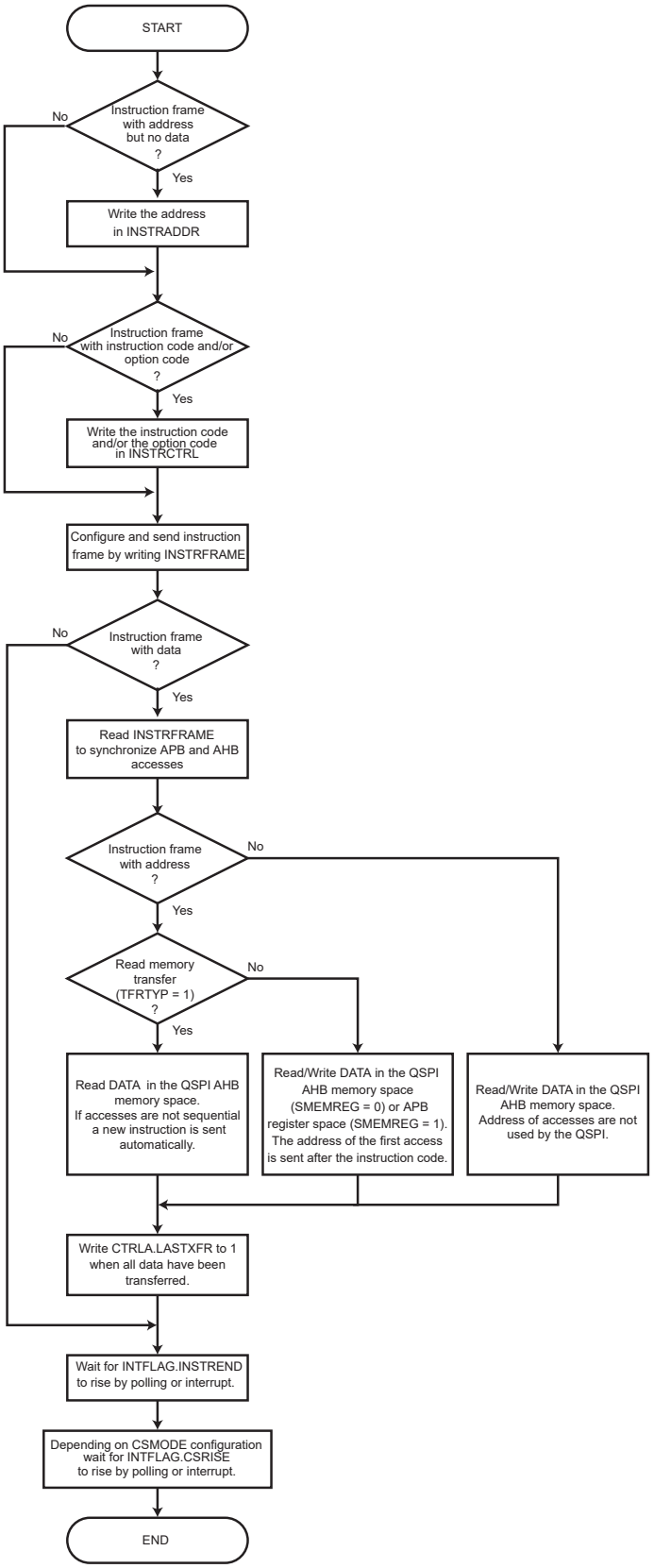
If TFRTYP has a value other than 0x1 and CTRLB.SMEMREG=0, the address sent in the instruction frame is the address of the first system bus accesses. The addresses of the subsequent access actions are not used by the QSPI. At each system bus access, an SPI transfer is performed with the same size. For example, a half-word system bus access leads to a 16-bit SPI transfer, and a byte system bus access leads to an 8-bit SPI transfer.

If CTRLB.SMEMREG=1, accesses are made via the QSPI registers and the address sent in the instruction frame is the address defined in the INSTRADDR register. Each time the INSTRFRAME or TXDATA registers are written, an SPI transfer is performed with a byte size. Another byte is read each time RXDATA register is read or written each time TXDATA register is written. The SPI transfer ends by writing the LASTXFER bit in Control A register (CTRLA.LASTXFER).

If TFRTYP=0x1, the address of the first instruction frame is the one of the first read access in the QSPI memory space. Each time the read accesses become non-sequential (addresses are not consecutive), a new instruction frame is sent with the last system bus access address. In this way, the system can read data at a random location in the serial memory. The size of the SPI transfers may differ from the size of the system bus read accesses.

When data transfer is not enabled, the end of the instruction frame is indicated when the INSTREND interrupt flag in the INTFLAG register is set. When data transfer is enabled, the user must indicate when data transfer is completed in the QSPI memory space by setting the bit LASTXFR in the CTRLA. The end of the instruction frame is indicated when the INSTREND interrupt flag in the INTFLAG register is set.

Figure 33-10. Instruction Transmission Flow Diagram



33.6.8.3 Read Memory Transfer

The user can access the data of the serial memory by sending an instruction with DATAEN=1 and TFRTYP=0x1 in the Instruction Frame register (INSTRFRAME).

In this mode the QSPI is able to read data at random address into the serial Flash memory, allowing the CPU to execute code directly from it (XIP execute-in-place).

In order to fetch data, the user must first configure the instruction frame by writing the INSTRFRAME. Then data can be read at any address in the QSPI address space mapping. The address of the system bus read accesses match the address of the data inside the serial Flash memory.

When Fetch Mode is enabled, several instruction frames can be sent before writing the bit LASTXFR in the CTRLA. Each time the system bus read accesses become non-sequential (addresses are not consecutive), a new instruction frame is sent with the corresponding address.

33.6.8.4 Continuous Read Mode

The QSPI is compatible with Continuous Read Mode (CRM) which is implemented in some Serial Flash memories.

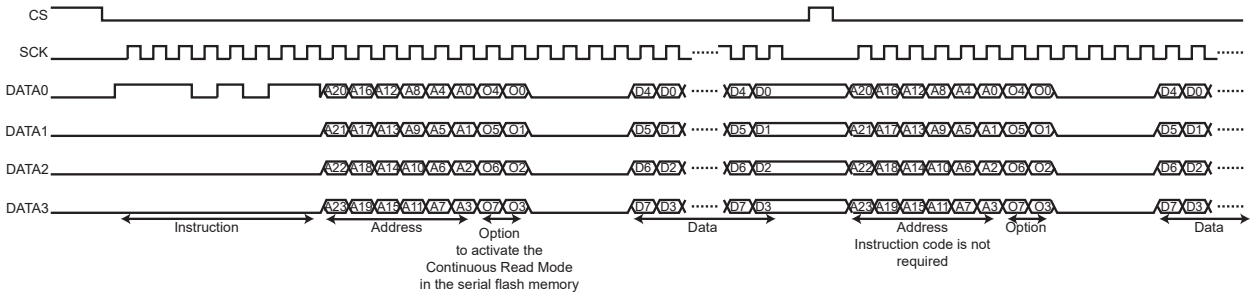
The CRM allows to reduce the instruction overhead by excluding the instruction code from the instruction frame. When CRM is activated in a Serial Flash memory (by a specific option code), the instruction code is stored in the memory. For the next instruction frames, the instruction code is not required, as the memory uses the stored one.

In the QSPI, CRM is used when reading data from the memory (INSTFRAME.TFRTYPE=0x1). The addresses of the system bus read accesses are often non-sequential, this leads to many instruction frames with always the same instruction code. By disabling the sending of the instruction code, the CRM reduces the access time of the data.

To be functional, this mode must be enabled in both the QSPI and the Serial Flash memory. The CRM is enabled in the QSPI by setting the CRM bit in the INSTRFRAME register (INSTFRAME.CRMODE=1, INSTFRAME.TFRTYPE must be 0x1). The CRM is enabled in the Serial Flash memory by sending a specific option code.

⚠ CAUTION If CRM is not supported by the Serial Flash memory or disabled, the CRMODE bit must not be set. Otherwise, data read out the Serial Flash memory is not valid.

Figure 33-11. Continuous Read Mode



33.6.8.5 Instruction Frame Transmission Examples

All waveforms in the following examples describe SPI transfers in SPI Clock mode 0 (BAUD.CPOL=0 and BAUD.CPHA=0). All system bus accesses described below refer to the system bus address phase. System bus wait cycles and system bus data phases are not shown.

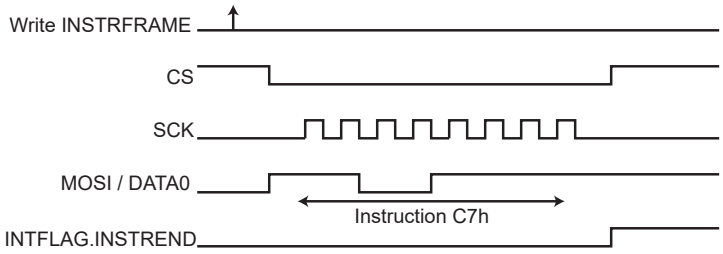
Example 33-1. Example 1

Instruction in Single-bit SPI, without address, without option, without data.

Command: CHIP ERASE (C7h).

- Write 0x0000_00C7 to INSTRCTRL register
- Write 0x0000_0010 to INSTRFRAME register
- Wait for INTFLAG.INSTREND to rise

Figure 33-12. Instruction Transmission Waveform 1



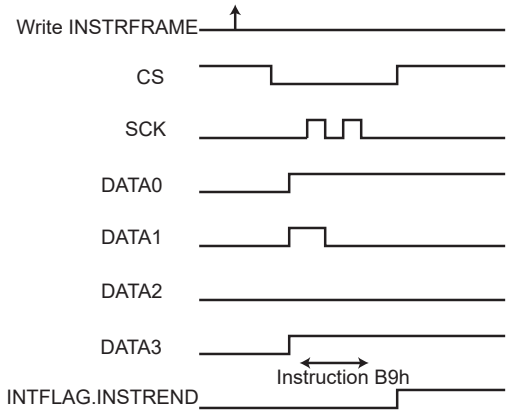
Example 33-2. Example 2

Instruction in Quad SPI, without address, without option, without data.

Command: POWER DOWN (B9h)

- Write 0x0000_00B9 to INSTRCTRL register
- Write 0x0000_0016 to INSTRFRAME register
- Wait for INTFLAG.INSTREND to rise

Figure 33-13. Instruction Transmission Waveform 2



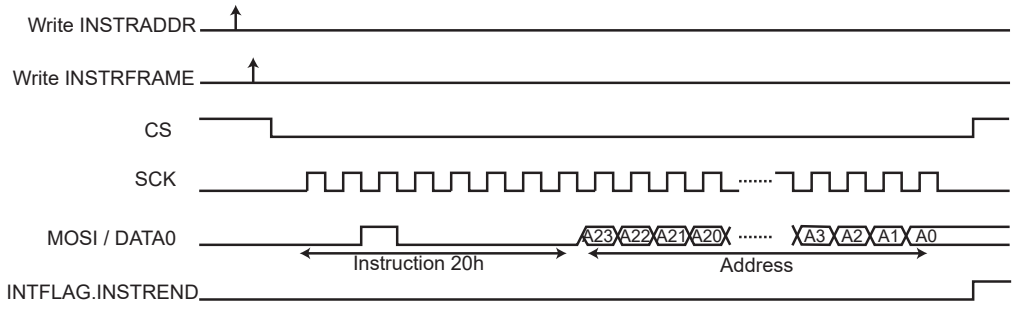
Example 33-3. Example 3

Instruction in Single-bit SPI, with address in Single-bit SPI, without option, without data.

Command: BLOCK ERASE (20h)

- Write the address (of the block to erase) to QSPI_AR
- Write 0x0000_0020 to INSTRCTRL register
- Write 0x0000_0030 toINSTRFRAME register
- Wait for INTFLAG.INSTREND to rise

Figure 33-14. Instruction Transmission Waveform 3



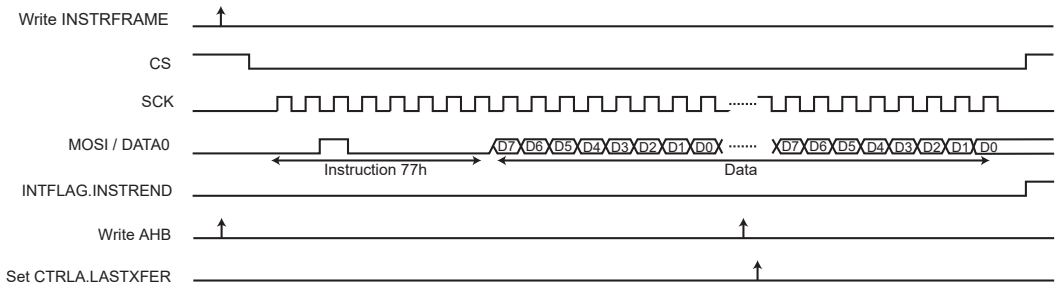
Example 33-4. Example 4

Instruction in Single-bit SPI, without address, without option, with data write in Single-bit SPI.

Command: SET BURST (77h)

- Write 0x0000_0077 to INSTRCTRL register.
- Write 0x0000_2090 to INSTRFRAME register.
- Read INSTRFRAME register (dummy read) to synchronize system bus accesses.
- Write data to the system bus memory space (0x0400_0000–0x0500_0000). The address of the system bus write accesses is not used.
- Write the LASTXFR bit in CTRLA register to '1'.
- Wait for INTFLAG.INSTREND to rise.

Figure 33-15. Instruction Transmission Waveform 4



Example 33-5. Example 5

Instruction in Single-bit SPI, with address in Dual SPI, without option, with data write in Dual SPI.

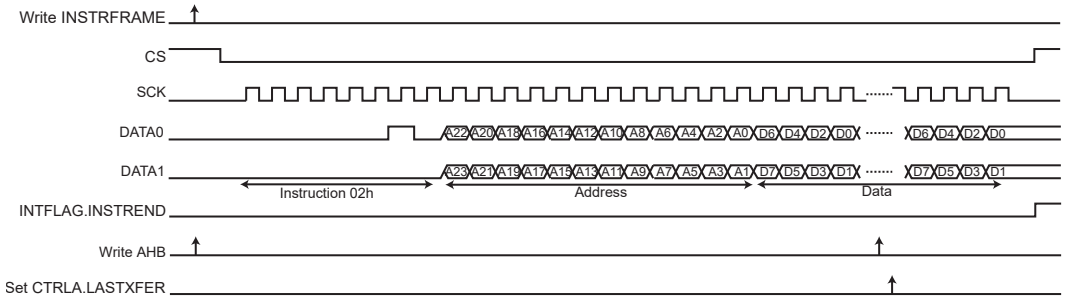
Command: BYTE/PAGE PROGRAM (02h)

- Write 0x0000_0002 to INSTRCTRL register.
- Write 0x0000_30B3 to INSTRFRAME register.
- Read INSTRFRAME register (dummy read) to synchronize system bus accesses.
- Write data to the QSPI system bus memory space (0x040 00000–0x0500_0000).

  The address of the first system bus write access is sent in the instruction frame.

  The address of the next system bus write accesses is not used.
- Write LASTXFR bit in CTRLA register to '1'.
- Wait for INTFLAG.INSTREND to rise.

Figure 33-16. Instruction Transmission Waveform 5



Example 33-6. Example 6

Instruction in Single-bit SPI, with address in Single-bit SPI, without option, with data read in Quad SPI, with eight dummy cycles.

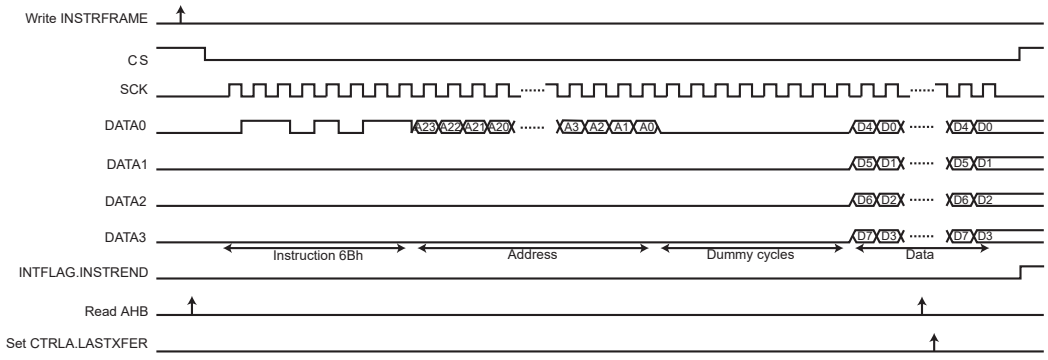
Command: QUAD_OUTPUT READ ARRAY (6Bh)

- Write 0x0000_006B to INSTRCTRL register.
- Write 0x0008_10B2 ti INSTRFRAME register.
- Read QSPI_IR (dummy read) to synchronize system bus accesses.
- Read data from the QSPI system bus memory space (0x040 00000–0x0500_0000).

  The address of the first system bus read access is sent in the instruction frame.

  The address of the next system bus read accesses is not used.
- Write the LASTXFR bit in CTRLA register to '1'.
- Wait for INTFLAG.INSTREND to rise.

Figure 33-17. Instruction Transmission Waveform 6



Example 33-7. Example 7

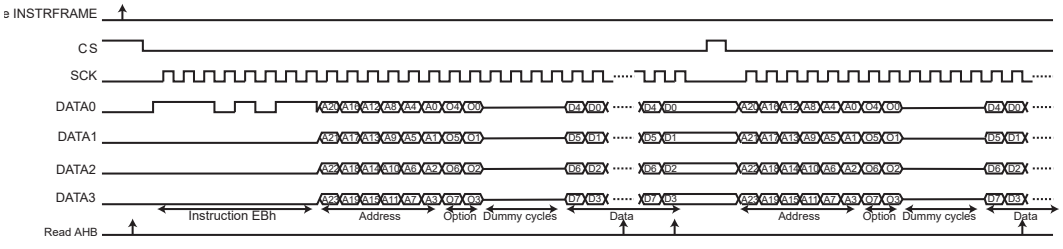
Instruction in Single-bit SPI, with address and option in Quad SPI, with data read from Quad SPI, with four dummy cycles, with fetch and continuous read.

Command: FAST READ QUAD I/O (EBh) - 8-BIT OPTION (0x30h)

- Write 0x0030_00EB to INSTRCTRL register.
- Write 0x0004_33F4 to INSTRFRAME register.
- Read INSTRFRAME register (dummy read) to synchronize system bus accesses.
- Read data from the QSPI system bus memory space (0x040 00000–0x0500_0000).

  Fetch is enabled, the address of the system bus read accesses is always used.
- Write LASTXFR bit in CTRLA register to '1'.
- Wait for INTFLAG.INSTREND to rise.

Figure 33-18. Instruction Transmission Waveform 7



Example 33-8. Example 8

Instruction in Quad SPI, with address in Quad SPI, without option, with data read from Quad SPI, with two dummy cycles, with fetch.

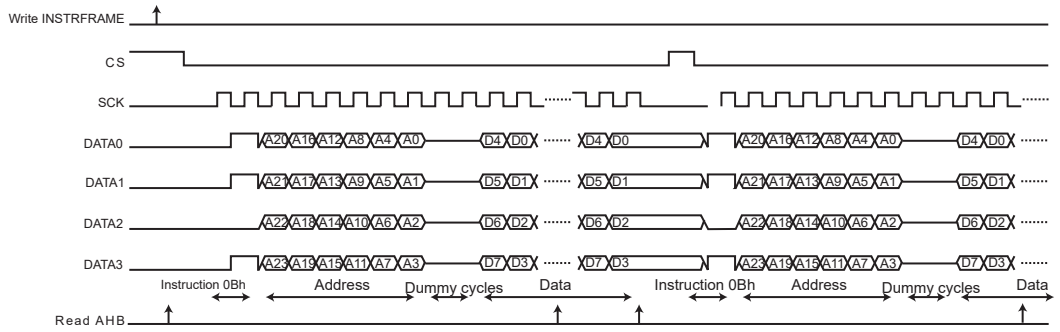
Command: HIGH-SPEED READ (0Bh)

- Write 0x0000_000B to INSTRCTRL register.

- Write 0x0002_20B6 to INSTRFRAME register.
- Read INSTRFRAME register (dummy read) to synchronize system bus accesses.
- Read data in the QSPI system bus memory space (0x040 00000–0x0500_0000).

  Fetch is enabled, the address of the system bus read accesses is always used.
- Write LASTXFR bit in CTRLA register to '1'.
- Wait for INTFLAG.INSTREND to rise.

Figure 33-19. Instruction Transmission Waveform 8



33.6.9 Scrambling/Unscrambling Function

The scrambling/unscrambling function cannot be performed on devices other than memories. Data is scrambled when written to memory and unscrambled when data is read.

The external data lines can be scrambled to prevent intellectual property data located in off-chip memories from being easily recovered by analyzing data at the package pin level of either the micro-controller or the QSPI client device (e.g., memory).

The scrambling/unscrambling function can be enabled by writing a '1' to the ENABLE bit in the Scrambling Control register (SCRAMBCTRL.ENABLE).

The scrambling and unscrambling are performed on-the-fly without impacting the throughput.

The scrambling method depends on the user-configurable Scrambling User Key in the Scrambling Key register (SCRAMBKEY.KEY). This register is only accessible in Write mode.

By default, the scrambling and unscrambling algorithm includes the scrambling user key, plus a device-dependent random value. This random value is not included when the Scrambling/Unscrambling Random Value Disable bit in the Scrambling Mode register (SCRAMBCTRL.RANDOMDIS) is written to '1'.

The random value is neither user-configurable nor readable. If SCRAMBCTRL.RANDOMDIS=0, data scrambled by a given circuit cannot be unscrambled by a different circuit.

If SCRAMBCTRL.RANDOMDIS=1, the scrambling/unscrambling algorithm includes only the scrambling user key, making it possible to manage data by different circuits.

The scrambling user key must be securely stored in a reliable Non-Volatile Memory to recover data from the off-chip memory. Any data scrambled with a given key cannot be recovered if the key is lost.

33.6.10 DMA Operation

The QSPI generates the following DMA requests:

- Data received (RX): The request is set when data is available in the RXDATA register, and cleared when RXDATA is read.
- Data transmit (TX): The request is set when the transmit buffer (TXDATA) is empty, and cleared when TXDATA is written.

Note: If DMA and RX memory modes are selected, a QSPI memory space read operation is required to force the first triggering.

If the CPU accesses the registers which are source of DMA request set/clear condition, the DMA request can be lost or the DMA transfer can be corrupted.

33.6.11 Interrupts

The QSPI has the following interrupt source:

- Interrupt Request (INTREQ): Indicates that at least one bit in the Interrupt Flag Status and Clear register (INTFLAG) is set to '1'.

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear (INTFLAG) register is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a '1' to the corresponding bit in the Interrupt Enable Set (INTENSET) register, and disabled by writing a '1' to the corresponding bit in the Interrupt Enable Clear (INTENCLR) register. An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled, or the QSPI is reset. All interrupt requests from the peripheral are ORed together on system level to generate one combined interrupt request to the NVIC. The user must read the INTFLAG register to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated.

33.7 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0							ENABLE	SWRST
		15:8								
		23:16								
		31:24								LASTXFER
0x04	CTRLB	7:0			CSMODE[1:0]		SMEMREG	WDRBT	LOOPEN	MODE
		15:8					DATALEN[3:0]			
		23:16	DLYBCT[7:0]							
		31:24	DLYCS[7:0]							
0x08	BAUD	7:0							CPHA	CPOL
		15:8	BAUD[7:0]							
		23:16	DLYBS[7:0]							
		31:24								
0x0C	RXDATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16								
		31:24								
0x10	TXDATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16								
		31:24								
0x14	INTENCLR	7:0					ERROR	TXC	DRE	RXC
		15:8						INSTREND		CSRISE
		23:16								
		31:24								
0x18	INTENSET	7:0					ERROR	TXC	DRE	RXC
		15:8						INSTREND		CSRISE
		23:16								
		31:24								
0x1C	INTFLAG	7:0					ERROR	TXC	DRE	RXC
		15:8						INSTREND		CSRISE
		23:16								
		31:24								
0x20	STATUS	7:0							ENABLE	
		15:8							CSSTATUS	
		23:16								
		31:24								
0x24 ... 0x2F	Reserved									
0x30	INSTRADDR	7:0	ADDR[7:0]							
		15:8	ADDR[15:8]							
		23:16	ADDR[23:16]							
		31:24	ADDR[31:24]							
0x34	INSTRCTRL	7:0	INSTR[7:0]							
		15:8								
		23:16	OPTCODE[7:0]							
		31:24								
0x38	INSTRFRAME	7:0	DATAEN	OPTCODEEN	ADDREN	INSTREN		WIDTH[2:0]		
		15:8	DDREN	CRMODE	TFRTYPE[1:0]			ADDRLEN	OPTCODELEN[1:0]	
		23:16				DUMMYLEN[4:0]				
		31:24								
0x3C ... 0x3F	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x40	SCRAMBCTRL	7:0							RANDOMDIS	ENABLE
		15:8								
		23:16								
		31:24								
0x44	SCRAMBKEY	7:0	KEY[7:0]							
		15:8	KEY[15:8]							
		23:16	KEY[23:16]							
		31:24	KEY[31:24]							

33.8 Register Description

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

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

See *Peripheral Access Controller (PAC)* from Related Links.

Some registers are enable-protected, meaning they can only be written when the QSPI is disabled. Enable-protection is denoted by the Enable-protected property in each individual register description.

Related Links

26. Peripheral Access Controller (PAC)

33.8.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00000000
Property: -

Control A

Bit	31	30	29	28	27	26	25	24
								LASTXFER
Access								W
Reset								0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							ENABLE	SWRST
Access							W	W
Reset							0	0

Bit 24 – LASTXFER Last Transfer
0: No effect.
1: The chip select will be de-asserted after the character written in TD has been transferred.

Bit 1 – ENABLE Enable
Writing a '0' to this bit disables the QSPI.
Writing a '1' to this bit enables the QSPI to transfer and receive data.
As soon as ENABLE is reset, QSPI finishes its transfer.
All pins are set in input mode and no data is received or transmitted.
If a transfer is in progress, the transfer is finished before the QSPI is disable.

Bit 0 – SWRST Software Reset
Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets the QSPI. A software-triggered hardware reset of the QSPI interface is performed.
DMAC channels are not affected by software reset.

33.8.2 Control B

Name: CTRLB
Offset: 0x04
Reset: 0x00000000
Property: PAC Write-Protection

Control B

Bit	31	30	29	28	27	26	25	24
	DLYCS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DLYBCT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
					DATALEN[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
			CSMODE[1:0]		SMEMREG	WDRBT	LOOPEN	MODE
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 31:24 – DLYCS[7:0] Minimum Inactive CS Delay

This bit field defines the minimum delay between the inactivation and the activation of CS. The DLYCS time guarantees the client minimum deselect time.
If DLYCS is 0x00, one CLK_QSPI_AHB period will be inserted by default.
Otherwise, the following equation determines the delay:
DLYCS = Minimum inactive × fperipheral clock

Bits 23:16 – DLYBCT[7:0] Delay Between Consecutive Transfers

This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.
When DLYBCT=0x00, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers. In Serial Memory mode (MODE=1), DLYBCT is ignored and no delay is inserted. Otherwise, the following equation determines the delay:
DLYBCT = (Delay Between Consecutive Transfers × fperipheral clock) / 32

Bits 11:8 – DATALEN[3:0] Data Length

The DATALEN field determines the number of data bits transferred. Reserved values must not be used.

Value	Name	Description
0x0	8BITS	8-bits transfer
0x1	9BITS	9-bits transfer
0x2	10BITS	10-bits transfer
0x3	11BITS	11-bits transfer
0x4	12BITS	12-bits transfer
0x5	13BITS	13-bits transfer
0x6	14BITS	14-bits transfer
0x7	15BITS	15-bits transfer
0x8	16BITS	16-bits transfer
0x9-0xF		Reserved

Bits 5:4 – CSMODE[1:0] Chip Select Mode

The CSMODE field determines how the chip select is de-asserted.

Value	Name	Description
0x0	NORELOAD	The chip select is de-asserted if TD has not been reloaded before the end of the current transfer.
0x1	LASTXFER	The chip select is de-asserted when the bit LASTXFER is written at 1 and the character written in TD has been transferred.
0x2	SYSTEMATICALLY	The chip select is de-asserted systematically after each transfer.
0x3		Reserved

Bit 3 – SMEMREG Serial Memory Register Mode

Value	Description
0	Serial memory registers are written via AHB access.
1	Serial memory registers are written via APB access. Reset the QSPI.

Bit 2 – WDRBT Wait Data Read Before Transfer

This bit determines the Wait Data Read Before Transfer option.

Bit 1 – LOOPEN Local Loopback Enable

This bit defines if the Local Loopback is enabled or disabled.

LOOPEN controls the local loopback on the data serializer for testing in SPI Mode only. (MISO is internally connected on MOSI).

Value	Description
0	Local Loopback is disabled.
1	Local Loopback is enabled.

Bit 0 – MODE Serial Memory Mode

This bit defines if the QSPI is in SPI Mode or Serial Memory Mode.

Value	Name	Description
0	SPI	SPI operating mode
1	MEMORY	Serial Memory operating mode

33.8.3 Baud Rate

Name: BAUD
Offset: 0x08
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	DLYBS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BAUD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
							CPHA	CPOL
Access							R/W	R/W
Reset							0	0

Bits 23:16 – DLYBS[7:0] Delay Before SCK

This field defines the delay from CS valid to the first valid SCK transition.
When DLYBS equals zero, the CS valid to SCK transition is 1/2 the SCK clock period.
Otherwise, the following equation determines the delay:

Equation 33-1. Delay Before SCK

$$Delay\ \ Before\ SCK = \frac{DLYBS}{MCK}$$

Bits 15:8 – BAUD[7:0] Serial Clock Baud Rate

The QSPI uses a modulus counter to derive the SCK baud rate from the module clock (MCK) CLK_QSPI_AHB. The Baud rate is selected by writing a value from 1 to 255 in the BAUD field. The following equation determines the SCK baud rate:

Equation 33-2. SCK Baud Rate

$$SCK\ \ Baud\ \ Rate = \frac{MCK}{(BAUD + 1)}$$

Bit 1 – CPHA Clock Phase

CPHA determines which edge of SCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce the required clock/data relationship between host and client devices.

Value	Description
0	Data is captured on the leading edge of SCK and changed on the following edge of SCK.
1	Data is changed on the leading edge of SCK and captured on the following edge of SCK.

Bit 0 – CPOL Clock Polarity

CPOL is used to determine the inactive state value of the serial clock (SCK). It is used with CPHA to produce the required clock/data relationship between host and client devices.

Value	Description
0	The inactive state value of SCK is logic level zero.

Value	Description
0	The inactive state value of SCK is logic level 'one'.

33.8.4 Receive Data

Name: RXDATA
Offset: 0x0C
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

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

Bits 15:0 – DATA[15:0] Receive Data
Data received by the QSPI is stored in this register right-justified. Unused bits read zero.

33.8.5 Transmit Data

Name: TXDATA
Offset: 0x10
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – DATA[15:0] Transmit Data
Data to be transmitted by the QSPI is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

33.8.6 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x14
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
						INSTREND		CSRISE
Access						R/W		R/W
Reset						0		0

Bit	7	6	5	4	3	2	1	0
					ERROR	TXC	DRE	RXC
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 10 – INSTREND Instruction End Interrupt Disable
Writing a '0' to this bit has no effect.
Writing a '1' will clear the corresponding interrupt request.

Value	Description
0	The INSTREND interrupt is disabled.
1	The INSTREND interrupt is enabled.

Bit 8 – CSRISE Chip Select Rise Interrupt Disable
Writing a '0' to this bit has no effect.
Writing a '1' will clear the corresponding interrupt request.

Value	Description
0	The CSRISE interrupt is disabled.
1	The CSRISE interrupt is enabled.

Bit 3 – ERROR Overrun Error Interrupt Disable
Writing a '0' to this bit has no effect.
Writing a '1' will clear the corresponding interrupt request.

Value	Description
0	The ERROR interrupt is disabled.
1	The ERROR interrupt is enabled.

Bit 2 – TXC Transmission Complete Interrupt Disable
Writing a '0' to this bit has no effect.
Writing a '1' will clear the corresponding interrupt request.

Value	Description
0	The TXC interrupt is disabled.
1	The TXC interrupt is enabled.

Bit 1 – DRE Transmit Data Register Empty Interrupt Disable

Writing a '0' to this bit has no effect.
Writing a '1' will clear the corresponding interrupt request.

Value	Description
0	The DRE interrupt is disabled.
1	The DRE interrupt is enabled.

Bit 0 – RXC Receive Data Register Full Interrupt Disable

Writing a '0' to this bit has no effect.
Writing a '1' will clear the corresponding interrupt request.

Value	Description
0	The RXC interrupt is disabled.
1	The RXC interrupt is enabled.

33.8.7 Interrupt Enable Set

Name: INTENSET
Offset: 0x18
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
						INSTREND		CSRISE
Access						R/W		R/W
Reset						0		0

Bit	7	6	5	4	3	2	1	0
					ERROR	TXC	DRE	RXC
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 10 – INSTREND Instruction End Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' will set the corresponding interrupt request.

Value	Description
0	The INSTREND interrupt is disabled.
1	The INSTREND interrupt is enabled.

Bit 8 – CSRISE Chip Select Rise Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' will set the corresponding interrupt request.

Value	Description
0	The CSRISE interrupt is disabled.
1	The CSRISE interrupt is enabled.

Bit 3 – ERROR Overrun Error Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' will set the corresponding interrupt request.

Value	Description
0	The ERROR interrupt is disabled.
1	The ERROR interrupt is enabled.

Bit 2 – TXC Transmission Complete Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' will set the corresponding interrupt request.

Value	Description
0	The TXC interrupt is disabled.
1	The TXC interrupt is enabled.

Bit 1 – DRE Transmit Data Register Empty Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' will set the corresponding interrupt request.

Value	Description
0	The DRE interrupt is disabled.
1	The DRE interrupt is enabled.

Bit 0 – RXC Receive Data Register Full Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' will set the corresponding interrupt request.

Value	Description
0	The RXC interrupt is disabled.
1	The RXC interrupt is enabled.

33.8.8 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x1C
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
						INSTREND		CSRISE
Access						R/W		R/W
Reset						0		0

Bit	7	6	5	4	3	2	1	0
					ERROR	TXC	DRE	RXC
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 10 – INSTREND Instruction End
This bit is set when an Instruction End has been detected.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 8 – CSRISE Chip Select Rise
The bit is set when a Chip Select Rise has been detected.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 3 – ERROR Overrun Error
This bit is set when an ERROR has occurred.
An ERROR occurs when RXDATA is loaded at least twice from the serializer.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the flag.

Bit 2 – TXC Transmission Complete
0: As soon as data is written in TXDATA.
1: TXDATA and internal shifter are empty. If a transfer delay has been defined, TXC is set after the completion of such delay.

Bit 1 – DRE Transmit Data Register Empty
0: Data has been written to TXDATA and not yet transferred to the serializer.
1: The last data written in the TXDATA has been transferred to the serializer.
This bit is '0' when the QSPI is disabled or at reset.
The bit is set as soon as ENABLE bit is set.

Bit 0 – RXC Receive Data Register Full
0: No data has been received since the last read of RXDATA.

1: Data has been received and the received data has been transferred from the serializer to RXDATA since the last read of RXDATA.

33.8.9 Status

Name: STATUS
Offset: 0x20
Reset: 0x00000200
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
							CSSTATUS	
Access							R	
Reset							1	

Bit	7	6	5	4	3	2	1	0
							ENABLE	
Access							R	
Reset							0	

Bit 9 – CSSTATUS Chip Select

Value	Description
0	Chip Select is asserted.
1	Chip Select is not asserted.

Bit 1 – ENABLE Enable

Value	Description
0	QSPI is disabled.
1	QSPI is enabled.

33.8.10 Instruction Address

Name: INSTRADDR
Offset: 0x30
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
	ADDR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 31:0 – ADDR[31:0] Instruction Address
Address to send to the serial Flash memory in the instruction frame.

33.8.11 Instruction Code

Name: INSTRCTRL
Offset: 0x34
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	OPTCODE[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

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

Bits 23:16 – OPTCODE[7:0] Option Code
These bits define the option code to send to the serial flash memory.

Bits 7:0 – INSTR[7:0] Instruction Code
Instruction code to send to the serial flash memory.

33.8.12 Instruction Frame

Name: INSTRFRAME
Offset: 0x38
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
				DUMMYLEN[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DDREN	CRMODE	TFRTYPE[1:0]			ADDRLEN	OPTCODELEN[1:0]	
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
	DATAEN	OPTCODEEN	ADDREN	INSTREN		WIDTH[2:0]		
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bits 20:16 – DUMMYLEN[4:0] Dummy Cycles Length
The DUMMYLEN field defines the number of dummy cycles required by the serial Flash memory before data transfer.

Bit 15 – DDREN Double Data Rate Enable
Note: Double Data Rate operating is only supported in Read.

Value	Description
0	Double Data Rate operating mode is disabled.
1	Double Data Rate operating mode is enabled.

Bit 14 – CRMODE Continuous Read Mode
This bit defines if the Continuous Read Mode is enabled or disabled.

Value	Description
0	Continuous Read Mode is disabled.
1	Continuous Read Mode is enabled.

Bits 13:12 – TFRTYPE[1:0] Data Transfer Type
These bits define the data type transfer.

Value	Name	Description
0x0	READ	Read transfer from the serial memory.Scrambling is not performed.Read at random location (fetch) in the serial flash memory is not possible.
0x1	READMEMORY	Read data transfer from the serial memory.If enabled, scrambling is performed.Read at random location (fetch) in the serial flash memory is possible.
0x2	WRITE	Write transfer into the serial memory.Scrambling is not performed.
0x3	WRITEMEMORY	Write data transfer into the serial memory. If enabled, scrambling is performed.

Bit 10 – ADDRLEN Address Length
The ADDRLEN bit determines the length of the address.

Value	Name	Description
0x0	24BITS	24-bits address length
0x1	32BITS	32-bits address length

Bits 9:8 – OPTCODELEN[1:0] Option Code Length

The OPTCODELEN field determines the length of the option code. The value written in OPTCODELEN must be coherent with value written in the field WIDTH. For example: OPTCODELEN=0 (1-bit option code) is not coherent with WIDTH=6 (option code sent with QuadSPI protocol, thus the minimum length of the option code is 4-bit).

Value	Name	Description
0x0	1BIT	1-bit length option code
0x1	2BITS	2-bits length option code
0x2	4BITS	4-bits length option code
0x3	8BITS	8-bits length option code

Bit 7 – DATAEN Data Enable

Value	Description
0	No data is sent/received to/from the serial flash memory.
1	Data is sent/received to/from the serial flash memory.

Bit 6 – OPTCODEEN Option Enable

Value	Description
0	The option is not sent to the serial flash memory
1	The option is sent to the serial flash memory.

Bit 5 – ADDREN Address Enable

Value	Description
0	The transfer address is not sent to the serial flash memory.
1	The transfer address is sent to the serial flash memory.

Bit 4 – INSTREN Instruction Enable

Value	Description
0	The instruction is not sent to the serial flash memory.
1	The instruction is sent to the serial flash memory.

Bits 2:0 – WIDTH[2:0] Instruction Code, Address, Option Code and Data Width

This field defines the width of the instruction code, the address, the option and the data.

Value	Name	Description
0x0	SINGLE_BIT_SPI	Instruction: Single-bit SPI / Address-Option: Single-bit SPI / Data: Single-bit SPI
0x1	DUAL_OUTPUT	Instruction: Single-bit SPI / Address-Option: Single-bit SPI / Data: Dual SPI
0x2	QUAD_OUTPUT	Instruction: Single-bit SPI / Address-Option: Single-bit SPI / Data: Quad SPI
0x3	DUAL_IO	Instruction: Single-bit SPI / Address-Option: Dual SPI / Data: Dual SPI
0x4	QUAD_IO	Instruction: Single-bit SPI / Address-Option: Quad SPI / Data: Quad SPI
0x5	DUAL_CMD	Instruction: Dual SPI / Address-Option: Dual SPI / Data: Dual SPI
0x6	QUAD_CMD	Instruction: Quad SPI / Address-Option: Quad SPI / Data: Quad SPI
0x7		Reserved

33.8.13 Scrambling Mode

Name: SCRAMBCTRL
Offset: 0x40
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							RANDOMDIS	ENABLE
Access							R/W	R/W
Reset							0	0

Bit 1 – RANDOMDIS Scrambling/Unscrambling Random Value Disable

Value	Description
0	The scrambling/unscrambling algorithm includes the scrambling user key plus a random value that may differ from chip to chip.
1	The scrambling/unscrambling algorithm includes only the scrambling user key.

Bit 0 – ENABLE Scrambling/Unscrambling Enable

This bit defines if the scrambling/unscrambling is enabled or disabled.

Value	Description
0	Scrambling/unscrambling is disabled.
1	Scrambling/unscrambling is enabled.

33.8.14 Scrambling Key

Name: SCRAMBKEY
Offset: 0x44
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	KEY[31:24]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	KEY[23:16]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	KEY[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	KEY[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – KEY[31:0] Scrambling User Key
This field defines the user key value.

34. Configurable Custom Logic (CCL)

34.1 Overview

The Configurable Custom Logic (CCL) is a programmable logic peripheral which can be connected to the device pins, to events, or to other internal peripherals. This allows the user to eliminate logic gates for simple glue logic functions on the PCB.

Each LookUp Table (LUT) consists of three inputs, a truth table, an optional synchronizer/filter, and an optional edge detector. Each LUT can generate an output as a user programmable logic expression with three inputs. Inputs can be individually masked.

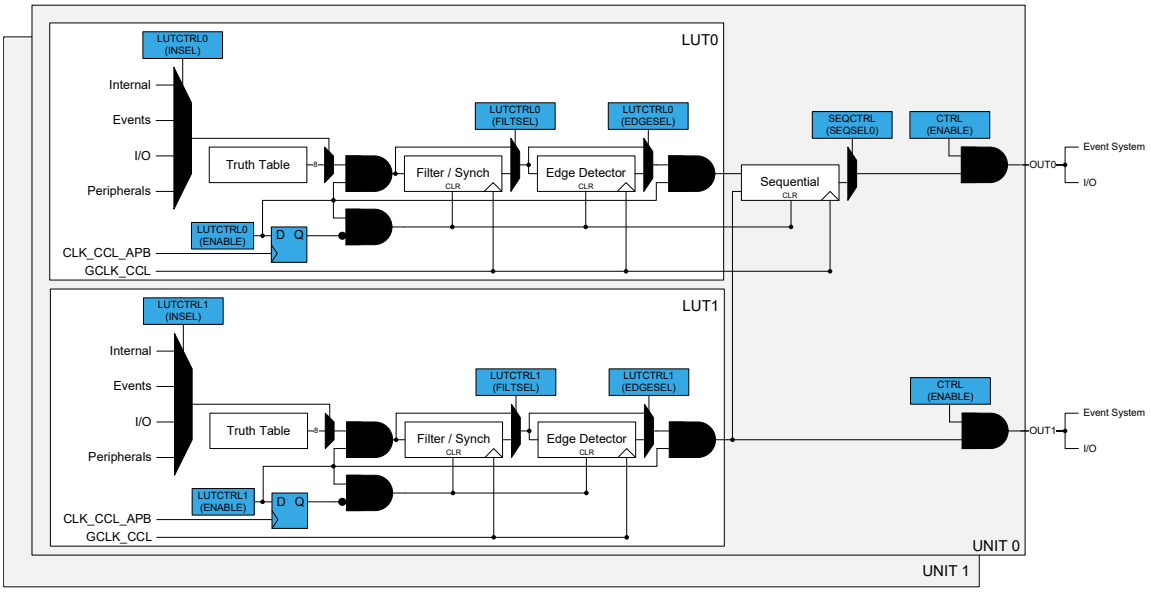
The output can be combinatorially generated from the inputs, and can be filtered to remove spikes. Optional sequential logic can be used. The inputs of the sequential module are individually controlled by two independent, adjacent LUT (LUT0/LUT1) outputs, enabling complex waveform generation.

34.2 Features

- Glue logic for general purpose PCB design
- Two programmable Look-up Tables (LUTs)
- Combinatorial logic functions: AND, NAND, OR, NOR, XOR, XNOR, NOT
- Sequential logic functions: Gated D Flip-Flop, JK Flip-Flop, gated D Latch, RS Latch
- Flexible LUT inputs selection:
  - I/Os
  - Events
  - Internal peripherals
  - Subsequent LUT output
- Output can be connected to the I/O pins or the Event System
- Optional synchronizer, filter or edge detector available on each LUT output

34.3 Block Diagram

Figure 34-1. Configurable Custom Logic



34.4 Signal Description

Pin Name	Type	Description
OUT[1:0]	Digital output	Output from lookup table
IN[5:0]	Digital input	Input to lookup table

For details on the pin mapping for this peripheral, see *I/O Ports and Peripheral Pin Select (PPS)* from Related Links. One signal can be mapped on several pins.

Related Links

6. I/O Ports and Peripheral Pin Select (PPS)

34.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

34.5.1 I/O Lines

The CCL can take inputs and generate output through I/O pins. For this to function properly, the I/O pins must be configured to be used by a Look Up Table (LUT).

34.5.2 Power Management

This peripheral can continue to operate in any Sleep mode where its source clock is running. Events connected to the event system can trigger other operations in the system without exiting Sleep modes.

34.5.3 Clocks

A generic clock (GCLK_CCL) is optionally required to clock the CCL. This clock must be configured and enabled in the Generic Clock Controller (GCLK) before using input events, filter, edge detection or sequential logic. GCLK_CCL is required when input events, a filter, an edge detector or a sequential sub-module is enabled.

This generic clock is asynchronous to the user interface clock (PB2_CLK).

34.5.4 DMA

Not applicable.

34.5.5 Interrupts

Not applicable.

34.5.6 Events

The CCL can use events from other peripherals and generate events that can be used by other peripherals. For this feature to function, the events have to be configured properly. Refer to the Related Links below for more information about the event users and event generators.

Related Links

28. Event System (EVSYS)

34.5.7 Debug Operation

When the CPU is halted in Debug mode the CCL continues normal operation. However, the CCL cannot be halted when the CPU is halted in Debug mode. If the CCL is configured in a way that requires it to be periodically serviced by the CPU, improper operation or data loss may result during debugging.

34.5.8 Register Access Protection

All registers with write access can be write-protected optionally by the Peripheral Access Controller (PAC). See *Peripheral Access Controller (PAC)* from Related Links.

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

PAC write protection does not apply to accesses through an external debugger.

Related Links

26. Peripheral Access Controller (PAC)

34.5.9 Analog Connections

Not applicable.

34.6 Functional Description

34.6.1 Principle of Operation

Configurable Custom Logic (CCL) is a programmable logic block that can use the device port pins, internal peripherals, and the internal Event System as both input and output channels. The CCL can serve as glue logic between the device and external devices. The CCL can eliminate the need for external logic component and can also help the designer overcome challenging real-time constrains by combining core independent peripherals in clever ways to handle the most time critical parts of the application independent of the CPU.

34.6.2 Operation

34.6.2.1 Initialization

The following bits are enable-protected, meaning that they can only be written when the corresponding even LUT is disabled (LUTCTRLx.ENABLE=0):

- Sequential Selection bits in the Sequential Control x (SEQCTRLx.SEQSEL) register

The following registers are enable-protected, meaning that they can only be written when the corresponding LUT is disabled (LUTCTRLx.ENABLE=0):

- LUT Control x (LUTCTRLx) register, except the ENABLE bit

Enable-protected bits in the LUTCTRLx registers can be written at the same time as LUTCTRLx.ENABLE is written to '1', but not at the same time as LUTCTRLx.ENABLE is written to '0'.

Enable-protection is denoted by the Enable-Protected property in the register description.

34.6.2.2 Enabling, Disabling, and Resetting

The CCL is enabled by writing a '1' to the Enable bit in the Control register (CTRL.ENABLE). The CCL is disabled by writing a '0' to CTRL.ENABLE.

Each LUT is enabled by writing a '1' to the Enable bit in the LUT Control x register (LUTCTRLx.ENABLE). Each LUT is disabled by writing a '0' to LUTCTRLx.ENABLE.

The CCL is reset by writing a '1' to the Software Reset bit in the Control register (CTRL.SWRST). All registers in the CCL will be reset to their initial state, and the CCL will be disabled. Refer to 34.8.1. CTRL for details.

34.6.2.3 Lookup Table Logic

The lookup table in each LUT unit can generate any logic expression OUT as a function of three inputs (IN[2:0]), as shown in Figure 34-2. One or more inputs can be masked. The truth table for the expression is defined by TRUTH bits in LUT Control x register (LUTCTRLx.TRUTH).

Figure 34-2. Truth Table Output Value Selection

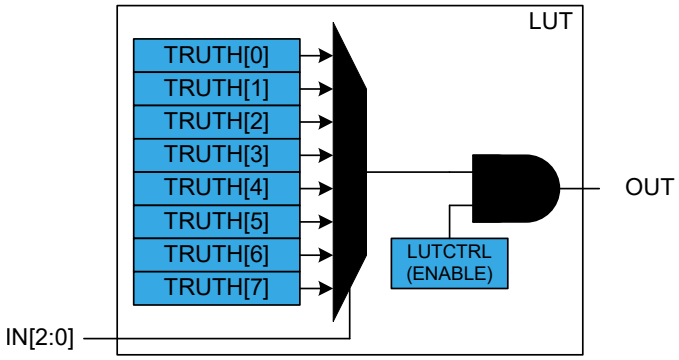


Table 34-1. Truth Table of LUT

IN[2]	IN[1]	IN[0]	OUT
0	0	0	TRUTH[0]
0	0	1	TRUTH[1]
0	1	0	TRUTH[2]
0	1	1	TRUTH[3]
1	0	0	TRUTH[4]
1	0	1	TRUTH[5]
1	1	0	TRUTH[6]
1	1	1	TRUTH[7]

34.6.2.4 Truth Table Inputs Selection

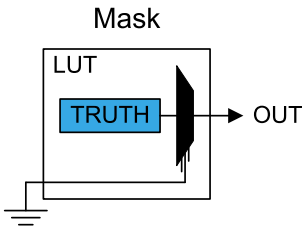
Input Overview
The inputs can be individually:

- Masked
- Driven by peripherals:
  - Analog comparator output (AC)
  - Timer/Counters waveform outputs (TC)
  - Serial Communication output transmit interface (SERCOM)
- Driven by internal events from Event System
- Driven by other CCL sub-modules

The Input Selection for each input ‘y’ of LUT x is configured by writing the Input ‘y’ Source Selection bit in the LUT x Control register (LUTCTRLx.INSELy).

Masked Inputs (MASK)
When a LUT input is masked (LUTCTRLx.INSELy = MASK), the corresponding TRUTH input (IN) is internally tied to zero, as shown in this figure:

Figure 34-3. Masked Input Selection



Internal Feedback Inputs (FEEDBACK)
When selected (LUTCTRLx.INSELy = FEEDBACK), the Sequential (SEQ) output is used as input for the corresponding LUT.

The output from an internal sequential sub-module can be used as input source for the LUT, see figure below for an example for LUT0 and LUT1. The sequential selection for each LUT follows the formula:

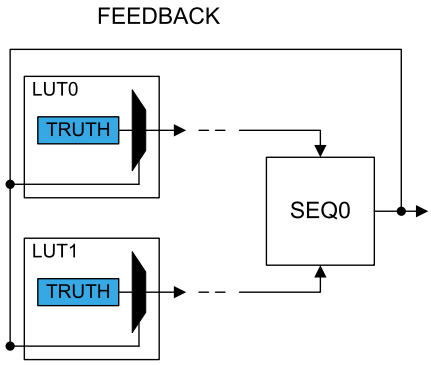
$IN[2N][i] = \mathrm{SEQ}[N]$

$IN[2N+1][i] = \mathrm{SEQ}[N]$

With *N* representing the sequencer number and *i*=0,1,2 representing the LUT input index.

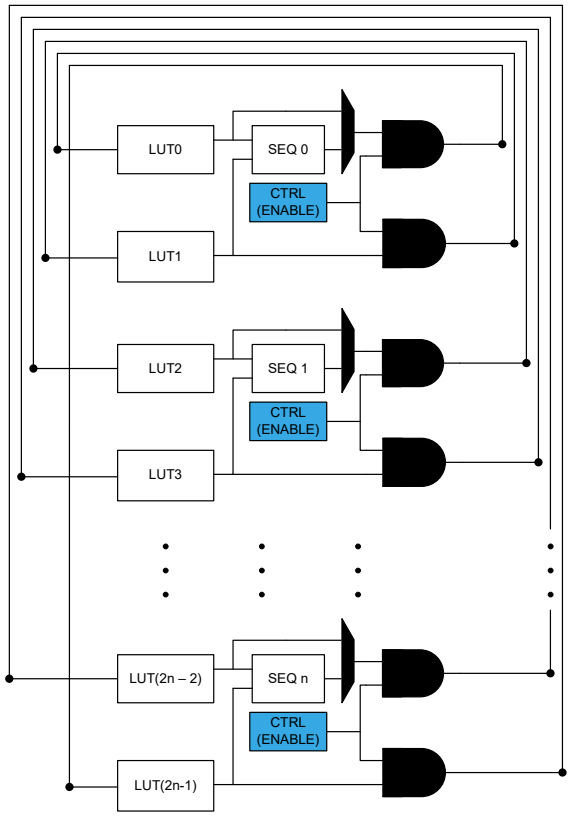
See *Sequential Logic* from Related Links.

Figure 34-4. Feedback Input Selection



Linked LUT (LINK)
When selected (LUTCTRLx.INSELy=LINK), the subsequent LUT output is used as the LUT input (for example, LUT2 is the input for LUT1), as shown in the figure below:

Figure 34-5. Linked LUT Input Selection



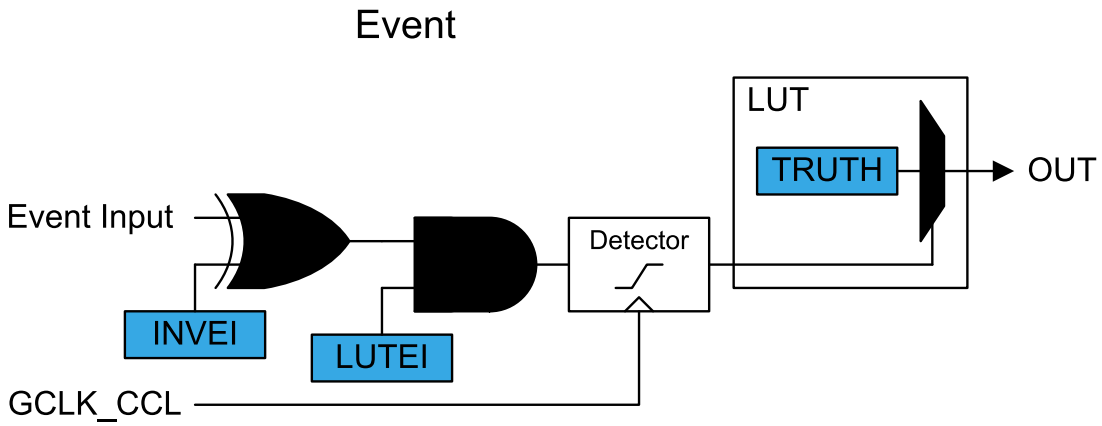
Internal Events Inputs Selection (EVENT)

Asynchronous events from the Event System can be used as input selection, as shown in the following figure. For each LUT, one event input line is available and can be selected on each LUT input. Before enabling the event selection by writing LUTCTRLx.INSELy=EVENT, the Event System must be configured first.

By default, CCL includes an edge detector. When the event is received, an internal strobe is generated when a rising edge is detected. The pulse duration is one GCLK_CCL clock cycle. Writing the LUTCTRLx.INSELy=ASYNCEVENT will disable the edge detector. In this case, it is possible to combine an asynchronous event input with any other input source. This is typically useful with event levels inputs for example, (external I/O pin events). The following steps ensure proper operation:

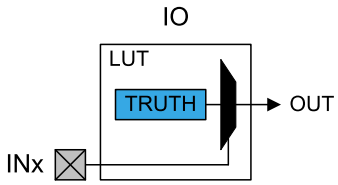
1. Enable the GCLK_CCL clock.
2. Configure the Event System to route the event asynchronously.
3. Select the event input type (LUTCTRLx.INSEL = ASYNCEVENT).
4. If a strobe must be generated on the event input falling edge, write a '1' to the Inverted Event Input Enable bit in the LUT Control register (LUTCTRLx.INVEI) .
5. Enable the event input by writing the Event Input Enable bit in the LUT Control register (LUTCTRLx.LUTEI) to '1'.

Figure 34-6. Event Input Selection



I/O Pin Inputs (IO)
When the I/O pin is selected as LUT input (LUTCTRLx.INSELy = IO), the corresponding LUT input will be connected to the pin, as shown in the figure below.

Figure 34-7. I/O Pin Input Selection



Analog Comparator Inputs (AC)
The AC outputs can be used as input source for the LUT (LUTCTRLx.INSELy=AC).

The analog comparator outputs are distributed following the formula:

$IN[N]\,[i] = AC[N\,\%\,ComparatorOutput\_Number]$

With *N* representing the LUT number and *i*=[0,1,2] representing the LUT input index.

Before selecting the comparator output, the AC must be configured first.

Figure 34-8. AC Input Selection

Timer/Counter Inputs (TC)
The TC waveform output WO[0] can be used as input source for the LUT (LUTCTRLx.INSELy = TC). Only consecutive instances of the TC, that is, TCx and the subsequent TC(x+1), are available as default and alternative TC selections (for example, TC0 and TC1 are sources for LUT0, TC1 and TC2 are sources for LUT1). See the figure below for an example for LUT0. More general, the Timer/Counter selection for each LUT follows the formula:

$\mathrm{IN}[N][i] = DefaultTC[N\,\%\,\mathrm{TC\_Instance\_Number}]$

$\mathrm{IN}[N][i] = AlternativeTC[(N+1)\,\%\,\mathrm{TC\_Instance\_Number}]$

Where *N* represents the LUT number and *i* represents the LUT input index (i=0,1,2).

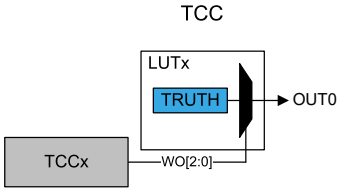
Before selecting the waveform outputs, the TC must be configured first.

Figure 34-9. TC Input Selection

Timer/Counter for Control Application Inputs (TCC)
The TCC waveform outputs can be used as input source for the LUT. Only WO[2:0] outputs can be selected and routed to the respective LUT input (that is, IN0 is connected to WO0, IN1 to WO1, and IN2 to WO2), as shown in the figure below.

Before selecting the waveform outputs, the TCC must be configured first.

Figure 34-10. TCC Input Selection



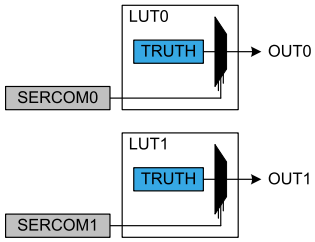
Serial Communication Output Transmit Inputs (SERCOM)
The serial engine transmitter output from Serial Communication Interface (SERCOM TX, TXd for USART, MOSI for SPI) can be used as input source for the LUT. The figure below shows an example for LUT0 and LUT1. The SERCOM selection for each LUT follows the formula:

$\mathrm{IN}[N][i] = SERCOM[N \,\%\, \mathrm{SERCOM\_Instance\_Number}]$

With *N* representing the LUT number and *i*=0,1,2 representing the LUT input index.

Before selecting the SERCOM as input source, the SERCOM must be configured first: the SERCOM TX signal must be output on SERCOMn/pad[0], which serves as input pad to the CCL.

Figure 34-11. SERCOM Input Selection



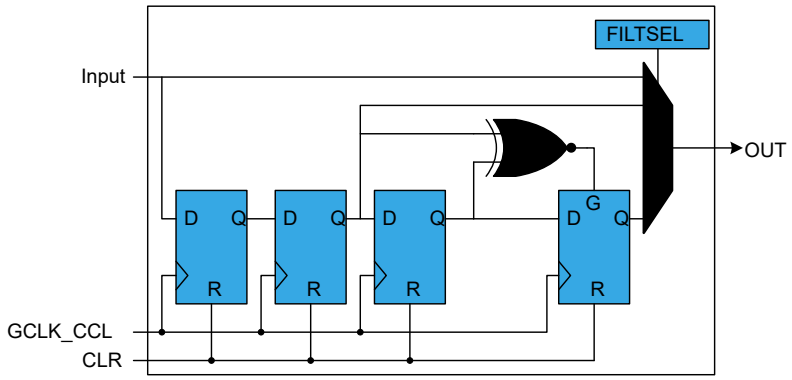
Related Links
34.6.2.7. Sequential Logic

34.6.2.5 Filter

By default, the LUT output is a combinatorial function of the LUT inputs. This may cause some short glitches when the inputs change value. These glitches can be removed by clocking through filters, if demanded by application needs.

The Filter Selection bits in LUT Control register (LUTCTRLx.FILTSEL) define the synchronizer or digital filter options. When a filter is enabled, the OUT output will be delayed by two to five GCLK cycles. One APB clock after the corresponding LUT is disabled, all internal filter logic is cleared.
Note: Events used as LUT input will also be filtered, if the filter is enabled.

Figure 34-12. Filter



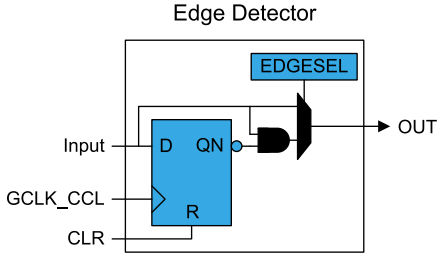
34.6.2.6 Edge Detector

The edge detector can be used to generate a pulse when detecting a rising edge on its input. To detect a falling edge, the TRUTH table must be inverted.

The edge detector is enabled by writing '1' to the Edge Selection bit in LUT Control register (LUTCTRLx.EDGESEL). In order to avoid unpredictable behavior, either the filter or synchronizer must be enabled.

Edge detection is disabled by writing a '0' to LUTCTRLx.EDGESEL. After disabling a LUT, the corresponding internal Edge Detector logic is cleared one APB clock cycle later.

Figure 34-13. Edge Detector



34.6.2.7 Sequential Logic

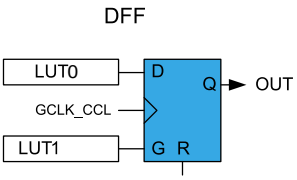
Each LUT pair can be connected to the internal sequential logic, which can be configured to work as D flip flop, JK flip flop, gated D-latch or RS-latch by writing the Sequential Selection bits on the corresponding Sequential Control x register (SEQCTRLx.SEQSEL). Before using sequential logic, the GCLK_CCL clock and optionally each LUT filter or edge detector must be enabled.

Note: While configuring the sequential logic, the even LUT must be disabled. When configured, the even LUT must be enabled.

Gated D Flip-Flop (DFF)

When the DFF is selected, the D-input is driven by the even LUT output LUT0, and the G-input is driven by the odd LUT output LUT1, as shown in the following figure.

Figure 34-14. D Flip Flop



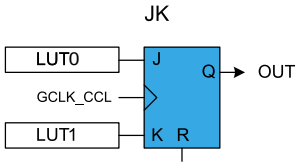
When the even LUT is disabled LUTCTRL0.ENABLE=0, the flip-flop is asynchronously cleared. The reset command (R) is kept enabled for one APB clock cycle. In all other cases, the flip-flop output (OUT) is refreshed on rising edge of the GCLK_CCL, as shown in the following table.

Table 34-2. DFF Characteristics

R	G	D	OUT
1	X	X	Clear
0	1	1	Set
		0	Clear
	0	X	Hold state (no change)

JK Flip-Flop (JK)
When this configuration is selected, the J-input is driven by the even LUT output LUT0, and the K-input is driven by the odd LUT output LUT1, as shown in the following figure.

Figure 34-15. JK Flip Flop



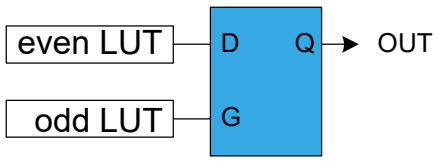
When the even LUT is disabled LUTCTRL0.ENABLE=0, the flip-flop is asynchronously cleared. The reset command (R) is kept enabled for one APB clock cycle. In all other cases, the flip-flop output (OUT) is refreshed on rising edge of the GCLK_CCL, as shown in the following table.

Table 34-3. JK Characteristics

R	J	K	OUT
1	X	X	Clear
0	0	0	Hold state (no change)
0	0	1	Clear
0	1	0	Set
0	1	1	Toggle

Gated D-Latch (DLATCH)
When the DLATCH is selected, the D-input is driven by the even LUT output LUT0, and the G-input is driven by the odd LUT output LUT1, as shown in the following figure.

Figure 34-16. D-Latch



When the even LUT is disabled LUTCTRL0.ENABLE=0, the latch output will be cleared. The G-input is forced enabled for one more APB clock cycle, and the D-input to zero. In all other cases, the latch output (OUT) is refreshed as shown in the following table.

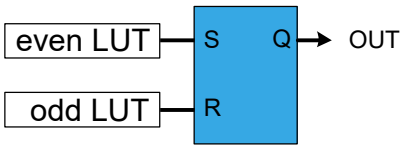
Table 34-4. D-Latch Characteristics

G	D	OUT
0	X	Hold state (no change)
1	0	Clear

.....continued		
G	**D**	**OUT**
1	1	Set

RS Latch (RS)
When this configuration is selected, the S-input is driven by the even LUT output LUT0, and the R-input is driven by the odd LUT output LUT1, as shown in the following figure.

Figure 34-17. RS-Latch



When the even LUT is disabled LUTCTRL0.ENABLE=0, the latch output will be cleared. The R-input is forced enabled for one more APB clock cycle and S-input to zero. In all other cases, the latch output (OUT) is refreshed as shown in the following table.

Table 34-5. RS-Latch Characteristics

S	R	OUT
0	0	Hold state (no change)
0	1	Clear
1	0	Set
1	1	Forbidden state

34.6.3 Events

The CCL can generate the following output events:

- LUTn where n=0-1: Lookup Table Output Value

Writing a '1' to the LUT Control Event Output Enable bit (LUTCTRL.LUTEO) enables the corresponding output event. Writing a '0' to this bit disables the corresponding output event.

The CCL can take the following actions on an input event:

- INSELx where x=0-2: The event is used as input for the TRUTH table. For additional information, refer to 34.5.6. Events.

Writing a '1' to the LUT Control Event Input Enable bit (LUTCTRL.LUTEI) enables the corresponding action on input event. Writing a '0' to this bit disables the corresponding action on input event.

Related Links
28. Event System (EVSYS)

34.6.4 Sleep Mode Operation

When using the GCLK_CCL internal clocking, writing the Run In Standby bit in the Control register (CTRL.RUNSTDBY) to '1' will allow GCLK_CCL to be enabled in Standby Sleep mode.

If CTRL.RUNSTDBY=0, the GCLK_CCL will be disabled in Standby Sleep mode. If the Filter, Edge Detector or Sequential logic are enabled, the LUT output will be forced to zero in STANDBY mode. In all other cases, the TRUTH table decoder will continue operation and the LUT output will be refreshed accordingly.

34.7 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRL	7:0		RUNSTDBY					ENABLE	SWRST
0x01 ... 0x03	Reserved									
0x04	SEQCTRLX	7:0					SEQSEL[3:0]			
0x05 ... 0x07	Reserved									
0x08	LUTCTRL0	7:0	EDGESEL		FILTSEL[1:0]				ENABLE	
		15:8			INSEL2[3:0]				INSEL1[3:0]	
		23:16		LUTEO	LUTEI	INVEI				
		31:24	TRUTH[7:0]							
0x0C	LUTCTRL1	7:0	EDGESEL		FILTSEL[1:0]				ENABLE	
		15:8			INSEL2[3:0]				INSEL1[3:0]	
		23:16		LUTEO	LUTEI	INVEI				
		31:24	TRUTH[7:0]							

34.8 Register Description

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

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the "PAC Write-Protection" property in each individual register description. See *Register Access Protection* from Related Links.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

Related Links

34.5.8. Register Access Protection

34.8.1 Control

Name: CTRL
Offset: 0x00
Reset: 0x00
Property: PAC Write-Protection

Note: CTRL register (except the bits ENABLE & SWRST) is Enable Protected when CCL.CTRL.ENABLE = 1.

Bit	7	6	5	4	3	2	1	0
		RUNSTDBY					ENABLE	SWRST
Access		R/W					R/W	W
Reset		0					0	0

Bit 6 – RUNSTDBY Run in Standby
This bit indicates if the GCLK_CCL clock must be kept running in standby mode. The setting is ignored for configurations where the generic clock is not required. For details refer to 34.6.4. Sleep Mode Operation.

Important: This bit must be written before enabling the CCL.

Value	Description
0	Generic clock is not required in standby sleep mode.
1	Generic clock is required in standby sleep mode.

Bit 1 – ENABLE Enable

Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

Bit 0 – SWRST Software Reset
Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the CCL to their initial state.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

34.8.2 Sequential Control X

Name: SEQCTRLX
Offset: 0x04
Reset: 0x00
Property: PAC Write-Protection, Enable-protected

Note: SEQCTRLX register is Enable-protected when CCL.LUTCTRL0.ENABLE = 1.

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

Bits 3:0 – SEQSEL[3:0] Sequential Selection
These bits select the sequential configuration:
Sequential Selection

Value	Name	Description
0x0	DISABLE	Sequential logic is disabled
0x1	DFF	D flip flop
0x2	JK	JK flip flop
0x3	LATCH	D latch
0x4	RS	RS latch
0x5 - 0xF	—	Reserved

34.8.3 LUT Control n

Name: LUTCTRL
Offset: 0x08 + n*0x04 [n=0..1]
Reset: 0x00000000
Property: PAC Write-Protection, Enable-protected

Note: The LUTCTRLn register is Enable Protected when CCL.LUTCTRLn.ENABLE = 1.

Bit	31	30	29	28	27	26	25	24
	TRUTH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		LUTEO	LUTEI	INVEI				
Access		R/W	R/W	R/W				
Reset		0	0	0				

Bit	15	14	13	12	11	10	9	8
			INSEL2[3:0]				INSEL1[3:0]	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EDGESEL		FILTSEL[1:0]				ENABLE	
Access	R/W		R/W	R/W			R/W	
Reset	0		0	0			0	

Bits 31:24 – TRUTH[7:0] Truth Table
These bits define the value of truth logic as a function of inputs IN[2:0].

Bit 22 – LUTEO LUT Event Output Enable

Value	Description
0	LUT event output is disabled.
1	LUT event output is enabled.

Bit 21 – LUTEI LUT Event Input Enable

Value	Description
0	LUT incoming event is disabled.
1	LUT incoming event is enabled.

Bit 20 – INVEI Inverted Event Input Enable

Value	Description
0	Incoming event is not inverted.
1	Incoming event is inverted.

Bits 8:11, 9:12, 10:13 – INSELx LUT Input x Source Selection
These bits select the LUT input x source:

Value	Name	Description
0x0	MASK	Masked input
0x1	FEEDBACK	Feedback input source
0x2	LINK	Linked LUT input source
0x3	EVENT	Event input source
0x4	IO	I/O pin input source
0x5	AC	AC input source: CMP[0] (LUT0) / CMP[1] (LUT1)

Value	Name	Description
0x6	TC	TC input source: TC0 WO[0] (LUT0) / TC1 WO[0] (LUT1)
0x7	ALTTC	Alternative TC input source: TC1 WO[0] (LUT0) / TC2 WO[0] (LUT1)
0x8	TCC	TCC input source: TCC0 (LUT0) / TCC1 (LUT1)
0x9	SERCOM	SERCOM input source: SERCOM0 PAD0 (LUT0) / SERCOM1 PAD0 (LUT1)
0xA	ALT2TC	1'b0
0xB	ASYNCEVENT	1'b0
0xC - 0xF	Reserved	Reserved

Bit 7 – EDGESEL Edge Selection

Value	Description
0	Edge detector is disabled.
1	Edge detector is enabled.

Bits 5:4 – FILTSEL[1:0] Filter Selection

These bits select the LUT output filter options:

Filter Selection

Value	Name	Description
0x0	DISABLE	Filter disabled
0x1	SYNCH	Synchronizer enabled
0x2	FILTER	Filter enabled
0x3	-	Reserved

Bit 1 – ENABLE LUT Enable

Value	Description
0	The LUT is disabled.
1	The LUT is enabled.

35. True Random Number Generator (TRNG)

35.1 Overview

The True Random Number Generator (TRNG) generates unpredictable random numbers that are not generated by an algorithm. It passes the American NIST Special Publication 800-22 and Diehard Random Tests Suites.

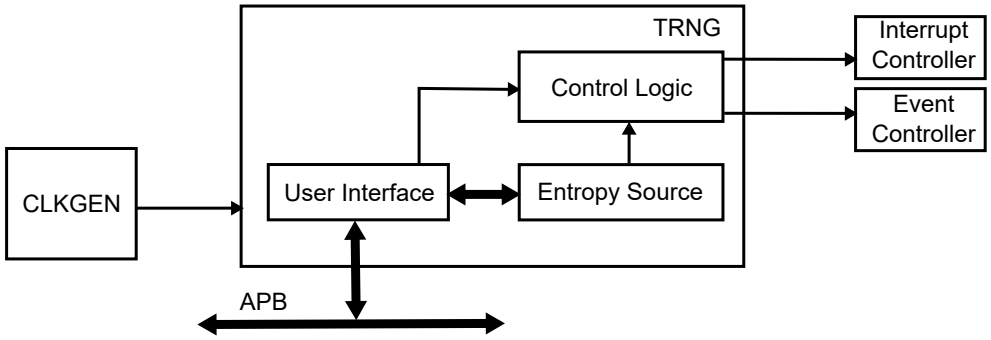
The TRNG may be used as an entropy source for seeding an NIST approved DRNG (Deterministic RNG) as required by FIPS PUB 140-2 and 140-3.

35.2 Features

- Passed NIST Special Publication 800-22 Tests Suite
- Passed Diehard Random Tests Suite
- May be used as Entropy Source for seeding an NIST approved DRNG (Deterministic RNG) as required by FIPS PUB 140-2 and 140-3
- Provides a 32-bit random number every 84 clock cycles

35.3 Block Diagram

Figure 35-1. TRNG Block Diagram.



35.4 Signal Description

Not applicable.

35.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described as follows.

35.5.1 I/O Lines

Not applicable.

35.5.2 Power Management

The functioning of TRNG depends on the sleep mode of device.

The TRNG interrupts can be used to wake up the device from sleep modes. Events connected to the event system can trigger other operations in the system without exiting sleep modes.

Related Links

35.6.5. Sleep Mode Operation

35.5.3 Clocks

The TRNG bus clock () can be enabled and disabled in the CRU module or PMD3.RNGMD bit (see *Peripheral Module Disable Register (PMD)* from Related Links).

Related Links

20. Peripheral Module Disable Register (PMD)

35.5.4 DMA

Not applicable.

35.5.5 Interrupts

The interrupt request line is connected to the interrupt controller. Using the TRNG interrupt(s) requires the interrupt controller to be configured first. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

35.5.6 Events

TRNG can generate Events that are used by the Event System (EVSYS) and EVSYS users.

TRNG cannot use any Events from other peripherals, as it is not an Event User.

Related Links

28. Event System (EVSYS)

35.5.7 Debug Operation

When the CPU is halted in debug mode the TRNG continues normal operation. If the TRNG is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

35.5.8 Register Access Protection

All registers with write access are optionally write-protected by the Peripheral Access Controller (PAC), except the following register:

- Interrupt Flag Status and Clear (INTFLAG) register

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

35.5.9 Analog Connections

Not applicable.

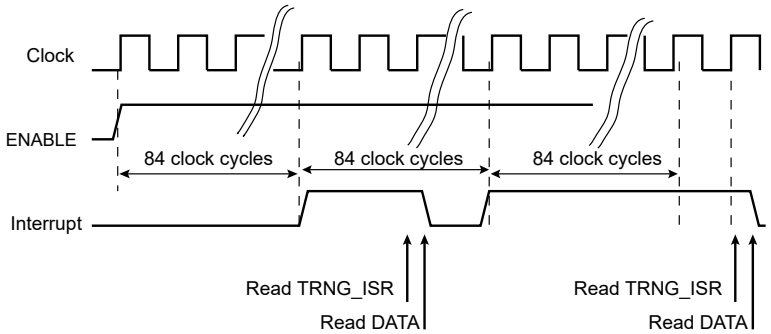
35.6 Functional Description

35.6.1 Principle of Operation

When the TRNG is enabled, the peripheral starts providing new 32-bit random numbers every 84 PB2_CLK clock cycles.

The TRNG can be configured to generate an interrupt or event when a new random number is available.

Figure 35-2. TRNG Data Generation Sequence



35.6.2 Basic Operation

35.6.2.1 Initialization

To operate the TRNG, do the following:

- Ensure PB2_CLK is enabled in the CRU and TRNG is enabled in the PMD3 register, PMD3.RNGMD bit.
- Optional: Enable the output event by writing a '1' to the EVCTRL.DATARDYEO bit.
- Optional: Enable the TRNG to Run in Standby sleep mode by writing a '1' to CTRLA.RUNSTDBY.
- Enable the TRNG operation by writing a '1' to CTRLA.ENABLE.

35.6.2.2 Enabling, Disabling and Resetting

The TRNG is enabled by writing '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The TRNG is disabled by writing a zero to CTRLA.ENABLE.

35.6.3 Interrupts

The TRNG has the following interrupt source:

- Data Ready (DATARDY): Indicates that a new random number is available in the DATA register and ready to be read.
  This interrupt is a synchronous wake-up source. See *Sleep Mode Controller* for details.

The interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.DATARDY) is set to '1' when the interrupt condition occurs. The interrupt can be enabled by writing a '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET.DATARDY), and disabled by writing a '1' to the corresponding bit in the Interrupt Enable Clear (INTENCLR) register.

An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, or the interrupt is disabled. See *INTFLAG* register from Related Links for details on how to clear interrupt flags.

Note that interrupts must be globally enabled for interrupt requests to be generated.

Related Links

35.8.5. INTFLAG

35.6.4 Events

The TRNG can generate the following output event:

- Data Ready (DATARDY): Generated when a new random number is available in the DATA register.

Writing '1' to the Data Ready Event Output bit in the Event Control Register (EVCTRL.DATARDYEO) enables the DTARDY event. Writing a '0' to this bit disables the corresponding output event. Refer to *EVSYS – Event System* for details on configuring the Event System.

Related Links

28. Event System (EVSYS)

35.6.5 Sleep Mode Operation

The Run in Standby bit in Control A register (CTRLA.RUNSTDBY) controls the behavior of the TRNG during standby sleep mode:

When this bit is '0', the TRNG is disabled during sleep, but maintains its current configuration.

When this bit is '1', the TRNG continues to operate during sleep and any enabled TRNG interrupt source can wake up the CPU.

35.7 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0		RUNSTDBY					ENABLE	
0x01 ... 0x03	Reserved									
0x04	EVCTRL	7:0								DATARDYEO
0x05 ... 0x07	Reserved									
0x08	INTENCLR	7:0								DATARDY
0x09	INTENSET	7:0								DATARDY
0x0A	INTFLAG	7:0								DATARDY
0x0B ... 0x1F	Reserved									
0x20	DATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							

35.8 Register Description

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

Some registers require synchronization when read and/or written. Synchronization is denoted by the “Read-Synchronized” and/or “Write-Synchronized” property in each individual register description.

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the “Enable-Protected” property in each individual register description.

35.8.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: PAC Write-Protection

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

Bit 6 – RUNSTDBY Run in Standby

This bit controls how the TRNG behaves during standby sleep mode:

Value	Description
0	The TRNG is halted during standby sleep mode.
1	The TRNG is not stopped in standby sleep mode.

Bit 1 – ENABLE Enable

Value	Description
0	The TRNG is disabled.
1	The TRNG is enabled.

35.8.2 Event Control

Name: EVCTRL
Offset: 0x04
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								DATARDYEO
Access								R/W
Reset								0

Bit 0 – DATARDYEO Data Ready Event Output
This bit indicates whether the Data Ready event output is enabled and whether an output event will be generated when a new random value is ready.

Value	Description
0	Data Ready event output is disabled and an event will not be generated.
1	Data Ready event output is enabled and an event will be generated.

35.8.3 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x08
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set (INTENSET) register.

Bit	7	6	5	4	3	2	1	0
								DATARDY
Access								R/W
Reset								0

Bit 0 – DATARDY Data Ready Interrupt Enable
Writing a '1' to this bit will clear the Data Ready Interrupt Enable bit, which disables the corresponding interrupt request.

Value	Description
0	The DATARDY interrupt is disabled.
1	The DATARDY interrupt is enabled.

35.8.4 Interrupt Enable Set

Name: INTENSET
Offset: 0x09
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

Bit	7	6	5	4	3	2	1	0
								DATARDY
Access								R/W
Reset								0

Bit 0 – DATARDY Data Ready Interrupt Enable
Writing a '1' to this bit will set the Data Ready Interrupt Enable bit, which enables the corresponding interrupt request.

Value	Description
0	The DATARDY interrupt is disabled.
1	The DATARDY interrupt is enabled.

35.8.5 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								DATARDY
Access								R/W
Reset								0

Bit 0 – DATARDY Data Ready

This flag is set when a new random value is generated, and an interrupt will be generated if INTENCLR/SET.DATARDY=1.

This flag is cleared by writing a '1' to the flag or by reading the DATA register. Writing a '0' to this bit has no effect.

35.8.6 Output Data

Name: DATA
Offset: 0x20
Reset: -
Property: -

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	–	–	–	–	–	–	–	–

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	–	–	–	–	–	–	–	–

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	–	–	–	–	–	–	–	–

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	–	–	–	–	–	–	–	–

Bits 31:0 – DATA[31:0] Output Data
These bits hold the 32-bit randomly generated output data.

36. Advanced Encryption Standard (AES)

36.1 Overview

The Advanced Encryption Standard peripheral (AES) provides a means for symmetric-key encryption of 128-bit blocks, in compliance to NIST specifications.

The symmetric-key algorithm requires the same key for both encryption and decryption.

Different key sizes are supported. The key size determines the number of repetitions of transformation rounds that convert the input (called the "plaintext") into the final output ("ciphertext"). The number of rounds of repetition is as follows:

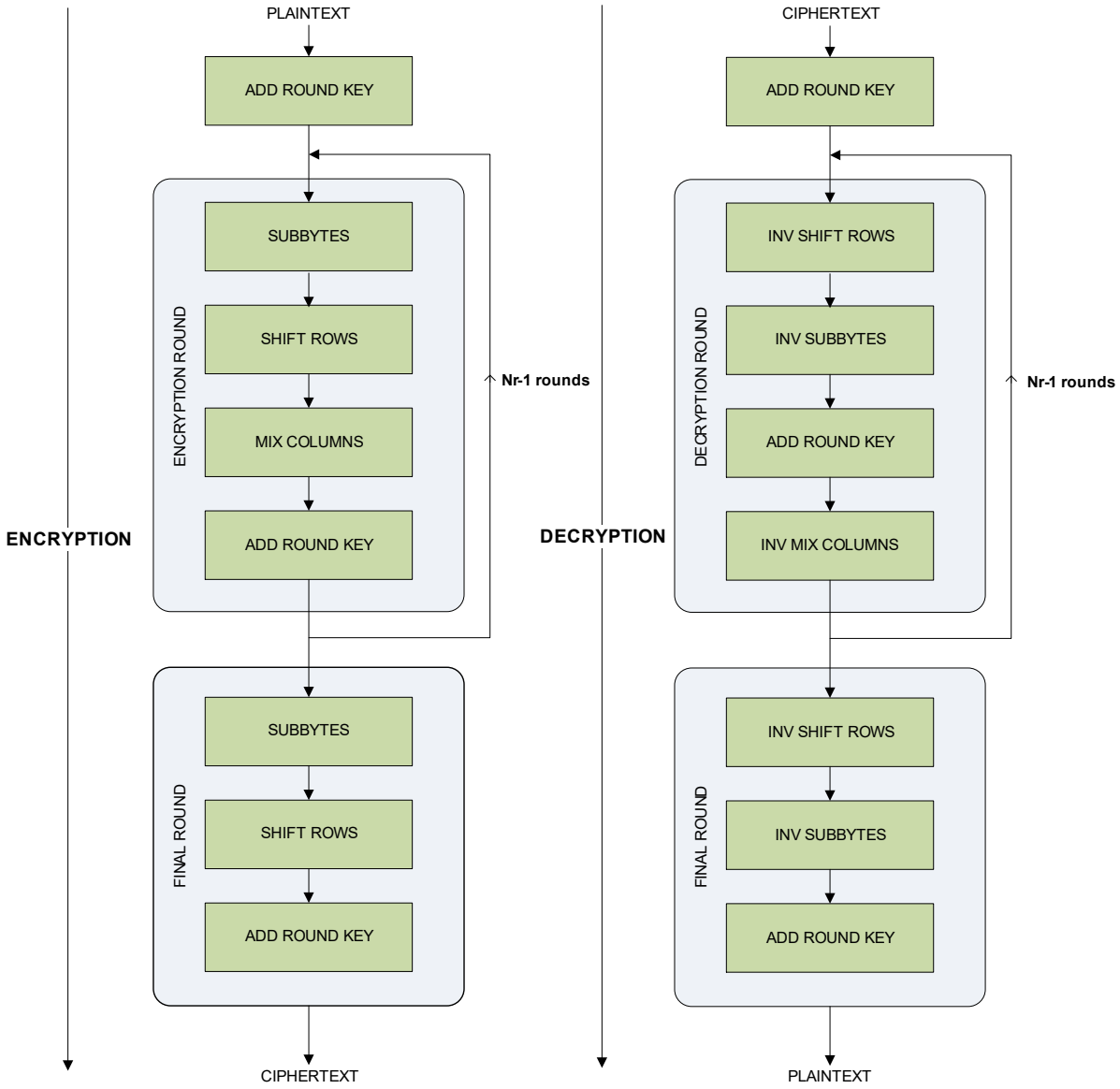
- 10 rounds of repetition for 128-bit keys
- 12 rounds of repetition for 192-bit keys
- 14 rounds of repetition for 256-bit keys

36.2 Features

- Compliant with FIPS Publication 197, Advanced Encryption Standard (AES)
- 128/192/256 bit cryptographic key supported
- Encryption time of 57/67/77 cycles with 128-bit/192-bit/256-bit cryptographic key
- Five confidentiality modes of operation as recommended in NIST Special Publication 800-38A
- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)
- Counter (CTR)
- Supports Counter with CBC-MAC (CCM/CCM*) mode for authenticated encryption
- 8, 16, 32, 64, 128-bit data sizes possible in CFB mode
- Galois Counter mode (GCM) encryption and authentication

36.3 Block Diagram

Figure 36-1. AES Block Diagram



36.4 Signal Description

Not applicable.

36.5 Product Dependencies

In order to use this AES module, other parts of the system must be configured correctly, as described below.

36.5.1 I/O Lines

Not applicable.

36.5.2 Power Management

The AES will continue to operate in Standby sleep mode, if it's source clock is running.

The AES interrupts can be used to wake up the device from Standby sleep mode. Refer to the Power Manager chapter for details on the different sleep modes.

AES is clocked only on the following conditions:

- When the DMA is enabled.
- Whenever there is an APB access for any read and write operation to the AES registers. (Not in Standby sleep mode.)
- When the AES is enabled & encryption/decryption is ongoing.

Related Links

15. Power Management Unit (PMU)

36.5.3 Clocks

The AES bus clock (PB2_CLK) can be enabled and disabled in the CRU module.

36.5.4 DMA

The AES has two DMA request lines; one for input data and one for output data. They are both connected to the DMA Controller (DMAC). These DMA request triggers will be acknowledged by the DMAC ACK signals. Using the AES DMA requests requires the DMA Controller to be configured first. See *Direct Memory Access Controller (DMAC)* from Related Links.

Related Links

22. Direct Memory Access Controller (DMAC)

36.5.5 Interrupts

The interrupt request line is connected to the interrupt controller. Using the AES interrupt requires the interrupt controller to be configured first. Refer to the Processor and Architecture chapter for details.

All the AES interrupts are synchronous wake-up sources. See *Sleep Mode Controller* for details.

Related Links

10. Processor and Architecture

36.5.6 Events

Not applicable.

36.5.7 Debug Operation

When the CPU is halted in debug mode, the AES module continues normal operation. If the AES module is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging. The AES module can be forced to halt operation during debugging.

36.5.8 Register Access Protection

All registers with write access are optionally write-protected by the peripheral access controller (PAC), except the following register:

- Interrupt Flag Register (INTFLAG)

Write protection is denoted by the Write-Protected property in the register description.

Write protection does not apply to accesses through an external debugger. See *Peripheral Access Controller (PAC)* from Related Links.

Related Links

26. Peripheral Access Controller (PAC)

36.5.9 Analog Connections

Not applicable.

36.6 Functional Description

36.6.1 Principle of Operation

The following is a high level description of the algorithm. These are the steps:

- KeyExpansion: Round keys are derived from the cipher key using Rijndael's key schedule.
- InitialRound:
  - AddRoundKey: Each byte of the state is combined with the round key using bitwise XOR.
- Rounds:
  - SubBytes: A non-linear substitution step where each byte is replaced with another according to a lookup table.
  - ShiftRows: A transposition step where each row of the state is shifted cyclically a certain number of steps.
  - MixColumns: A mixing operation which operates on the columns of the state, combining the four bytes in each column.
  - AddRoundKey
- Final Round (no MixColumns):
  - SubBytes
  - ShiftRows
  - AddRoundKey

The relationship between the module's clock frequency and throughput (in bytes per second) is given by:

Clock Frequency = (Throughput/2) x (Nr+1) for 2 byte parallel processing

Clock Frequency = (Throughput/4) x (Nr+1) for 4 byte parallel processing

where Nr is the number of rounds, depending on the key length.

36.6.2 Basic Operation

36.6.2.1 Initialization

The following register is enable-protected:

- Control A (CTRLA)

Enable-protection is denoted by the Enable-Protected property in the register description.

36.6.2.2 Enabling, Disabling, and Resetting

The AES module is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The module is disabled by writing a zero to CTRLA.ENABLE. The module is reset by writing a one to the Software Reset bit in the Control A register (CTRLA.SWRST).

36.6.2.3 Basic Programming

The CIPHER bit in the Control A Register (CTRLA.CIPHER) allows selection between the encryption and the decryption processes. The AES is capable of using cryptographic keys of 128/192/256 bits to encrypt and decrypt data in blocks of 128 bits. The Key Size (128/192/256) can be programmed in the KEYSIZE field in the Control A Register (CTRLA.KEYSIZE). This 128-bit/192-bit/256-bit key is defined in the Key Word Registers (KEYWORD). By setting the XORKEY bit of CTRLA register, keyword can be updated with the resulting XOR value of user keyword and previous keyword content.

The input data for processing is written to a data buffer consisting of four 32-bit registers through the Data register address. The data buffer register (note that input and output data shares the same data buffer register) that is written to when the next write is performed is indicated by the Data Pointer in the Data Buffer Pointer (DATABUFPTR) register. This field is incremented by one or wrapped by hardware when a write to the INDATA register address is performed. This field can also be programmed, allowing the user direct control over which input buffer register to write. Note that when AES module is in the CFB operation mode with the data segment size less than 128 bits, the input data must be written to the first (DATABUFPTR = 0) and second (DATABUFPTR = 1) input buffer registers (see Table 36-1).

The input to the encryption processes of the CBC, CFB and OFB modes includes, in addition to the plaintext, a 128-bit data block called the Initialization Vector (IV), which must be set in the Initialization Vector Registers (INTVECT). Additionally, the GCM mode 128-bit authentication data needs to be programmed. The Initialization Vector is used in the initial step in the encryption of a message and in the corresponding decryption of the message. The Initialization Vector Registers are also used by the Counter mode to set the counter value.

It is necessary to notify AES module whenever the next data block it is going to process is the beginning of a new message. This is done by writing a one to the New Message bit in the Control B register (CTRLB.NEWMSG).

The AES modes of operation are selected by setting the AESMODE field in the Control A Register (CTRLA.AESMODE). In Cipher Feedback Mode (CFB), five data sizes are possible (8, 16, 32, 64 or 128 bits), configurable by means of the CFBS field in the Control A Register (CTRLA.CFBS). In Counter mode, the size of the block counter embedded in the module is 16 bits. Therefore, there is a rollover after processing 1 megabyte of data. The data pre-processing, post-processing and data chaining for the concerned modes are automatically performed by the module.

When data processing has completed, the Encryption Complete bit in the Interrupt Flag register (INTFLAG.ENCCMP) is set by hardware (which triggers an interrupt request if the corresponding interrupt is enabled). The processed output data is read out through the Output Data register (INDATA) address from the data buffer consisting of four 32-bit registers. The data buffer register that is read when the next read is performed is indicated by the Data Pointer field in the Data Buffer Pointer register (DATABUFPTR). This field is incremented by one or wrapped by hardware when a read from the INDATA register address is performed. This field can be programmed, giving the user direct control over which output buffer register to read from. Note that when AES module is in the CFB operation mode with the data segment size less than 128 bits, the output data must be read from the first (DATABUFPTR = 0) and second (DATABUFPTR = 1) output buffer registers (see Table 36-1). The Encryption Complete bit (INTFLAG.ENCCMP) is cleared by hardware after the processed data has been read from the relevant output buffer registers.

Table 36-1. Relevant Input/Output Data Registers for Different Confidentiality Modes

Confidentiality Mode	Relevant Input / Output Data Registers
ECB	All
CBC	All
OFB	All
128-bit CFB	All
64-bit CFB	First and Second
32-bit CFB	First
16-bit CFB	First
8-bit CFB	First
CTR	All

36.6.2.4 Start Modes

The Start mode field in the Control A Register (CTRLA.STARTMODE) allows the selection of encryption start mode.

1. Manual Start Mode
   In the Manual Start Mode the sequence is as follows:
   a. Write the 128/192/256 bit key in the Key Register (KEYWORD)
   b. Write the initialization vector or counter in the Initialization Vector Register (INTVECT). The initialization vector concerns all modes except ECB
   c. Enable interrupts in Interrupt Enable Set Register (INTENSET), depending on whether an interrupt is required or not at the end of processing.
   d. Write the data to be encrypted or decrypted in the Data Registers (INDATA).
   e. Set the START bit in Control B Register (CTRLB.START) to begin the encryption or the decryption process.
   f. When the processing completes, the Encryption Complete bit in the Interrupt Flag Register (INTFLAG.ENCCMP) raises. If Encryption Complete interrupt has been enabled, the interrupt line of the AES is activated.
   g. When the software reads one of the Output Data Registers (INDATA), INTFLAG.ENCCMP bit is automatically cleared.
2. Auto start Mode
   The Auto Start Mode is similar to the manual one, but as soon as the correct number of input data registers is written, processing is automatically started without setting the START bit in the Control B Register. DMA operation uses this mode.
3. Last Output Data Mode (LOD)
   This mode is used to generate message authentication code (MAC) on data in CCM mode of operation. The CCM mode combines counter mode for encryption and CBC-MAC generation for authentication.

When LOD is disabled in CCM mode then counter mode of encryption is performed on the input data block.

When LOD is enabled in CCM mode then CBC-MAC generation is performed. Zero block is used as the initialization vector by the hardware. Reading from the Output Data Register (INDATA) is not required to clear the ENCCMP flag. The ENCCMP flag is automatically cleared by writing into the Input Data Register (INDATA). This allows retrieval of only the last data in several encryption/decryption processes. No output data register reads are necessary between each block of encryption/decryption process.

Note that assembling message depending on the security level identifier in CCM* has to be done in software.

36.6.2.5 Computation of last Nk words of expanded key

The AES algorithm takes the cryptographic key provided by the user and performs a Key Expansion routine to generate an expanded key. The expanded key contains a total of 4(Nr + 1) 32-bit words, where the first Nk (4/6/8 for a 128-/192-/256-bit key) words are the user-provided key. For data encryption, the expanded key is used in the forward direction, i.e., the first four words are used in the initial round of data processing, the second four words in the first round, the third four words in the second round, and so on. On the other hand, for data decryption, the expanded key is used in the reverse direction, i.e.,the last four words are used in the initial round of data processing, the last second four words in the first round, the last third four words in the second round, and so on.

To reduce gate count, the AES module does not generate and store the entire expanded key prior to data processing. Instead, it computes on-the-fly the round key (four 32-bit words) required for the current round of data processing. In general, the round key for the current round of data processing can be computed from the Nk words of the expanded key generated in the previous rounds. When AES module is operating in the encryption mode, the round key for the initial round of data processing is simply the user-provided key written to the KEY registers. On the other hand, when AES module is operating in the decryption mode, the round key for the initial round of data processing is the last four words of the expanded key, which is not available unless AES module has performed at least one encryption process prior to operating in the decryption mode.

In general, the last Nk words of the expanded key must be available before decryption can start. If desired, AES module can be instructed to compute the last Nk words of the expanded key in advance by writing a one to the Key Generate (KEYGEN) bit in the CTRLA register (CTRLA.KEYGEN). The computation takes Nr clock cycles. Alternatively, the last Nk words of the expanded key can be automatically computed by AES module when a decryption process is initiated if they have not been computed in advance or have become invalid. Note that this will introduce a latency of Nr clock cycles to the first decryption process.

36.6.2.6 Hardware Countermeasures against Differential Power Analysis Attacks

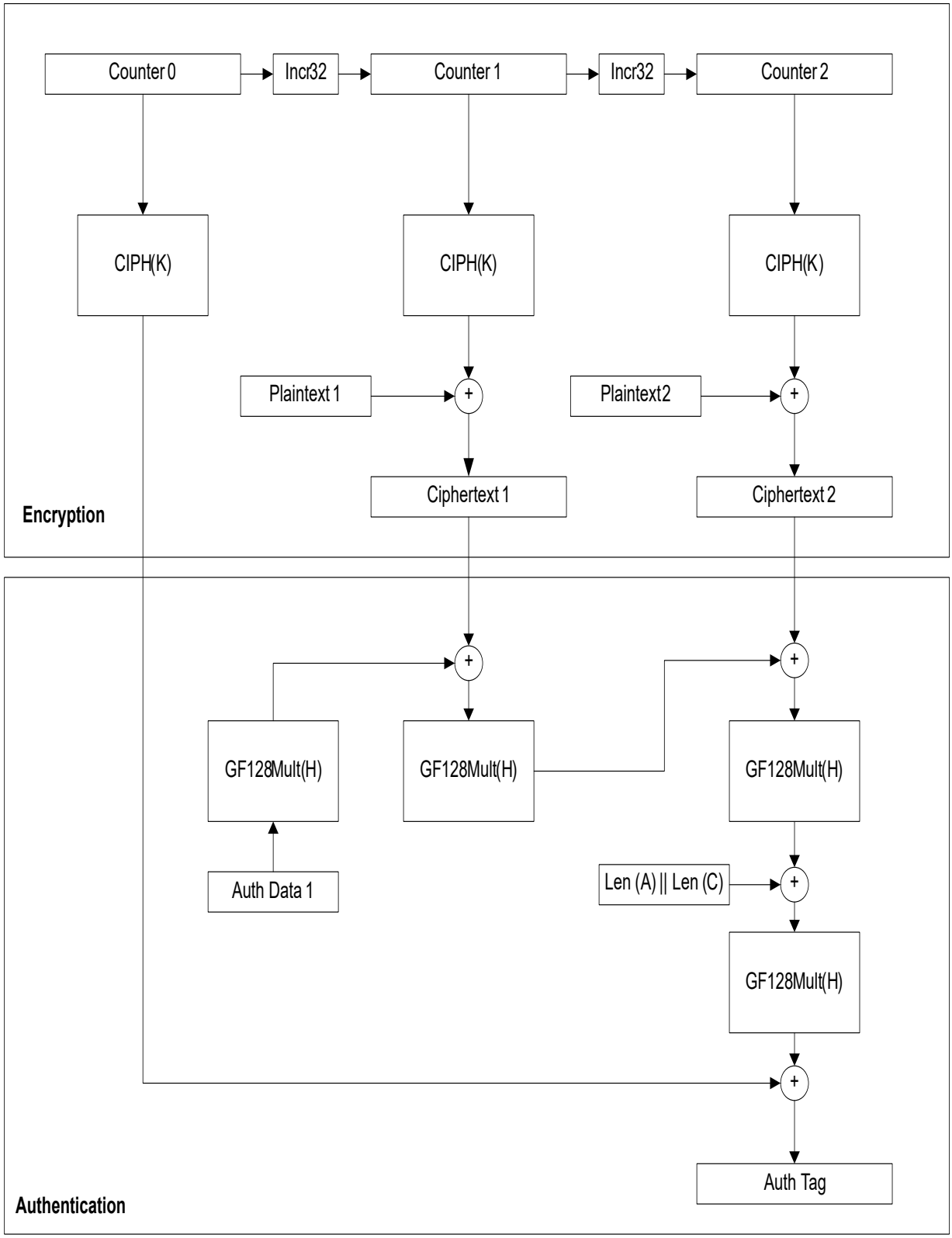
The AES module features four types of hardware countermeasures that are useful for protecting data against differential power analysis attacks:

- Type 1: Randomly add one cycle to data processing
- Type 2: Randomly add one cycle to data processing (other version)
- Type 3: Add a random number of clock cycles to data processing, subject to a maximum of 11/13/15 clock cycles for key sizes of 128/192/256 bits
- Type 4: Add random spurious power consumption during data processing

By default, all countermeasures are enabled, but require a write in DRNGSEED register to be effective. One or more of the countermeasures can be disabled by programming the Countermeasure Type field in the Control A (CTRLA.CTYPE) register. The countermeasures use random numbers generated by a deterministic random number generator embedded in AES module. The seed for the random number generator is written to the RANDSEED register. Note also that a new seed must be written after a change in the keysize. Note that enabling countermeasures reduces AES module's throughput. In short, the throughput is highest with all the countermeasures disabled. On the other hand, with all of the countermeasures enabled, the best protection is achieved but the throughput is worst.

36.6.3 Galois Counter Mode (GCM)

GCM is comprised of the AES engine in CTR mode along with a universal hash function (GHASH engine) that is defined over a binary Galois field to produce a message authentication tag. The GHASH engine processes data packets after the AES operation. GCM provides assurance of the confidentiality of data through the AES Counter mode of operation for encryption. Authenticity of the confidential data is assured through the GHASH engine. Refer to the NIST Special Publication 800-38D Recommendation for more information.


Counter 0
Incr32
Counter 1
Incr32
Counter 2
CIPH(K)
CIPH(K)
CIPH(K)
Plaintext 1
Plaintext2
Ciphertext 1
Ciphertext 2
Encryption
GF128Mult(H)
GF128Mult(H)
GF128Mult(H)
Auth Data 1
Len (A) || Len (C)
GF128Mult(H)
Auth Tag
Authentication

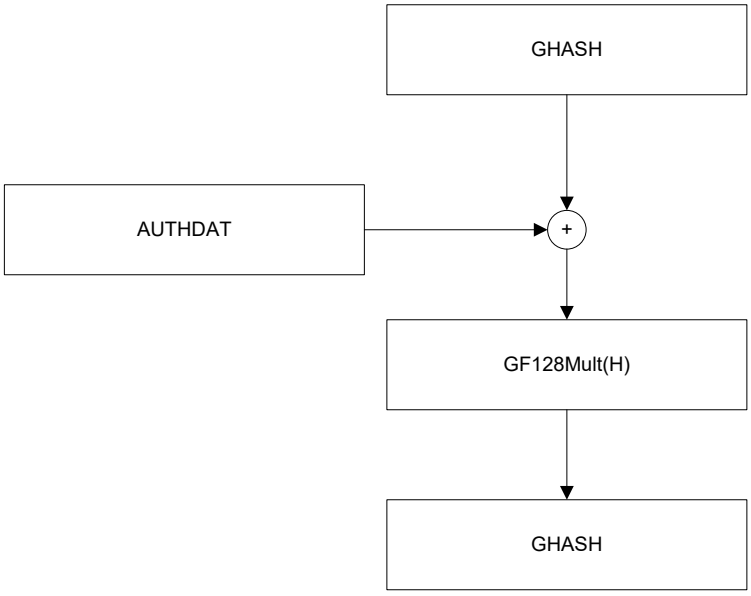
36.6.3.1 GCM Operation

36.6.3.1.1 Hashkey Generation

- Configure CTRLA register as follows:
  a. CTRLA.STARTMODE as Manual (Auto for DMAC)
  b. CTRLA.CIPHER as Encryption
  c. CTRLA.KEYSIZE as per the key used
  d. CTRLA.AESMODE as ECB
  e. CTRLA.CTYPE as per the countermeasures required.
- Set CTRLA.ENABLE
- Write zero to CIPLEN reg.
- Write the key in KEYWORD register
- Write the zeros to INDATA reg
- Set CTRLB.Start.
- Wait for INTFLAG.ENCCMP to be set
- AES Hardware generates Hash Subkey in HASHKEY register.

36.6.3.1.2 Authentication Header Processing

- Configure CTRLA register as follows:
  a. CTRLA.STARTMODE as Manual
  b. CTRLA.CIPHER as Encryption
  c. CTRLA.KEYSIZE as per the key used
  d. CTRLA.AESMODE as GCM
  e. CTRLA.CTYPE as per the countermeasures required.
- Set CTRLA.ENABLE
- Write the key in KEYWORD register
- Set CTRLB.GFMUL
- Write the Authdata to INDATA reg
- Set CTRLB.START as1
- Wait for INTFLAG.GFMCMP to be set.
- AES Hardware generates output in GHASH register
- Continue steps 4 to 7 for remaining Authentication Header.
  Note: If the Auth data is less than 128 bit, it has to be padded with zero to make it 128 bit aligned.



36.6.3.1.3 Plain text Processing

- Set CTRLB.NEWMSG for the new set of plain text processing.
- Load CIPLEN reg.
- Load (J0+1) in INTVECT register.
- As described in NIST documentation J 0 = IV || 0 31 || 1 when len(IV)=96 and J0 =$GHASH_H$ (IV || 0 s+64 || [len(IV)] 64) (s is the minimum number of zeroes that must be padded with the Initialization Vector to make it a multiple of 128) if len(IV) != 96.
- Load plain text in INDATA register.
- Set CTRLB.START as 1.
- Wait for INTFLAG.ENCCMP to be set.
- AES Hardware generates output in INDATA register.
- Intermediate GHASH is stored in GHASH register and Cipher Text available in INDATA register.
- Continue 3 to 6 till the input of plain text to get the cipher text and the Hash keys.
- At the last input, set CTRLB.EOM.
- Write last in-data to INDATA reg.
- Set CTRLB.START as 1.
- Wait for INTFLAG.ENCCMP to be set.
- AES Hardware generates output in INDATA register and final Hash key in GHASH register.
- Load [LEN(A)]64||[LEN(C)]64 in INDATA register and set CTRLB.GFMUL and CTRLB.START as 1.
- Wait for INTFLAG.GFMCMP to be set.
- AES Hardware generates final GHASH value in GHASH register.

36.6.3.1.4 Plain text processing with DMAC

- Set CTRLB.NEWMSG for the new set of plain text processing.
- Load CIPLEN reg.
- Load (J0+1) in INTVECT register.
- Load plain text in INDATA register.
- Wait for INTFLAG.ENCCMP to be set.
- AES Hardware generates output in INDATA register.
- Intermediate GHASH is stored in GHASH register and Cipher Text available in INDATA register.
- Continue 3 to 5 till the input of plain text to get the cipher text and the Hash keys.
- At the last input, set CTRLB.EOM.
- Write last in-data to INDATA reg.
- Wait for INTFLAG.ENCCMP to be set.
- AES Hardware generates output in INDATA register and final Hash key in GHASH register.
- Load [LEN(A)]64||[LEN(C)]64 in INDATA register and set CTRLB.GFMUL and CTRLB.START as 1.
- Wait for INTFLAG.GFMCMP to be set.
- AES Hardware generates final GHASH value in GHASH register.

36.6.3.1.5 Tag Generation

- Configure CTRLA
  a. Set CTRLA.ENABLE to 0
  b. Set CTRLA.AESMODE as CTR
  c. Set CTRLA.ENABLE to 1
- Load J0 value to INITVECTV reg.
- Load GHASH value to INDATA reg.
- Set CTRLB.NEWMSG and CTRLB.START to start the Counter mode operation.
- Wait for INTFLAG.ENCCMP to be set.
- AES Hardware generates the GCM Tag output in INDATA register.