

## Introduction

The PIC32CX-BZ2 family is a general purpose, low-cost, 32-bit Microcontroller (MCU) family of devices supporting multi-protocol wireless interfaces (Bluetooth<sup>®</sup> and Zigbee<sup>®</sup>), 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<sup>2</sup>C), 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<sup>®</sup> Cortex<sup>®</sup> -M4

- 3.35 Coremark<sup>®</sup>/MHz
- 4 KB Combined Instruction Cache and Data Cache
- 8-Zone Memory Protection Unit (MPU)
- Thumb<sup>®</sup>-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<sup>®</sup> 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  $\pm$ 20 ppm Crystal Oscillator (External Low Cost Crystal)
- Two PA Design Architecture (LPA (+5 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:
  - -97 dBm for Bluetooth Low Energy 1 Mbps
  - -94 dBm for Bluetooth Low Energy 2 Mbps
  - -104 dBm for Bluetooth Low Energy 125 Kbps
  - -101 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

- -100 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<sup>2</sup>C up to 1 MHz (three SERCOMs support I<sup>2</sup>C)
  - 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)
- One 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,  $\pm 20$  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,  $\pm 250$  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
- PIC32CX1012BZ24032 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 3.0 Radio.

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

### WBZ45 Module Variants

- WBZ451 Module based on (PIC32CX1012BZ25048 SoC)
  - WBZ451PE (PCB antenna)

- WBZ451UE (u.FL)
- WBZ450 Module based on (PIC32CX1012BZ24032 SoC)
  - WBZ450PE (PCB antenna)
  - 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

- WBZ451 Module Certified to FCC, ISED, CE, UKCA, MIC, KCC, NCC and SRRC Radio Regulations
- WBZ450 Module Certified to FCC, ISED, CE, UKCA, MIC, KCC, NCC and SRRC 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

Introduction.....	1
PIC32CX-BZ2 SoC Family Features.....	1
Operating Conditions of MCUs.....	1
Core: 64 MHz ARM® Cortex® -M4.....	1
Memories.....	1
System.....	2
Supported Connectivity Standards.....	2
Power Supply.....	2
2.4 GHz RF Transceiver.....	2
Bluetooth.....	3
802.15.4/Zigbee Modulation Scheme.....	3
Proprietary.....	4
High Performance Peripherals.....	4
System Peripherals.....	5
Advanced Analog.....	5
Security.....	6
Oscillators.....	6
I/O.....	6
Package.....	6
WBZ45 Module Features.....	6
WBZ45 Module Variants.....	6
Antenna.....	7
Clock Management.....	7
System Peripheral, Advanced Analog and Security.....	7
Package and Operating Conditions.....	7
Certifications.....	7
1. Ordering Information.....	18
1.1. PIC32CX-BZ2 SoC Ordering Information.....	18
1.2. WBZ45 Module Ordering Information.....	18
2. Configuration Summary.....	20
3. PIC32CX-BZ2 SoC Description.....	21
3.1. PIC32CX-BZ2 SoC Block Diagram.....	21
3.2. Pinout Diagram.....	21
4. WBZ45 Module Description.....	24
4.1. Pinout Diagram.....	24
4.2. Basic Connection Requirement.....	26
4.3. WBZ45 Module Placement Guidelines.....	28
4.4. WBZ45 Module Routing Guidelines.....	29
4.5. WBZ45 Module RF Considerations.....	30
4.6. WBZ45 Module Antenna Considerations.....	30
4.7. WBZ45 Module Reflow Profile Information.....	36
4.8. WBZ45 Module Assembly Considerations.....	37

5. Pinout and Signal Descriptions List.....	38
6. I/O Ports and Peripheral Pin Select (PPS).....	40
6.1. Control Registers.....	41
6.2. Peripheral Pin Select (PPS).....	44
6.3. Function Priority for Device Pins.....	56
6.4. I/O Ports Control Registers.....	61
6.5. Operation in Power Saving Modes.....	76
6.6. Results of Various Resets.....	77
7. Power Subsystem.....	78
7.1. Block Diagram.....	78
7.2. Voltage Regulators.....	79
7.3. Power Supply Modes.....	79
7.4. Typical Power Supply Connection for SoC.....	80
7.5. Typical Power Supply Connection for Module.....	80
7.6. Power-Up Sequence.....	81
8. Product Memory Mapping Overview.....	82
9. Prefetch Cache (PCHE).....	85
9.1. Introduction.....	85
9.2. Features.....	85
9.3. Overview.....	85
9.4. Prefetch Behavior.....	87
9.5. Configurations.....	87
9.6. Predictive Prefetch Behavior.....	88
9.7. Coherency Support.....	88
9.8. Effects of Reset.....	88
9.9. Error Conditions.....	89
9.10. Operation in Power-saving Modes.....	90
9.11. Register Summary (PCHE).....	91
9.12. Register Description.....	91
10. Processor and Architecture.....	98
10.1. Cortex M4 Processor.....	98
10.2. Nested Vector Interrupt Controller (NVIC).....	100
10.3. High-Speed Bus System.....	104
11. Cortex M Cache Controller (CMCC).....	107
11.1. Overview.....	107
11.2. Features.....	107
11.3. Block Diagram.....	108
11.4. Signal Description.....	109
11.5. Product Dependencies.....	109
11.6. Functional Description.....	110
11.7. DEBUG Mode.....	112
11.8. RAM Properties.....	113
11.9. Register Summary.....	114
11.10. Register Description.....	114

12. Device Service Unit (DSU).....	127
12.1. Overview.....	127
12.2. Features.....	127
12.3. Block Diagram.....	127
12.4. Signal Description.....	127
12.5. Product Dependencies.....	128
12.6. Debug Operation.....	129
12.7. Chip Erase.....	130
12.8. Programming.....	131
12.9. Intellectual Property Protection.....	131
12.10. Device Identification.....	132
12.11. Functional Description.....	133
12.12. DSU Register Summary.....	136
12.13. Register Description.....	137
13. Clock and Reset Unit (CRU).....	158
13.1. Overview.....	158
13.2. Features.....	158
13.3. Block Diagram.....	159
13.4. System and Peripheral Clock Generation (CLKGEN).....	162
13.5. Idle Mode.....	162
13.6. Dream Mode.....	162
13.7. FRCDIV.....	163
13.8. RFPLL Wrapper.....	163
13.9. Start-up Considerations.....	163
13.10. Fail-Safe Clock Monitor.....	163
13.11. Fast RC Oscillator.....	163
13.12. Secondary Oscillator.....	164
13.13. Low Power RC Oscillator (LPRC).....	164
13.14. Reference Clock Generator.....	164
13.15. Resets.....	165
13.16. Register Summary.....	174
13.17. Register Description.....	176
14. RAM Error Correction Code (RAMECC).....	200
14.1. Overview.....	200
14.2. Features.....	200
14.3. Block Diagram.....	200
14.4. Signal Description.....	200
14.5. Product Dependencies.....	200
14.6. Functional Description.....	201
14.7. Register Summary - RAMECC.....	203
14.8. Register Description.....	203
15. Power Management Unit (PMU).....	210
15.1. Overview.....	210
15.2. Power Modes.....	210
15.3. PMU Register Summary.....	212
15.4. Deep Sleep Control Register .....	213

16. Watchdog Timer (WDT).....	215
16.1. Overview.....	215
16.2. Features.....	215
16.3. Applications.....	215
16.4. Block Diagram.....	217
16.5. Configuration.....	217
16.6. Register Summary - WDT.....	218
16.7. Register Description.....	218
17. Deadman Timer (DMT).....	220
17.1. Overview.....	220
17.2. Features.....	220
17.3. Block Diagram.....	220
17.4. DMT Operation.....	221
17.5. Register Summary.....	224
17.6. Register Description.....	224
18. System Configuration and Register Locking (CFG).....	234
18.1. Overview.....	234
18.2. Applications.....	235
18.3. CFG Register Summary.....	237
18.4. Register Description.....	238
19. Register Locking.....	272
19.1. System Lock Register.....	272
19.2. Register Summary.....	274
19.3. Register Description.....	274
20. Peripheral Module Disable Register (PMD).....	276
20.1. Overview.....	276
20.2. Enabling Peripherals.....	276
20.3. Registers and Bits.....	276
20.4. PMD Register.....	276
20.5. PMDx Initialization Values by Variant Name.....	279
21. Real-Time Counter and Calendar (RTCC).....	281
21.1. Overview.....	281
21.2. Features.....	281
21.3. Block Diagram.....	282
21.4. Signal Description.....	283
21.5. Product Dependencies.....	283
21.6. Functional Description.....	284
21.7. Register Summary - Mode 0 - 32-Bit Counter.....	296
21.8. Register Description - Mode 0 - 32-Bit Counter.....	297
21.9. Register Summary - Mode 1 - 16-Bit Counter.....	318
21.10. Register Description - Mode 1 - 16-Bit Counter.....	319
21.11. Register Summary - Mode 2 - Clock/Calendar.....	340
21.12. Register Description - Mode 2 - Clock/Calendar.....	341
22. Direct Memory Access Controller (DMAC).....	363

22.1. Overview.....	363
22.2. Features.....	363
22.3. Block Diagram.....	365
22.4. Signal Description.....	365
22.5. Product Dependencies.....	365
22.6. Functional Description.....	366
22.7. DMAC Register Summary.....	390
22.8. Register Description.....	394
22.9. DMAC Register Summary (SRAM).....	423
22.10. Register Description - SRAM.....	423
23. External Interrupt Controller (EIC).....	430
23.1. Overview.....	430
23.2. Features.....	430
23.3. Block Diagram.....	430
23.4. Signal Description.....	430
23.5. Product Dependencies.....	431
23.6. Functional Description.....	432
23.7. EIC Register Summary.....	438
23.8. Register Description.....	438
24. Flash Memory.....	453
24.1. Overview.....	453
24.2. Features.....	453
24.3. Functional Block Diagram.....	454
24.4. Flash Memory Addressing.....	454
24.5. Memory Configuration.....	454
24.6. Boot Flash Memory (BFM) Partitions.....	455
24.7. Program Flash Memory (PFM) Partitions.....	456
24.8. Error Correcting Code (ECC) and Flash Programming.....	456
24.9. Interrupts.....	457
24.10. Error Detection .....	457
24.11. NVMKEY Register Unlocking Sequence.....	458
24.12. Word Programming.....	459
24.13. Quad Word Programming.....	460
24.14. Row Programming.....	461
24.15. Page Erase.....	462
24.16. Program Flash Memory (PFM) Erase.....	464
24.17. Pre-Program.....	465
24.18. Device Code Protection bit (CP).....	465
24.19. Operation in Power-Saving Modes.....	465
24.20. Operation in Debug Mode.....	465
24.21. Effects of Various Resets.....	465
24.22. Control Registers.....	465
25. Integrity Check Monitor (ICM).....	484
25.1. Overview.....	484
25.2. Features.....	484
25.3. Block Diagram.....	485



25.4.	Signal Description.....	485
25.5.	Product Dependencies.....	485
25.6.	Functional Description.....	486
25.7.	Register Summary - ICM.....	499
25.8.	Register Description.....	500
26.	Peripheral Access Controller (PAC).....	518
26.1.	Overview.....	518
26.2.	Features.....	518
26.3.	Block Diagram.....	518
26.4.	Product Dependencies.....	518
26.5.	Functional Description.....	519
26.6.	Register Summary.....	523
26.7.	Register Description.....	523
27.	Frequency Meter (FREQM).....	541
27.1.	Overview.....	541
27.2.	Features.....	541
27.3.	Block Diagram.....	541
27.4.	Signal Description.....	541
27.5.	Product Dependencies.....	541
27.6.	Functional Description.....	542
27.7.	Register Summary - FREQM.....	545
27.8.	Register Description.....	545
28.	Event System (EVSYS).....	555
28.1.	Overview.....	555
28.2.	Features.....	555
28.3.	Block Diagram.....	555
28.4.	Product Dependencies.....	556
28.5.	Functional Description.....	557
28.6.	Register Summary.....	563
28.7.	Register Description.....	567
29.	Serial Communication Interface (SERCOM).....	583
29.1.	Overview.....	583
29.2.	Features.....	583
29.3.	Block Diagram.....	584
29.4.	Signal Description.....	584
29.5.	Product Dependencies.....	584
29.6.	Functional Description.....	585
30.	SERCOM Synchronous and Asynchronous Receiver and Transmitter (SERCOM USART).....	591
30.1.	Overview.....	591
30.2.	USART Features.....	591
30.3.	Block Diagram.....	592
30.4.	Signal Description.....	592
30.5.	Product Dependencies.....	592
30.6.	Functional Description.....	594
30.7.	Register Summary.....	612

30.8. Register Description.....	612
31. SERCOM Serial Peripheral Interface (SERCOM SPI).....	639
31.1. Overview.....	639
31.2. Features.....	639
31.3. Block Diagram.....	640
31.4. Signal Description.....	640
31.5. Product Dependencies.....	640
31.6. Functional Description.....	642
31.7. Register Summary.....	652
31.8. Register Description.....	652
32. SERCOM Inter-Integrated Circuit (SERCOM I <sup>2</sup> C).....	671
32.1. Overview.....	671
32.2. Features.....	671
32.3. Block Diagram.....	672
32.4. Signal Description.....	672
32.5. Product Dependencies.....	672
32.6. Functional Description.....	674
32.7. Register Summary - I <sup>2</sup> C Client.....	692
32.8. Register Description - I <sup>2</sup> C Client.....	692
32.9. Register Summary - I <sup>2</sup> C Host.....	708
32.10. Register Description - I <sup>2</sup> C Host.....	708
33. Quad Serial Peripheral Interface (QSPI).....	727
33.1. Overview.....	727
33.2. Features.....	727
33.3. Block Diagram.....	728
33.4. Signal Description.....	728
33.5. Product Dependencies.....	728
33.6. Functional Description.....	730
33.7. Register Summary.....	746
33.8. Register Description.....	747
34. Configurable Custom Logic (CCL).....	768
34.1. Overview.....	768
34.2. Features.....	768
34.3. Block Diagram.....	768
34.4. Signal Description.....	769
34.5. Product Dependencies.....	769
34.6. Functional Description.....	770
34.7. Register Summary.....	780
34.8. Register Description.....	780
35. True Random Number Generator (TRNG).....	785
35.1. Overview.....	785
35.2. Features.....	785
35.3. Block Diagram.....	785
35.4. Signal Description.....	785
35.5. Product Dependencies.....	785

35.6. Functional Description.....	786
35.7. Register Summary.....	789
35.8. Register Description.....	789
36. Advanced Encryption Standard (AES).....	796
36.1. Overview.....	796
36.2. Features.....	796
36.3. Block Diagram.....	797
36.4. Signal Description.....	798
36.5. Product Dependencies.....	798
36.6. Functional Description.....	799
36.7. Register Summary.....	807
36.8. Register Description.....	809
37. Public Key Cryptography Controller (PUKCC).....	825
37.1. Overview.....	825
37.2. Product Dependencies.....	825
37.3. Functional Description.....	826
38. Analog-to-Digital Converter (ADC).....	937
38.1. Overview.....	937
38.2. ADC Operation.....	938
38.3. ADC Module Configuration.....	941
38.4. Additional ADC Functions.....	951
38.5. Interrupts.....	957
38.6. Power-Saving Modes of Operation.....	959
38.7. Effects of Reset.....	961
38.8. Transfer Function.....	961
38.9. ADC Sampling Requirements.....	961
38.10. Connection Considerations.....	962
38.11. Register Description.....	963
39. Analog Comparators (AC).....	1007
39.1. Overview.....	1007
39.2. Features.....	1007
39.3. Block Diagram.....	1008
39.4. Product Dependencies.....	1008
39.5. Functional Description.....	1010
39.6. Register Summary.....	1019
39.7. Register Description.....	1019
40. Timer/Counter (TC).....	1035
40.1. Overview.....	1035
40.2. Features.....	1035
40.3. Block Diagram.....	1036
40.4. Signal Description.....	1036
40.5. Product Dependencies.....	1037
40.6. Functional Description.....	1038
40.7. Register Description.....	1054

41. Timer/Counter for Control Applications (TCC).....	1114
41.1. Overview.....	1114
41.2. Features.....	1114
41.3. Block Diagram.....	1115
41.4. Signal Description.....	1116
41.5. Product Dependencies.....	1116
41.6. Functional Description.....	1117
41.7. Register Summary.....	1150
41.8. Register Description.....	1152
42. Zigbee Bluetooth Radio Subsystem (ZBT).....	1190
42.1. Overview.....	1190
42.2. Features.....	1190
42.3. Wireless Subsystem Top Level Diagram.....	1193
42.4. Bluetooth Link Controller .....	1194
42.5. Zigbee/Proprietary Data Rate Link Controller.....	1195
42.6. Radio Arbiter.....	1196
42.7. RF Physical Layer .....	1196
42.8. Frequency Synthesizer .....	1197
42.9. RFLDO.....	1197
43. Electrical Characteristics.....	1198
43.1. Absolute Maximum Electrical Characteristics.....	1198
43.2. DC Electrical Characteristics.....	1198
43.3. Thermal Specifications.....	1199
43.4. Power Supply DC Module Electrical Specifications.....	1199
43.5. Active Current Consumption DC Electrical Specifications (85°C).....	1202
43.6. Active Current Consumption DC Electrical Specifications (125°C).....	1202
43.7. Idle Current Consumption DC Electrical Specifications (85°C).....	1204
43.8. Idle Current Consumption DC Electrical Specifications (125°C).....	1205
43.9. Sleep Current Consumption DC Electrical Specifications (85°C).....	1206
43.10. Sleep Current Consumption DC Electrical Specifications (125°C).....	1207
43.11. Deep Sleep Current Consumption DC Electrical Specifications (85°C).....	1210
43.12. Deep Sleep Current Consumption DC Electrical Specifications (125°C).....	1210
43.13. XDS (Extreme Deep Sleep) Current Consumption DC Electrical Specifications (85°C).....	1212
43.14. XDS (Extreme Deep Sleep) Current Consumption DC Electrical Specifications (125°C).....	1213
43.15. I/O PIN AC/DC Electrical Specifications.....	1214
43.16. External XTAL and Clock AC Electrical Specifications.....	1215
43.17. XOSC32 AC Electrical Specifications.....	1217
43.18. Low Power Internal 32 kHz RC Oscillator AC Electrical Specifications.....	1219
43.19. FRC AC Electrical Specifications.....	1219
43.20. Frequency AC Electrical Specifications.....	1220
43.21. ADC Electrical Specifications.....	1221
43.22. Comparator AC Electrical Specifications.....	1224
43.23. SPI Module Electrical Specifications.....	1225
43.24. UART AC Electrical Specifications.....	1228
43.25. I <sup>2</sup> C Module Electrical Specifications.....	1229
43.26. QSPI Module Electrical Specifications.....	1233
43.27. TCx Timer Capture Module AC Electrical Specifications.....	1235

43.28.	TCCx Timer Capture Module AC Electrical Specifications.....	1236
43.29.	FLASH NVM AC Electrical Specifications.....	1237
43.30.	Frequency Meter AC Electrical Specifications.....	1237
43.31.	Bluetooth Low Energy RF Characteristics.....	1238
43.32.	Zigbee RF Characteristics.....	1250
44.	Packaging Information.....	1261
44.1.	PIC32CX1012BZ25048 SoC Packaging Information.....	1262
44.2.	PIC32CX1012BZ24032 SoC Packaging Information.....	1265
44.3.	WBZ451 Module Packaging Information.....	1268
44.4.	WBZ450 Module Packaging Information .....	1271
45.	Appendix A: Regulatory Approval.....	1274
45.1.	United States.....	1276
45.2.	Canada.....	1277
45.3.	Europe.....	1278
45.4.	Japan.....	1280
45.5.	Korea.....	1280
45.6.	Taiwan.....	1281
45.7.	China.....	1282
45.8.	UKCA (UK Conformity Assessed).....	1283
45.9.	Other Regulatory Information.....	1284
46.	Document Revision History.....	1285
	Microchip Information.....	1286
	The Microchip Website.....	1286
	Product Change Notification Service.....	1286
	Customer Support.....	1286
	Product Identification System.....	1287
	Microchip Devices Code Protection Feature.....	1287
	Legal Notice.....	1287
	Trademarks.....	1288
	Quality Management System.....	1289
	Worldwide Sales and Service.....	1290

## 1. Ordering Information

This chapter provides the ordering information of the PIC32CX-BZ2 SoC and the WBZ45 Module.

### 1.1 PIC32CX-BZ2 SoC Ordering Information

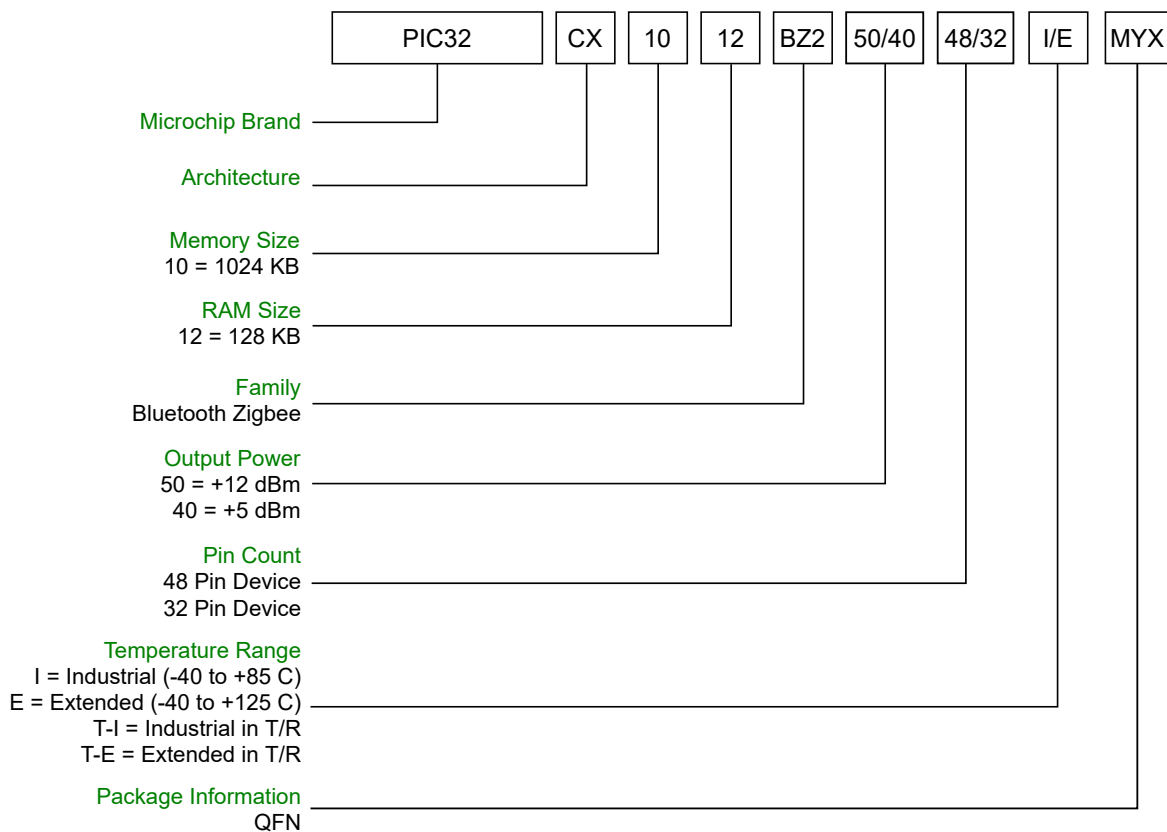
The following table describes the ordering information of the PIC32CX-BZ2 SoC.

**Table 1-1.** PIC32CX-BZ2 SoC Ordering Details

SoC Name	Pin and Package	Description	Ordering Code
PIC32CX1012BZ25048	48-pin and QFN (7 mm x 7 mm x 0.9 mm)	32-bit ARM®Cortex™-M4 with MCU Bluetooth/Zigbee Connectivity	PIC32CX1012BZ25048-I/MYX
PIC32CX1012BZ24032	32-pin and QFN (5 mm x 5 mm x 1 mm)		PIC32CX1012BZ24032-I/MYX

The following figure illustrates the details of the PIC32CX-BZ2 ordering information.

**Figure 1-1.** PIC32CXBZ2 Ordering Information



### 1.2 WBZ45 Module Ordering Information

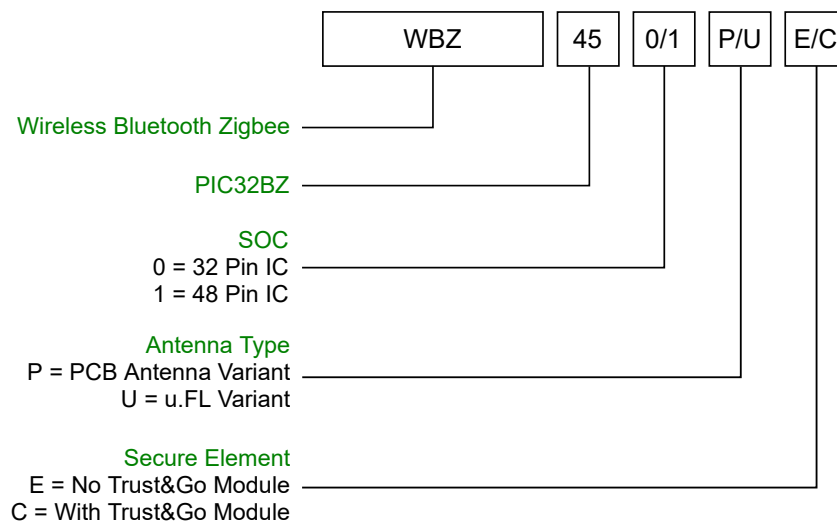
The following table describes the ordering information of the WBZ45 Module.

**Table 1-2.** WBZ45 Module Ordering Details

Model No.	Module SoC	Description	Regulatory Certification	Ordering Code
WBZ450PE	PIC32CX1012BZ24032-I/MYX	WBZ450 module with PCB antenna	FCC, ISED, CE, UKCA, MIC, KCC, NCC and SRRC	WBZ450PE-I
WBZ450UE		WBZ450 module with U.FL connector for external antenna	FCC, ISED, CE, UKCA, MIC, KCC, NCC and SRRC	WBZ450UE-I
WBZ451PE	PIC32CX1012BZ25048-I/MYX	WBZ451 module with PCB antenna	FCC, ISED, CE, UKCA, MIC, KCC, NCC and SRRC	WBZ451PE-I
WBZ451UE		WBZ451 module with U.FL connector for external antenna	FCC, ISED, CE, UKCA, MIC, KCC, NCC and SRRC	WBZ451UE-I

The following figure illustrates the details of the WBZ45 ordering information.

**Figure 1-2.** WBZ45 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 Meter	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 <sup>(1)</sup>	2	8	Y	Y	Y	Y	Y	12	Y	Y
PIC32CX1012BZ24032			32	QFN	2												16 <sup>(1)</sup>	1	6						4		
WBZ451			39	LGA	4												29	2	8						12		
WBZ450			30	LGA	2												16	1	6						4		

**Note:**

1. Only when SOSC is not used.



### 3. PIC32CX-BZ2 SoC Description

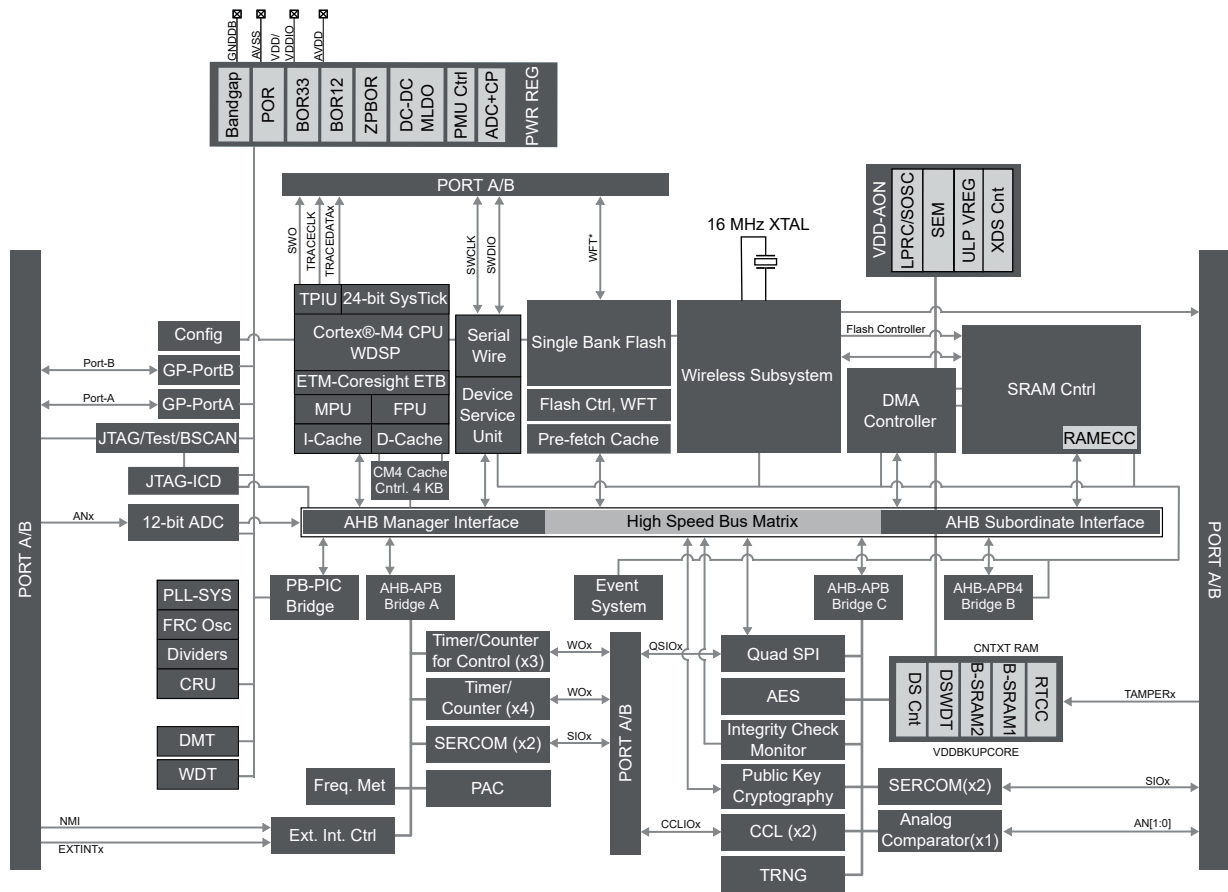
This chapter contains device-specific information for the PIC32CX-BZ2 SoC.

**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

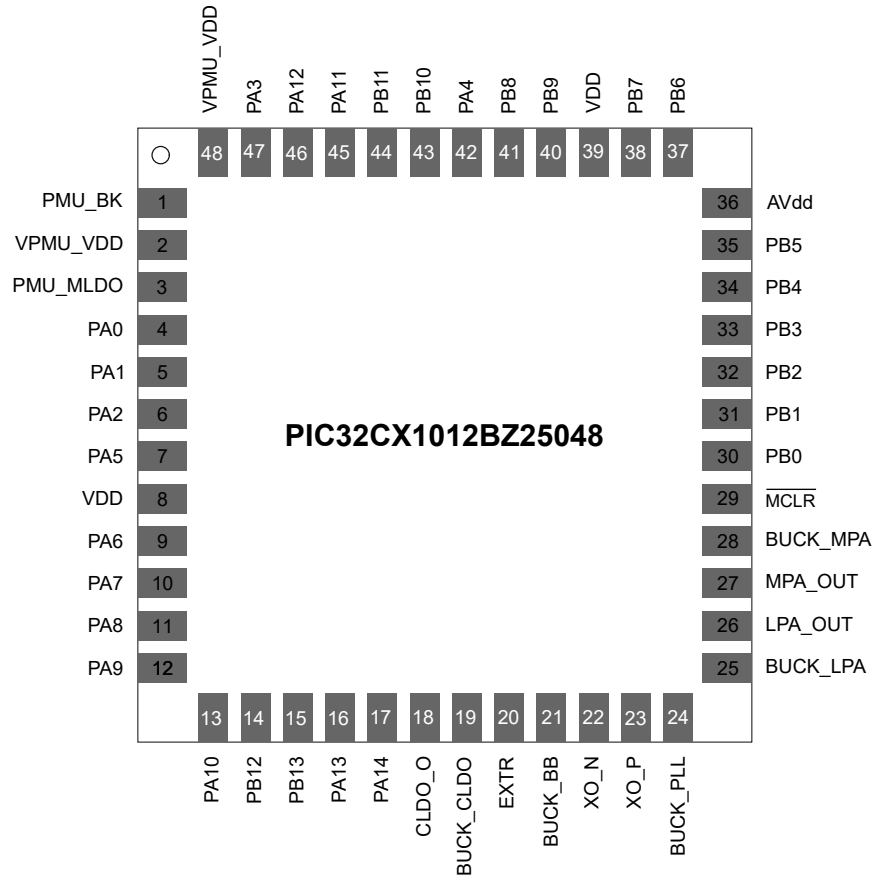
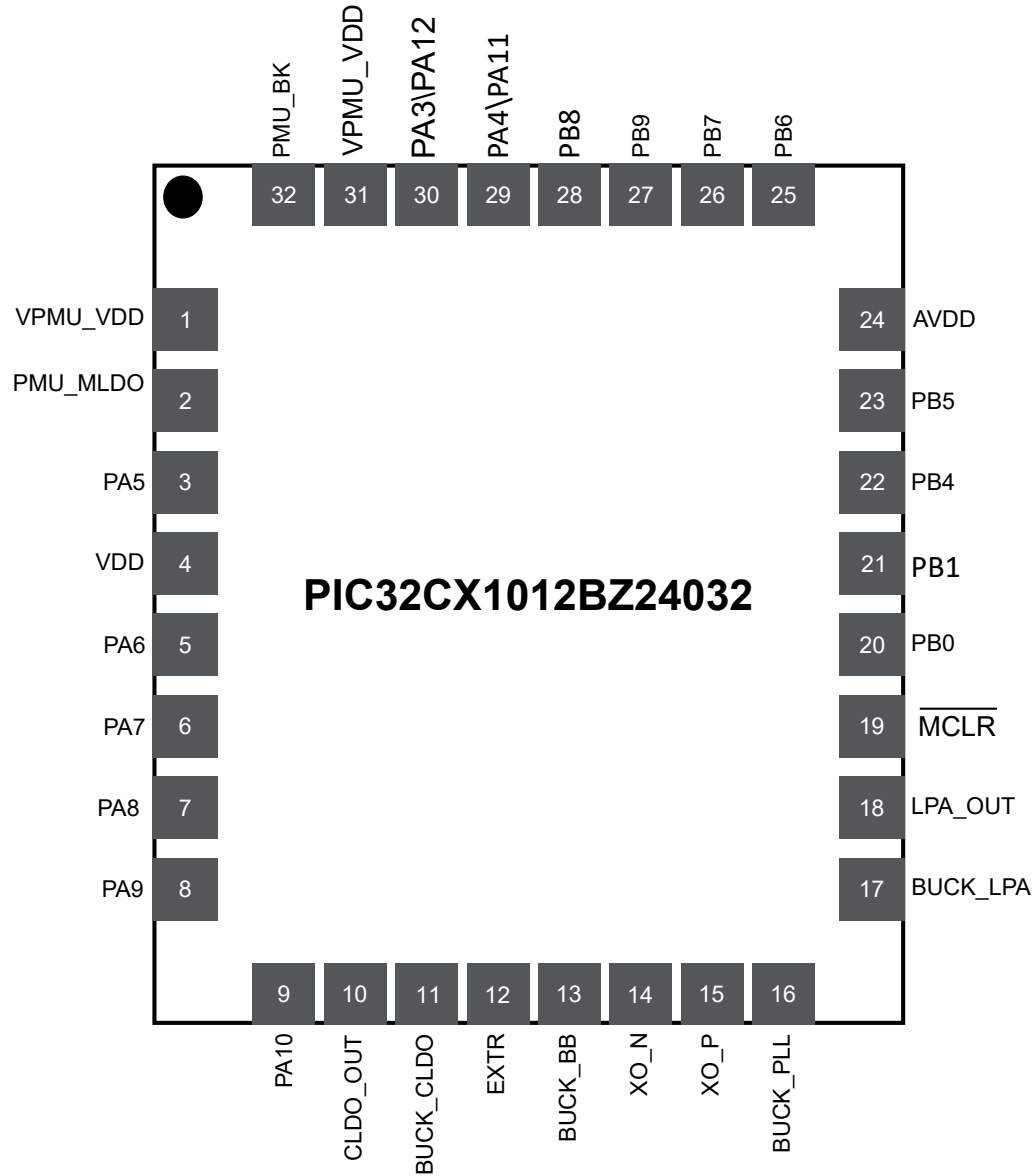


Figure 3-3. PIC32CX1012BZ24032 SoC Pin Diagram



**Note:**

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

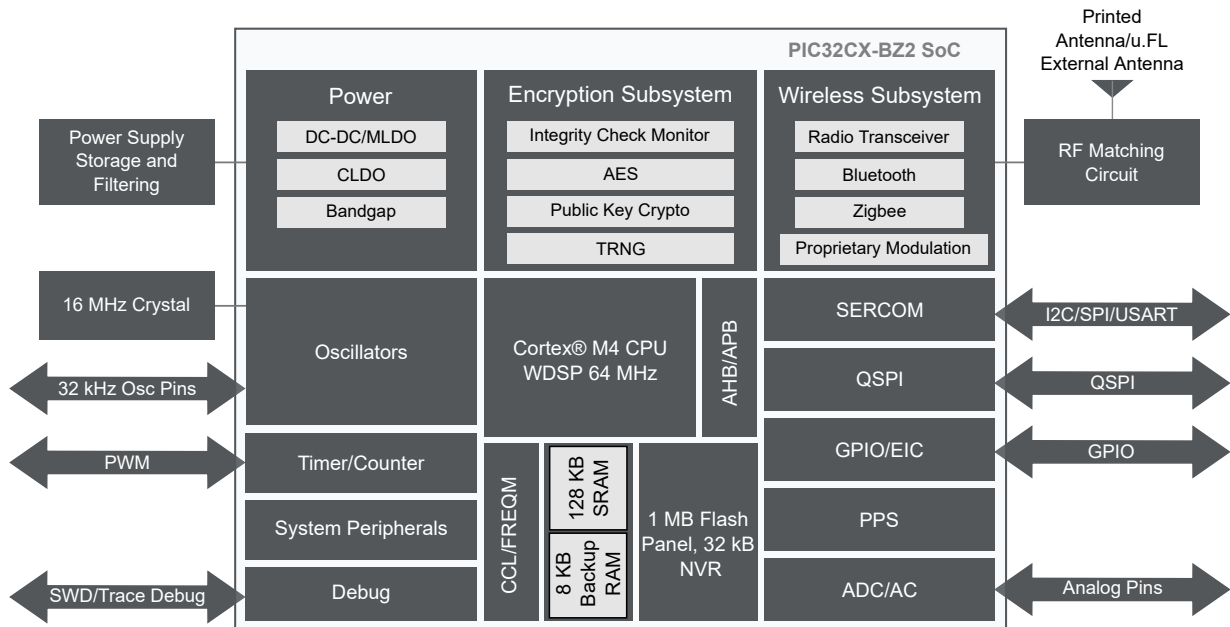
## 4. WBZ45 Module Description

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

- PCB antenna
- u.FL connector for external antenna

The following figure illustrates the WBZ45 Module block diagram.

**Figure 4-1.** WBZ45 Module Block Diagram



### 4.1 Pinout Diagram

The following figures illustrate the module pinout diagram.

Figure 4-2. WBZ450 Module Pin Diagram

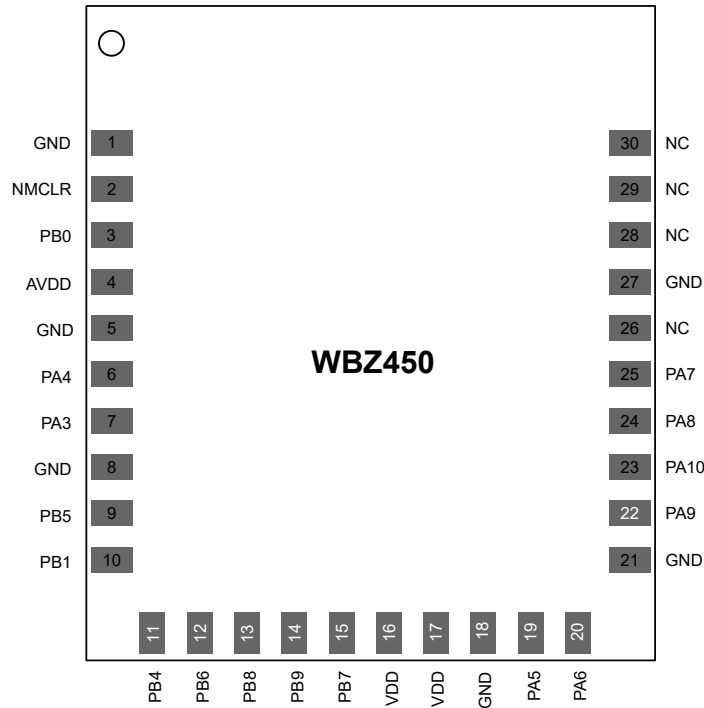
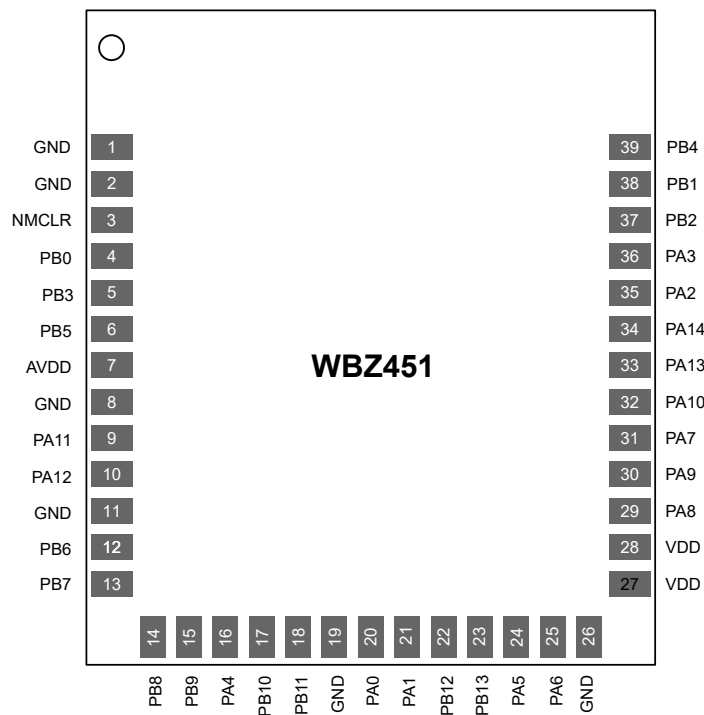


Figure 4-3. WBZ451 Module Pin Diagram



**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 4-4. WBZ450 Module Basic Connection and Interface Diagram

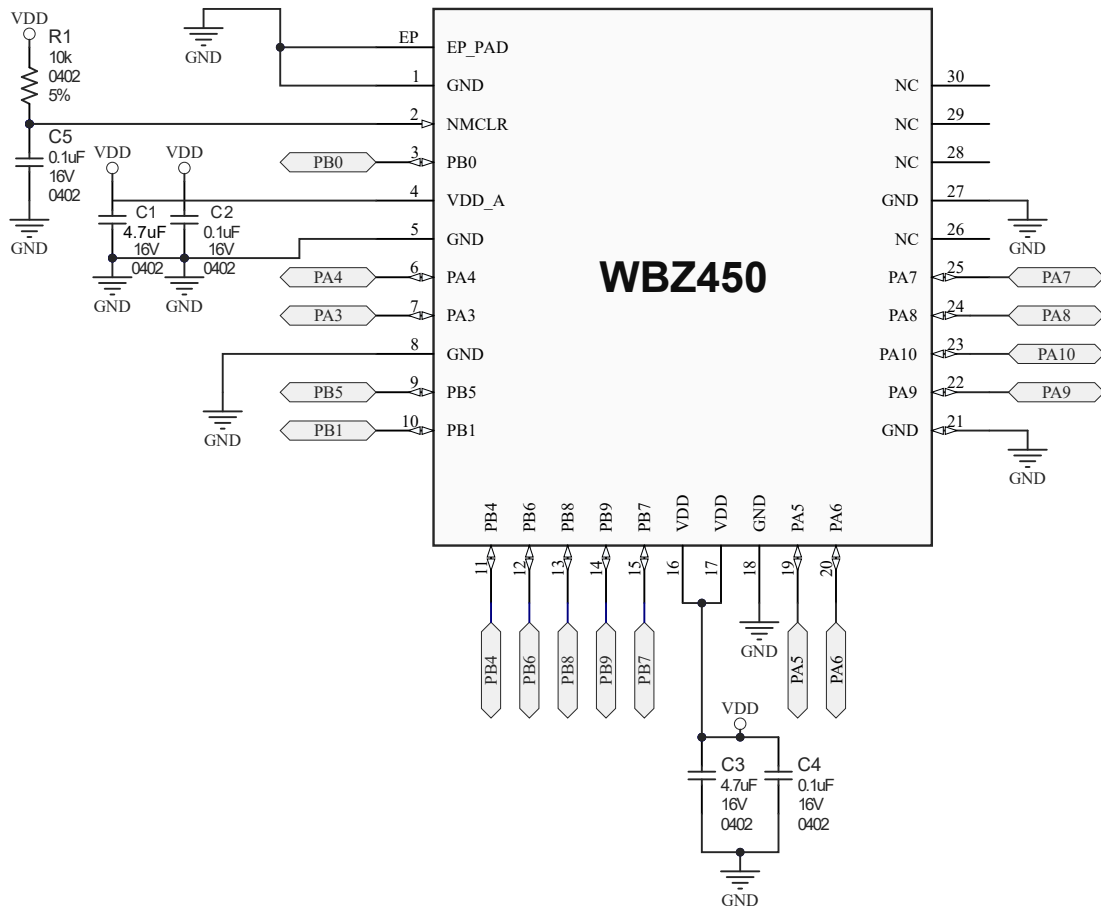
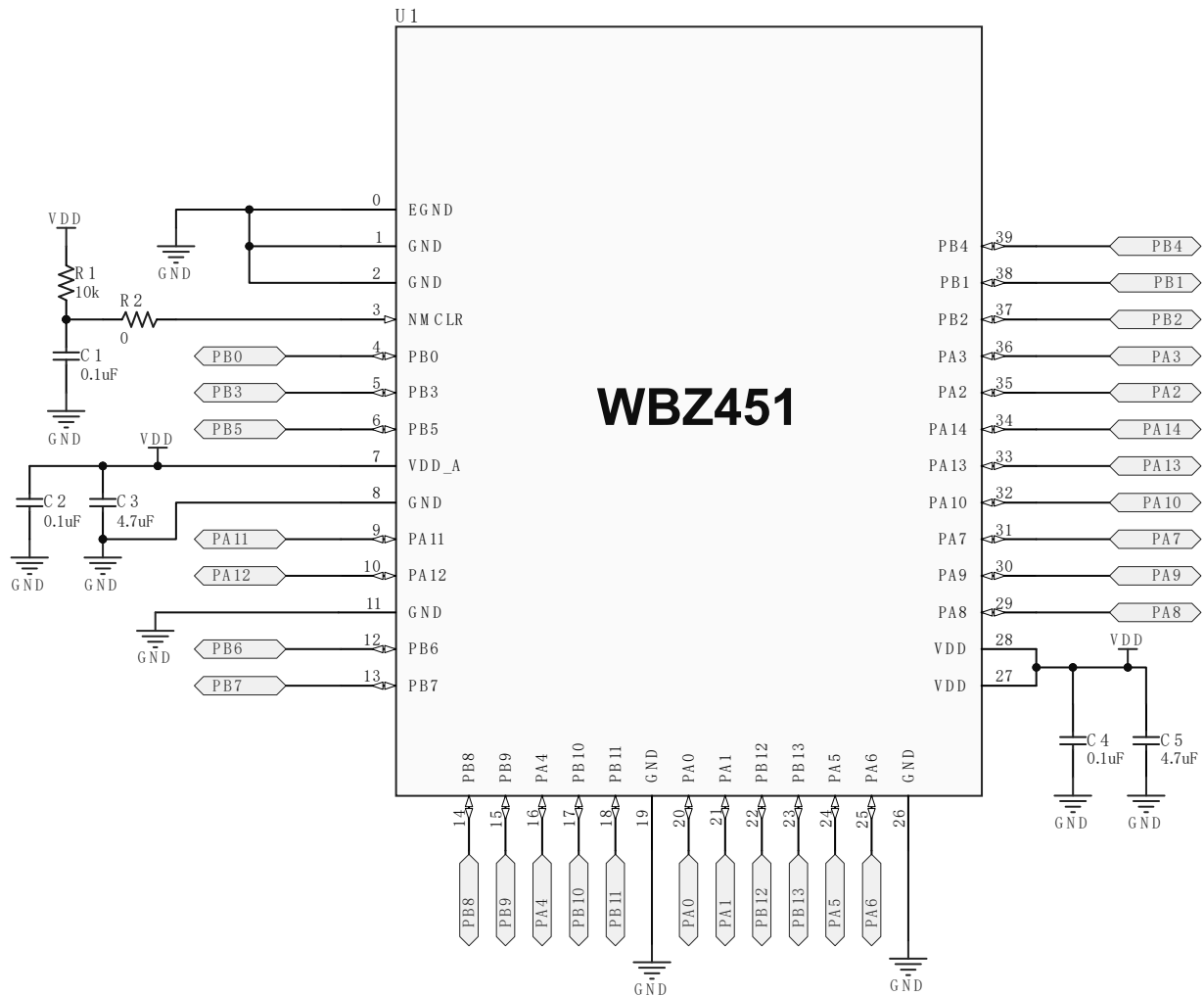


Figure 4-5. WBZ451 Module Basic Connection and Interface Diagram



#### 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  $\mu$ F be on the VDD\_A pin and 4.7  $\mu$ F and a 0.1  $\mu$ F 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  $\mu$ F) 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  $\Omega$ s, not to exceed 100 $\Omega$ .

**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.

##### 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

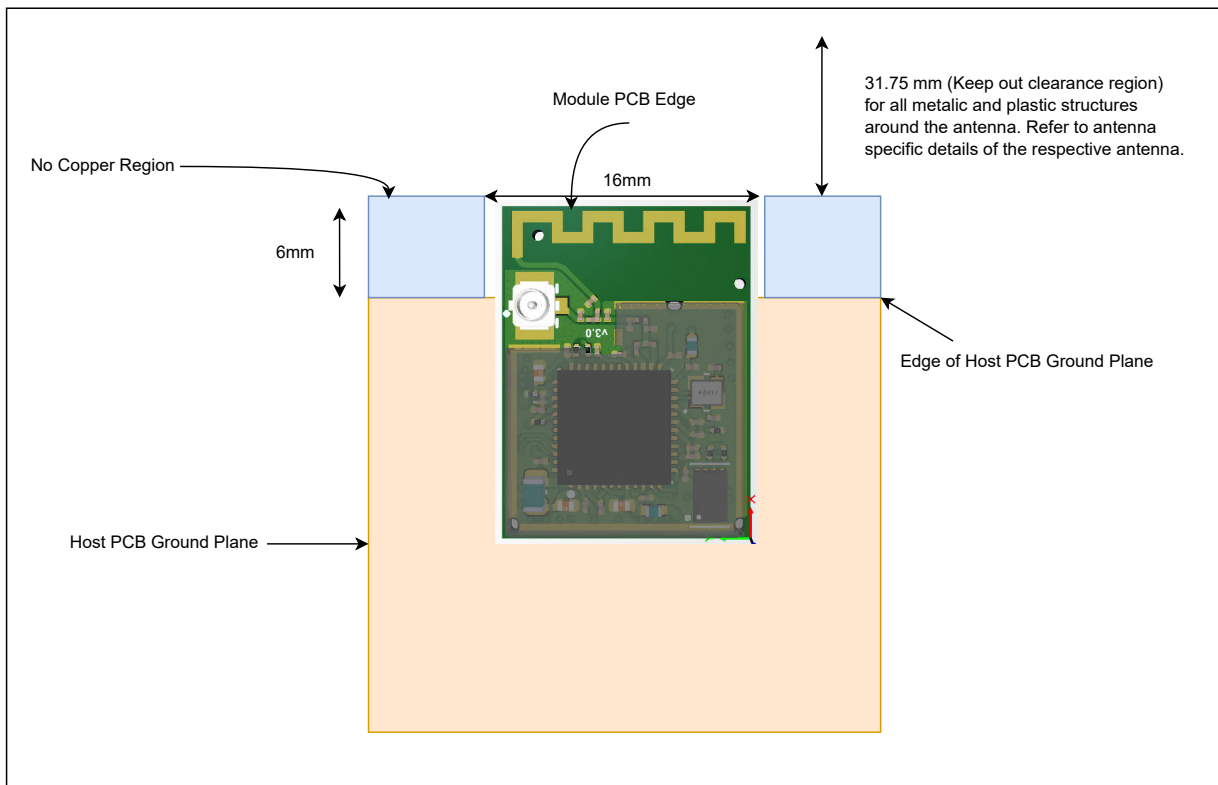
- For any Bluetooth Low Energy/Zigbee product, the antenna placement affects the performance of the whole system. The antenna requires free space to radiate RF signals and it must not be surrounded by the ground plane. Thus, for the best PCB antenna performance, it is recommended that the WBZ45 module 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 4-6. Module Placement Guidelines**



#### Related Links

[4.4. WBZ45 Module Routing Guidelines](#)

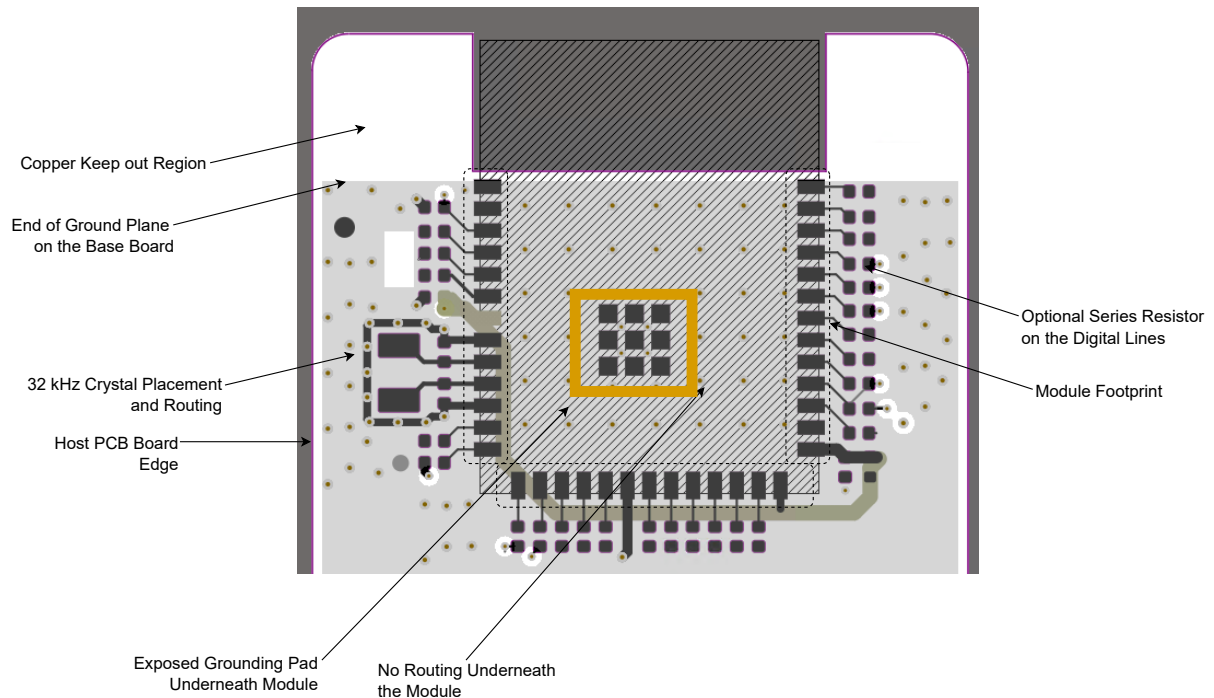
[4.3. WBZ451 Module Packaging Information](#)

## 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 the edge 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 4-7.** 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 WBZ450 Module

Parameter	Specification
Operating frequency	2400 to 2480 MHz
Peak gain	4.14 dBi at 2440 MHz
Efficiency	50%

**Table 4-2.** PCB Antenna Specification for WBZ451 Module

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

### PCB Antenna Radiation Pattern

The following figures illustrate the antenna radiation patterns for WBZ450 and WBZ451 modules.

**Figure 4-8.** WBZ450 Antenna Radiation Pattern when  $\Phi = 0^\circ$

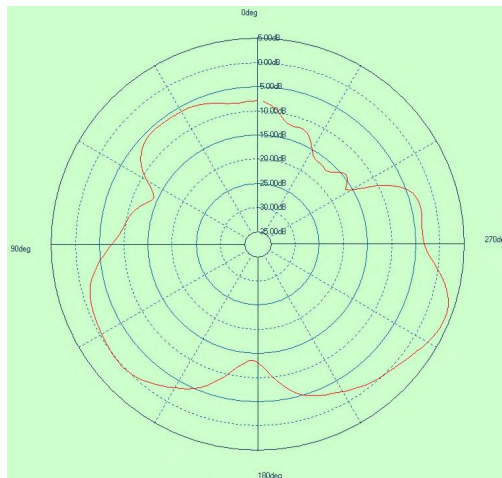


Figure 4-9. WBZ450 Antenna Radiation Pattern when  $\Phi = 90^\circ$

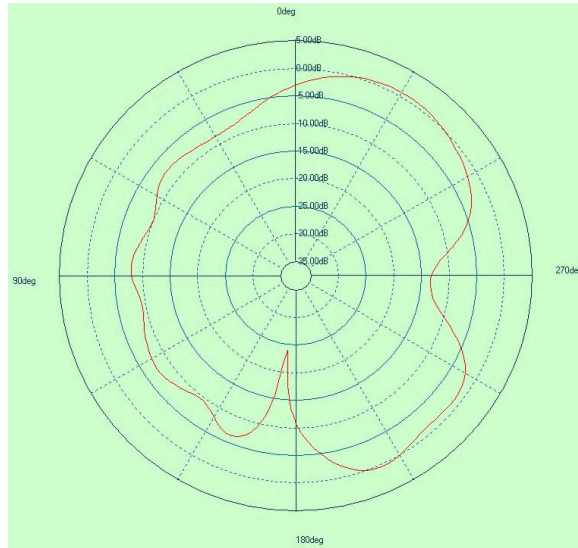


Figure 4-10. WBZ450 Antenna Radiation Pattern when  $\Theta = 90^\circ$

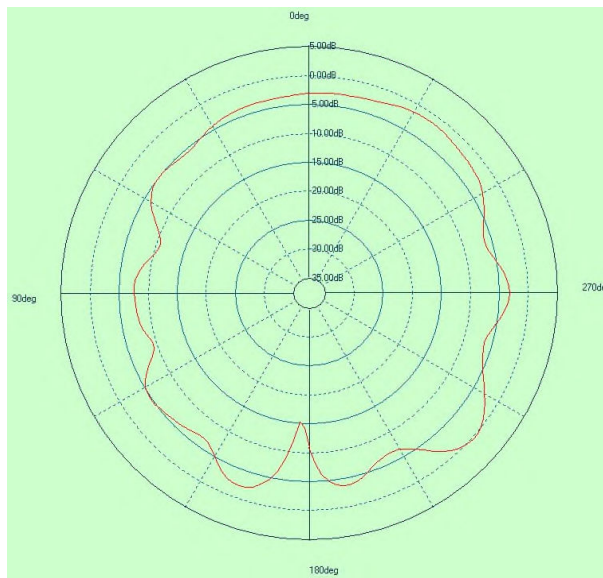


Figure 4-11. WBZ451 Antenna Radiation Pattern when Phi = 0°

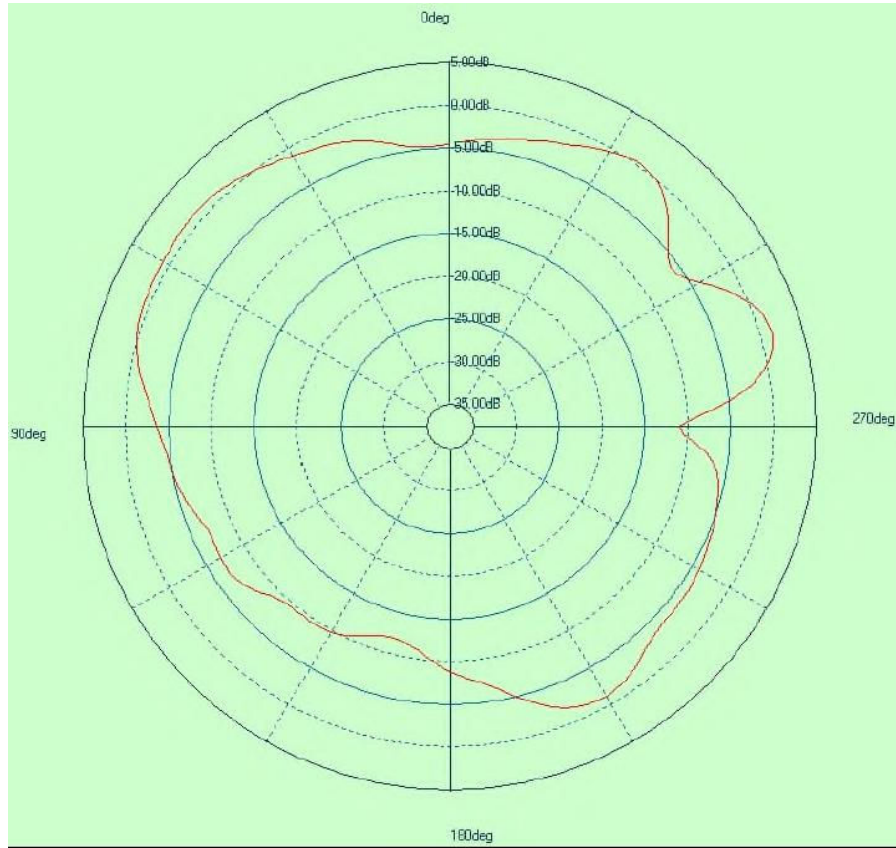


Figure 4-12. WBZ451 Antenna Radiation Pattern when Phi = 90°

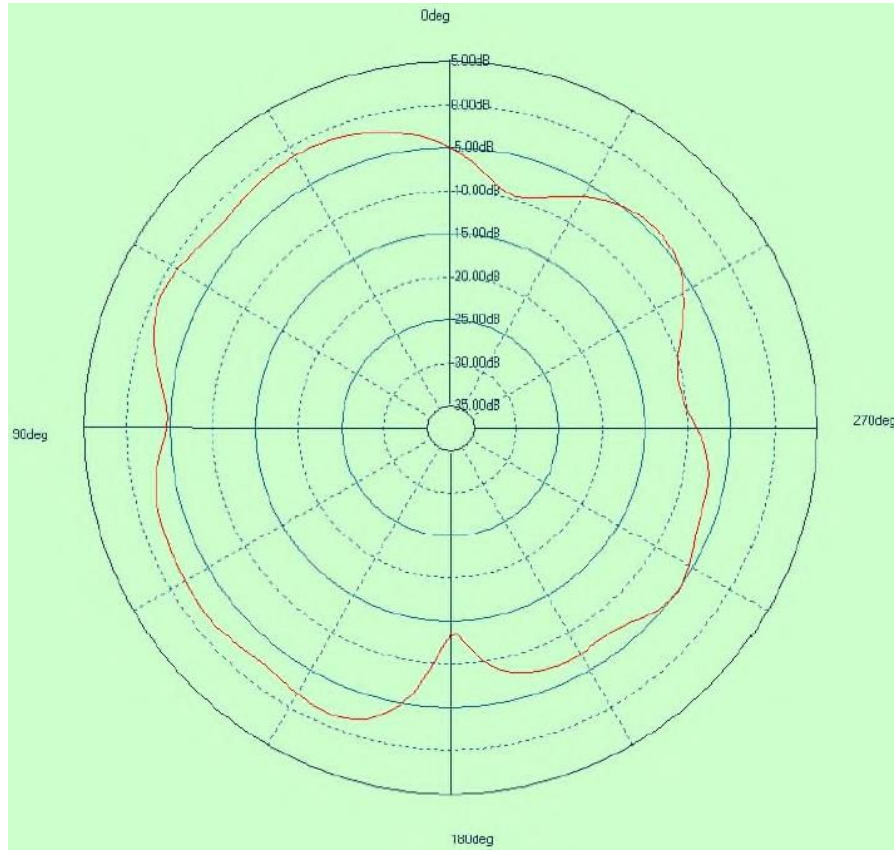
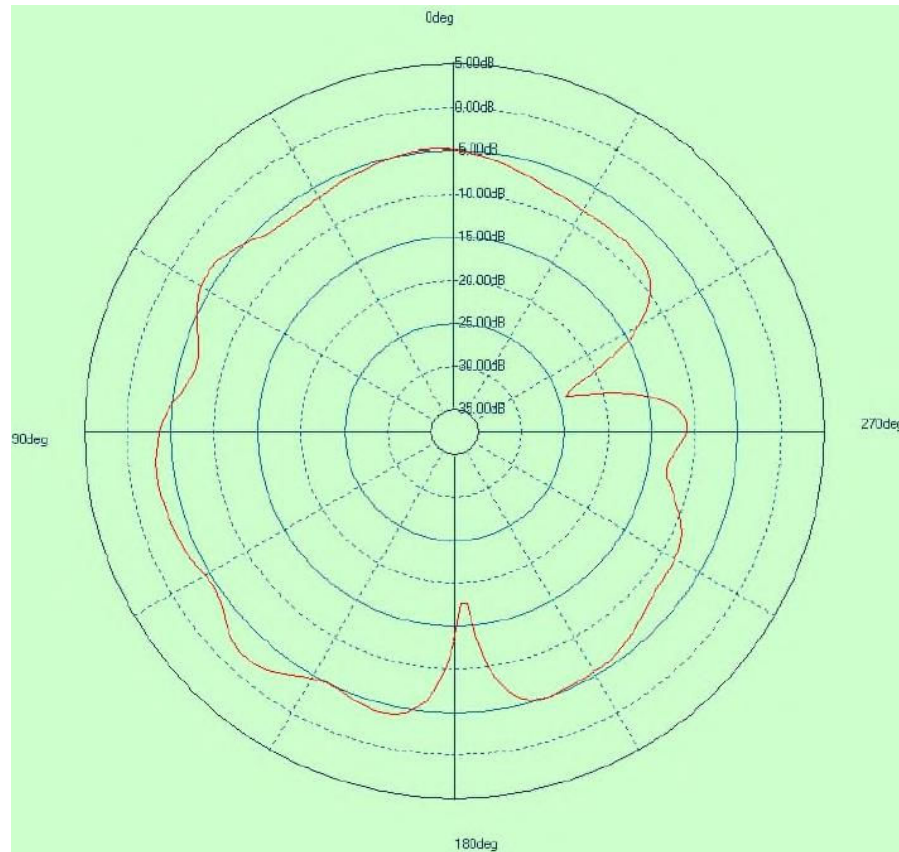




Figure 4-13. WBZ451 Antenna Radiation Pattern when Theta = 90°



#### 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 4-14. WBZ45 Antenna Placement Guidelines

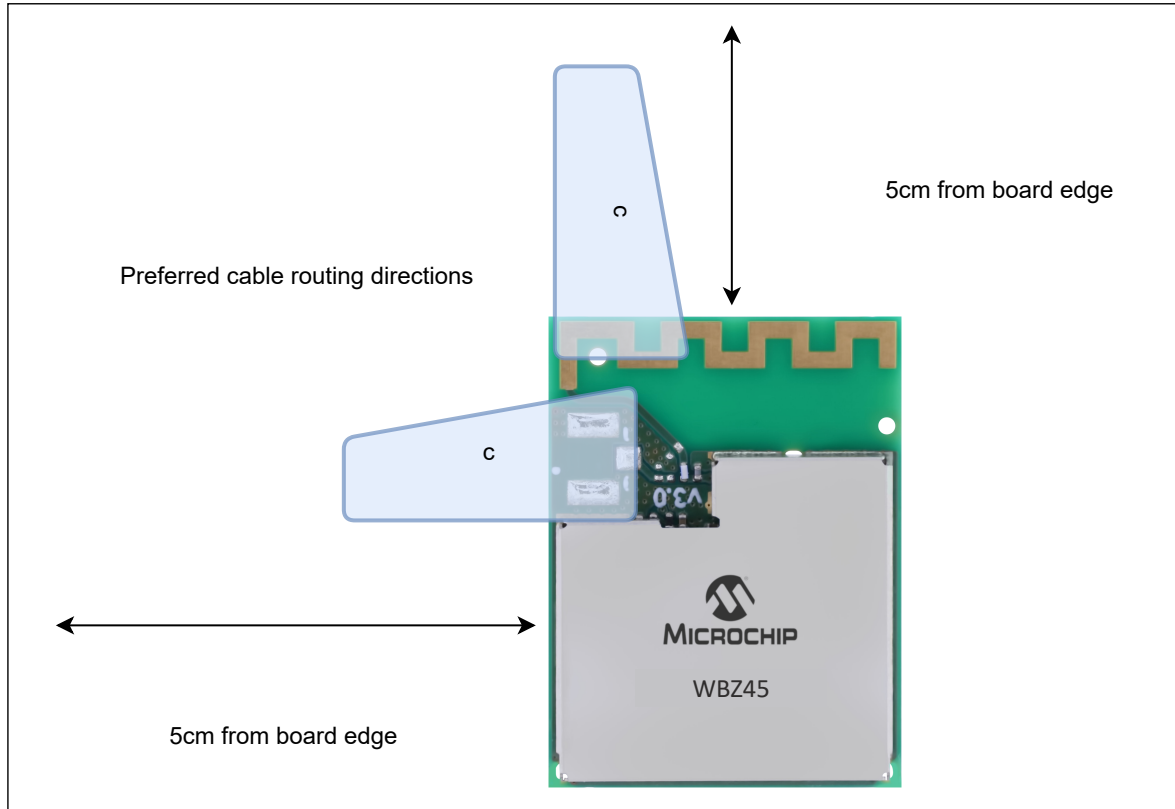


Table 4-3. List of Certified Antenna

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 <sup>(4)</sup>	GPIO <sup>(1,2)</sup>	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	—	—	—
6	10	25	31	PA7	—	—	—	RA7	—	—	SERCOM1_PAD0	—	—	TRACECLK
7	11	24	29	PA8	—	—	—	RA8	—	—	SERCOM1_PAD1	—	FECTRL0	—
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_OUT	—	—	—	—	—	—	—	—	—	—
11	19			BUCK_CLDO	—	—	—	—	—	—	—	—	—	—
12	20			EXTR <sup>(9)</sup>	—	—	—	—	—	—	—	—	—	—
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	—	—	—	—	—	—	—	—	—	—
19	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 <sup>(10)</sup>	RB4	—	—	—	—	FECTRL3	TRACEDATA3
23	35	9	6	PB5	—	AN1	—	RB5	—	—	—	—	FECTRL4	TRACEDATA0
24	36	4	7	AVDD	—	—	—	—	—	—	—	—	—	—

.....continued														
PIC32CX-BZ2 SoC		WBZ45 Module		Pad Name	Peripherals									
24032	25048	450	451		AC	ADC	EIC <sup>(4)</sup>	GPIO <sup>(1,2)</sup>	QSPI	RTCC	SERCOM	OSC	RF	DEBUG
25	37	12	12	PB6	—	AN0,AN2	—	RB6	—	—	—	—	FECTRL5	TRACEDATA1
26	38	15	13	PB7	LVDIN	AN3	—	RB7	—	—	—	—	—	TRACEDATA2, CM4_SWO
	39			VDD	—	—	—	—	—	—	—	—	—	—
27	40	14	15	PB9	—	—	—	RB9	—	—	—	—	—	CM4_SWDIO
28	41	13	14	PB8	—	—	—	RB8	—	—	—	—	—	CM4_SWCLK
29 <sup>(6)</sup>	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 <sup>(6)</sup>	—	—	—	SOSCI	—	—
30	46		10	PA12	—	—	—	RA12 <sup>(6)</sup>	—	—	—	SOSCO	—	—
30 <sup>(6)</sup>	47	7	36	PA3	—	—	—	RA3	—	RTC_IN0	SERCOM0_PAD2	SCLKI	—	—
31	48			PMU_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 (IOCA<sub>n</sub> and IOCB<sub>n</sub>) except RA11 and RA12. Please refer to Table 6-13 Port A Register Map for RA11 and RA12. CN register.
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. For 25048 only, pin 29 and pin 30 can be configured as PA11 and PA12 by setting up CFGCON2.SOSCESEL = 0.
8. External resistor used to set internal reference current of the SOC.
9. Disable trace data output for SWD or 4-wire trace output are shared. To disable trace data output, use CFGCON0.SWOEN=0. To disable 4-wire trace output, use CFGCON0.TROEN=0.
10. INT0 can be used as a wake-up source from Deep Sleep or Extreme Deep Sleep Low Power modes, as well as an ADC trigger source. The INT0 can be configured using Configuration Control Register 0 (CFGCON0). INT0 functionality on PB4 cannot be remapped using PPS. The software SDK and operational stacks provided by Microchip handles the operation of INT0 as a wake-up source in Deep Sleep Low Power Mode.
11. 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.

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

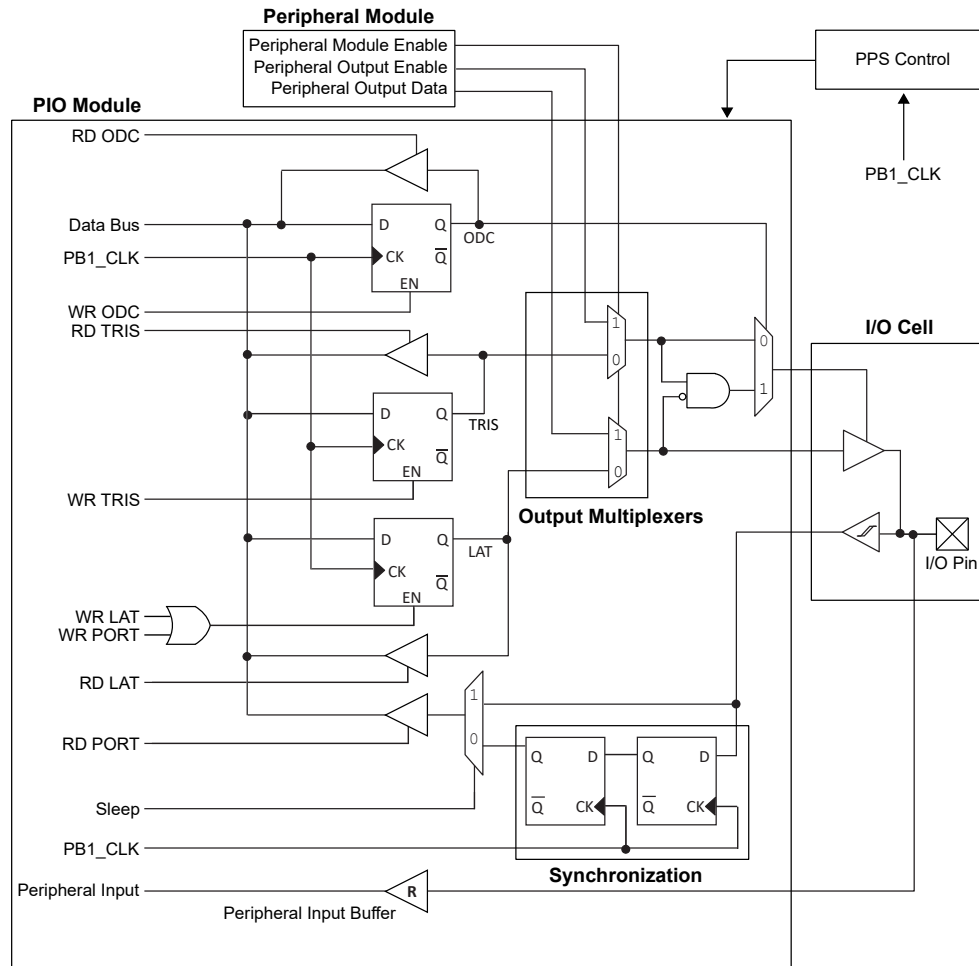
General purpose I/O (GPIO) pins allow the PIC32CX-BZ2 devices to monitor and control other devices. To add flexibility and functionality, some pins are multiplexed with alternate function(s). These functions depend on which peripheral features are on the device. In general, when a peripheral is functioning, 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<sup>2</sup>C pins). I<sup>2</sup>C pins are already open-drain pins; therefore, the ODCx settings do not influence the I<sup>2</sup>C 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/CNNEEx)
- 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/CNNEEx 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, CNNEEx 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<sup>(1)</sup>

SRCON1x	SRCON0x	Description
1	1	Slew rate control is enabled and is set to the slowest edge rate
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<sup>2</sup>C 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

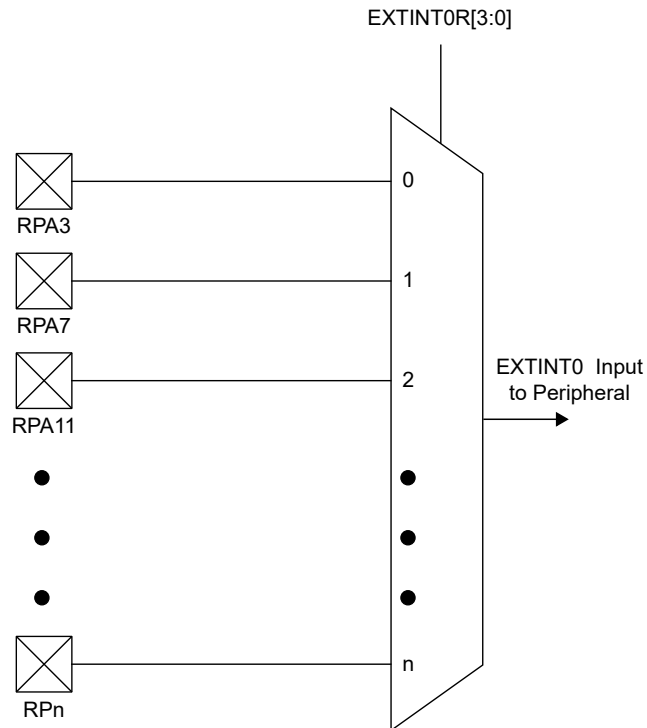
[6.2.7.3. Remappable Input Example](#)

#### 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
SERCOM0_PAD3	SCOM0P3R	SCOM0P3R[3:0]	0001 = RPA3
SERCOM1_PAD2	SCOM1P2R	SCOM1P2R[3:0]	0010 = RPA7
SERCOM2_PAD1	SCOM2P1R	SCOM2P1R[3:0]	0011 = RPA11
SERCOM3_PAD0	SCOM3P0R	SCOM3P0R[3:0]	0100 = RPB0
QSCK	QSCKR	QSCKR[3:0]	0101 = RPB4
QD1	QD1R	QD1R[3:0]	0110 = RPB8
REFI	REFIR	REFIR[3:0]	0111 = RPB12
CCLIN0	CCLIN0R	CCLIN0R[3:0]	1000 = RPA2
CCLIN3	CCLIN3R	CCLIN3R[3:0]	1001 = RPA6
TC0_WO0G1	TC0WO0G1R	TC0WO0G1R[3:0]	1010 = RPA10
TC1_WO0G1	TC1WO0G1R	TC1WO0G1R[3:0]	1011 = RPA14
TC2_WO0G1	TC2WO0G1R	TC2WO0G1R[3:0]	1100 = RPB3
TC3_WO0G1	TC3WO0G1R	TC3WO0G1R[3:0]	1101 = RPB7
			1110 = RPB11
			1111 = RPA9

**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
SERCOM0_PAD0	SCOM0P0R	SCOM0P0R[3:0]	0001 = RPA4
SERCOM1_PAD3	SCOM1P3R	SCOM1P3R[3:0]	0010 = RPA8
SERCOM2_PAD2	SCOM2P2R	SCOM2P2R[3:0]	0011 = RPA12
SERCOM3_PAD1	SCOM3P1R	SCOM3P1R[3:0]	0100 = RPB1
QD2	QD2R	QD2R[3:0]	0101 = RPB5
CCLIN1	CCLIN1R	CCLIN1R[3:0]	0110 = RPB9
CCLIN4	CCLIN4R	CCLIN4R[3:0]	0111 = RPB13
TC0_WO0G2	TC0WO0G2R	TC0WO0G2R[3:0]	1000 = RPA3
TC1_WO1G2	TC1WO1G2R	TC1WO1G2R[3:0]	1001 = RPA7
TC2_WO1G2	TC2WO1G1R	TC2WO1G1R[3:0]	1010 = RPA11
TC3_WO1G2	TC3WO1G1R	TC3WO1G1R[3:0]	1011 = RPB0
			1100 = RPB4
			1101 = RPB8
			1110 = RPB12
			1111 = RPA0

**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
SERCOM0_PAD1	SCOM0P1R	SCOM0P1R[3:0]	0001 = RPA5
SERCOM1_PAD0	SCOM1P0R	SCOM1P0R[3:0]	0010 = RPA9
SERCOM2_PAD3	SCOM2P3R	SCOM2P3R[3:0]	0011 = RPA13
SERCOM3_PAD2	SCOM3P2R	SCOM3P2R[3:0]	0100 = RPB2
QD3	QD3R	QD3R[3:0]	0101 = RPB6
CCLIN2	CCLIN2R	CCLIN2R[3:0]	0110 = RPB10
CCLIN5	CCLIN5R	CCLIN5R[3:0]	0111 = RPA0
TC0_WO1G3	TC0WO1G3R	TC0WO1G3R[3:0]	1000 = RPA4
TC2_WO0G3	TC2WO0G3R	TC2WO0G3R[3:0]	1001 = RPA8
TC3_WO0G3	TC3WO0G3R	TC3WO0G3R[3:0]	1001 = RPA8
			1010 = RPA12
			1011 = RPB1
			1100 = RPB5
			1101 = RPB9
			1110 = RPB13
			1111 = RPA1

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

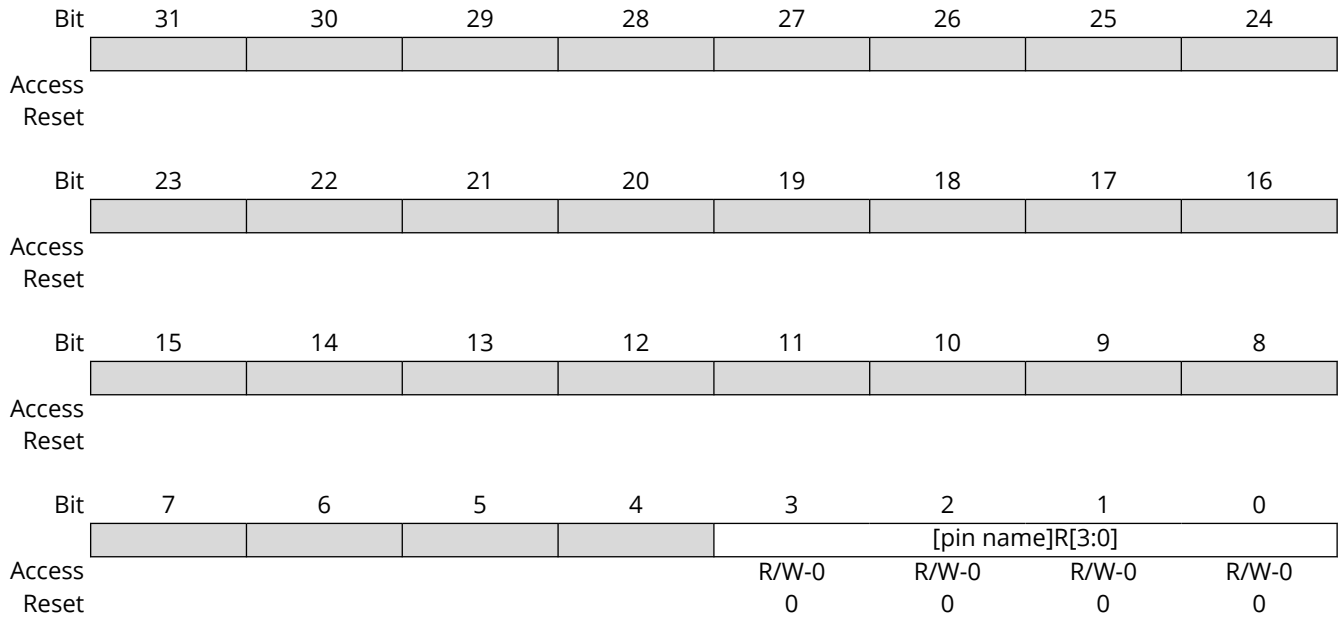
Peripheral Pin	[pin name]R SFR	[pin name]R bits	[pin name]R Value to RPN Pin Selection
EXTINT3	EXTINT3R	EXTINT3R[3:0]	0000 = OFF
NMI	NMIR	NMIR[3:0]	0001 = RPA6
SERCOM0_PAD2	SCOM0P2R	SCOM0P2R[3:0]	0010 = RPA10
SERCOM1_PAD1	SCOM1P1R	SCOM1P1R[3:0]	0011 = RPA14
SERCOM2_PAD0	SCOM2P0R	SCOM2P0R[3:0]	0100 =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

### 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.



**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](#)

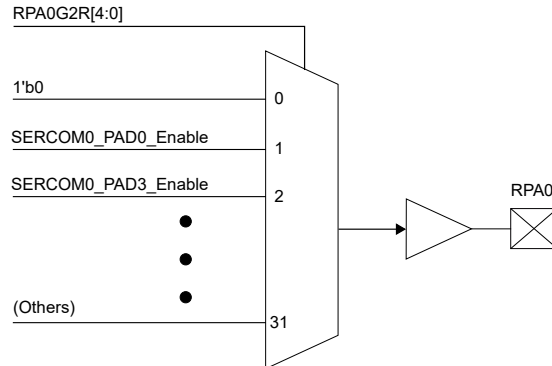
## 6.2.9 Pin Output RP Registers

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



**Table 6-8. Remappable Output Pin Configuration – Group2**

Output Pin	[pin name]R SFR	[pin name]R bits	[pin name]R Value to RPh Pin Selection
RPA0	RPA0G2R	RPA0G2R[4:0]	00000 = OFF
RPA3	RPA3G2R	RPA3G2R[4:0]	00001 = SERCOM0_PAD0
RPA4	RPA4G2R	RPA4G2R[4:0]	00010 = SERCOM0_PAD3
RPA6	RPA6G2R	RPA6G2R[4:0]	00011 = SERCOM0_PAD2
RPA7	RPA7G2R	RPA7G2R[4:0]	00100 = SERCOM1_PAD1
RPA8	RPA8G2R	RPA8G2R[4:0]	00101 = SERCOM1_PAD3
RPA8	RPA8G2R	RPA8G2R[4:0]	00110 = SERCOM1_PAD2
RPB0	RPB0G2R	RPB0G2R[4:0]	00111 = SERCOM2_PAD1
RPB1	RPB1G2R	RPB1G2R[4:0]	01000 = SERCOM2_PAD2
RPB4	RPB4G2R	RPB4G2R[4:0]	01001 = SERCOM3_PAD1
RPB5	RPB5G2R	RPB5G2R[4:0]	01010 = SERCOM3_PAD0
RPB8	RPB8G2R	RPB8G2R[4:0]	01011 = TCC0_WO1
RPB9	RPB9G2R	RPB9G2R[4:0]	01100 = TCC0_WO5
RPB12	RPB12G2R	RPB12G2R[4:0]	01101 = TCC0_WO3
RPB13	RPB13G2R	RPB13G2R[4:0]	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

**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
RPA1	RPA1G3R	RPA1G3R[4:0]	00001 = SERCOM0_PAD1 00010 = SERCOM0_PAD0
RPA3	RPA3G3R	RPA3G3R[4:0]	00011 = SERCOM0_PAD3
RPA4	RPA4G3R	RPA4G3R[4:0]	00100 = SERCOM1_PAD2
RPA5	RPA5G3R	RPA5G3R[4:0]	00101 = SERCOM1_PAD0
RPA8	RPA8G3R	RPA8G3R[4:0]	00110 = SERCOM1_PAD3
RPA9	RPA9G3R	RPA9G3R[4:0]	00111 = SERCOM2_PAD2
RPA13	RPA13G3R	RPA13G3R[4:0]	01000 = SERCOM2_PAD3
RPB1	RPB1G3R	RPB1GX3R[4:0]	01001 = SERCOM3_PAD2
RPB2	RPB2G3R	RPB2G3R[4:0]	01010 = SERCOM3_PAD1
RPB5	RPB5G3R	RPB5G3R[4:0]	01011 = TCC0_WO2
RPB6	RPB6G3R	RPB6G3R[4:0]	01100 = TCC0_WO0
RPB9	RPB9G3R	RPB9G3R[4:0]	01101 = TCC0_WO4
RPB10	RPB10G3R	RPB10G3R[4:0]	01110 = TCC1_WO2
RPB13	RPB13G3R	RPB13G3R[4:0]	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

**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
RPA2	RPA2G4R	00001 = SERCOM0_PAD2
RPA4	RPA4G4R	00010 = SERCOM0_PAD1
RPA5	RPA5G4R	00011 = SERCOM0_PAD0
RPA6	RPA6G4R	00100 = SERCOM1_PAD3
RPA8	RPA8G4R	00101 = SERCOM1_PAD1
RPA9	RPA9G4R	00110 = SERCOM1_PAD0
RPA10	RPA10G4R	00111 = SERCOM2_PAD3
RPA13	RPA13G4R	01000 = SERCOM2_PAD0
RPA14	RPA14G4R	01001 = SERCOM3_PAD3
RPB2	RPB2G4R	01010 = SERCOM3_PAD2
RPB3	RPB3G4R	01011 = TCC0_WO3
RPB6	RPB6G4R	01100 = TCC0_WO1
RPB7	RPB7G4R	01101 = TCC0_WO5
RPB10	RPB10G4R	01110 = TCC1_WO3
RPB11	RPB11G4R	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

**Related Links**

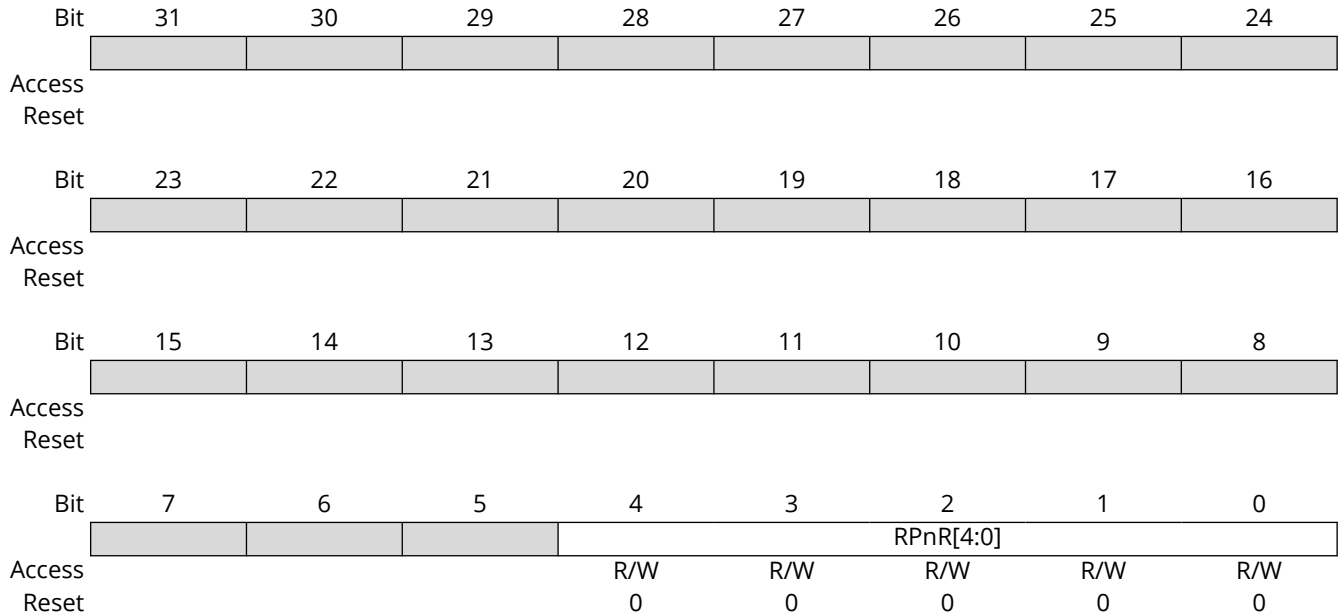
[6.2.9.1. RPNR](#)

### 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.



**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 <sup>(1)</sup>	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	SCLK1	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 <sup>(1)</sup> /SMB
	RPA5	PPS		I/O	DIG/ST
	IOCA5	Change Notification		I	ST
	RA5	GPIO		I/O	DIG/ST
pa6	PGC2ENTRY	DEBUG	9	I	ST
	SERCOM0_PAD1	SERCOM0		I/O	DIG/ST/ I2C <sup>(1)</sup> /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 <sup>(1)</sup> /SMB
	RPA7	PPS		I/O	DIG/ST
	IOCA7	Change Notification		I	ST
	RA7	GPIO		I/O	DIG/ST

.....continued

Pin Name <sup>(1)</sup>	Functions in Priority Order	Reference Peripheral	Pin Number (48-pin)	I/O	Type
pa8	PGD2ENTRY	DEBUG	11	I	ST
	FECTRL0	RF Radio		O	DIG
	SERCOM1_PAD1	SERCOM1		I/O	DIG/ST/ I2C <sup>(1)</sup> /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	—	—
pa13	COEXCTRL0	RF Radio	16	I/O	DIG/ST
	SERCOM2_PAD0	SERCOM2		I/O	DIG/ST/ I2C <sup>(1)</sup> /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 <sup>(1)</sup> /SMB
	RPA14	PPS		I/O	DIG/ST
	IOCA14	Change Notification		I	ST
	RA14	GPIO		I/O	DIG/ST

**Note:**

1. SERCOM<sub>x</sub>\_PAD<sub>x</sub> 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
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/O	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

.....continued

Pin Name (1)	Functions in Priority Order	Reference Peripheral	Pin Number (48-pin)	I/O	Type
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
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
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_2500)	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
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
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		
005Ch	SCOM1P3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	—	SCOM1P3R[3:0]	0000		
0060h	SCOM2P0R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	—	SCOM2P0R[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
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		
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		

.....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
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
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



.....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	
00D4h	TC2WO1G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	—	TC2WO1G4R [3:0]			0000
00D8h	TC3WO0G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	—	TC3WO0G1R [3:0]			0000
00DCh	TC3WO0G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	—	TC3WO0G3R[3:0]			0000
00E0h	TC3WO1G2R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	—	TC3WO1G2R [3:0]			0000
00E4h	TC3WO1G4R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	—	TC3WO1G4R[3:0]			0000

**Table 6-16. Peripheral Pin Select Output Registers**

Virtual Address (4400_1000) <sup>(1)</sup>	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
0230h	RPA5G1R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	RPA5G1R[4:0]				0000
0234h	RPA5G3R	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	RPA5G3R[4:0]				0000

.....continued

Virtual Address (4400_1000) <sup>(1)</sup>	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	
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	
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	

.....continued

Virtual Address (4400_1000) <sup>(b)</sup>	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	
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
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

.....continued

Virtual Address (4400_1000) <sup>(1)</sup>	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	
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
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

.....continued

Virtual Address (4400_1000) <sup>(1)</sup>	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	
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

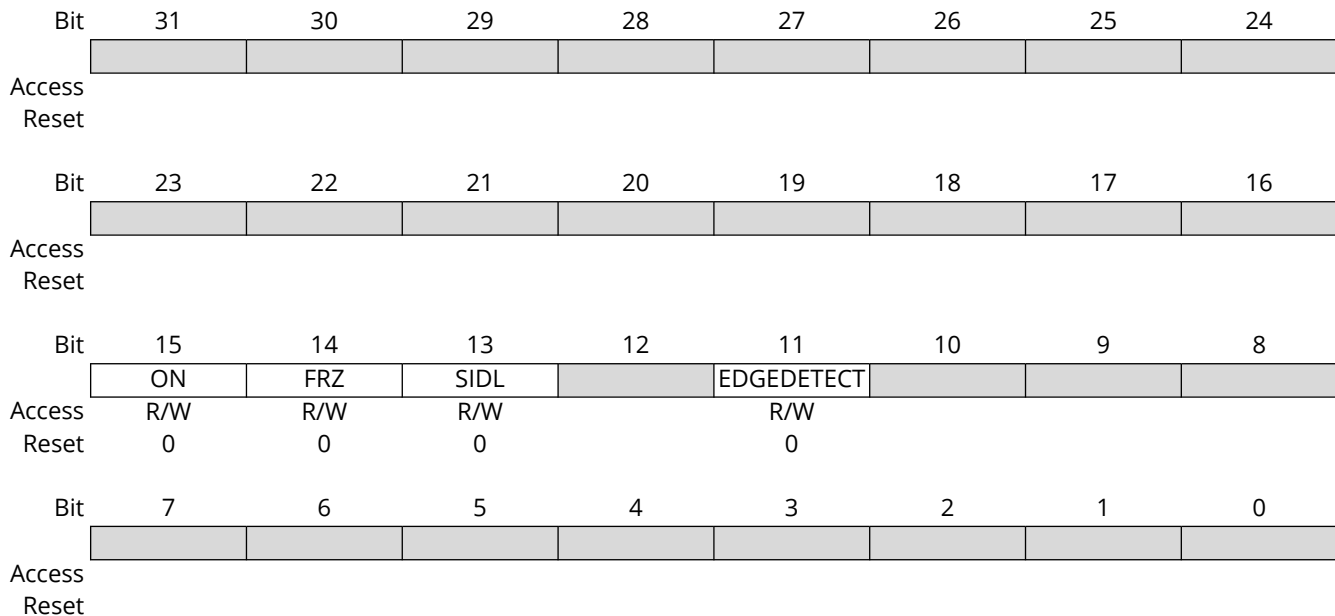
[6.1.9. CLR, SET and INV Registers](#)

### 6.4.1 Change Notice Control for PORTx Register

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

**Note:**

- 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 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

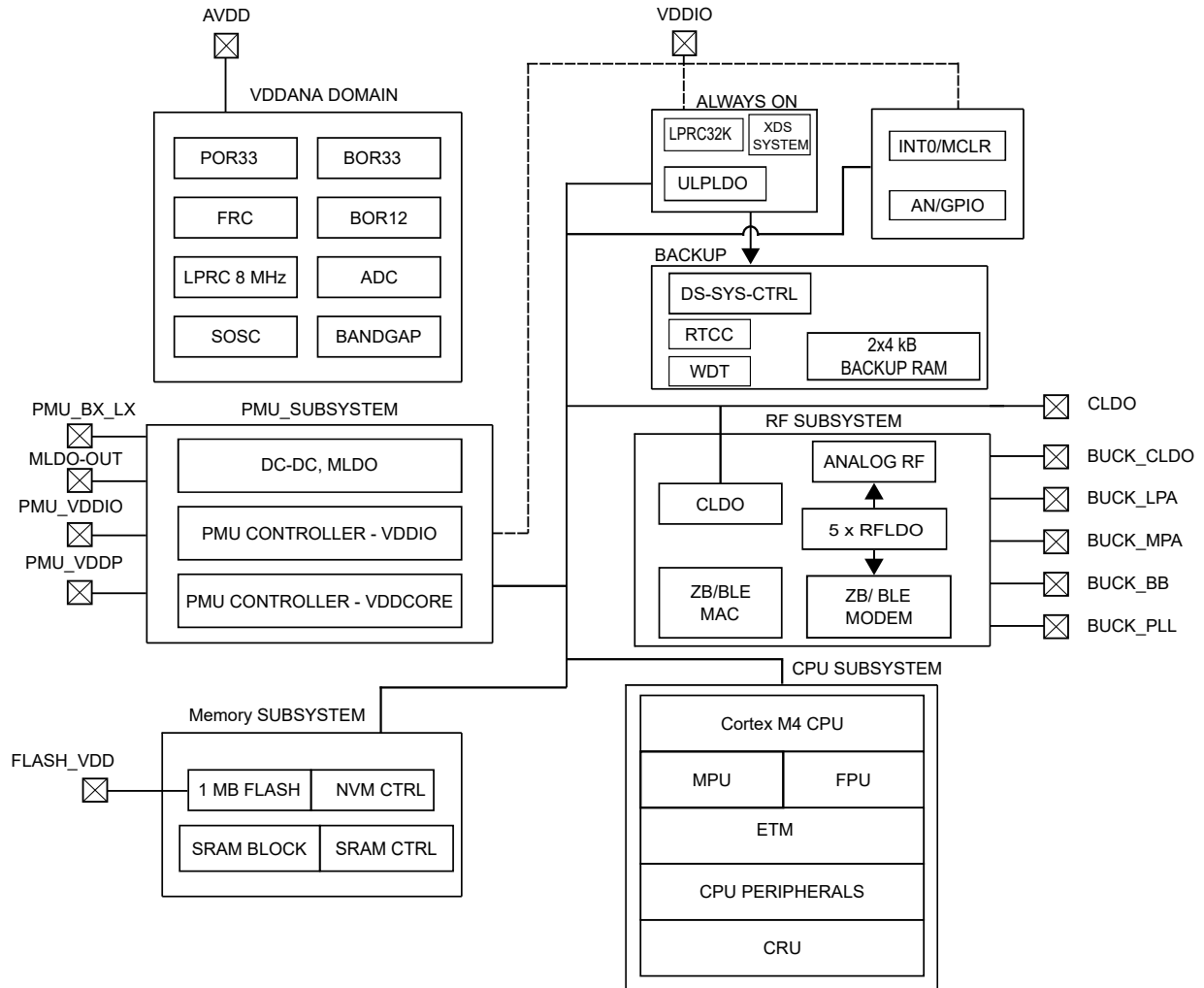
**Table 6-17.** Results of Resets Available

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

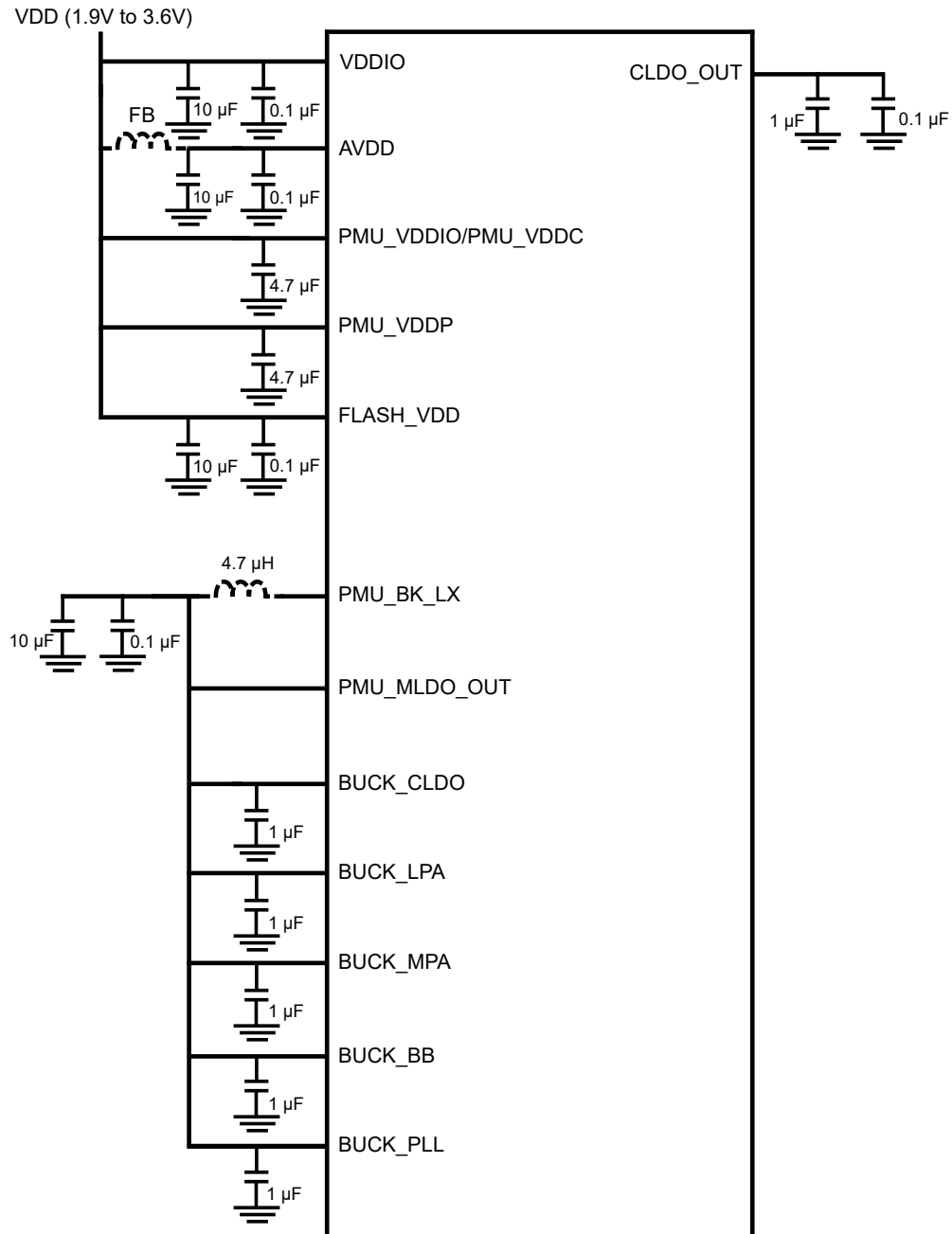
The PIC32CX-BZ2 supports a single power supply from 1.9V to 3.6V. The IO supply cannot be decoupled from the main supply. The same voltage must be applied to VDDx, PMU\_VDDIO, VPMU\_VDDC and AVDD with different levels of filtering. The internal voltage regulator has the following four different modes:

- 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

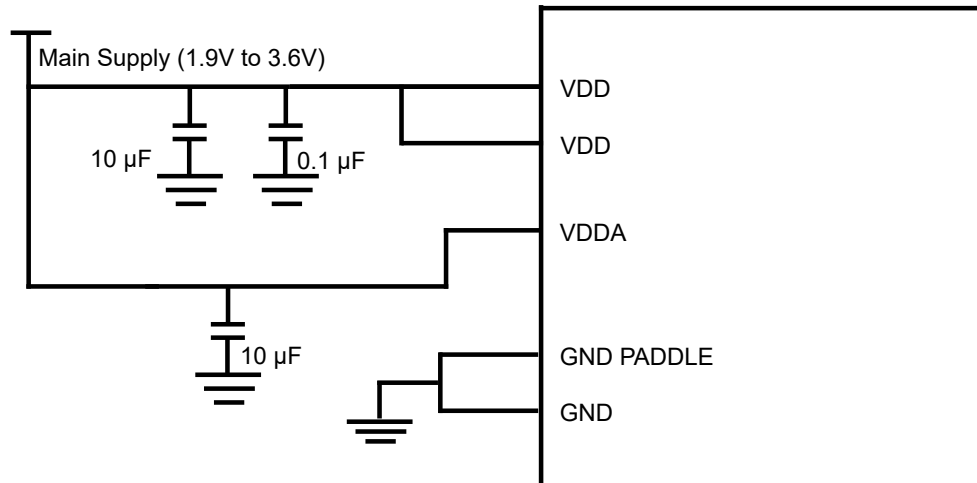


## 7.5 Typical Power Supply Connection for Module

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

System	Peripheral Bridge-A	Peripheral Bridge-B	Peripheral Bridge-C	Peripheral Bus-PIC					
0x00000000	Boot Flash (Alias)	0x40000000	PAC	0x41000000	DSU	0x42000000	QSPI	0x44000000	Device Config Registers
0x00005000	Device Config/Boot (Alias)	0x40000400	FREM	0x41002000	CMCC	0x42000400	AES	0x44000500	WDT
0x00006000	OTP Page (Alias)	0x40000800	EIC	0x41004000	DMAC	0x42000800	Reserved	0x44000600	FC
0x00007000	Unimplemented	0x40000C00	SERCOM-0	0x41006000	EVSYS	0x42000C00	SERCOM-2	0x44000800	PFW
0x00040000	Unimplemented	0x40001000	SERCOM-1	0x41008000	RECC	0x42001000	SERCOM-3	0x44000A00	CRU
0x00045000	Unimplemented	0x40001400	TC-0	0x4100A000	Reserved	0x42001400	Reserved	0x44000C00	BOR_CMP
0x00047000	Unimplemented	0x40001800	TC-1	0x41010000	WBT	0x42001800	CCL	0x44000E00	DMT
0x00048000	Unimplemented	0x40001C00	TC-2	0x41050000	Reserved (ZBT)	0x42001C00	AC	0x44001000	PPS
0x01000000	Flash	0x40002000	TC-3	0x41020000	Reserved	0x42002000	Reserved	0x44001400	ADCCON
0x01100000	Unimplemented	0x40002400	TCC-0	0x41FFFFF		0x42002400	HMTX	0x44001A00	ADCCFG
0x02000000	PUKCC	0x40002800	TCC-1			0x42002800	TRNG	0x44001E00	ADCOUT
0x02012000	Unimplemented	0x40002C00	TCC-2			0x42002C00	ICM	0x44002200	PORT A
0x03000000	CM4CCTCM	0x40003000	Reserved			0x42003000	PUKC	0x44002300	PORT B
0x03001000	Unimplemented	0x40FFFFFF				0x42003400	Reserved	0x44002400	Reserved
0x04000000	QSPI					0x42010000	RTCC	0x44010000	Reserved
0x05000000	Unimplemented					0x42011000	DSCN	0x44012000	ICD
0x20000000	System RAM					0x42012000	Reserved	0x44012400	PCHE
0x20020000	Unimplemented					0x42FFFFFF		0x44012800	Reserved
0x40000000	PB-A							0x44013E00	PMU
0x41000000	PB-B							0x44014000	Backup RAM
0x42000000	PB-C							0x44016000	Reserved
0x43000000	Unimplemented							0x4401FFFF	
0x44000000	PB-(PIC)								
0x44020000	Unimplemented								
0xE0000000	CM4F System Registers								
0xFFFF3FFF									

**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	
<b>System Configuration</b>					
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 <sup>2</sup>	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
<b>Boot Configuration</b>					
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	
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](https://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

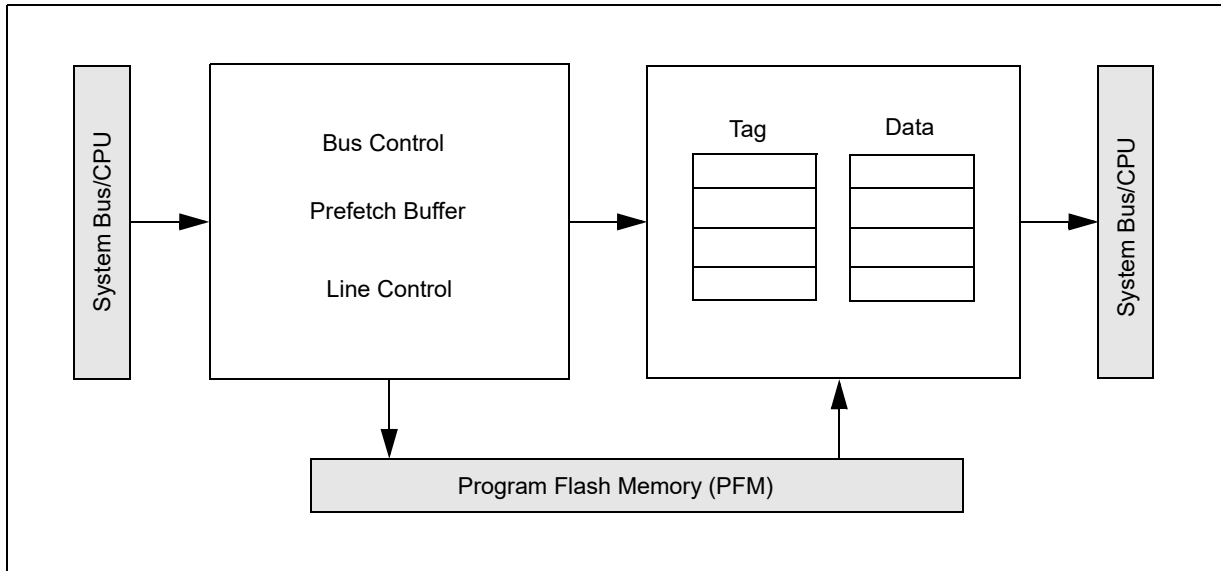
When running the Prefetch module at high-clock rates, insert the Wait states into Program Flash Memory (PFM) read transactions to meet the access time of the PFM. The user can hide the Wait states to the core by prefetching and storing the instructions in a temporary holding area that the CPU can access 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

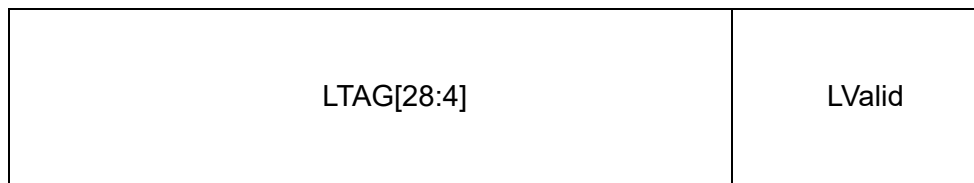
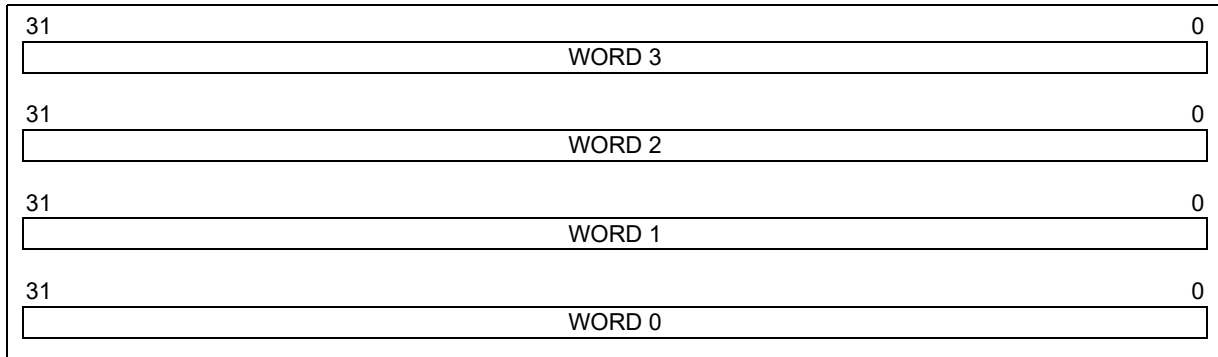


Figure 9-3. Data Line



## 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 PFMDDED 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					PFMEDD	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
Reset				0				0
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



Value	Description
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

Address Wait state is hard wired to '0' for higher performance at the clock frequency.

**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
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
Access					PFMDED	PFMSEC		
Reset					HS, R/C	HS, R/W		
					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
Access	PFMSECCNT[7:0]							
Reset	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
	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

.....continued

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

### 10.1.4 Cortex-M4 Core Peripherals

<b>Nested Vectored Interrupt Controller</b>	The Nested Vector Interrupt Controller (NVIC) is an embedded interrupt controller that supports low latency interrupt processing.
<b>System Control Block</b>	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 <i>Cortex-M4 Technical Reference Manual</i> for more details ( <a href="http://www.arm.com">http://www.arm.com</a> ).
<b>System Timer</b>	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 <i>Cortex-M4 Technical Reference Manual</i> for more details ( <a href="http://www.arm.com">http://www.arm.com</a> ).
<b>Memory Protection Unit</b>	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 <i>Cortex-M4 Technical Reference Manual</i> for more details ( <a href="http://www.arm.com">http://www.arm.com</a> ).
<b>Floating-Point Unit</b>	The Floating Point Unit (FPU) provides IEEE 754-compliant operations on single-precision, 32-bit, floating-point values. Refer to the <i>Cortex-M4 Technical Reference Manual</i> for more details ( <a href="http://www.arm.com">http://www.arm.com</a> ).

### 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](http://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 <sup>(1)</sup> Order: USART, I2CM, I2CS, SPI	0	12
	1	
	2	
	3	
	4	
	5	
	6	
	7	
SERCOM1 – Serial Communication Interface 1 <sup>(1)</sup> Order: USART, I2CM, I2CS, SPI	0	13
	1	
	2	
	3	
	4	
	5	
	6	
	7	
SERCOM2 – Serial Communication Interface 2 <sup>(1)</sup> Order: USART, I2CM, I2CS, SPI	0	14
	1	
	2	
	3	
	4	
	5	
	6	
	7	
SERCOM3 – Serial Communication Interface 3 <sup>(1)</sup> Order: USART, I2CM, I2CS, SPI	0	15
	1	
	2	
	3	
	4	
	5	
	6	
	7	

.....continued

Module	Source	Line
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		
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	

.....continued		
Module	Source	Line
TC0 – Basic Timer Counter 0	ERR A	19
	MC 0	
	MC 1	
	OVF	
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	ISO	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

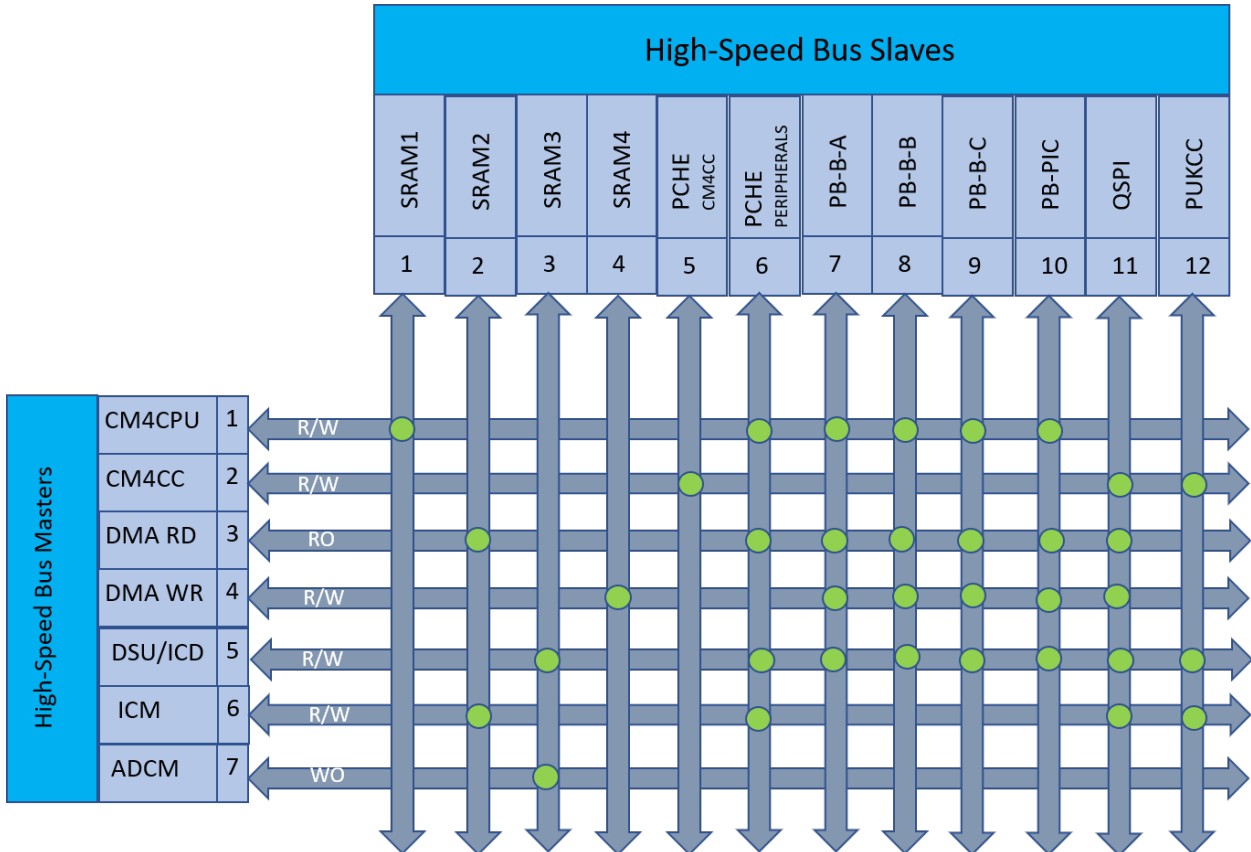


Table 10-1. High Speed Bus Matrix Host

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
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

.....continued

High-Speed Bus Matrix Client	Client ID
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.

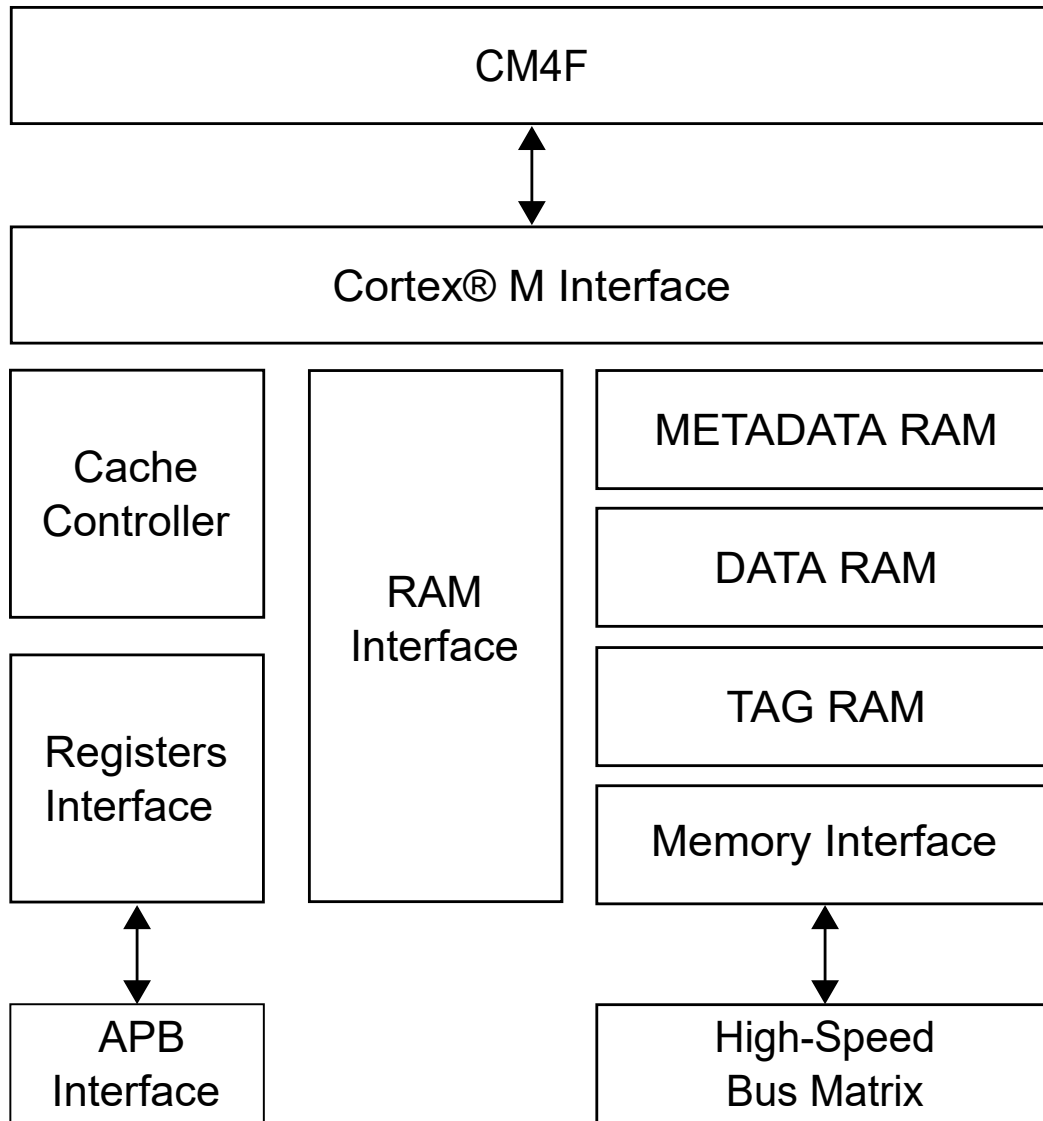
The CMCC interfaces with the CPU through the AHB, and is connected to the APB bus interface for its configuration.

### 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

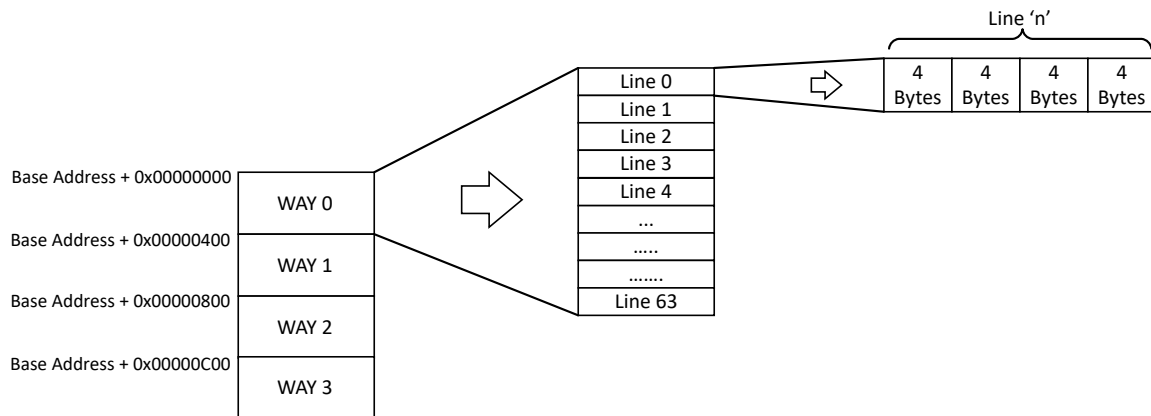
### 11.3 Block Diagram

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 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., `NOB`) 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 to 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 <sup>(1)</sup> 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 <sup>(1)</sup> 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]					INDEX[7:4]			
		15:8									
		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
Access			CLSIZE[2:0]			CSIZE[2:0]		
Reset			R	R	R	R	R	R
			0	1	0	0	1	0
Bit	7	6	5	4	3	2	1	0
Access	LCKDOWN	WAYNUM[1:0]		RRP	LRUP	RANDP	GCLK	AP
Reset	R	R	R	R	R	R	R	R
	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

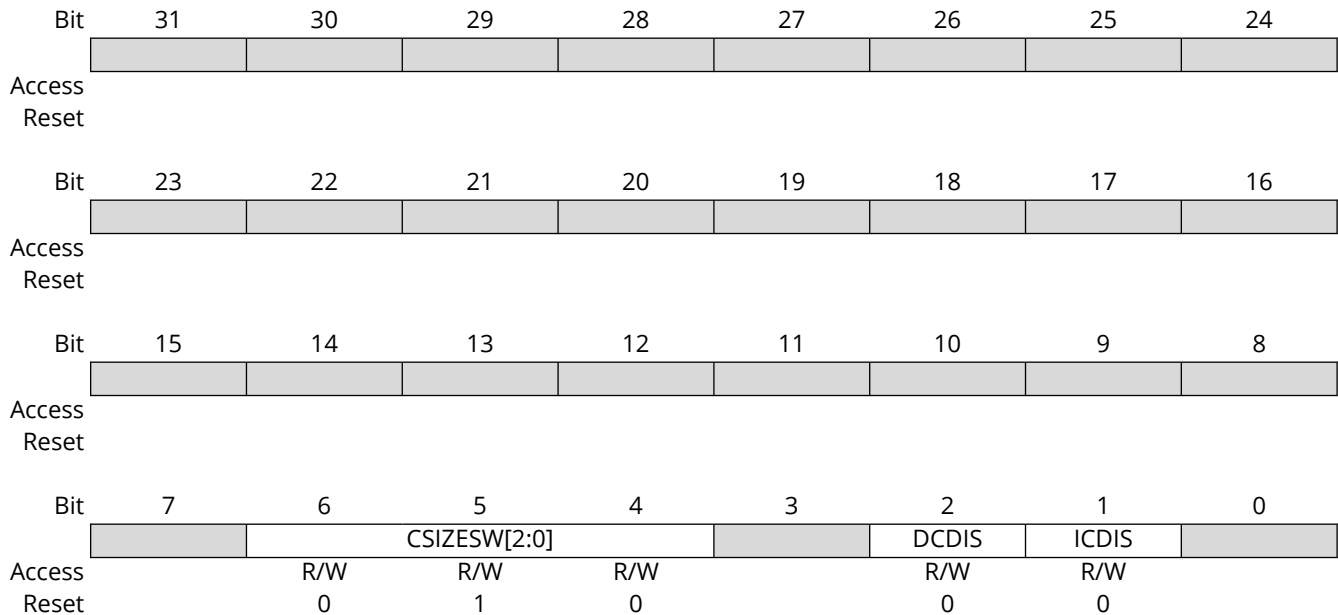
Writing a '0' to this bit disables the Access Port Access Allowed.

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



### Bits 6:4 – CSIZESW[2:0] Cache Size Configured by Software

This field configures the cache size.

Value	Name	Description
0x0	CONF_CS_SIZE_1KB	The Cache Size is configured to 1KB
0x1	CONF_CS_SIZE_2KB	The Cache Size is configured to 2KB
0x2	CONF_CS_SIZE_4KB	The Cache Size is configured to 4KB
0x3	CONF_CS_SIZE_8KB	The Cache Size is configured to 8KB
0x4	CONF_CS_SIZE_16KB	The Cache Size is configured to 16KB
0x5	CONF_CS_SIZE_32KB	The Cache Size is configured to 32KB
0x6	CONF_CS_SIZE_64KB	The Cache Size is configured to 64KB
0x7		Reserved

### Bit 2 – DCDIS Data Cache Disable

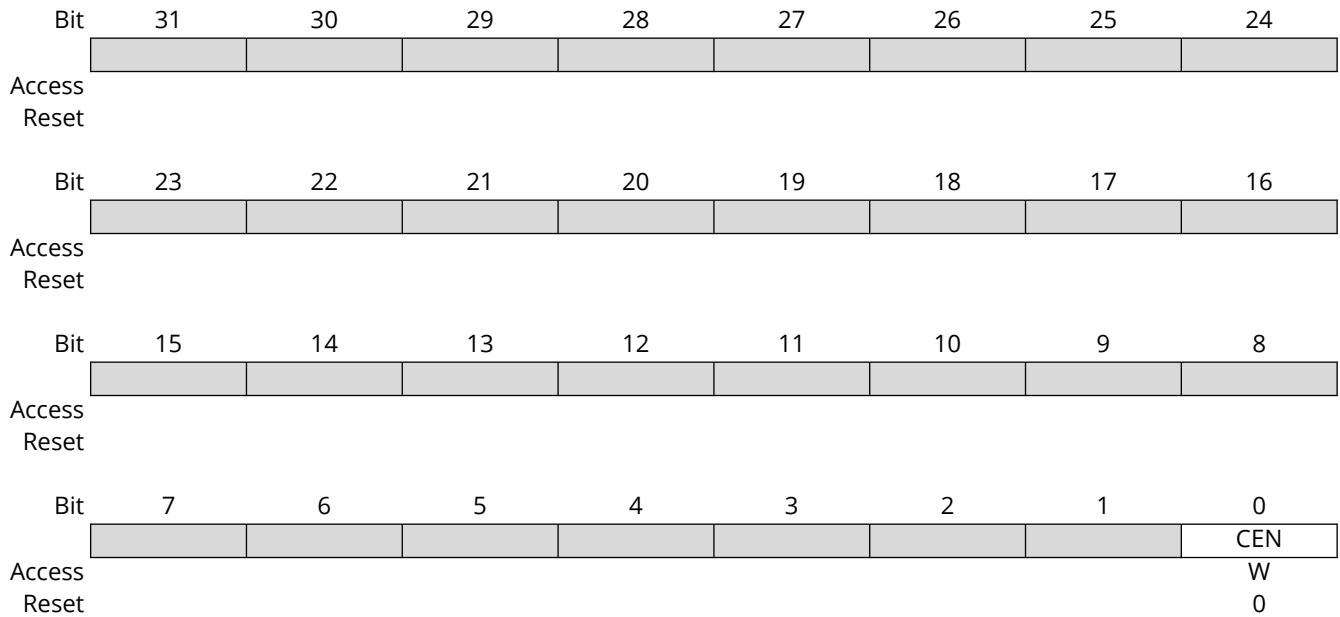
Writing a '0' to this bit enables data caching.  
Writing a '1' to this bit disables data caching.

### 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

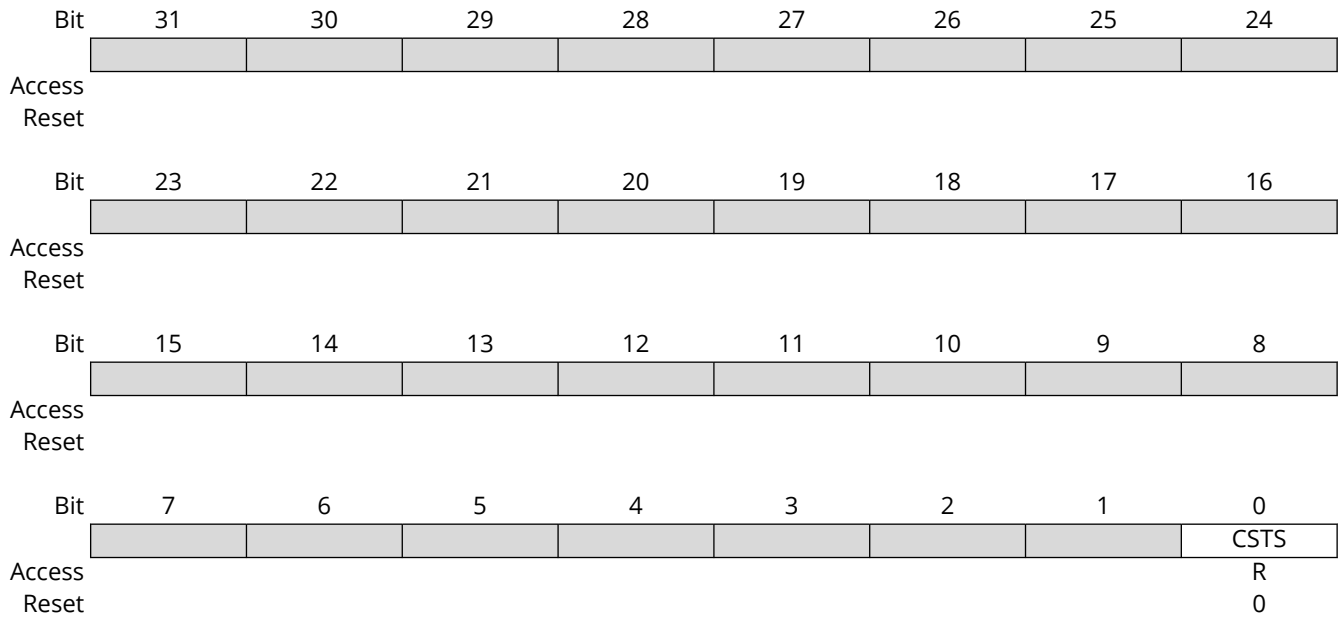
**Name:** CTRL  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** Write-only



**Bit 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

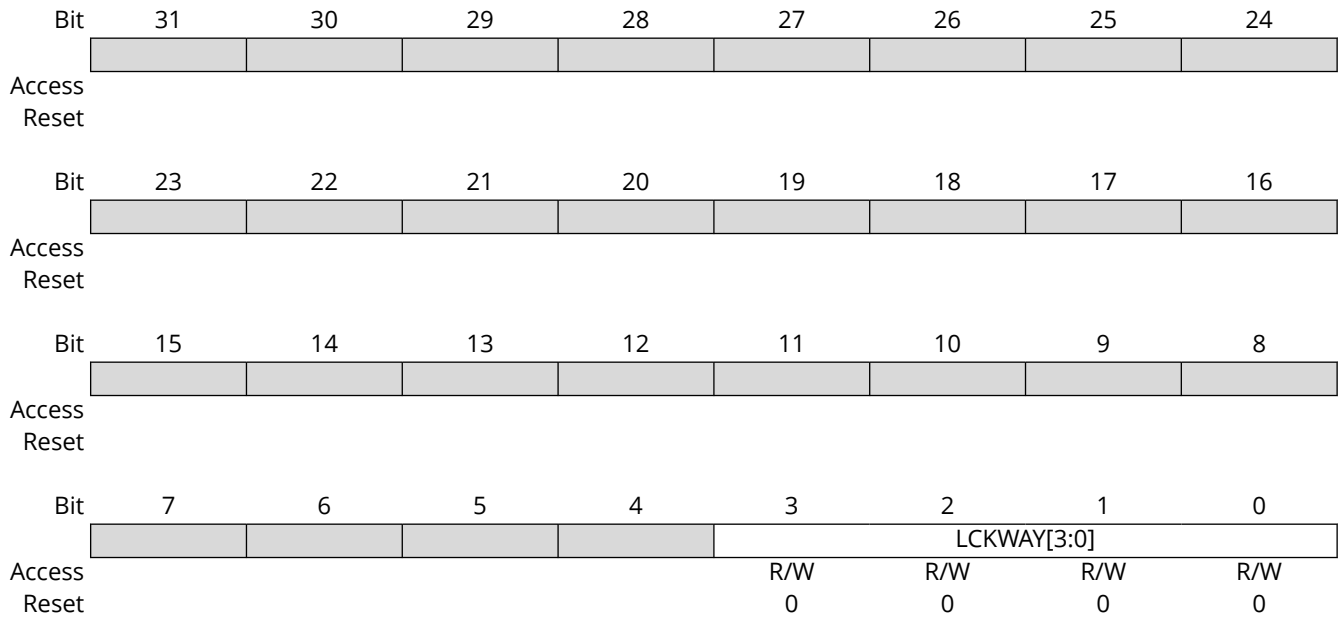
**Name:** SR  
**Offset:** 0x0C  
**Reset:** 0x00000000  
**Property:** Read-only



**Bit 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

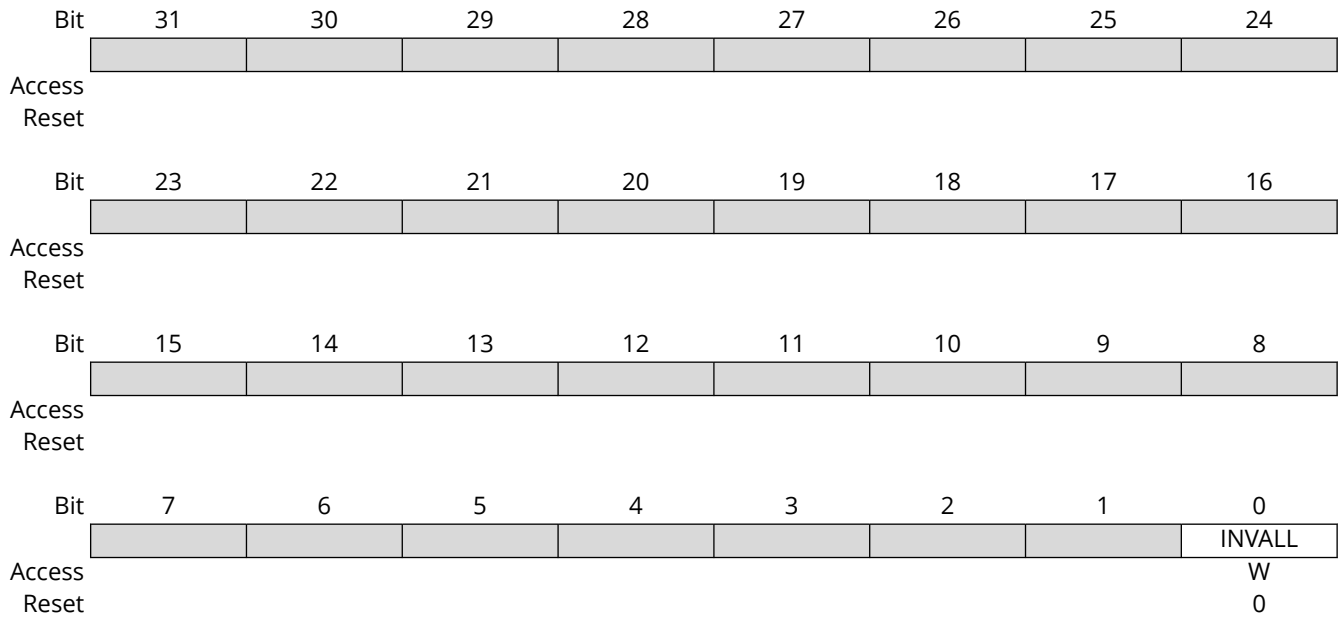
**Name:** LCKWAY  
**Offset:** 0x10  
**Reset:** 0x00000000  
**Property:** Read/Write



**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 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

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

#### Bits 31:28 – WAY[3:0] Invalidate Way

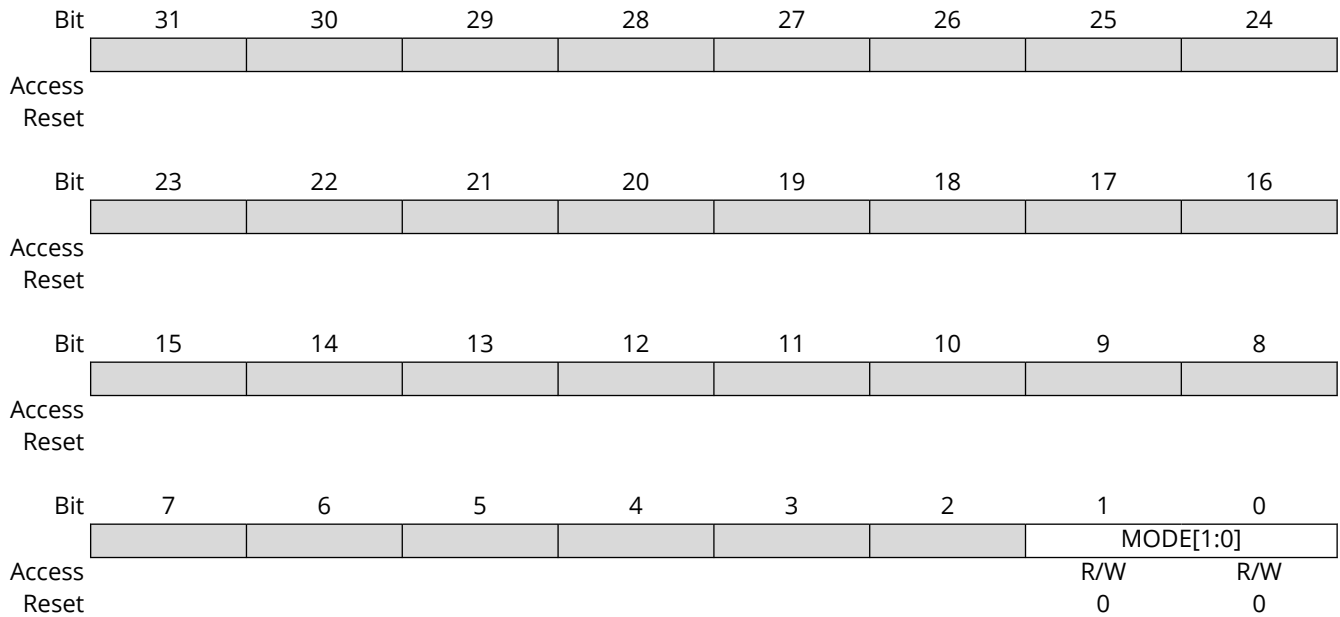
Value	Name	Description
0x0	WAY0	Way 0 is selection for index invalidation
0x1	WAY1	Way 1 is selection for index invalidation
0x2	WAY2	Way 2 is selection for index invalidation
0x3	WAY3	Way 3 is selection for index invalidation
0x4–0xF		Reserved

#### 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



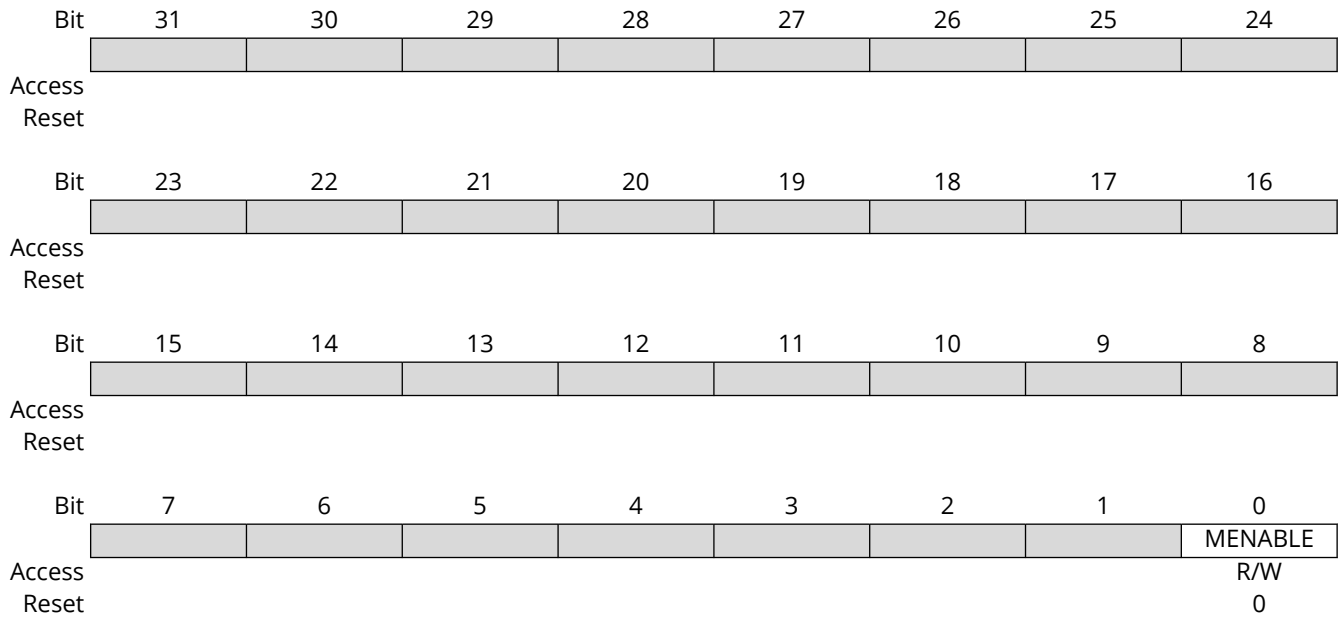
#### Bits 1:0 – MODE[1:0] Cache Controller Monitor Counter Mode

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

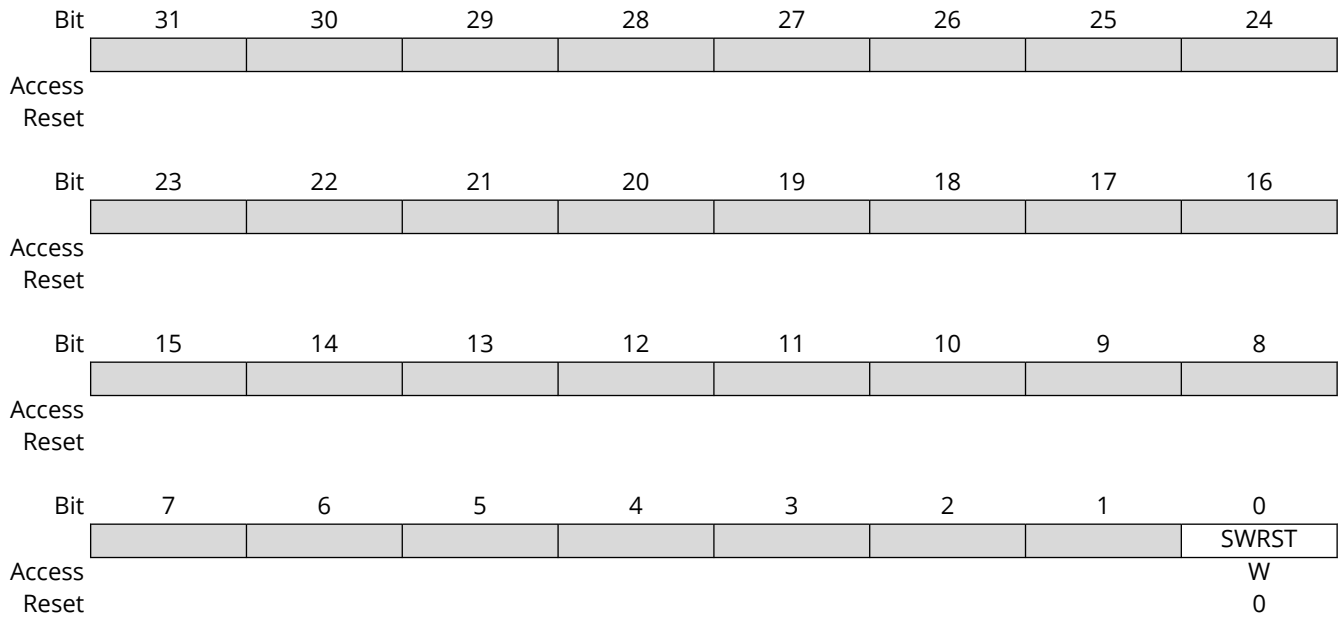


**Bit 0 - MENABLE** Cache Controller Monitor Enable  
 Writing a '0' to this bit disables the monitor counter.  
 Writing a '1' to this bit enables the monitor counter.



### 11.10.10 Cache Monitor Control

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



**Bit 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

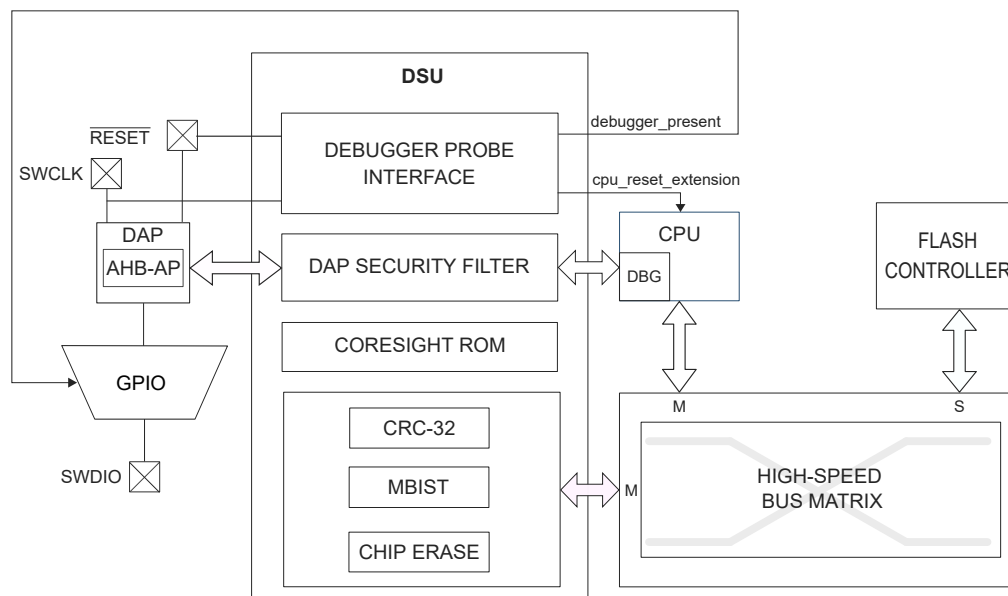
The Device Service Unit (DSU) provides a means of detecting debugger probes. It enables the ARM Debug Access Port (DAP) to have control over multiplexed debug pads and CPU Reset. The DSU also provides system-level services to debug adapters in an ARM debug system. It implements a CoreSight Debug ROM 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
RESET	Digital Input	External Reset pin

.....continued

Signal Name	Type	Description
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:

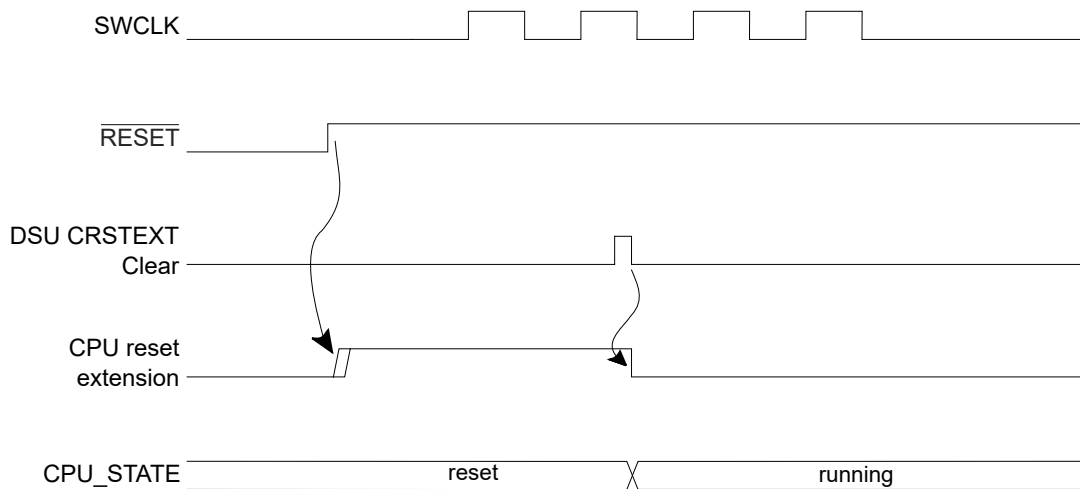
- CPU Reset extension
- Debugger probe detection

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

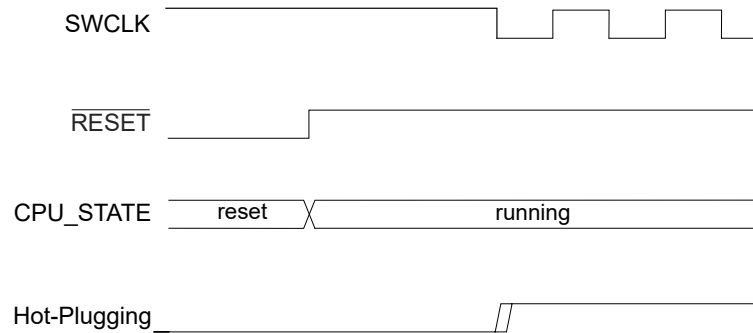
Cold-Plugging is the detection of a debugger when the system is in Reset. Cold-Plugging is detected when the CPU Reset extension is requested, as described above.

#### 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

[13.15.3.2. 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](http://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

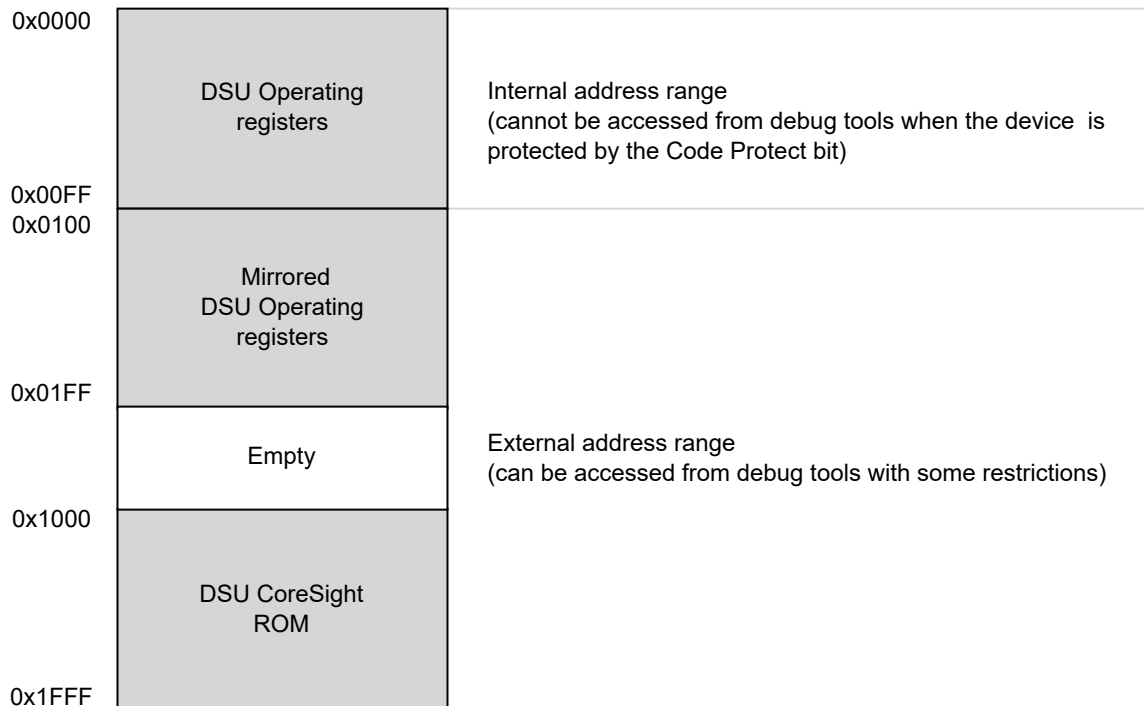
For security reasons, DSU features have limitations when used from a debug adapter. To differentiate external accesses from internal ones, the first 0x100 bytes of the DSU register map has been mirrored at offset 0x100:

- 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**

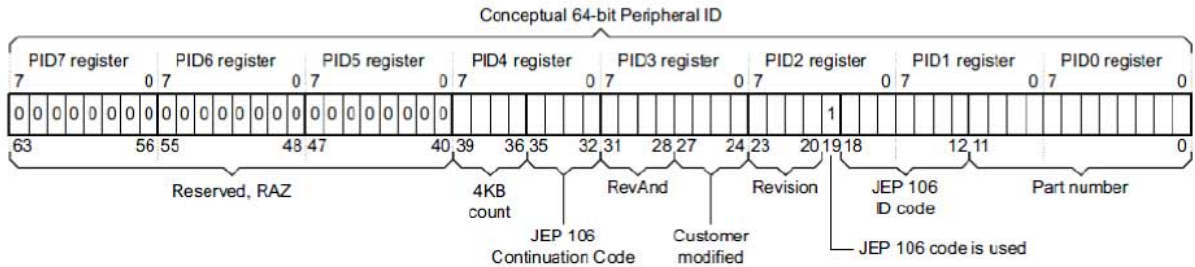
Device identification relies on the ARM CoreSight component identification scheme, which allows the chip to be identified as a SAM device implementing a DSU. The DSU contains identification registers to differentiate the device.

**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	0b000000	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 (DCC0 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									
0x1FD4 ... 0x1FDF	Reserved										

.....continued

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

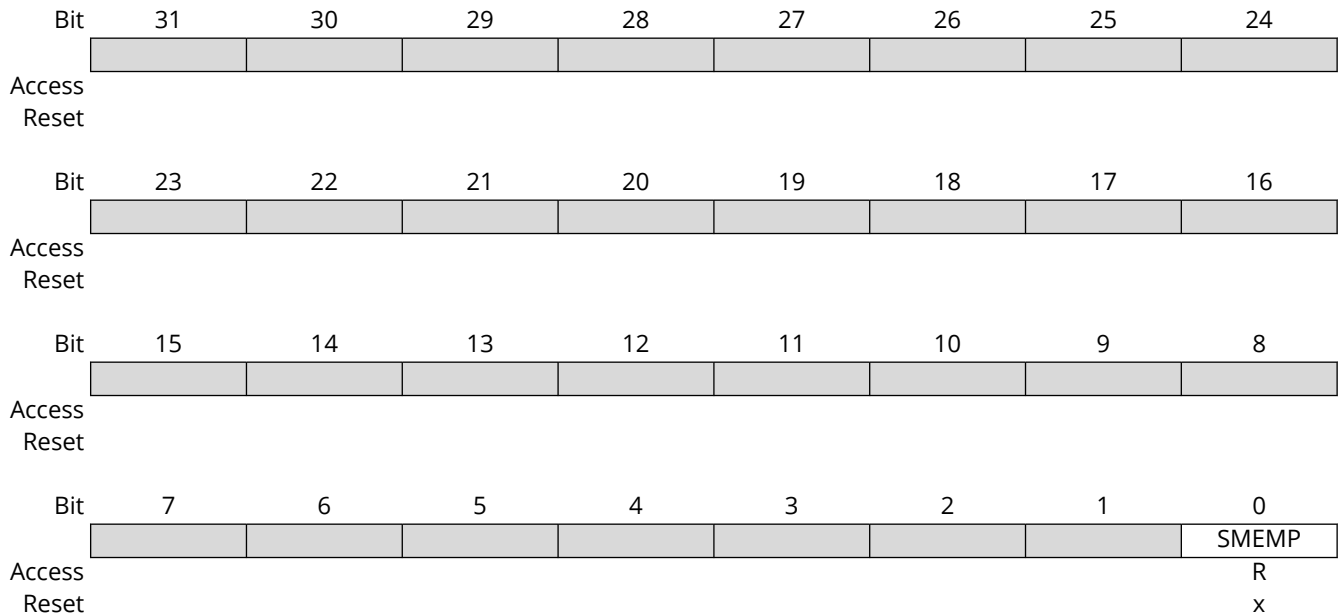
**Name:** END  
**Offset:** 0x1008  
**Reset:** 0x00000000  
**Property:** -

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:** x determined by Debug Access Level (DAL)  
**Property:** -



#### Bit 0 - SMEMP System Memory Present

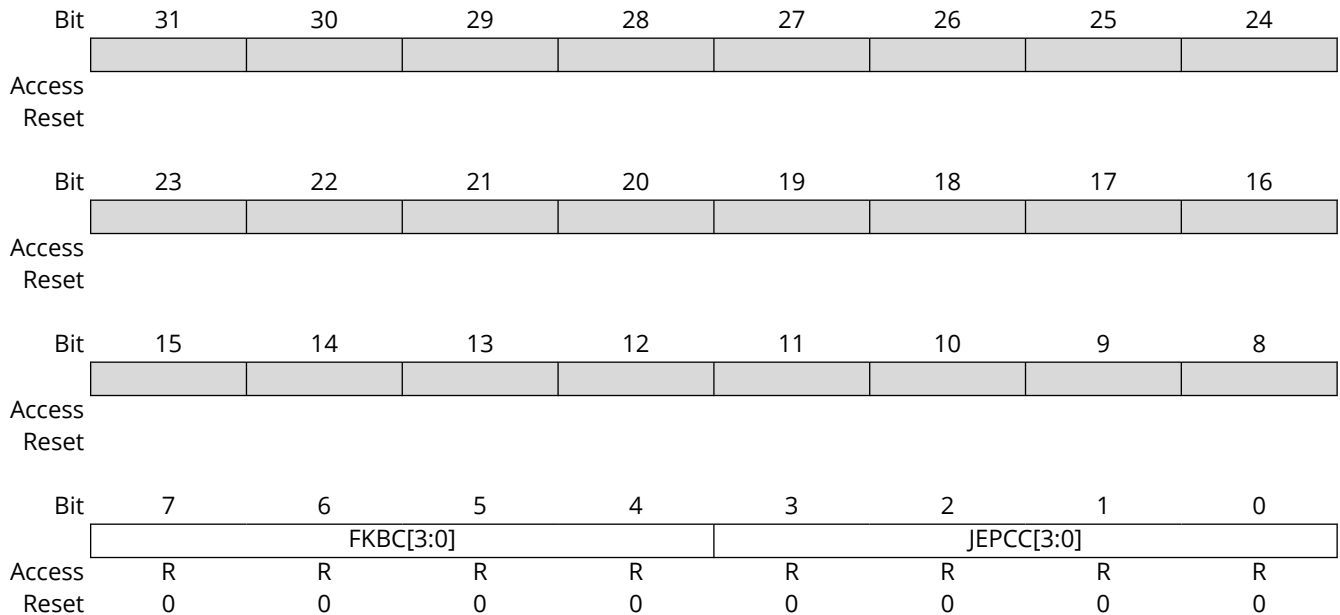
This bit indicates whether system memory is present on the bus that connects to the ROM table. This bit is set at power-up if the device is not protected, indicating that the system memory is accessible from a debug adapter.

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:** -



**Bits 7:4 – FKBC[3:0]** 4KB Count

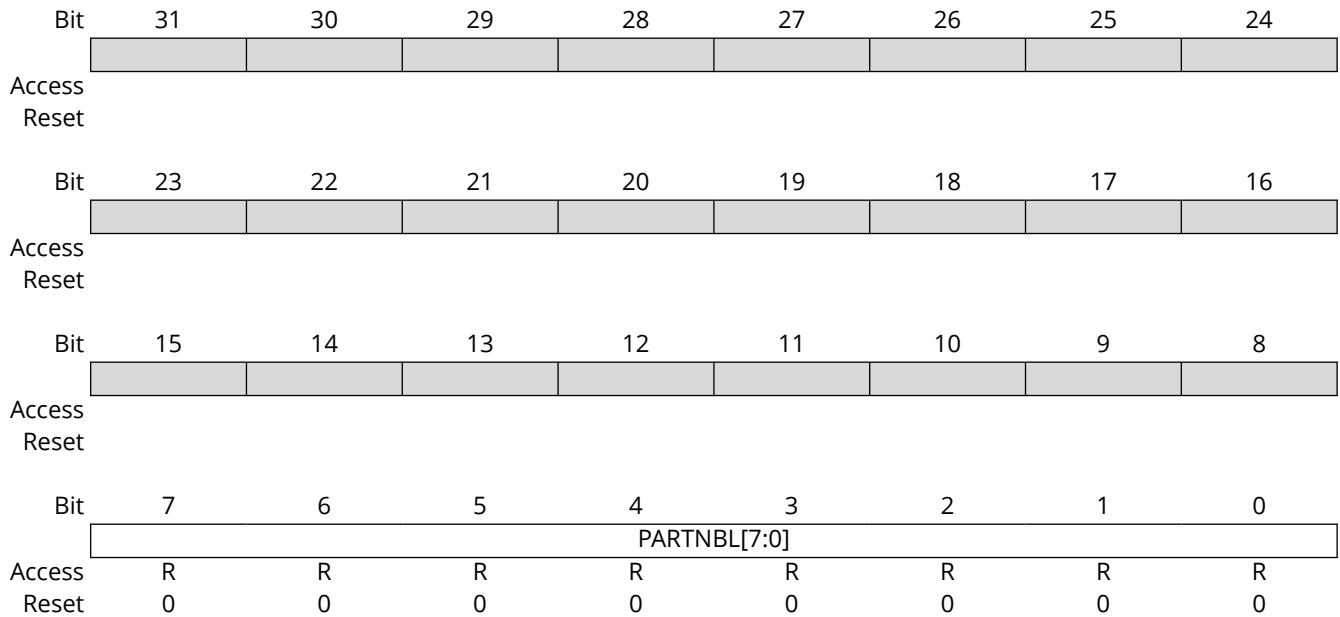
These bits will always return zero when read, indicating that this debug component occupies one 4KB block.

**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:** -

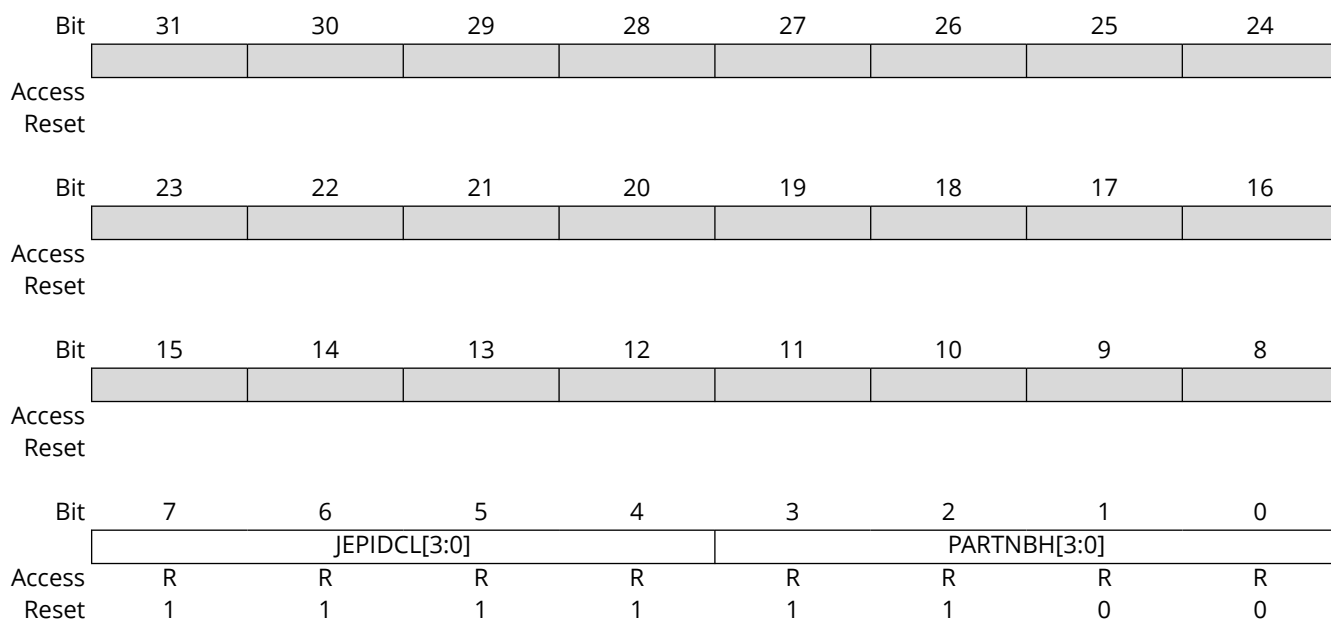


**Bits 7:0 – PARTNBL[7:0] Part Number Low**

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:** -

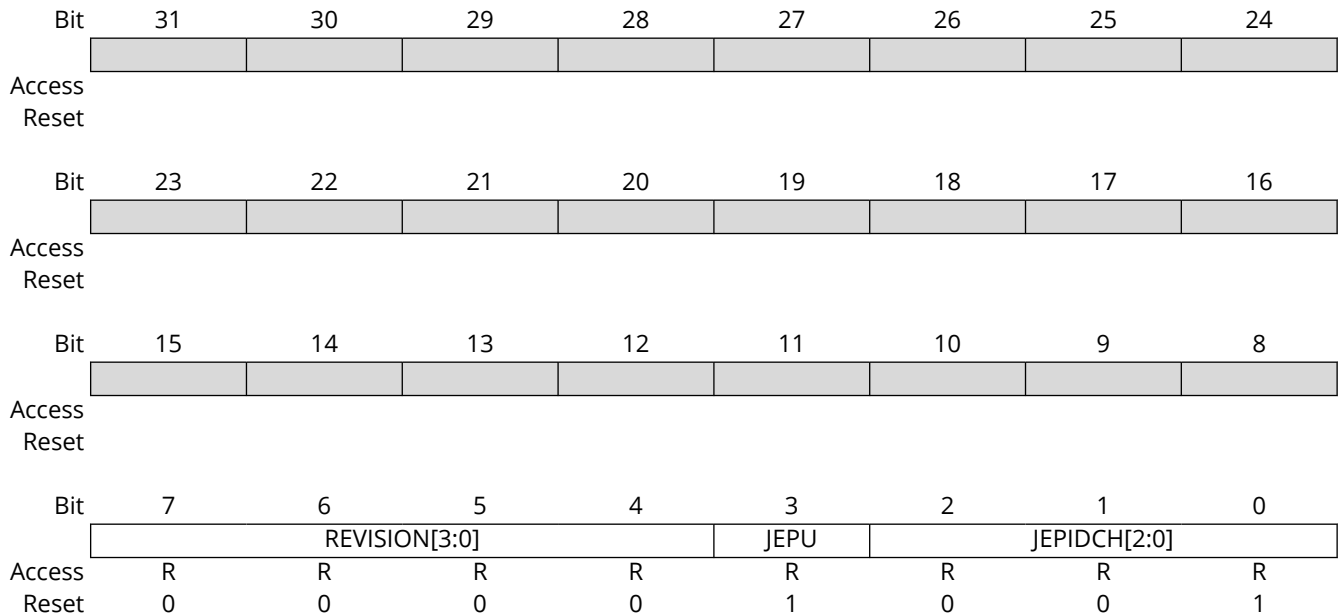


**Bits 7:4 – JEPIDCL[3:0]** Low Part of the JEP-106 Identity Code  
 These bits will always return 0xF when read (JEP-106 identity code is 0x1F).

**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:** -



**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

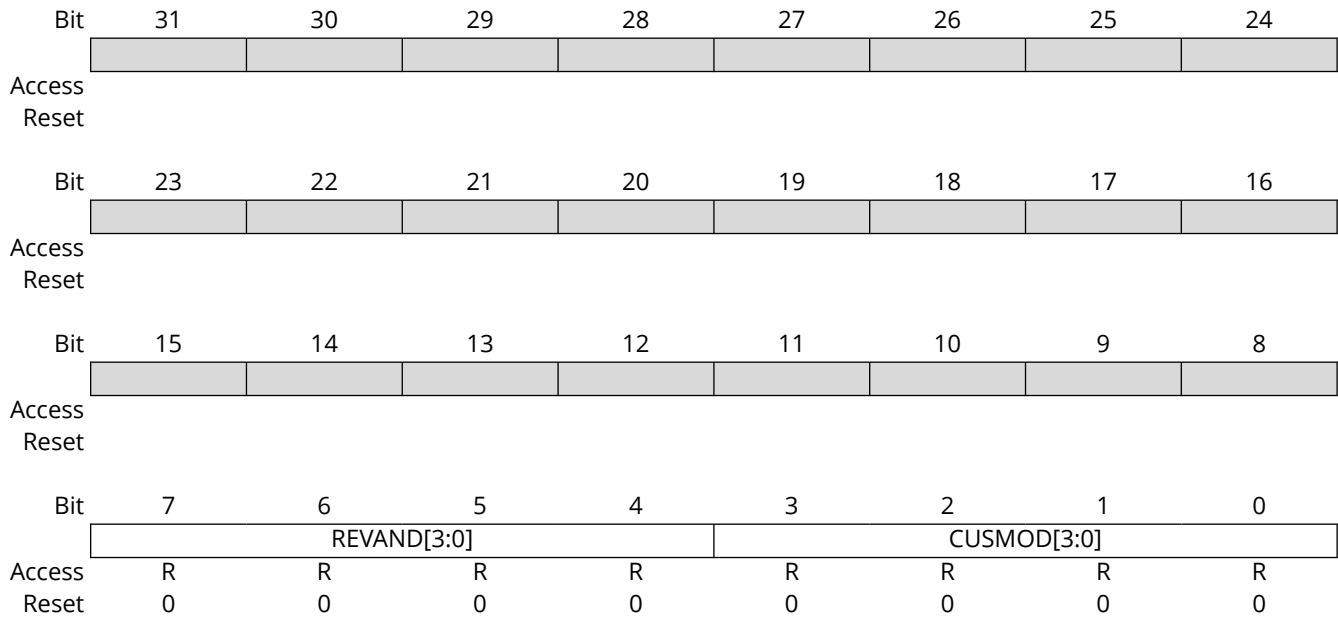
This bit will always return one when read, indicating that JEP-106 code is used.

**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:** -

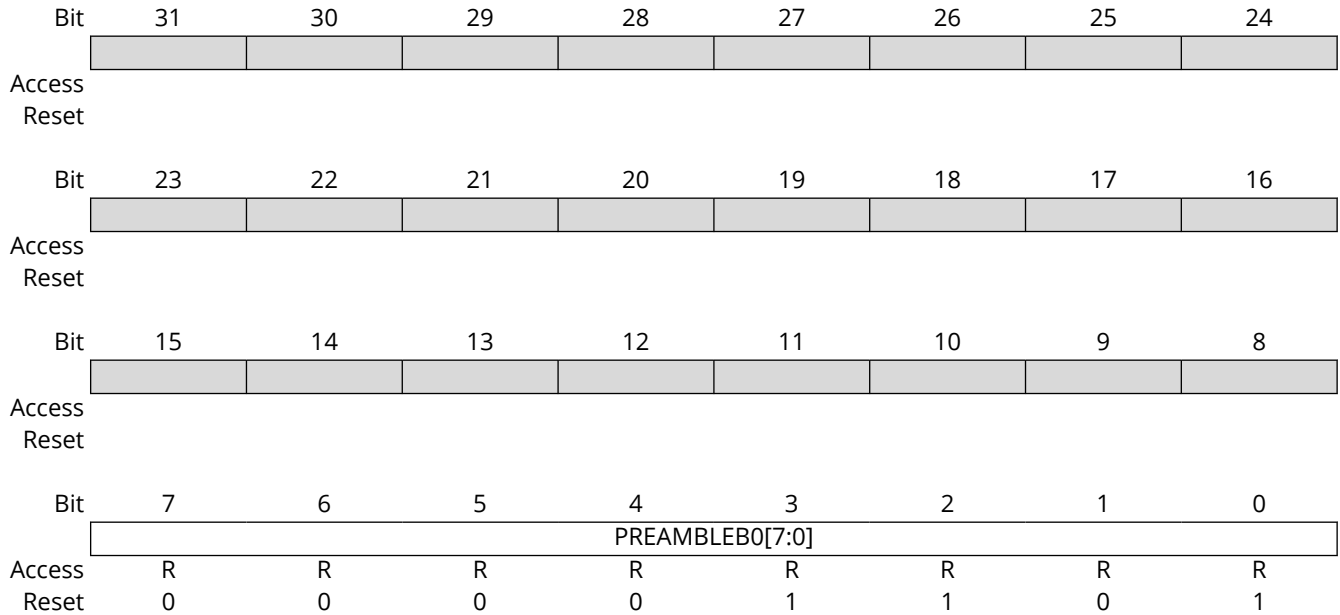


**Bits 7:4 – REVAND[3:0]** Revision Number  
 These bits will always return 0x0 when read.

**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:** -



**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
Access	CCLASS[3:0]				PREAMBLE[3:0]			
Reset	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

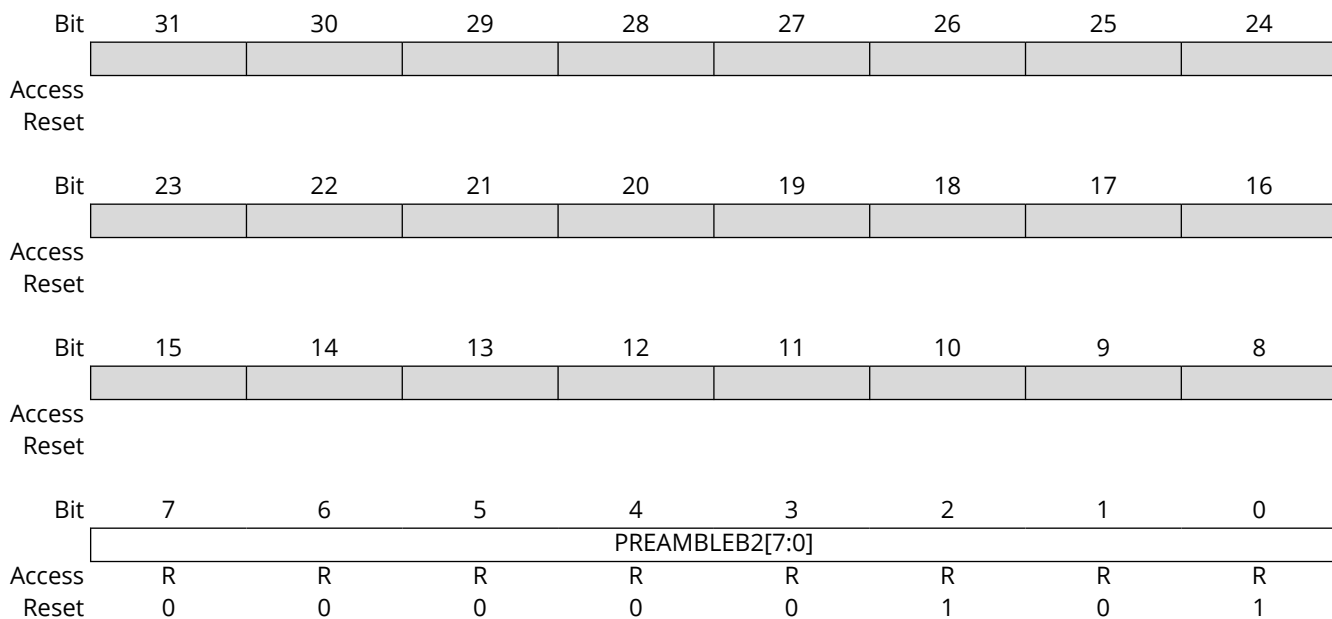
These bits will always return 0x1 when read indicating that this ARM CoreSight component is ROM table (For more details, refer to the *ARM Debug Interface v5 Architecture Specification* at <http://www.arm.com>).

#### 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:** -

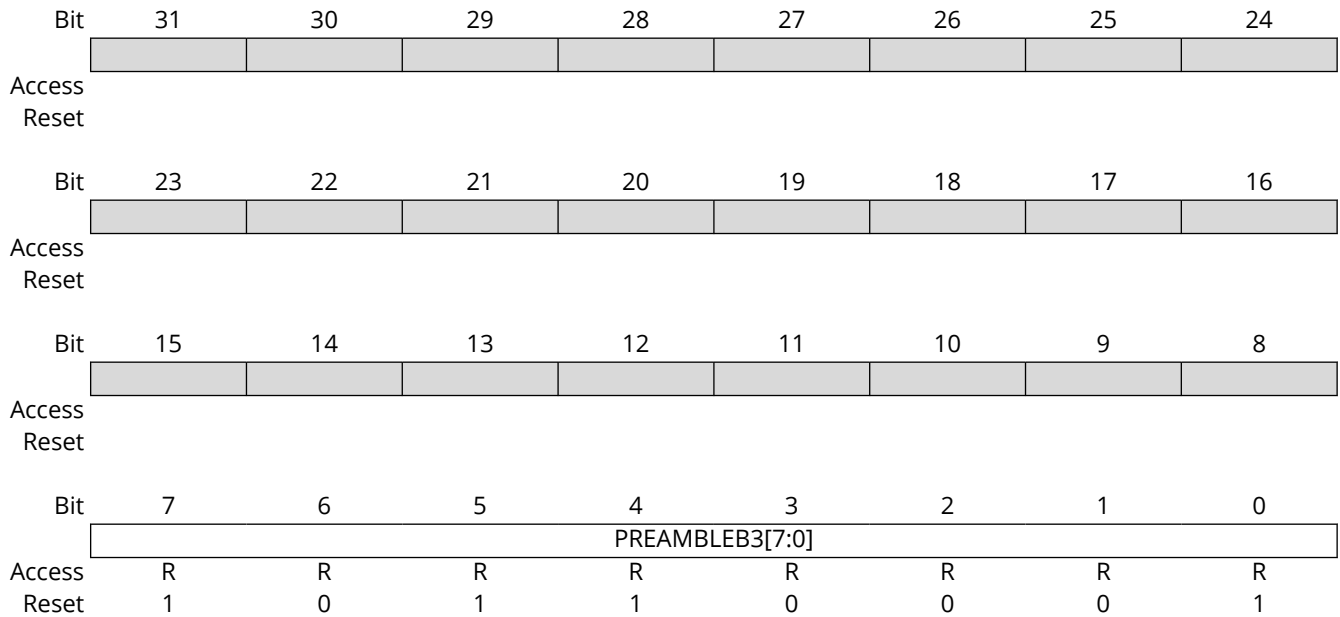


**Bits 7:0 – PREAMBLEB2[7:0]** Preamble Byte 2  
 These bits will always return 0x00000005 when read.



### 12.13.20 Component Identification 3

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



**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

[13.15. 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

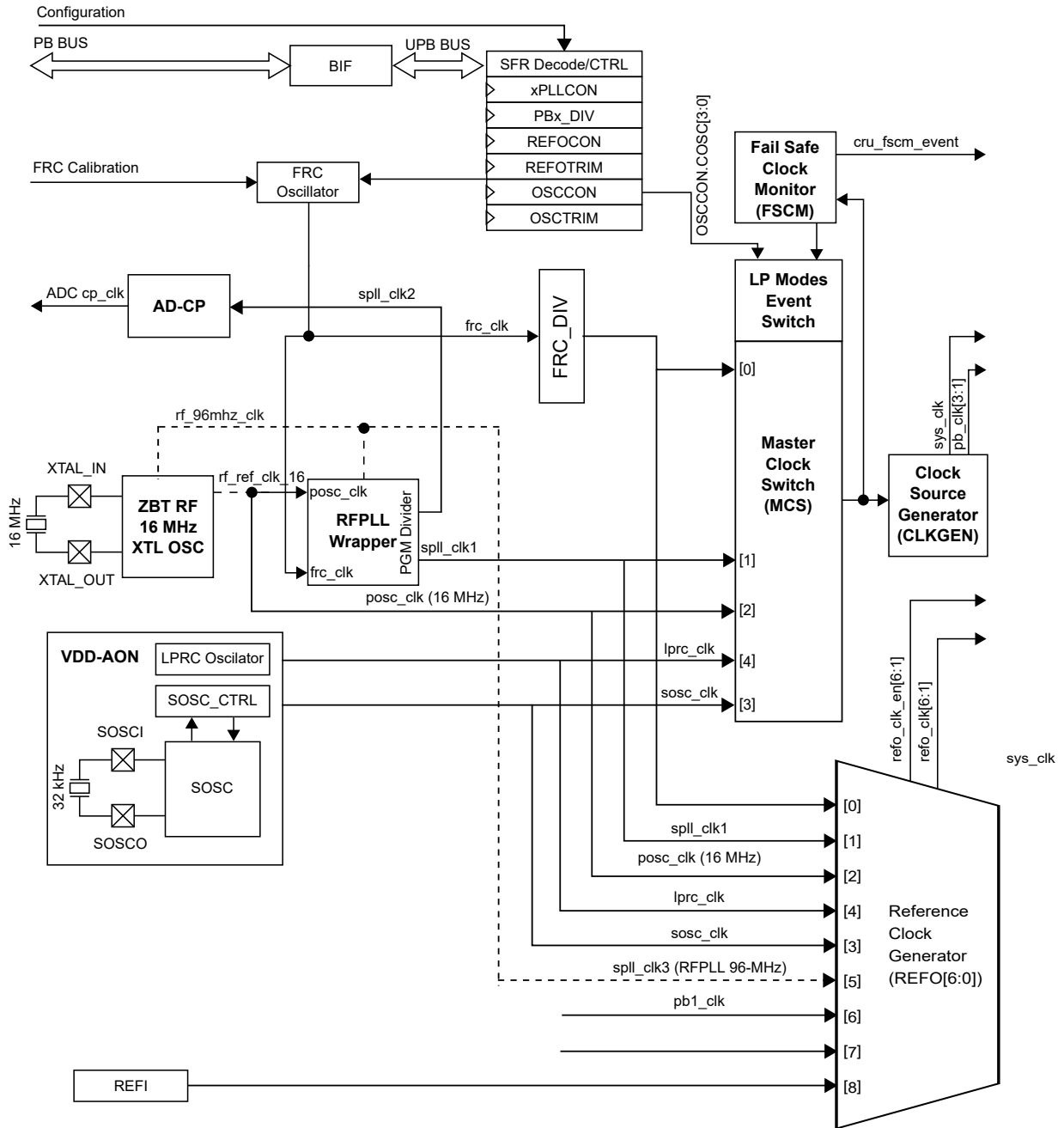


Figure 13-2. RFPLL Wrapper

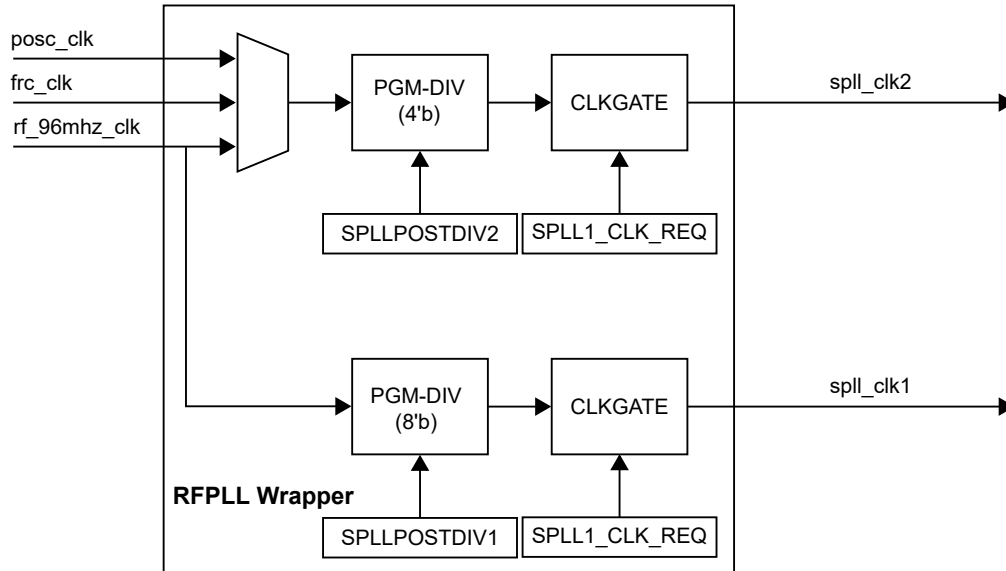
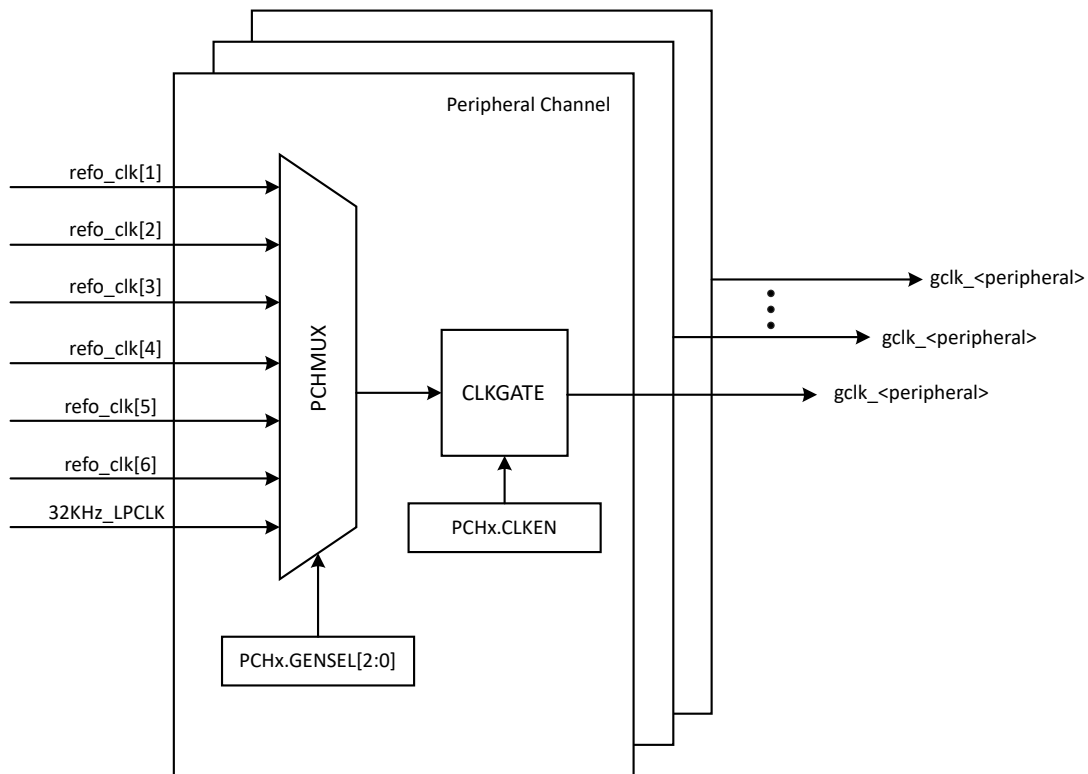


Figure 13-3. Peripheral Clock Generation



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  $\pm 0.25\%$  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  $\pm 1.5\%$  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\_<Peripheral>) 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



.....continued

Peripheral Clock	Pchannel Index
GCLK_SERCOM2_CORE, GCLK_SERCOM3_CORE	4
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 Resets

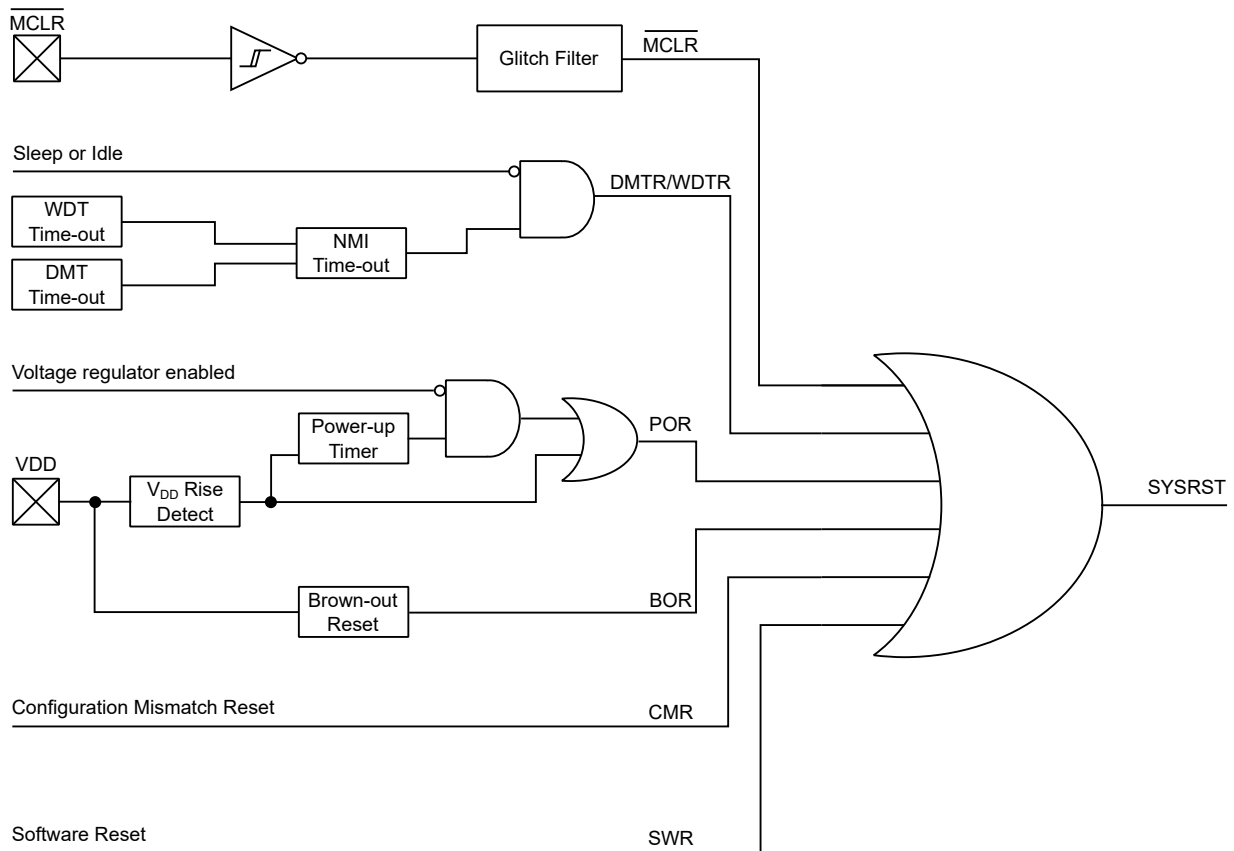
The Reset module combines all Reset sources and controls the device Master Reset signal, SYSRST. The device Reset sources are as follows:

- 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



### 13.15.1 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 Register 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**

[13.17.4. RCON](#)

[13.15.2. Reset Control Registers](#)

[13.15.4.3. Using the RCON Status Bits](#)

### 13.15.2 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	

### 13.15.3 Modes of Operation

#### 13.15.3.1 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.

#### 13.15.3.2 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](#)

### 13.15.3.3 Master Clear Reset

Whenever the master clear pin ( $\overline{MCLR}$ ) is driven low, the Reset event is synchronized with the system clock, SYSCLK, before asserting the system Reset, SYSRST, provided the input pulse on  $\overline{MCLR}$  is longer than a certain minimum width, as specified in the *Electrical Specifications*.

The  $\overline{MCLR}$  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  $\overline{MCLR}$  Reset.

The  $\overline{MCLR}$  pin can be configured to generate a POR event, rather than a normal SYSRST event. This is configured through the SMCLR bit (CFGCON1[14]).

### 13.15.3.4 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  $\overline{MCLR}$  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.

### 13.15.3.5 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 \(CRU\)](#)

### 13.15.3.6 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](#)

### 13.15.3.7 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.

### 13.15.3.8 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 \(DMT\)](#)

### 13.15.3.9 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 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.

### 13.15.3.10 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.

### 13.15.4 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<sup>(1)</sup>

Condition	Program Counter	EXTR	SWR	WDTO	DMTO	SLEEP <sup>(2)</sup>	IDLE <sup>(2)</sup>	CMR	BOR	POR
Power-on Reset or 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
MCLR Reset during the Run mode		1	u	u	u	u	u	u	u	u
MCLR Reset during the Idle mode		1	u	u	u	u	1	u	u	u
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
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

- Legends:
  - u = unchanged
- The SLEEP or IDLE states are entered when the correct sequence plus WAIT instruction is executed.

#### 13.15.4.1 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

[13.15.4. Effects of Various Resets](#)

#### 13.15.4.2 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.

#### 13.15.4.3 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



.....continued

Flag Bit	Set by	Cleared by
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])	NMCLR 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
BCFGERR (RCON[27])	Recoverable error in primary configuration words	User Software, POR, BOR

### 13.15.5 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		2SPDSL						
		31:24					FRCDIV[2:0]				
0x04 ... 0x0F	Reserved										
0x10	OSCTRM	7:0	TUN[5:0]								
		15:8									
		23:16									
		31:24									
0x14 ... 0x1F	Reserved										
0x20	SPLLCON	7:0			SPLLST	SPLLFLOCK	SPLLPWDN				
		15:8	SPLL1POSTDIV1[7:0]								
		23:16				SPLL2POSTDIV2[3:0]					
		31:24	SPLL_BYP[1:0]								
0x24 ... 0x2F	Reserved										
0x30	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	
0x34 ... 0x3F	Reserved										
0x40	RSWRST	7:0								SWRST	
		15:8									
		23:16									
		31:24									
0x44 ... 0x4F	Reserved										
0x50	RNMICON	7:0	NMICNT[7:0]								
		15:8	NMICNT[15:8]								
		23:16	SWNMI				EXT	PLVD	CF	WDTS	
		31:24							DMTO	WDTR	
0x54 ... 0x6F	Reserved										
0x70	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	
0x74 ... 0x7F	Reserved										
0x80	REFO1TRIM	7:0									
		15:8									
		23:16	ROTRIM0								
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1	
0x84 ... 0x8F	Reserved										
0x90	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	

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
0x94 ... 0x9F	Reserved										
0xA0	REFO2TRIM	7:0									
		15:8									
		23:16	ROTRIM0								
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1	
0xA4 ... 0xAF	Reserved										
0xB0	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	
0xB4 ... 0xBF	Reserved										
0xC0	REFO3TRIM	7:0									
		15:8									
		23:16	ROTRIM0								
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1	
0xC4 ... 0xCF	Reserved										
0xD0	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	
0xD4 ... 0xDF	Reserved										
0xE0	REFO4TRIM	7:0									
		15:8									
		23:16	ROTRIM0								
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1	
0xE4 ... 0xEF	Reserved										
0xF0	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	
0xF4 ... 0xFF	Reserved										
0x0100	REFOSTRIM	7:0									
		15:8									
		23:16	ROTRIM0								
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1	
0x0104 ... 0x010F	Reserved										
0x0110	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	
0x0114 ... 0x011F	Reserved										

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
0x0120	REFO6TRIM	7:0									
		15:8									
		23:16	ROTRIM0								
		31:24	ROTRIM8	ROTRIM7	ROTRIM6	ROTRIM5	ROTRIM4	ROTRIM3	ROTRIM2	ROTRIM1	
0x0124 ... 0x012F	Reserved										
0x0130	PB1DIV	7:0					PB1DIV[6:0]				
		15:8	PB1DIVON				PB1DIVRDY				
		23:16									
		31:24									
0x0134 ... 0x013F	Reserved										
0x0140	PB2DIV	7:0					PB1DIV[6:0]				
		15:8	PB1DIVON				PB1DIVRDY				
		23:16									
		31:24									
0x0144 ... 0x014F	Reserved										
0x0150	PB3DIV	7:0					PB1DIV[6:0]				
		15:8	PB1DIVON				PB1DIVRDY				
		23:16									
		31:24									
0x0154 ... 0x015F	Reserved										
0x0160	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]				
0x0164 ... 0x016F	Reserved										
0x0170	CLKSTAT	7:0		SPLL3RDY		LPRCRDY	SOSCRDY	POSCRDY	SPLL1RDY	FRCRDY	
		15:8									
		23:16									
		31:24									
0x0174 ... 0x018F	Reserved										
0x0190	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		2SPDSL					
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 – 2SPDSL 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 – SOSSEN** 32 kHz Secondary (LP) Oscillator Enable bit

Value	Description
1	Enable Secondary Oscillator
0	Disable Secondary Oscillator

**Bit 0 – OSWEN** Oscillator Switch Enable bit

**Notes:**

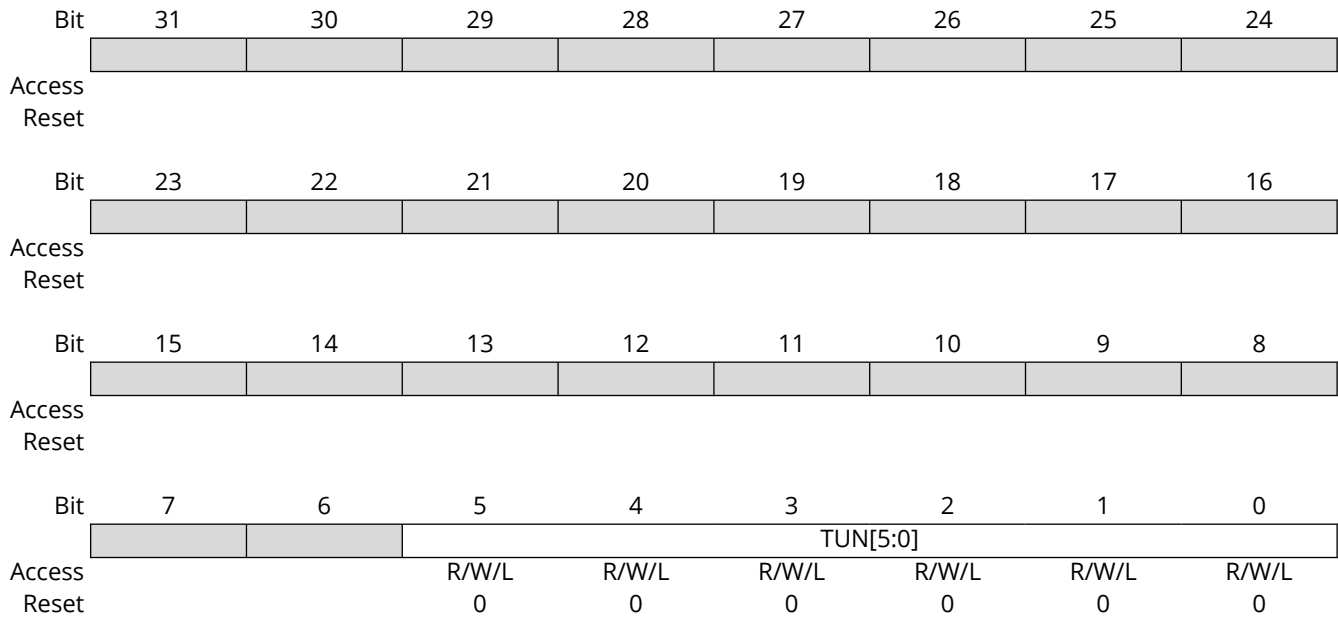
- A Write of value '0' has no effect.
- Cleared by hardware after a successful clock switch
- Cleared by hardware after a redundant clock switch (NOSC = COSC)
- Cleared by hardware after FSCM switches the oscillator to Fail-Safe Clock Source (FRC)

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:** 0x10  
**Reset:** 0x00000000



#### 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:** 0x20  
**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
			SPLL_RST	SPLL_FLOCK	SPLL_PWDN			
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 ≤ SPLL2POSTDIV2 ≤ 15	Divide-by SPLL2POSTDIV2
0	No Clock; Clock disabled

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

Value	Description
2 ≤ SPLL1POSTDIV1 ≤ 255	Divide-by SPLL1POSTDIV1
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 – SPLLLOCK** 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 – SPLLWPN** PLL Power Down Register bit

Value	Description
1	PLL is powered down
0	PLL is active

### 13.17.4 Reset Control Register

**Name:** RCON  
**Offset:** 0x30  
**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 – DMT0** 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.

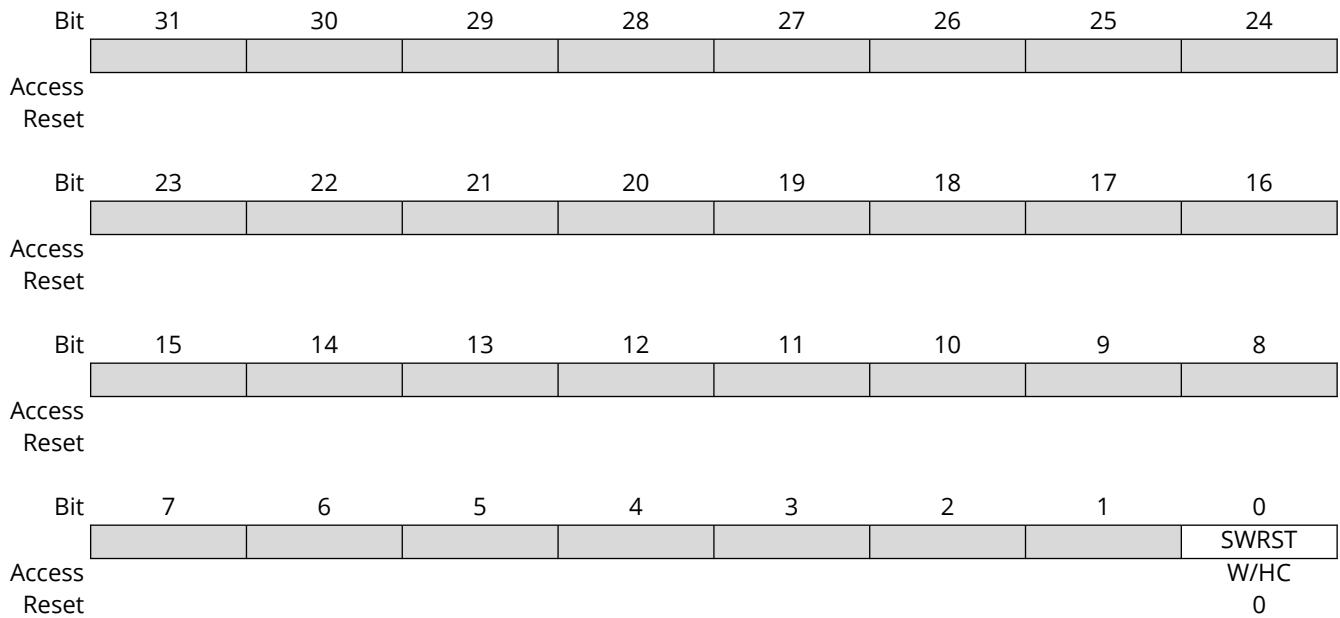
**Note:** User may write this bit to '1'. Does not cause a POR.

Value	Description
1	A Power-on Reset has occurred
0	A Power-on Reset has not occurred

### 13.17.5 Software Reset Register

**Name:** RSWRST  
**Offset:** 0x40  
**Reset:** 0x00000000  
**Property:** -

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



**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'.

### 13.17.6 NMI Control Register

**Name:** RNMICON  
**Offset:** 0x50  
**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.

### 13.17.7 Reference Oscillator x Control

**Name:** REFOxCON  
**Offset:** 0x70 + (x-1)\*0x20 [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
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

### 13.17.8 Reference Oscillator x Trim

**Name:** REFOxTRIM  
**Offset:**  $0x80 + (x-1)*0x20$  [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

### 13.17.9 PBx Clock Divisor Control

**Name:** PBxDIV  
**Offset:** 0x0130 + (x-1)\*0x10 [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
Access	PB1DIVON				PB1DIVRDY			
Reset	R				R			
Reset	1				1			
Bit	7	6	5	4	3	2	1	0
Access	PB1DIV[6:0]							
Reset		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 <b>Note:</b> 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

### 13.17.10 Slew Rate Control for Clock Switching

**Name:** SLEWCON  
**Offset:** 0x160  
**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 <sup>1</sup> ), then no divisor
010	Divide by 4 (2 <sup>2</sup> ), then by 2, then no divisor
011	Divide by 8 (2 <sup>3</sup> ), then by 4, then by 2, then no divisor

Value	Description
100	Divide by 16 (2 <sup>4</sup> ), then by 8, then by 4, then by 2, then no divisor
...	...
111	Divide by 128 (2 <sup>7</sup> ), 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

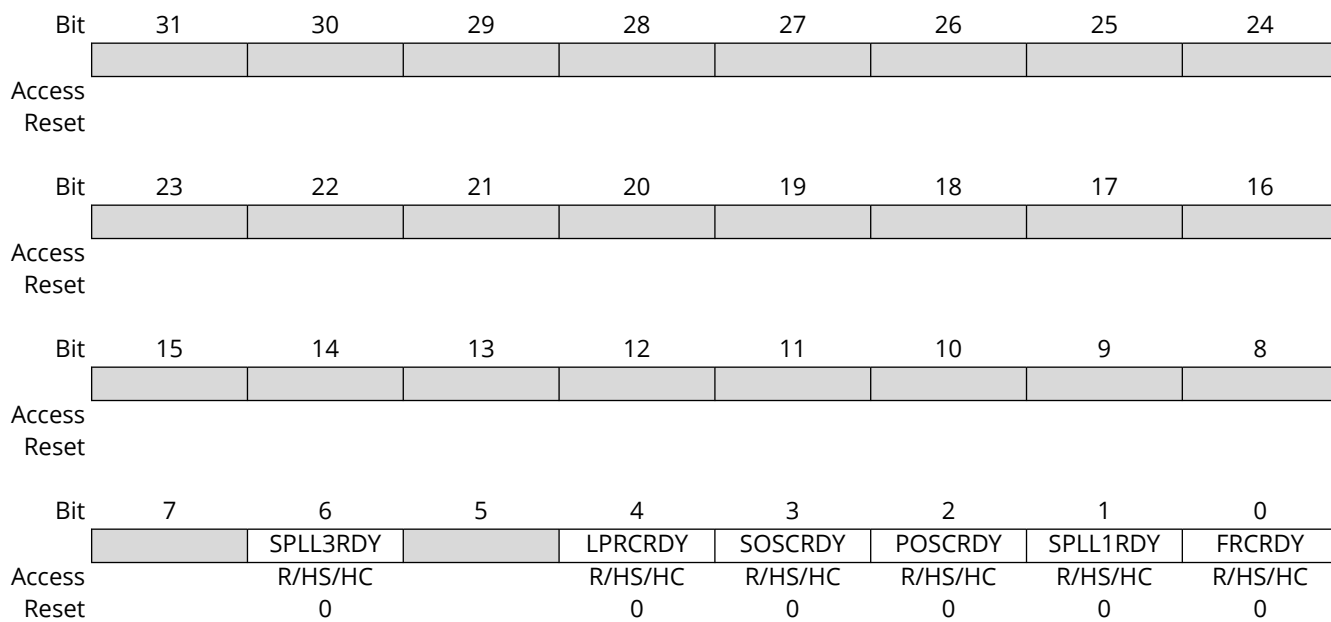
**Bit 0 – SLW\_BUSY** Clock Switch Slewing Active Status Bit – Read-Only

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

### 13.17.11 Clock Status

**Name:** CLKSTAT  
**Offset:** 0x170  
**Reset:** 0x00000000

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



#### 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
0	FRC is not stable and not ready

### 13.17.12 User Clock Diagnostics Control

**Name:** CLK\_DIAG  
**Offset:** 0x190  
**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

## 14. RAM Error Correction Code (RAMECC)

### 14.1 Overview

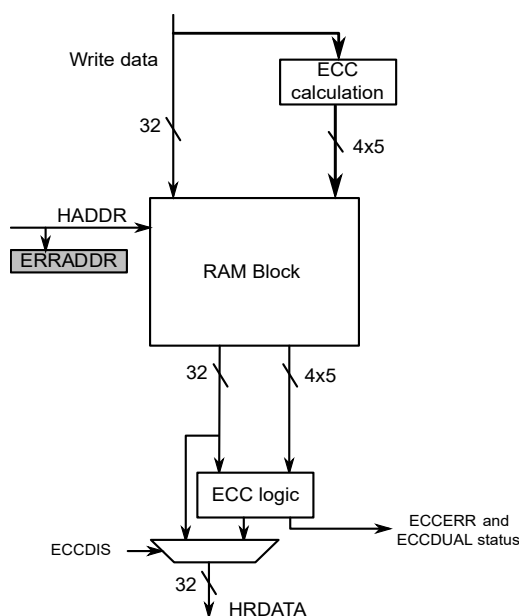
Single bit error correction and dual bit error detection is available for RAM.

### 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.SINGLEEE are cleared. If a dual bit error occurs while INTFLAG.SINGLEEE is set, the ERRADDR register is updated with the dual bit error information and INTFLAG.DUALE is also set.

The INTFLAG.SINGLEEE 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 (SINGLEEE): 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	<a href="#">INTENCLR</a>	7:0							DUALE	SINGLEE
0x01	<a href="#">INTENSET</a>	7:0							DUALE	SINGLEE
0x02	<a href="#">INTFLAG</a>	7:0							DUALE	SINGLEE
0x03	<a href="#">STATUS</a>	7:0								ECCDIS
0x04	<a href="#">ERRADDR</a>	7:0	ERRADDR[7:0]							
		15:8	ERRADDR[15:8]							
		23:16								
		31:24	ERRADDR[16]							
0x08	Reserved									
...										
0x0E										
0x0F	<a href="#">DBGCTRL</a>	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	SINGLEEE
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 – SINGLEEE Single Bit ECC Error Interrupt

This flag is set on the occurrence of a single bit ECC error.

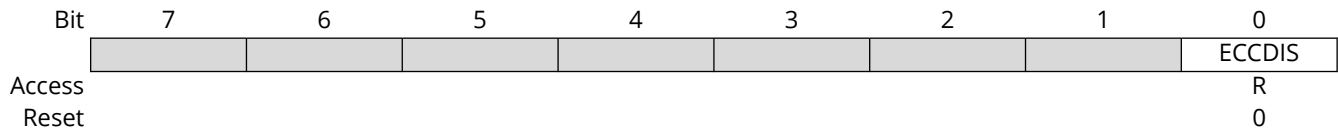
Writing a '0' to this bit has no effect.

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, Write-Protected



##### 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:** R

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access								ERRADDR[16]
Reset								R 0
Bit	15	14	13	12	11	10	9	8
Access	ERRADDR[15:8]							
Reset	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
Access	ERRADDR[7:0]							
Reset	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.SINGLEEE 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
Access							ECCELOG	ECCDIS
Reset							R/W	R/W
							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 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 <sup>1</sup>	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	On Demand	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 <sup>(1)</sup>
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 <sup>(2)</sup> , 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.
3. For more information, see [DSCON](#) register from Related Links.

**Related Links**

- [15.4. DSCON](#)
- [20.4.2. Register Description](#)
- [43. Electrical Characteristics](#)

### 15.3 PMU Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	DSCON	7:0							ZPBOR	DSSR
		15:8	DSEN		XSEMAEN	RTCMD				RTCCWDIS
		23:16								
		31:24								



## 15.4 Deep Sleep Control Register

**Name:** DSCON  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** -

**Note:**

1. All register bits are reset only in the case of a  $V_{DDBAT}$  POR event.

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	DSEN		XSEMAEN	RTCMD				RTCCWDIS
Reset	R/W/HC		R/W	R/W				R/W
Reset	0		0	0				0
Bit	7	6	5	4	3	2	1	0
Access							ZPBOR	DSSR
Reset							R/W/C/HS	R/C/HS/HC
Reset							0	0

### Bit 15 – DSEN Deep Sleep Enable Bit

This bit is only writable when `cfg_deep_sleep_en = 1`.

Value	Description
1	Deep Sleep mode is entered on a SLEEP/WAIT command
0	Sleep mode is entered on a SLEEP/WAIT command

### Bit 13 – XSEMAEN Extended Semaphore Enable Bit

Value	Description
1	Extended semaphores retention is enabled in Deep Sleep mode
0	Indeterminate extended semaphore retention in Deep Sleep

### Bit 12 – RTCMD RTCC Module Disable Bit

Value	Description
1	RTCC is not enabled
0	RTCC is enabled

### Bit 8 – RTCCWDIS RTCC Wake-up Disable Bit

Value	Description
1	Wake-up from RTCC is disabled
0	Wake-up from RTCC is enabled

### Bit 1 – ZPBOR Zero-Power BOR Event Bit

**Note:**

Unlike all other events, a Zero-Power BOR event will not cause a wake-up from Deep Sleep. This bit is present only as a status bit.

Value	Description
1	The ZPBOR was active and a BOR event was detected during Deep Sleep
0	The ZPBOR was not active or was active, but did not detect a BOR event during Deep Sleep

**Bit 0 – DSSR** Deep Sleep State Restored Bit

Clearing this bit will cause the `xds_keepctrl_en_lv` output to be negated, indicating it is safe to release all Deep Sleep configuration keeper cells. If the wake-up source was something other than an MCLR or ICD Reset, the `xds_keepio_en_lv` output will also be negated at the same time, indicating it is safe to release all I/O keeper cells. This bit must be cleared by software.

## 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 WDT PSS[4:0] inputs for the Sleep Mode Counter. These values range from 1 to 2<sup>20</sup> 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.

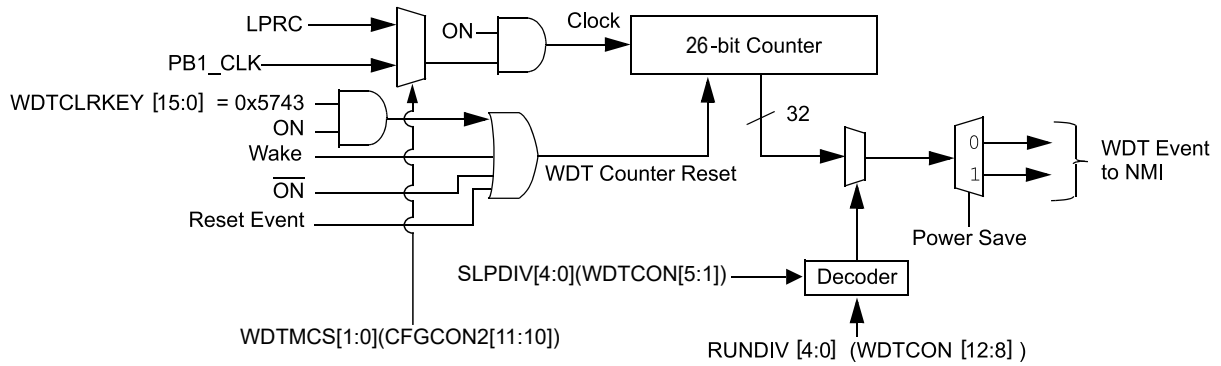
The WDT is reset after the count is exceeded and continues to run.

### 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 WDT PSS[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.

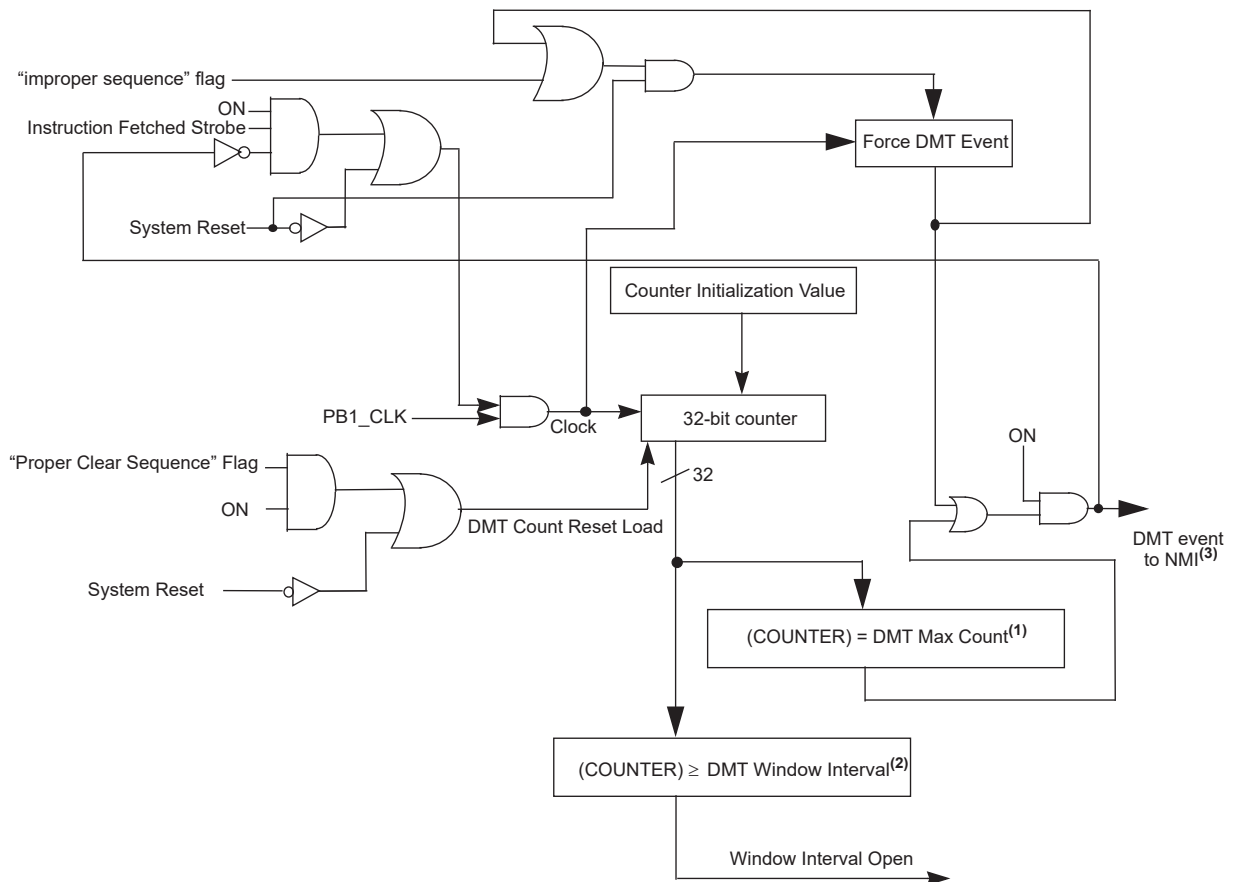
### 17.2 Features

- 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**[13.15. 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.

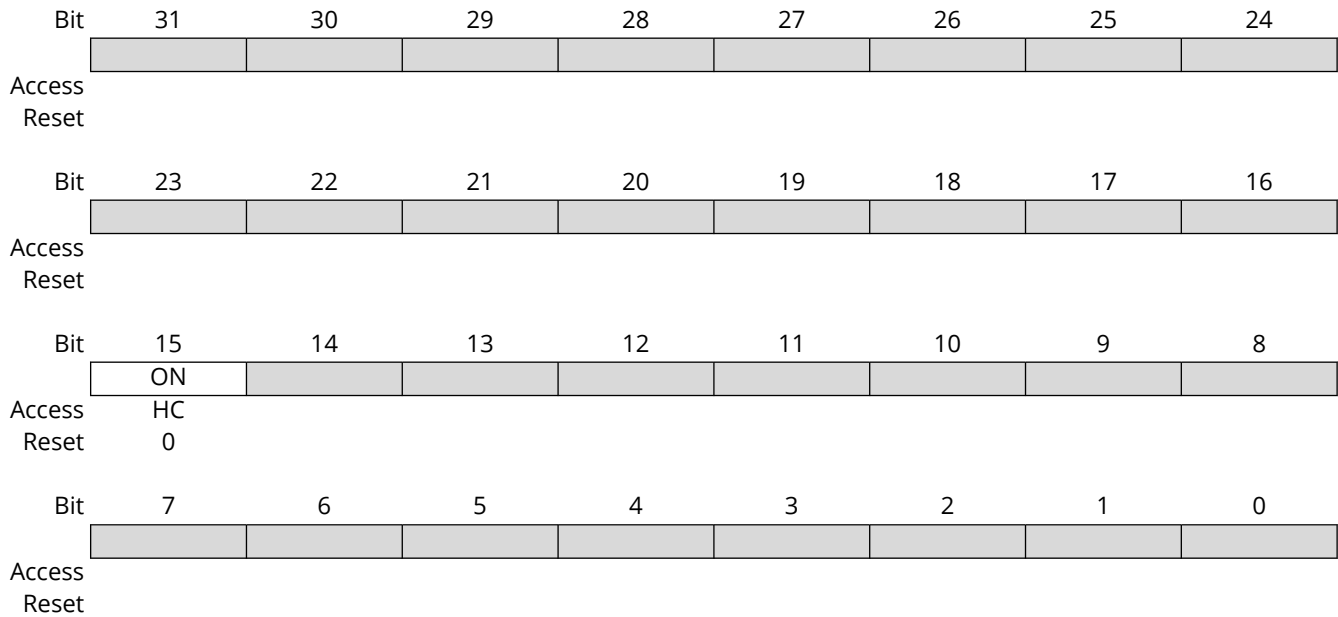
**Note:** All registers in this table have corresponding CLR, SET and INV registers at their virtual addresses plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET and INV Registers* from Related Links

### Related Links

[6.1.9. CLR, SET and INV Registers](#)

### 17.6.1 Deadman Timer Control

**Name:** DMTCON  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** -



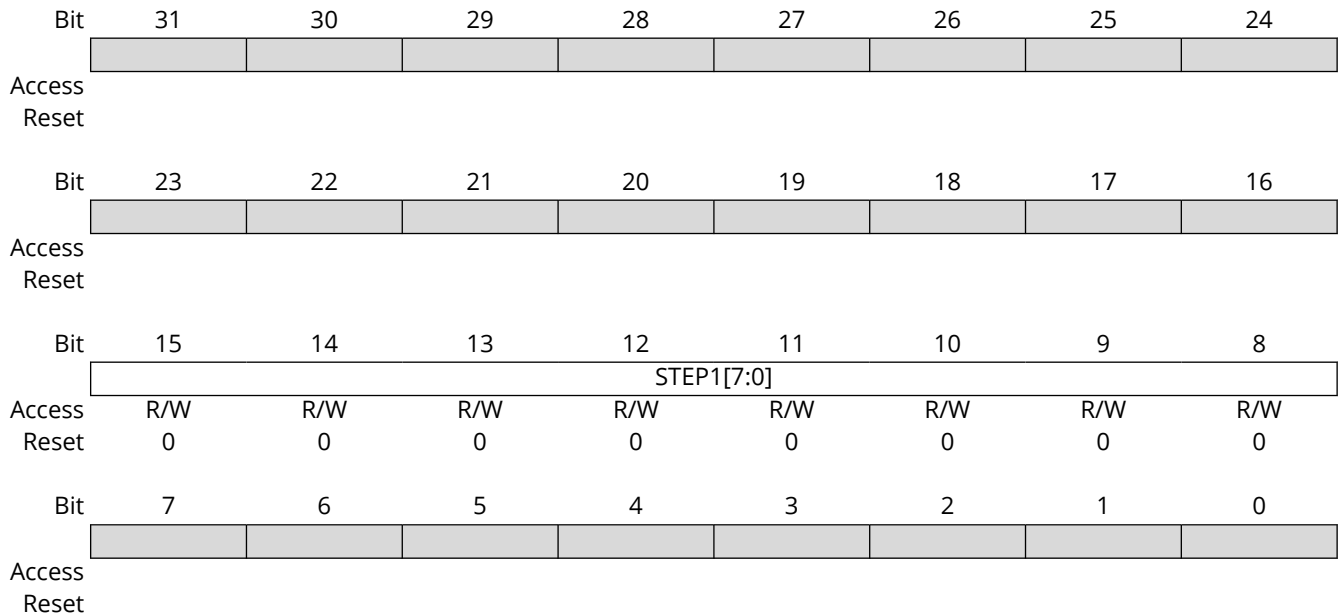
#### Bit 15 - ON On bit

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:** -

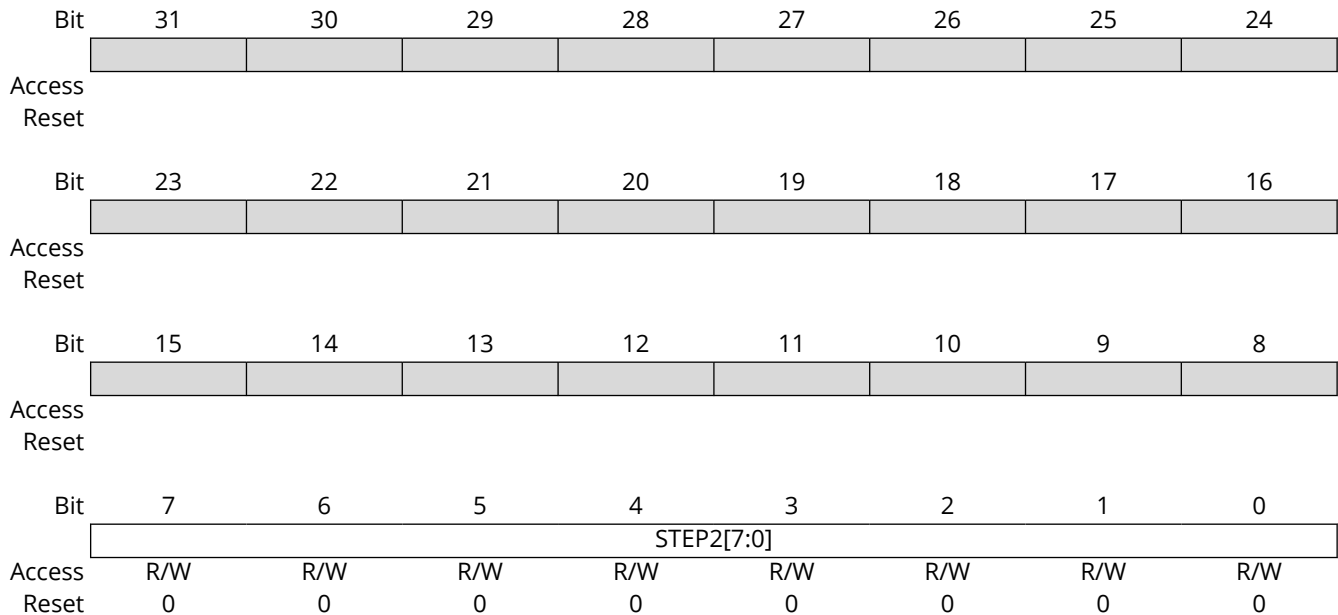


### Bits 15:8 – STEP1[7:0] Pre-Clear Enable bit when write pattern is:

Value	Description
01000000	Enables the Deadman Timer Pre-Clear (STEP 1).
all other write patterns	Sets DMTSTAT.BAD1 flag to '1'. <b>Note:</b> 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:** -



#### 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. <b>Note:</b> 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
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
Access	UPRCNT[15:8]							
Reset	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
Access	UPRCNT[7:0]							
Reset	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	ALTFCFG0(0004_5E9C)
DEVCFG0	ALTDEVCFG0(0004_5E98)
DEVCFG1	ALTDEVCFG1(0004_5E94)
DEVCFG2	ALTDEVCFG2(0004_5E90)
DEVCFG4	ALTDEVCFG4(0004_5E8C)

.....continued

Register	Alternate Register
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	ACCOMP1_ALT EN	GPSOSCE*	ADCOPVR	JTAGEN	TROEN	SWOEN	TDOEN
		15:8	CFGLOCK[1:0]		IOLOCK	PMDLOCK	PGLOCK	PMULOCK	RTCOUT_ALTE N	RTCIN0_ALTE N
		23:16	SLRTEN2	SLRTEN1	SLRTEN0	HPLUGDIS	SMBUSEN2	SMBUSEN1	SMBUSEN0	VBCMODE
		31:24		FRECCDIS	FECCON/ECCTTL[1:0]		ADCFEN	INTOP	INTOE	PCM
0x04 ... 0x0F	Reserved									
0x10	CFGCON1(L)	7:0	ZBTWKSYS		TRCEN	ICESEL[1:0]			DEBUG[1:0]	
		15:8	QSCHEN	SMCLR	SLRCTRL2	SLRCTRL1	SLRCTRL0	CLASSBDIS	CMP1_OE	CMP0_OE
		23:16	I2CDSEL2	I2CDSEL1	I2CDSEL0	CCL_OE	SCOM2_HSEN	SCOM1_HSEN	SCOM0_HSEN	QSPI_HSEN
		31:24		CLKZBREF	QSPIDDRM	WDTPSS[4:0]				
0x14 ... 0x1F	Reserved									
0x20	CFGCON2(L)	7:0	PMUTEST_VD D_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_MO D	VBKP_DIVSEL	VBKP_32KCSEL[1:0]		VBKP_1KCSEL	RTCEVENT_E N	RTCEVENTSEL[1:0]	
		23:16	DSWDTPS[2:0]		DSZPBOREN		CPEN_DLY[2:0]		RTCEVTYPE	
		31:24	RTCNTM_CSE L	LPOSCEN	UVREGROVR	DSBITEN	DSWDTEN	DSWDTLPRC	DSWDTPS[4:3]	
0x44 ... 0x4F	Reserved									
0x50	CFGPGQOS	7:0	ICDQOS[1:0]		ICDPG[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	S01CD	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	EVSYS2SEL[2:0]		C1D		EVSYS1SEL[2:0]		
		15:8	C4D	EVSYS4SEL[2:0]		C3D		EVSYS3SEL[2:0]		
		23:16	C6D	EVSYS6SEL[2:0]		C5D		EVSYS5SEL[2:0]		
		31:24	C8D	EVSYS8SEL[2:0]		C7D		EVSYS7SEL[2:0]		
0x74 ... 0x7F	Reserved									
0x80	CFGPCLKGEN3	7:0	C10D	EVSYS10SEL[2:0]		C9D		EVSYS9SEL[2:0]		
		15:8	C12D	EVSYS12SEL[2:0]		C11D		EVSYS11SEL[2:0]		
		23:16	TCC0CD	TCC0CSEL[2:0]		ACCD		ACCLKSEL[2:0]		
		31:24	TC1CD	TC1CSEL[2:0]		TC0CD		TC0CSEL[2:0]		

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	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/ECCTL[1:0]	ADFCEN	INTOP	INTOE	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_ALT EN	RTCIN0_ALTE N
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_ALT EN	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/ECCTL[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

**Notes:**

- This bit is only writable when CFGLOCK[1:0] is '00'.
- This bit is only applicable in 48-pin variants.

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  
**Notes:**

- This bit is only writable when CFGLOCK[1:0] is '00'.
- This bit is only applicable in 48-pin variants.

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 variants.

**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	I2CDSELO	CCL_OE	SCOM2_HSE N	SCOM1_HSE N	SCOM0_HSE N	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 is enabled
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
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

**Notes:**

- This bit is only writable when CFGLOCK[1:0] is '00'.
- This bit is only applicable in 48-pin variants.

Value	Description
1	I <sup>2</sup> C delay is enabled
0	I <sup>2</sup> C delay is 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 <sup>2</sup> C delay is enabled
0	I <sup>2</sup> C 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 <sup>2</sup> C delay is enabled
0	I <sup>2</sup> C 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

**Bit 19 – SCOM2\_HSEN** SERCOM2 (Direct) Enable

**Notes:**

- This bit is only writable when CFGLOCK[1:0] is '00'.
- This bit is only applicable in 48-pin variants.

Value	Description
1	Direct mode (High-Speed) is enabled
0	Via PPS is enabled

**Bit 18 – SCOM1\_HSEN** SERCOM1 (Direct) Enable

**Note:**

- This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Direct mode (High-Speed) is enabled
0	Via PPS is enabled

**Bit 17 – SCOM0\_HSEN** SERCOM0 (Direct) Enable

**Note:**

- This bit is only writable when CFGLOCK[1:0] is '00'.

Value	Description
1	Direct mode (High-Speed) is enabled
0	Via PPS is enabled

**Bit 16 – QSPI\_HSEN** QSPI (Direct) Enable

**Notes:**

- This bit is only writable when CFGLOCK[1:0] is '00'.
- This bit is only applicable in 48-pin variants.

Value	Description
1	Direct Mode (High-Speed) is enabled
0	Via PPS is enabled

**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

**Notes:**

- This bit is only writable when CFGLOCK[1:0] is '00'.
- This bit is only applicable in 48-pin variants.

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	SOSCSSEL	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_VD D_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 <sup>8</sup> for cfg_dmt_cnt[31:0]
00001	Counter value is 2 <sup>9</sup> for cfg_dmt_cnt[31:0]
...	...
10100	Counter value is 2 <sup>28</sup> for cfg_dmt_cnt[31:0]
10101	Counter value is 2 <sup>29</sup> for cfg_dmt_cnt[31:0]
10110	Counter value is 2 <sup>30</sup> for cfg_dmt_cnt[31:0]
10111	Counter value is 2 <sup>31</sup> for cfg_dmt_cnt[31:0]

Value	Description
11000 –	Reserved
11111	

**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 – WDTE** 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

Value	Description
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 – SOSSEL** 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]
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 $32\mu\text{s} * \text{ACMP\_CYCLE} + 1$ 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_MO	VBKP_DIVSEL	VBKP_32KSEL[1:0]		VBKP_1KSEL	RTCEVENT_E	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 <sup>36</sup> (25.7 days)
11110	1:2 <sup>35</sup> (12.8 days)
11101	1:2 <sup>34</sup> (6.4 days)
11100	1:2 <sup>33</sup> (77.0 hours)
11011	1:2 <sup>32</sup> (38.5 hours)
11010	1:2 <sup>31</sup> (19.2 hours)
11001	1:2 <sup>30</sup> (9.6 hours)
11000	1:2 <sup>29</sup> (4.8 hours)
10111	1:2 <sup>28</sup> (2.4 hours)
10110	1:2 <sup>27</sup> (72.2 minutes)
10101	1:2 <sup>26</sup> (36.1 minutes)
10100	1:2 <sup>25</sup> (18.0 minutes)
10011	1:2 <sup>24</sup> (9.0 minutes)
10010	1:2 <sup>23</sup> (4.5 minutes)
10001	1:2 <sup>22</sup> (135.3 s)
10000	1:2 <sup>21</sup> (67.7 s)
01111	1:2 <sup>20</sup> (33.825 s)
01110	1:2 <sup>19</sup> (16.912 s)
01101	1:2 <sup>18</sup> (8.456 s)
01100	1:2 <sup>17</sup> (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)

Value	Description
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\_32KSEL[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\_1KSEL** 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
	S01CD	SERCOM01CSEL[2:0]			MCD	FREQMMSEL[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 CM4 Trace 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] CM4 Trace 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 TC2 and TC3 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] TC2 and TC3 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 23 – TCC12CD** TCC1 and TCC2 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]** TCC1 and TCC2 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** SERCOM2 and SERCOM3 Peripheral Clock Disable

**Notes:**

- This field is only writable when CFGLOCK[1:0] = '00'.
- This bit is only applicable in 48-pin variants.

Value	Description
0	Clock is disabled
1	Clock is enabled

**Bits 18:16 – SERCOM23CSEL[2:0]** SERCOM2 and SERCOM3 Peripheral Clock Selection

**Notes:**

- This field is only writable when CFGLOCK[1:0] = '00'.
- This bit is only applicable in 48-pin variants.

Value	Description
0	No clock is selected
1-6	REFO1-6 clock is selected
7	Low power clock is selected

**Bit 15 – S01CD** SERCOM0 and SERCOM1 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]** SERCOM0 and SERCOM1 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** FREQM Measurement 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 – FREQMMSEL[2:0]** FREQM Measurement 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** FREQM Reference 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]** FREQM Reference 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** EIC 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]** EIC 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.

Note that the following bits EVSYSCX range from 1-8 which corresponds to channel 0 to 7. 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	EVSYS8SEL[2:0]			C7D	EVSYS7SEL[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	EVSYS6SEL[2:0]			C5D	EVSYS5SEL[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	EVSYS4SEL[2:0]			C3D	EVSYS3SEL[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	EVSYS2SEL[2:0]			C1D	EVSYS1SEL[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 EVSYSC8 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] EVSYSC8 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 EVSYSC7 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] EVSYSC7 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 23 – C6D** EVSYSC6 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]** EVSYSC6 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** EVSYSC5 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]** EVSYSC5 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** EVSYSC4 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]** EVSYSC4 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** EVSYSC3 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]** EVSYSC3 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** EVSYSC2 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]** EVSYSC2 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** EVSYSC1 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]** EVSYSC1 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.

Note that the following bits EVSYSCX range from 9-12 which corresponds to channel 8 to 11. 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	EVSYS12SEL[2:0]			C11D	EVSYS11SEL[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	EVSYS10SEL[2:0]			C9D	EVSYS9SEL[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 TC1 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] TC1 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 TC0 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] TC0 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 23 – TCC0CD** TCC0 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]** TCC0 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** AC 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]** AC 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** EVSYSC12 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]** EVSYSC12 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** EVSYSC11 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]** EVSYSC11 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** EVSYSC10 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]** EVSYSC10 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** EVSYSC9 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]** EVSYSC9 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 PCHE 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.SPILLCON
- 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 = PMD<sub>x</sub>=0 AND Module Enable=1, Disable =PMD<sub>x</sub>=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
Access			SQIMD					
Reset			R/W/L					
Reset			0					
Bit	23	22	21	20	19	18	17	16
Access								RTCCMD
Reset								R/W/L
Reset								0
Bit	15	14	13	12	11	10	9	8
Access								ADCSARMD
Reset								R/W/L
Reset								0
Bit	7	6	5	4	3	2	1	0
Access	ADCMD	ACMD		PLVDMD	LPAMD	MPAMD	BTMD	ZBMD
Reset	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

Value	Description
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																RTCCMD	ADCSARMD	ADCMD	ADCM	PLVDMD			LPAMD	MPAMD	BTMD	ZBMD										
PIC32CX1 012BZ250 48	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	000 0_0 000			
PartName	31															24	23						16	15						8	7						0	Hex
		PMD2																REFO4MD	REFO3MD	REFO2MD	REFO1MD	REFO6MD			REFO5MD													
PIC32CX1 012BZ250 48	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	000 0_0 000			
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		
PIC32CX1 012BZ250 48	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	000 0_0 000			



## 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, or from the wake inputs.

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 30.5 $\mu$ s, 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
- Two 32-bit or four 16-bit compare values
- 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)

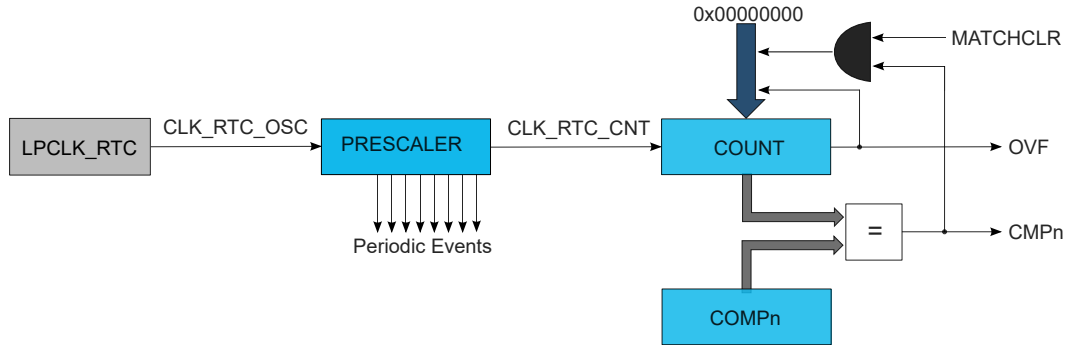


Figure 21-2. RTCC Block Diagram (Mode 1 — 16-Bit Counter)

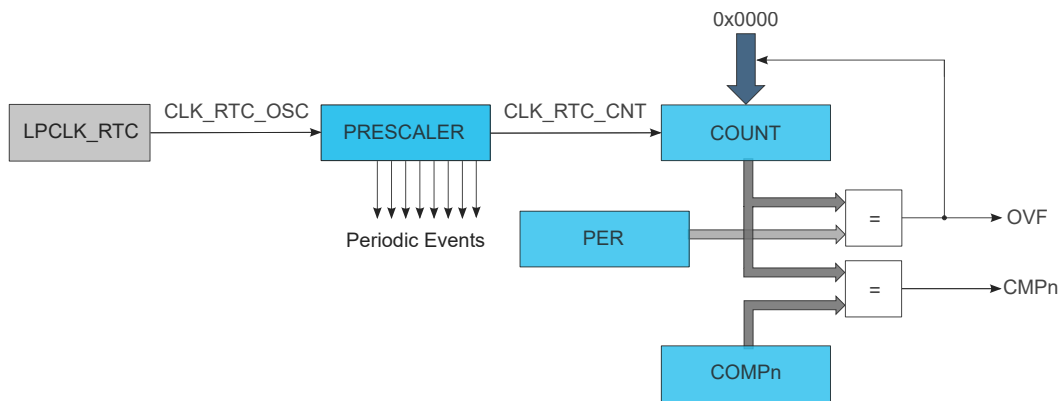


Figure 21-3. RTCC Block Diagram (Mode 2 — Clock/Calendar)

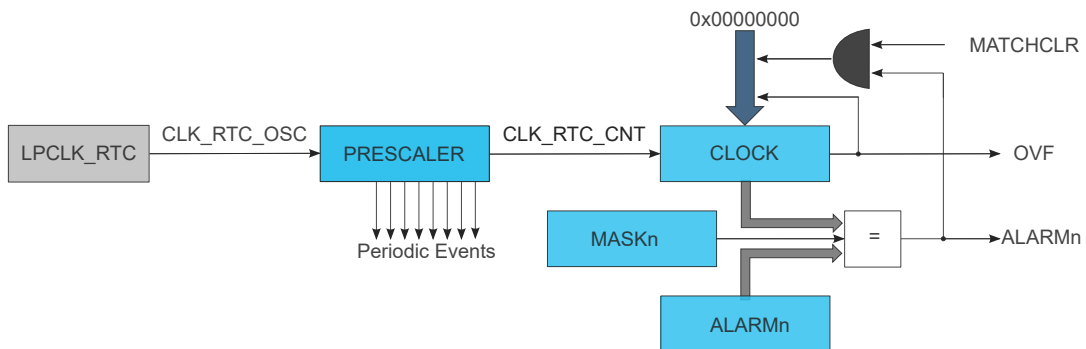
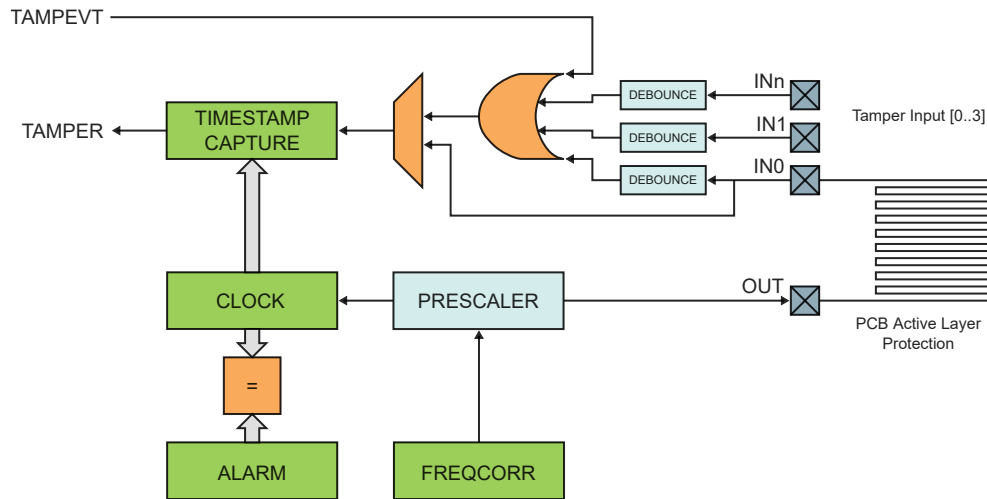


Figure 21-4. RTCC Block Diagram (Tamper Detection)



## 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
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, SOSOC 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

The DMA request lines (or line if only one request) are connected to the DMA Controller (DMAC). Using the RTC DMA requests requires the DMA Controller to be configured first.

#### 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)
- Tamper Control register (TAMPCTRL)

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:

$$f_{\text{CLK\_RTC\_CNT}} = \frac{f_{\text{CLK\_RTC\_OSC}}}{2^{\text{PRESCALER}}}$$

The frequency of the oscillator clock, CLK\_RTC\_OSC, is given by  $f_{\text{CLK\_RTC\_OSC}}$ , and  $f_{\text{CLK\_RTC\_CNT}}$  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 registers (COMPn, n=0–1). When a compare match occurs, the Compare n Interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG.CMPn) 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 COMPn 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.CMPn and INTFLAG.OVF will both be set simultaneously on a compare match with COMPn.

#### 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 registers (ALARMn, n=0-1). When an alarm match occurs, the Alarm n Interrupt flag in the Interrupt Flag Status and Clear registers (INTFLAG.ALARMn, n=0..1) 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 n Mask register (MASKn.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 ALARMn 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 ALARMn.

#### Related Links

[21.3. Block Diagram](#)

[21.6.8.1. Periodic Intervals](#)

### 21.6.3 DMA Operation

The RTC generates the following DMA request:

- Tamper (TAMPER): The request is set on capture of the timestamp. The request is cleared when the Timestamp register is read.

If the CPU accesses the registers which are source for DMA request set/clear condition, the DMA request can be lost or the DMA transfer can be corrupted, if enabled.

### 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.
- Tamper (TAMPER): Indicates detection of valid signal on a tamper input pin or tamper event input.
- 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.

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.
- Tamper (TAMPER): Generated on detection of valid signal on a tamper input pin or tamper event input.
- 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
- Count Read Synchronization bit in Control A register (CTRLA.COUNTSYNC)
- Clock Read Synchronization bit in Control A register (CTRLA.CLOCKSYNC)

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 General Purpose n registers (GPn)

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'
- The Timestamp Value register (TIMESTAMP)

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

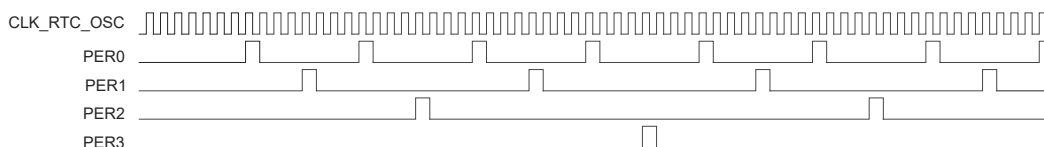
The RTC prescaler can generate interrupts and events at periodic intervals, allowing flexible system tick creation. Any of the upper eight bits of the prescaler (bits 2 to 9) can be the source of an interrupt/event. When one of the eight Periodic Event Output bits in the Event Control register (EVCTRL.PEREO[n=0..7]) 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:

$$f_{\text{PERIODIC}(n)} = \frac{f_{\text{CLK\_RTC\_OSC}}}{2^{n+3}}$$

$f_{\text{CLK\_RTC\_OSC}}$  is the frequency of the internal prescaler clock CLK\_RTC\_OSC, and n is the position of the EVCTRL.PEREO[n] 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:** The same applies for interrupts enabled in INTENSET/CLR.

### 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 8192 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 128 of these periods. The resulting correction is as follows:

$$\text{Correction in ppm} = \frac{\text{FREQCORR.VALUE}}{8192 \cdot 128} \cdot 10^6 \text{ ppm}$$

This results in a resolution of 0.95367ppm.

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 or when tamper detection occurs while CTRLA.GPTRST=1, 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).

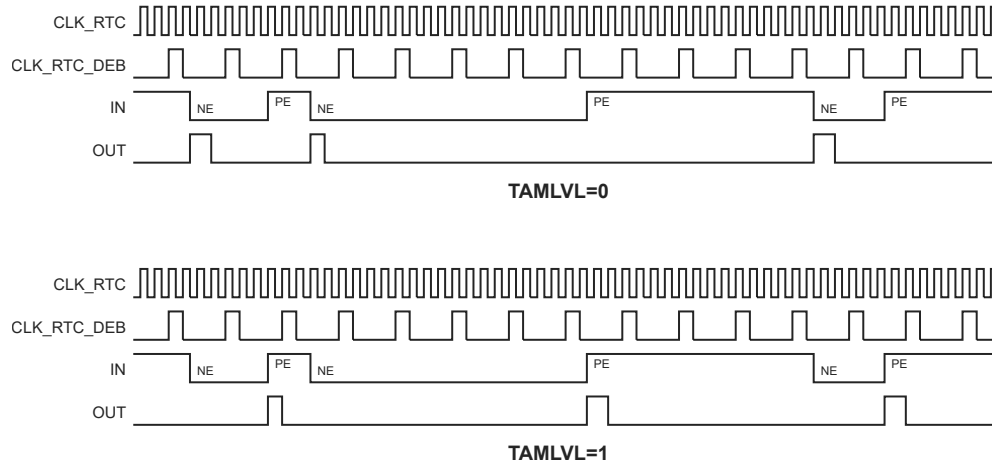
Up to four polarity external inputs (INn) can be used for tamper detection. The polarity for each input is selected with the Tamper Level bits in the Tamper Control register (TAMPCTRL.TAMPLVLn).

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.

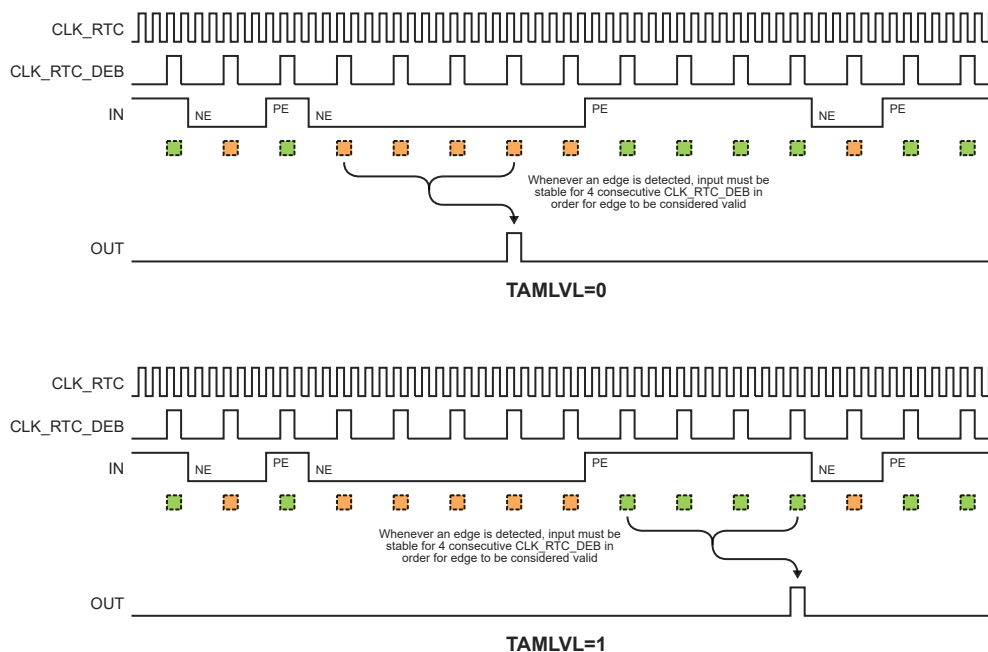
**Figure 21-6.** Edge Detection with Debouncer Disabled



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.

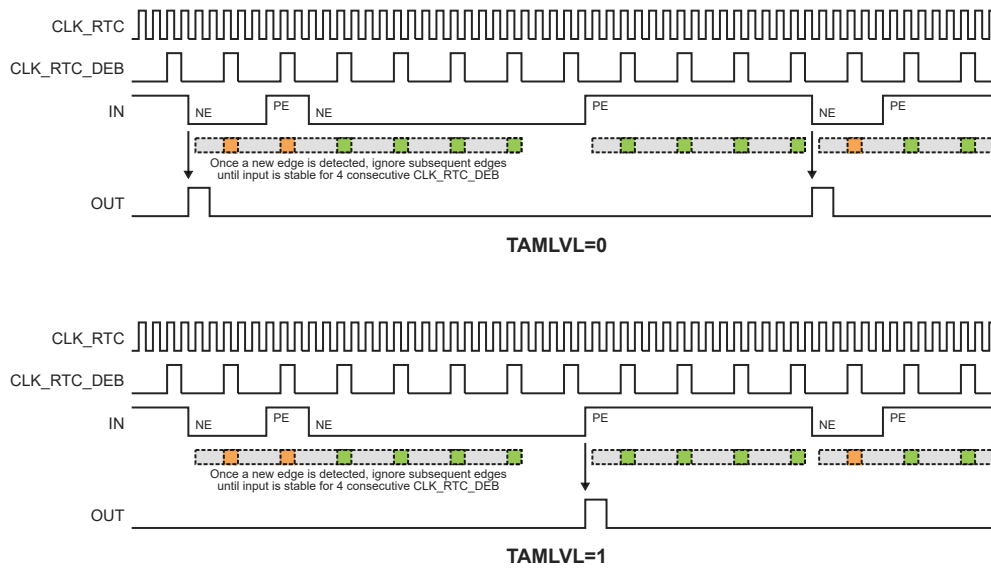
**Figure 21-7.** Edge Detection with Synchronous Stability Debouncing





- Asynchronous (CTRLB.DEBASYN = 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

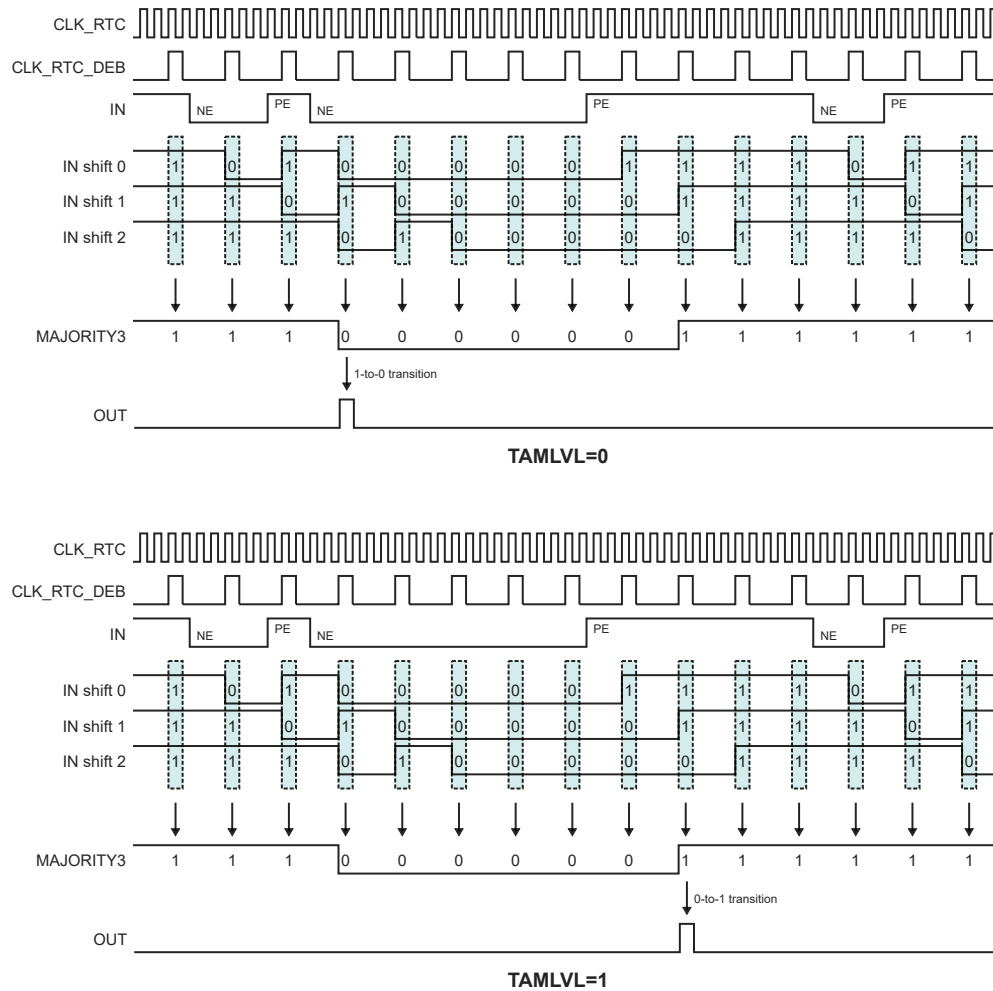


Table 21-3. Debouncer Configuration

TAMPCTRL. DEBNCn	CTRLB. DEBMAJ	CTRLB. DEBASYN	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	GPTRST	BKTRST		PRESCALER[3:0]			
0x02	CTRLB	7:0	DMAEN	RTCOUT	DEBASYNC	DEBMAJ			GP2EN	GP0EN
		15:8			ACTF[2:0]			DEBF[2:0]		
0x04	EVCTRL	7:0	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
		15:8	OVFEO	TAMPERO				CMPEOn[1:0]		
		23:16								TAMPEVEI
0x08	INTENCLR	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF	TAMPER					CMP1	CMP0
0x0A	INTENSET	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF	TAMPER					CMP1	CMP0
0x0C	INTFLAG	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF	TAMPER					CMP1	CMP0
0x0E	DBGCTRL	7:0								DBGRUN
0x0F	Reserved									
0x10	SYNCBUSY	7:0		COMP1	COMP0		COUNT	FREQCORR	ENABLE	SWRST
		15:8	COUNTSYNC							
		23:16					GP3	GP2	GP1	GP0
		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]							
0x4C	GP3	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
0x50 ... 0x5F	Reserved									
0x60	TAMPCTRL	7:0	IN3ACT[1:0]		IN2ACT[1:0]		IN1ACT[1:0]		INOACT[1:0]	
		15:8								
		23:16					TAMLVL3	TAMLVL2	TAMLVL1	TAMLVL0
		31:24					DEBNC3	DEBNC2	DEBNC1	DEBNC0
0x64	TIMESTAMP	7:0	COUNT[7:0]							
		15:8	COUNT[15:8]							
		23:16	COUNT[23:16]							
		31:24	COUNT[31:24]							
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	GPTRST	BKTRST		PRESCALER[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	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 14 – GPTRST GP Registers Reset On Tamper Enable

Only GP registers enabled by the CTRLB.GPnEN bits are affected. This bit can be written only when the peripheral is disabled.

This bit is not synchronized.

### 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

Value	Name	Description
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
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			GP2EN	GP0EN
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	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 1 – GP2EN** General Purpose 2 Enable

Value	Description
0	COMP1 compare function enabled. GP2/GP3 disabled.
1	COMP1 compare function disabled. GP2/GP3 enabled.

**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								TAMPEVEI
Reset								R/W 0
Bit	15	14	13	12	11	10	9	8
Access	OVFEO	TAMPEREO					CMPEOn[1:0]	
Reset	R/W 0	R/W 0					R/W 0	R/W 0
Bit	7	6	5	4	3	2	1	0
Access	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
Reset	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0

#### Bit 16 – TAMPEVEI Tamper Event Input Enable

Value	Description
0	Tamper event input is disabled and incoming events will be ignored.
1	Tamper event input is enabled and incoming events will capture the COUNT value.

#### 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 14 – TAMPEREO Tamper Event Output Enable

Value	Description
0	Tamper event output is disabled and will not be generated.
1	Tamper event output is enabled and will be generated for every tamper input.

#### Bits 9:8 – CMPEOn[1:0] Compare n Event Output Enable [n = 1..0]

Value	Description
0	Compare n event is disabled and will not be generated.
1	Compare n event is enabled and will be generated for every compare match.

#### Bits 0, 1, 2, 3, 4, 5, 6, 7 – PEREO<sub>n</sub> 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	TAMPER					CMP1	CMP0
Access	R/W	R/W					R/W	R/W
Reset	0	0					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 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.

### Bit 14 – TAMPER Tamper Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Tamper Interrupt Enable bit, which disables the Tamper interrupt.

Value	Description
0	The Tamper interrupt is disabled.
1	The Tamper interrupt is enabled.

### Bits 8, 9 – CMPn Compare n Interrupt Enable [n = 1..0]

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Compare n Interrupt Enable bit, which disables the Compare n interrupt.

Value	Description
0	The Compare n interrupt is disabled.
1	The Compare n 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	TAMPER					CMP1	CMP0
Access	R/W	R/W					R/W	R/W
Reset	0	0					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 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.

### Bit 14 – TAMPER Tamper Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Tamper Interrupt Enable bit, which enables the Tamper interrupt.

Value	Description
0	The Tamper interrupt is disabled.
1	The Tamper interrupt is enabled.

### Bits 8, 9 – CMPn Compare n Interrupt Enable [n = 1..0]

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Compare n Interrupt Enable bit, which and enables the Compare n interrupt.

Value	Description
0	The Compare n interrupt is disabled.
1	The Compare n 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					CMP1	CMP0
Access	R/W	R/W					R/W	R/W
Reset	0	0					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 8, 9 – CMPn Compare n [n = 1..0]

This flag is cleared by writing a '1' to the flag.

This flag is set on the next CLK\_RTC\_CNT cycle after a match with the compare condition, and an interrupt request will be generated if INTENCLR/SET.COMPn is one.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Compare n 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					GP3	GP2	GP1	GP0
Reset					R	R	R	R
					0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	COUNTSYNC							
Reset	R							
	0							
Bit	7	6	5	4	3	2	1	0
Access		COMP1	COMP0		COUNT	FREQCORR	ENABLE	SWRST
Reset		R	R		R	R	R	R
		0	0		0	0	0	0

### Bits 16, 17, 18, 19 – GPn General Purpose n Synchronization Busy Status

Value	Description
0	Write synchronization for GPn register is complete.
1	Write synchronization for GPn register is ongoing.

### 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.

### Bits 5, 6 – COMPn Compare n Synchronization Busy Status [n = 1..0]

Value	Description
0	Write synchronization for COMPx register is complete.
1	Write synchronization for COMPx 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.



### 21.8.9 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]		IN1ACT[1:0]		IN0ACT[1:0]	
Access								
Reset	0	0	0	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, 2:3, 4:5, 6:7 – 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 Timestamp

**Name:**       TIMESTAMP  
**Offset:**     0x64  
**Reset:**       0x0  
**Property:**   -

Bit	31	30	29	28	27	26	25	24
	COUNT[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
	COUNT[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
	COUNT[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
	COUNT[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

### Bits 31:0 – COUNT[31:0] Count Timestamp Value

The 32-bit value of COUNT is captured by the TIMESTAMP when a tamper condition occurs

## 21.8.16 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



### 21.8.17 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	GPTRST	BKTRST		PRESCALER[3:0]				
0x02	CTRLB	7:0	DMAEN	RTCOUT	DEBASYNC	DEBMAJ			GP2EN	GP0EN	
		15:8			ACTF[2:0]			DEBF[2:0]			
0x04	EVCTRL	7:0	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0	
		15:8	OVFEO	TAMPERO			CMPEO3	CMPEO2	CMPEO1	CMPEO0	
		23:16								TAMPEVEI	
		31:24									
0x08	INTENCLR	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0	
		15:8	OVF	TAMPER			CMP3	CMP2	CMP1	CMP0	
0x0A	INTENSET	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0	
		15:8	OVF	TAMPER			CMP3	CMP2	CMP1	CMP0	
0x0C	INTFLAG	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0	
		15:8	OVF	TAMPER			CMP3	CMP2	CMP1	CMP0	
0x0E	DBGCTRL	7:0								DBGRUN	
0x0F	Reserved										
0x10	SYNCBUSY	7:0	COMP2	COMP1	COMP0	PER	COUNT	FREQCORR	ENABLE	SWRST	
		15:8	COUNTSYNC							COMP3	
		23:16					GP3	GP2	GP1	GP0	
		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]								
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]		IN1ACT[1:0]		INOACT[1:0]	
		15:8								
		23:16					TAMLVL3	TAMLVL2	TAMLVL1	TAMLVL0
		31:24					DEBNC3	DEBNC2	DEBNC1	DEBNC0
0x64	TIMESTAMP	7:0	COUNT[7:0]							
		15:8	COUNT[15:8]							
		23:16								
		31:24								
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	GPTRST	BKTRST		PRESCALER[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	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 14 – GPTRST GP Registers Reset On Tamper Enable

Only GP registers enabled by the CTRLB.GPnEN bits are affected. This bit can be written only when the peripheral is disabled.

This bit is not synchronized.

Value	Description
0	GPn registers will not reset when a tamper condition occurs.
1	GPn registers will reset when a tamper condition occurs.

#### 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

Value	Name	Description
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
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	DEBASYN	DEBMAJ			GP2EN	GP0EN
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	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 – DEBASYN 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 1 – GP2EN** General Purpose 2 Enable

Value	Description
0	COMP1 compare function enabled. GP2/GP3 disabled.
1	COMP1 compare function disabled. GP2/GP3 enabled.

**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								TAMPEVEI
Reset								R/W 0
Bit	15	14	13	12	11	10	9	8
Access	OVFEO	TAMPEREO			CMPEO3	CMPEO2	CMPEO1	CMPEO0
Reset	R/W 0	R/W 0			R/W 0	R/W 0	R/W 0	R/W 0
Bit	7	6	5	4	3	2	1	0
Access	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
Reset	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0

#### Bit 16 – TAMPEVEI Tamper Event Input Enable

Value	Description
0	Tamper event input is disabled, and incoming events will be ignored
1	Tamper event input is enabled, and incoming events will capture the COUNT value

#### 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 14 – TAMPEREO Tamper Event Output Enable

Value	Description
0	Tamper event output is disabled, and will not be generated.
1	Tamper event output is enabled, and will be generated for every tamper input.

#### Bits 8, 9, 10, 11 – CMPEOn Compare n Event Output Enable [n = 3..0]

Value	Description
0	Compare n event is disabled and will not be generated.
1	Compare n event is enabled and will be generated for every compare match.

#### Bits 0, 1, 2, 3, 4, 5, 6, 7 – PEREO<sub>n</sub> 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	TAMPER			CMP3	CMP2	CMP1	CMP0
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
	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.

### Bit 14 – TAMPER Tamper Interrupt Enable

Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Tamper Interrupt Enable bit, which disables the Tamper interrupt.

Value	Description
0	The Tamper interrupt is disabled.
1	The Tamper interrupt is enabled.

### Bits 8, 9, 10, 11 – CMPn Compare n Interrupt Enable [n = 3..0]

Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Compare n Interrupt Enable bit, which disables the Compare n interrupt.

Value	Description
0	The Compare n interrupt is disabled.
1	The Compare n 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	TAMPER			CMP3	CMP2	CMP1	CMP0
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
	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.

#### Bit 14 – TAMPER Tamper Interrupt Enable

Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Tamper Interrupt Enable bit, which enables the Tamper interrupt.

Value	Description
0	The Tamper interrupt is disabled.
1	The Tamper interrupt is enabled.

#### Bits 8, 9, 10, 11 – CMPn Compare n Interrupt Enable [n = 3..0]

Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Compare n Interrupt Enable bit, which and enables the Compare n interrupt.

Value	Description
0	The Compare n interrupt is disabled.
1	The Compare n 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	TAMPER			CMP3	CMP2	CMP1	CMP0
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
	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

This flag is set after a tamper condition occurs, and an interrupt request will be generated if INTENCLR.TAMPER/ INTENSET.TAMPER is one.

Writing a '0' to this bit has no effect.

Writing a one to this bit clears the Tamper interrupt flag.

### Bits 8, 9, 10, 11 – CMPn Compare n [n = 3..0]

This flag is cleared by writing a '1' to the flag.

This flag is set on the next CLK\_RTC\_CNT cycle after a match with the compare condition, and an interrupt request will be generated if INTENCLR/SET.COMPn is one.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Compare n 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					GP3	GP2	GP1	GP0
Reset					R	R	R	R
					0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	COUNTSYNC							COMP3
Reset	R							R
	0							0
Bit	7	6	5	4	3	2	1	0
Access	COMP2	COMP1	COMP0	PER	COUNT	FREQCORR	ENABLE	SWRST
Reset	R	R	R	R	R	R	R	R
	0	0	0	0	0	0	0	0

#### Bits 16, 17, 18, 19 – GPn General Purpose n Synchronization Busy Status

Value	Description
0	Write synchronization for GPn register is complete.
1	Write synchronization for GPn register is ongoing.

#### 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.

#### Bits 5, 6, 7, 8 – COMPn Compare n Synchronization Busy Status [n = 3..0]

Value	Description
0	Write synchronization for COMPn register is complete.
1	Write synchronization for COMPn register 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.

Value	Description
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]		IN1ACT[1:0]		IN0ACT[1:0]	
Access								
Reset	0	0	0	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.

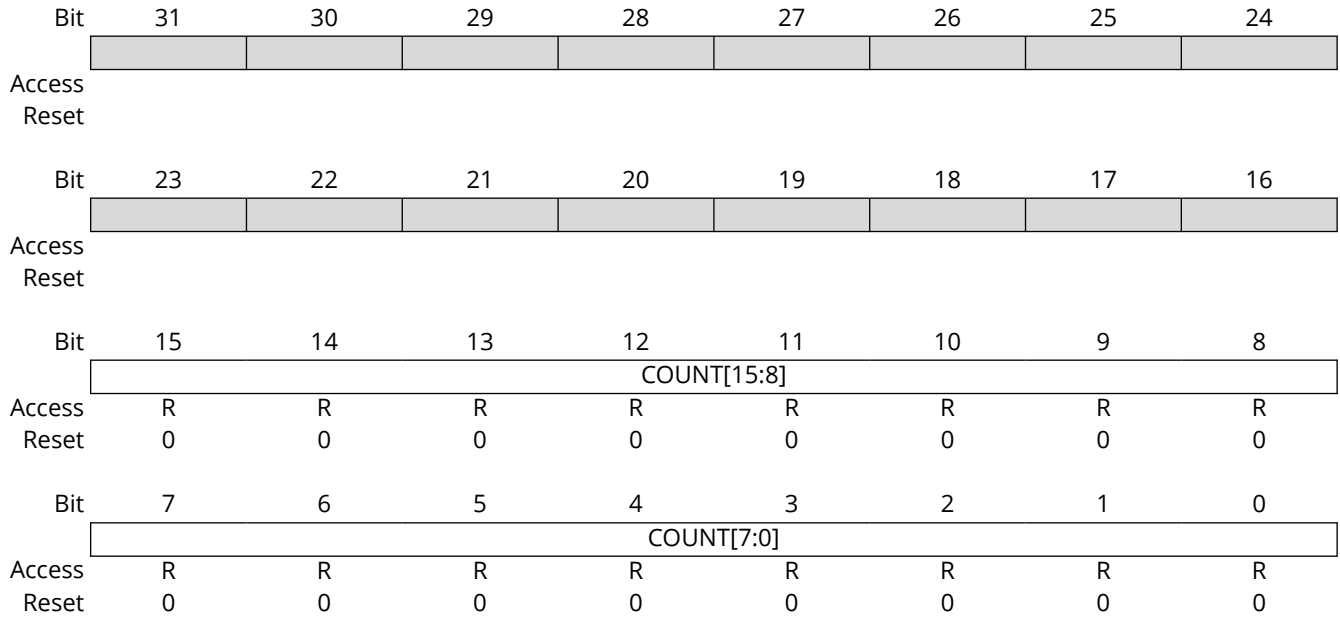
### Bits 0:1, 2:3, 4:5, 6:7 – INnACT Tamper Channel n Action [n=0..3]

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 Timestamp

**Name:**       TIMESTAMP  
**Offset:**     0x64  
**Reset:**      0x0000  
**Property:**   -



**Bits 15:0 – COUNT[15:0]** Count Timestamp Value

The 16-bit value of COUNT is captured by the TIMESTAMP when a tamper condition occurs.

## 21.10.16 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

### 21.10.17 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	GPTRST	BKTRST		PRESCALER[3:0]			
0x02	CTRLB	7:0	DMAEN	RTCOUT	DEBASYN	DEBMAJ			GP2EN	GP0EN
		15:8			ACTF[2:0]			DEBF[2:0]		
0x04	EVCTRL	7:0	PERE07	PERE06	PERE05	PERE04	PERE03	PERE02	PERE01	PERE00
		15:8	OVFEO	TAMPERO					ALARMEO1	ALARMEO0
		23:16								TAMPEVEI
31:24										
0x08	INTENCLR	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF	TAMPER					ALARM1	ALARM0
0x0A	INTENSET	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF	TAMPER					ALARM1	ALARM0
0x0C	INTFLAG	7:0	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0
		15:8	OVF	TAMPER					ALARM1	ALARM0
0x0E	DBGCTRL	7:0								DBGRUN
0x0F	Reserved									
0x10	SYNCBUSY	7:0		ALARM1	ALARM0		CLOCK	FREQCORR	ENABLE	SWRST
		15:8	CLOCKSYNC			MASK1	MASK0			
		23:16					GP3	GP2	GP1	GP0
		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]							
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]	IN1ACT[1:0]	INOACT[1:0]				
		15:8								
		23:16				TAMLVL3	TAMLVL2	TAMLVL1	TAMLVL0	
		31:24				DEBNC3	DEBNC2	DEBNC1	DEBNC0	
0x64	TIMESTAMP	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]	
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	GPTRST	BKTRST		PRESCALER[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	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 14 – GPTRST GP Registers Reset On Tamper Enable

Only GP registers enabled by the CTRLB.GPnEN bits are affected. This bit can be written only when the peripheral is disabled.

This bit is not synchronized.

#### 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

Value	Name	Description
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
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			GP2EN	GPOEN
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	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 1 – GP2EN** General Purpose 2 Enable

Value	Description
0	COMP1 compare function enabled. GP2/GP3 disabled.
1	COMP1 compare function disabled. GP2/GP3 enabled.

**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								TAMPEVEI
Reset								R/W 0
Bit	15	14	13	12	11	10	9	8
Access	OVFEO	TAMPEREO					ALARMEO1	ALARMEO0
Reset	R/W 0	R/W 0					R/W 0	R/W 0
Bit	7	6	5	4	3	2	1	0
Access	PEREO7	PEREO6	PEREO5	PEREO4	PEREO3	PEREO2	PEREO1	PEREO0
Reset	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0

#### Bit 16 – TAMPEVEI Tamper Event Input Enable

Value	Description
0	Tamper event input is disabled, and incoming events will be ignored.
1	Tamper event input is enabled, and all incoming events will capture the CLOCK value.

#### 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 14 – TAMPEREO Tamper Event Output Enable

Value	Description
0	Tamper event output is disabled, and will not be generated
1	Tamper event output is enabled, and will be generated for every tamper input.

#### 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 – PEREO<sub>n</sub> 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	TAMPER					ALARM1	ALARM0
Access	R/W	R/W					R/W	R/W
Reset	0	0					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 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.

### Bit 14 – TAMPER Tamper Interrupt Enable

### Bits 8, 9 – ALARMn Alarm n Interrupt Enable [n = 1..0]

Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Alarm n Interrupt Enable bit, which disables the Alarm n interrupt.

Value	Description
0	The Alarm n interrupt is disabled.
1	The Alarm n 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	TAMPER					ALARM1	ALARM0
Access	R/W	R/W					R/W	R/W
Reset	0	0					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 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.

### Bit 14 – TAMPER Tamper Interrupt Enable

Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Tamper Interrupt Enable bit, which enables the Tamper interrupt.

Value	Description
0	The Tamper interrupt is disabled.
1	The Tamper interrupt is enabled.

### Bits 8, 9 – ALARMn Alarm n Interrupt Enable [n = 1..0]

Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Alarm n Interrupt Enable bit, which enables the Alarm n interrupt.

Value	Description
0	The Alarm n interrupt is disabled.
1	The Alarm n 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	TAMPER					ALARM1	ALARM0
Access	R/W	R/W					R/W	R/W
Reset	0	0					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

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 8, 9 - ALARMn Alarm n [n = 1..0]

This flag is cleared by writing a '1' to the flag.

This flag is set on the next CLK\_RTC\_CNT cycle after a match with the compare condition, and an interrupt request will be generated if INTENCLR/SET.ALARMn is one.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Alarm n 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.

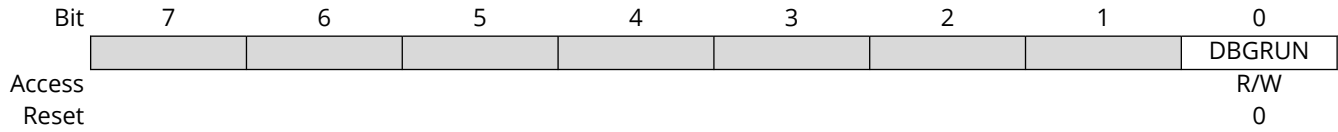
This flag is set on the 0-to-1 transition of prescaler bit [n+2], and an interrupt request will be generated if INTENCLR/SET.PERx is '1'.

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 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					GP3	GP2	GP1	GP0
Reset					R	R	R	R
Bit	15	14	13	12	11	10	9	8
Access	CLOCKSYNC			MASK1	MASK0			
Reset	R			R	R			
Bit	7	6	5	4	3	2	1	0
Access		ALARM1	ALARM0		CLOCK	FREQCORR	ENABLE	SWRST
Reset		R	R		R	R	R	R
Bit	7	6	5	4	3	2	1	0
Reset		0	0		0	0	0	0

### Bits 16, 17, 18, 19 – GPn General Purpose n Synchronization Busy Status

Value	Description
0	Write synchronization for GPn register is complete.
1	Write synchronization for GPn register is ongoing.

### Bit 15 – CLOCKSINC Clock Read Sync Enable Synchronization Busy Status

Value	Description
0	Write synchronization for CTRLA.CLOCKSINC bit is complete.
1	Write synchronization for CTRLA.CLOCKSINC bit is ongoing.

### Bits 11, 12 – MASKn Mask n Synchronization Busy Status [n = 1..0]

Value	Description
0	Write synchronization for MASKx register is complete.
1	Write synchronization for MASKx register is ongoing.

### Bits 5, 6 – ALARMn Alarm n Synchronization Busy Status [n = 1..0]

Value	Description
0	Write synchronization for ALARMx register is complete.
1	Write synchronization for ALARMx register 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.

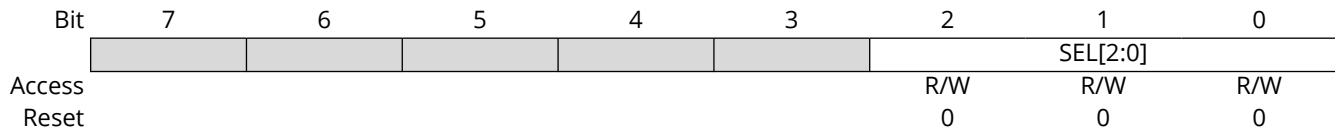
#### Bits 5:0 – SECOND[5:0] Second

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



#### 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]		IN1ACT[1:0]		IN0ACT[1:0]	
Access								
Reset	0	0	0	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, 2:3, 4:5, 6:7 – 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 Timestamp Value

**Name:**       TIMESTAMP  
**Offset:**     0x64  
**Reset:**       0  
**Property:**   -

Bit	31	30	29	28	27	26	25	24
	YEAR[5:0]					MONTH[3:2]		
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
	MONTH[1:0]		DAY[4:0]				HOUR[4]	
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
	HOUR[3:0]			MINUTE[5:2]				
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
	MINUTE[1:0]		SECOND[5:0]					
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

### Bits 31:26 – YEAR[5:0] Year

The year value is captured by the TIMESTAMP when a tamper condition occurs.

### Bits 25:22 – MONTH[3:0] Month

The month value is captured by the TIMESTAMP when a tamper condition occurs.

### Bits 21:17 – DAY[4:0] Day

The day value is captured by the TIMESTAMP when a tamper condition occurs.

### Bits 16:12 – HOUR[4:0] Hour

The hour value is captured by the TIMESTAMP when a tamper condition occurs.

### Bits 11:6 – MINUTE[5:0] Minute

The minute value is captured by the TIMESTAMP when a tamper condition occurs.

### Bits 5:0 – SECOND[5:0] Second

The second value is captured by the TIMESTAMP when a tamper condition occurs.

## 21.12.16 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

### 21.12.17 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

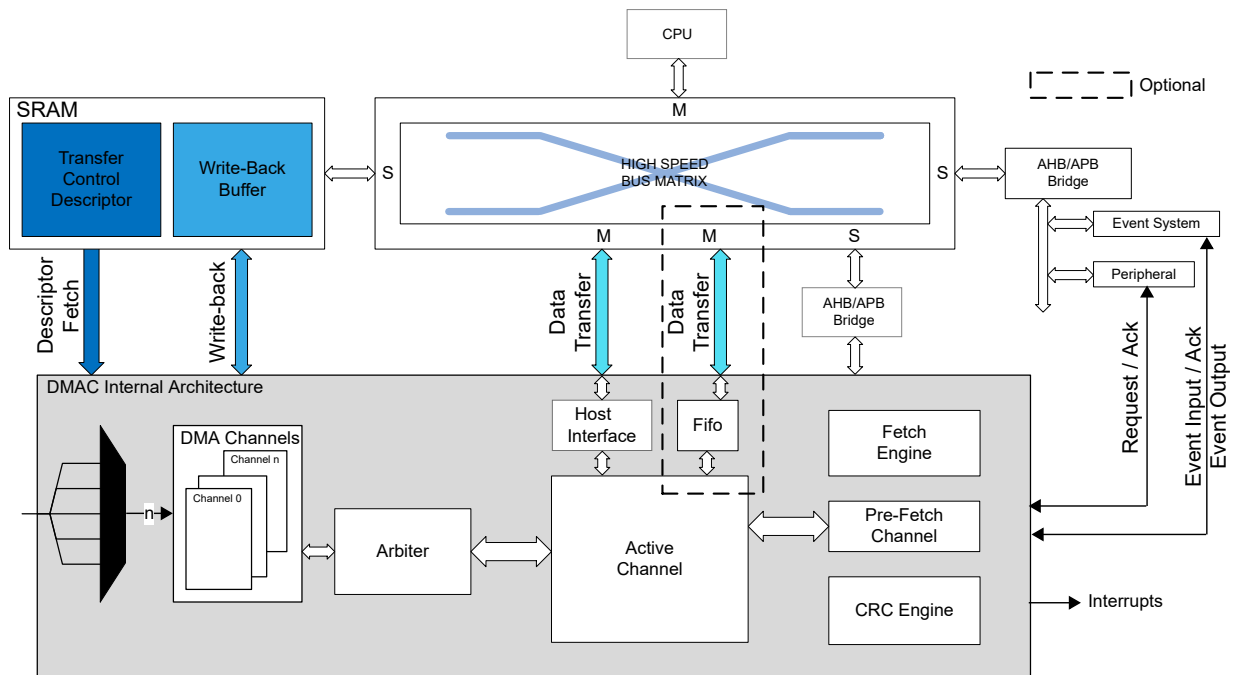
- Data transfer from:
  - Peripheral to peripheral
  - Peripheral to memory
  - Memory to peripheral
  - Memory to memory
- Transfer trigger sources

- 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.

#### 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 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

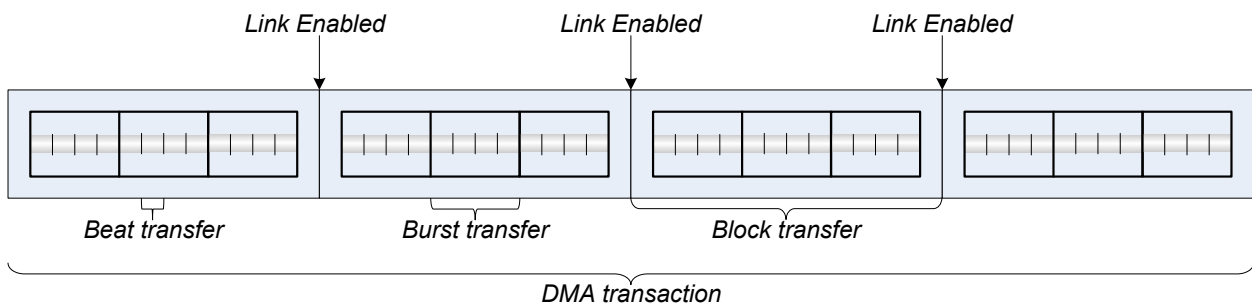
### 22.6.1 Principle of Operation

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.LVLEN $x$ =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 be 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 (CTRL.DMAENABLE) register to '1'. The DMAC is disabled by writing a '0' to the CTRL.DMAENABLE register.

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 value to the CRC Source bits in the Control register (CRCCTRL.CRCSRC). The CRC is disabled by writing a '0' to CRCCTRL.CRCSRC.

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 (CHCTRLA.SWRST), after the corresponding channel is configured. 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

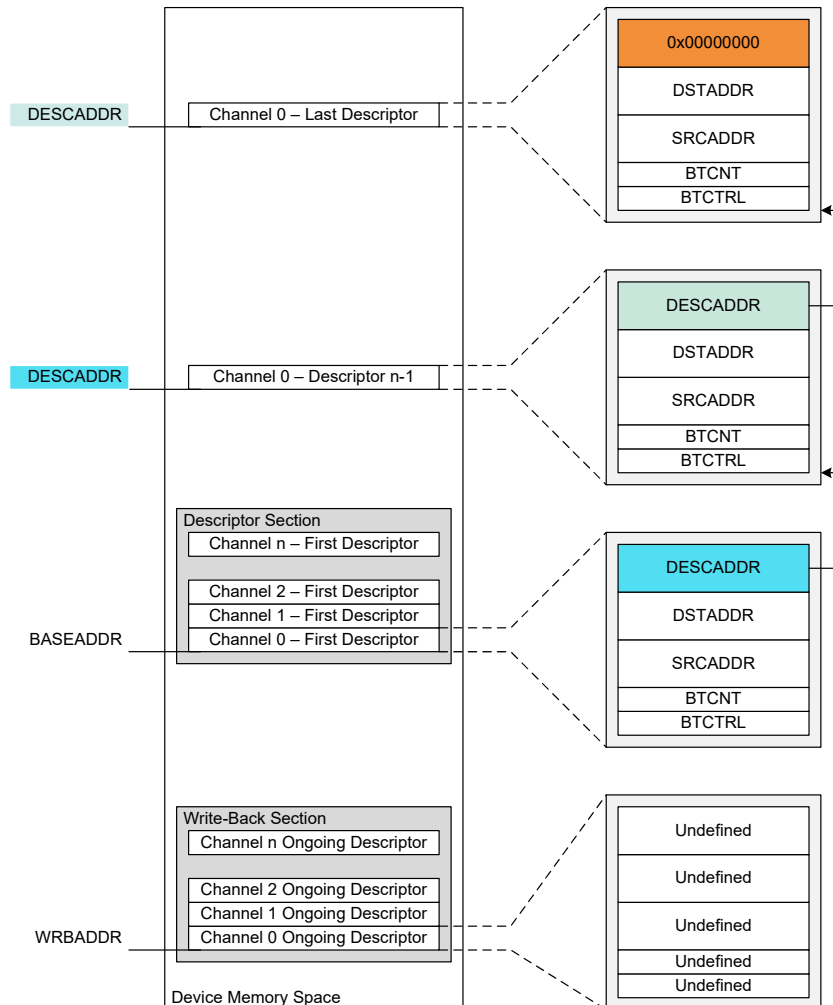
The transfer descriptors, together with the channel configurations, decide how a block transfer must be executed. Before a DMA channel is enabled (CHCTRLA.ENABLE is written to one) and receives a transfer trigger, its first transfer descriptor must be initialized and valid (BTCTRL.VALID). The first transfer 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

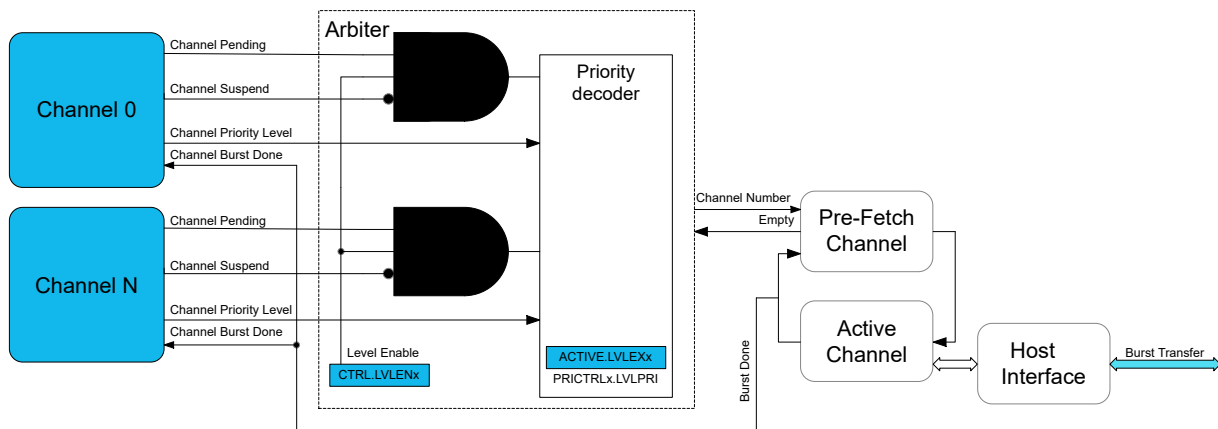
active channel is the DMA channel being granted access to perform its next burst transfer. When the Active Channel has completed a burst transfer, the descriptor stored in the Pre-Fetch Channel is transferred to the Active Channel and a new burst will take place.

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.** Arbitrer 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).

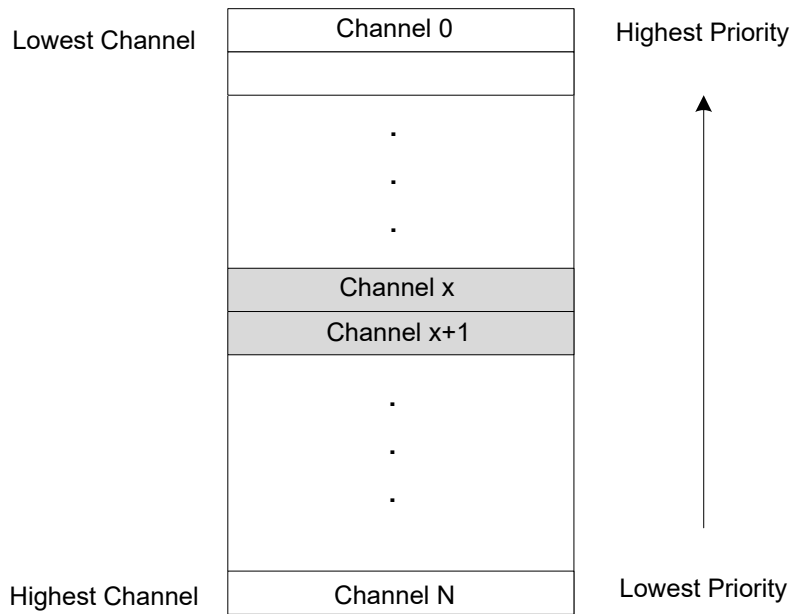
Each DMA channel supports up to 4-level priority scheme.

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

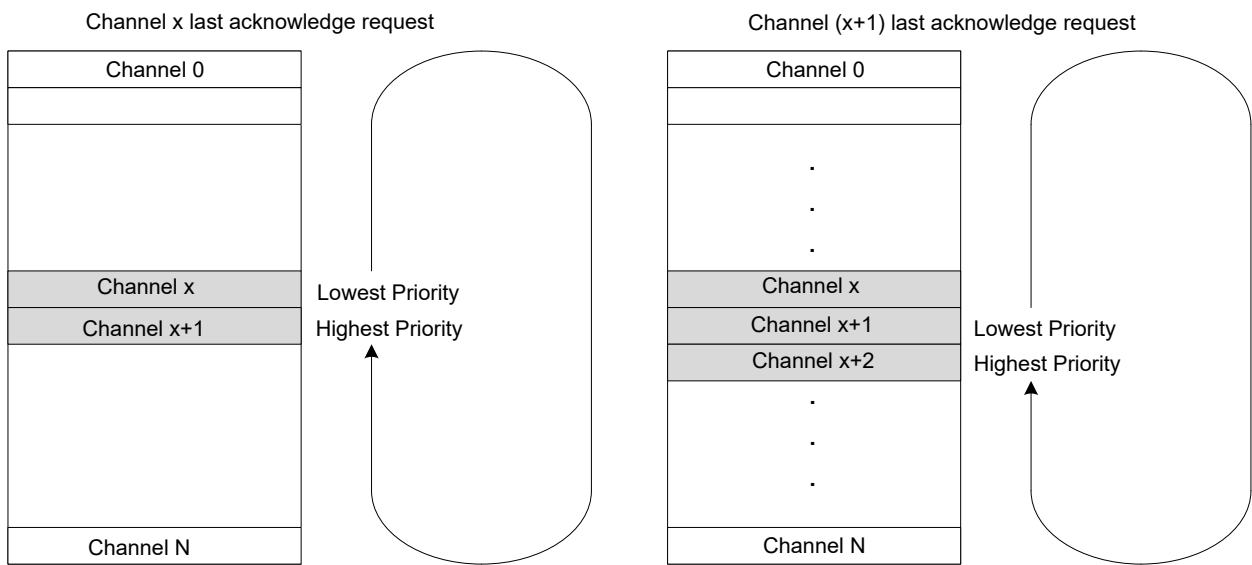
scheme, there is a risk of high channel numbers never being granted access as the active channel. This can be avoided using a dynamic arbitration scheme.

**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](#)

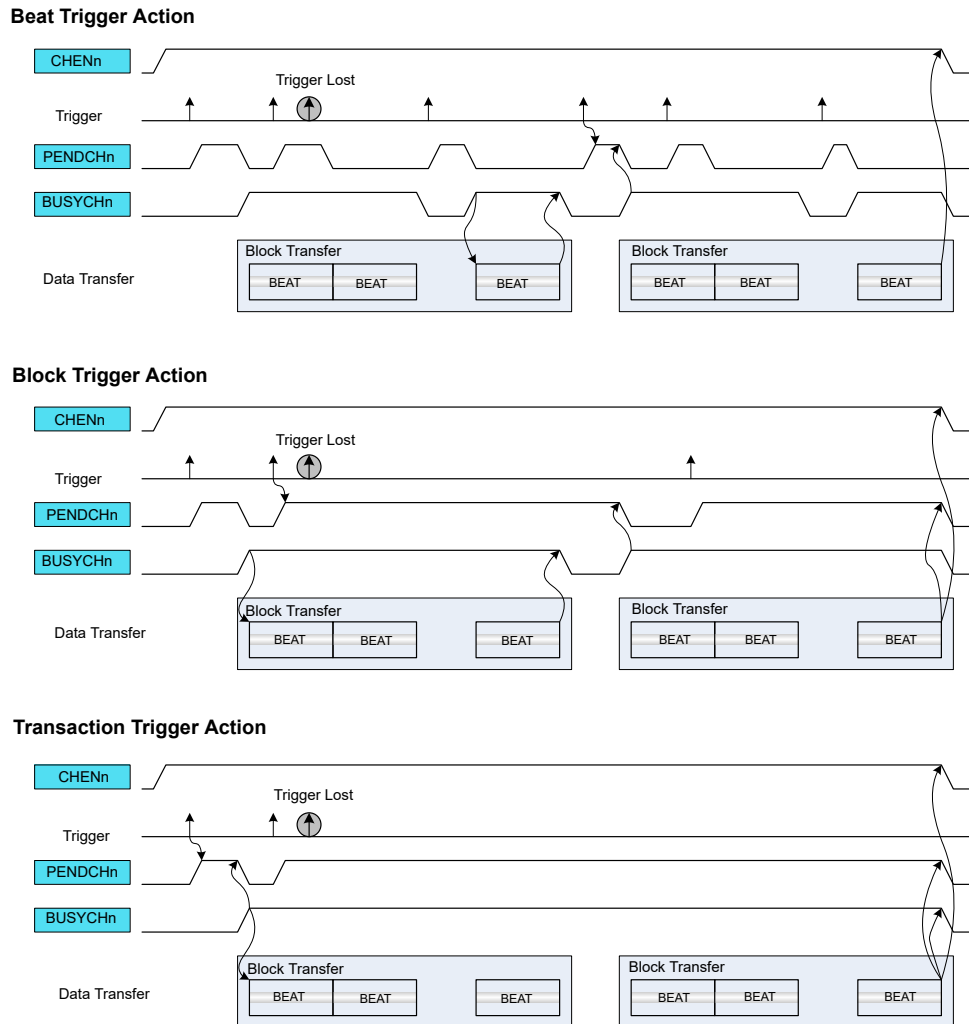
### 22.6.2.6 Transfer Triggers and Actions

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



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 (DSTADDR) 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:

$$\text{SRCADDR} = \text{SRCADDR}_{\text{START}} + \text{BTCNT} \cdot (\text{BEATSIZE} + 1) \cdot 2^{\text{STEPSIZE}}$$

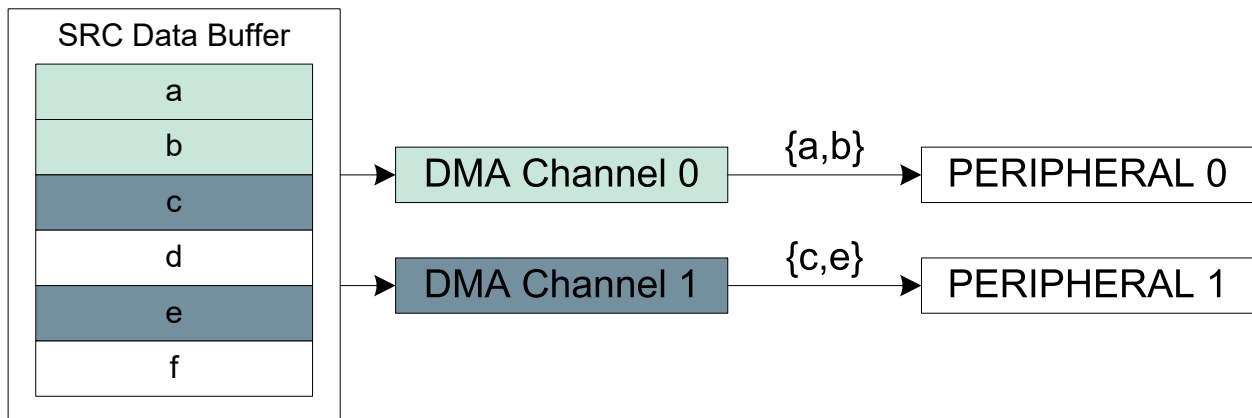
If BTCTRL.STEPSEL=0:

$$\text{SRCADDR} = \text{SRCADDR}_{\text{START}} + \text{BTCNT} \cdot (\text{BEATSIZE} + 1)$$

- SRCADDR<sub>START</sub> 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:

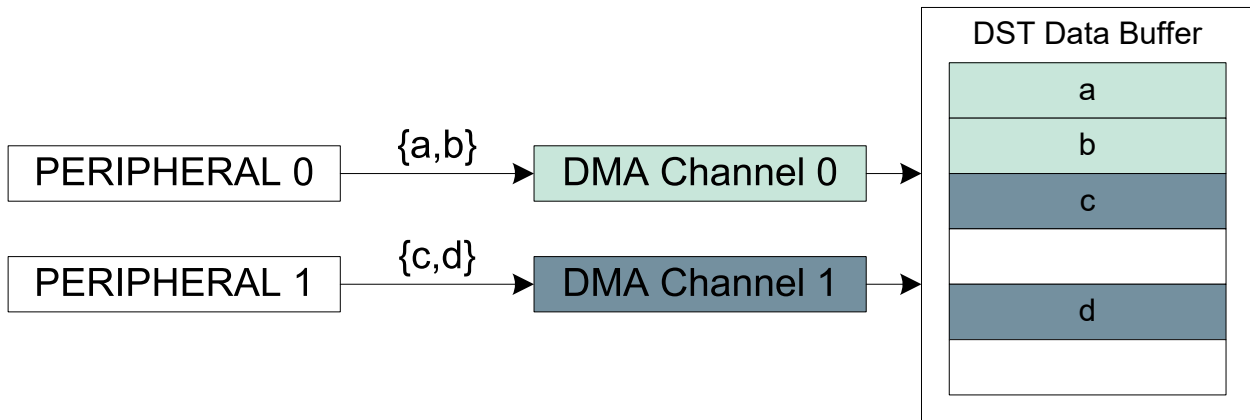
$\text{DSTADDR} = \text{DSTADDR}_{\text{START}} + \text{BTCNT} \cdot (\text{BEATSIZE} + 1) \cdot 2^{\text{STEPSIZE}}$	where BTCTRL.STEPSEL is zero
$\text{DSTADDR} = \text{DSTADDR}_{\text{START}} + \text{BTCNT} \cdot (\text{BEATSIZE} + 1)$	where BTCTRL.STEPSEL is one

- DSTADDR<sub>START</sub> 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 (CHCTRLx.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.

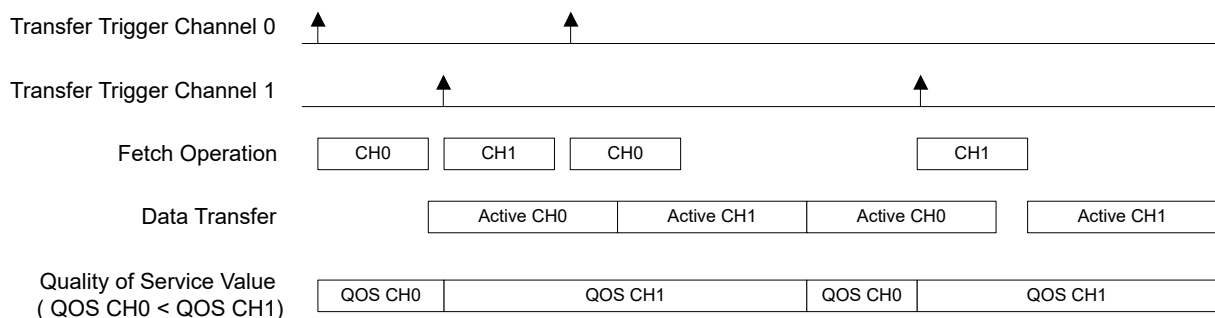
#### Related Links

[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).

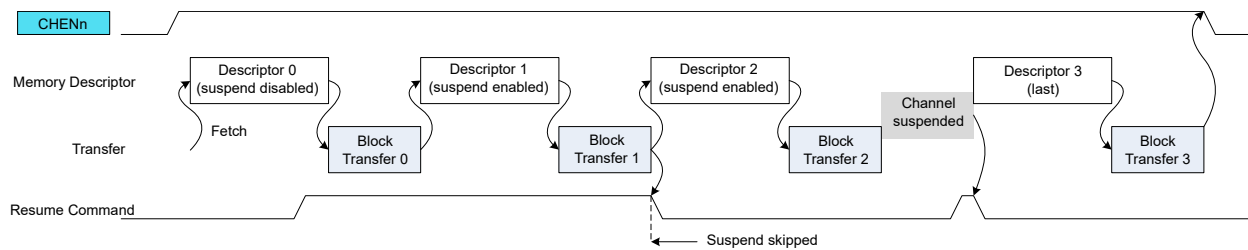
### Related Links

[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

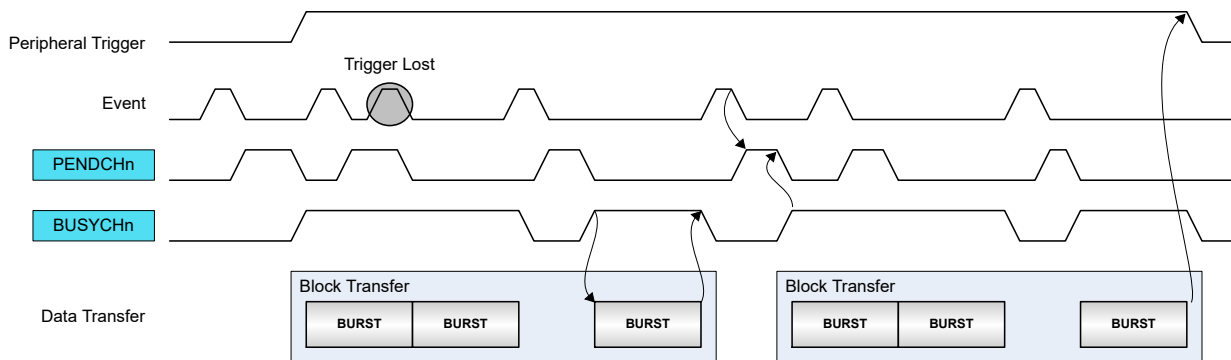
The event input is used to trigger a beat or burst transfer on peripherals.

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



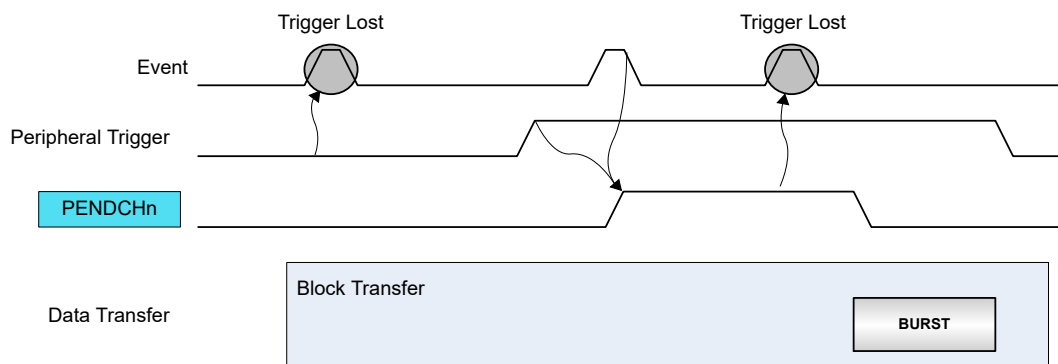
### Conditional Transfer on Strobe

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

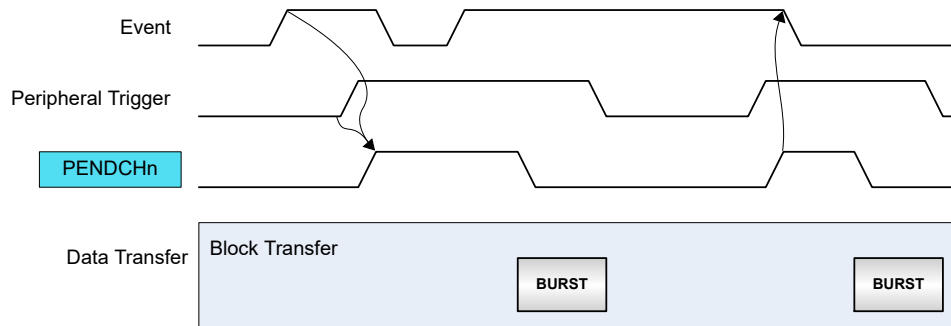
The event input is used to trigger a conditional transfer on peripherals with pending transfer requests. As example, this type of event can be used for peripheral-to-peripheral transfers, where one peripheral is the source of event and the second peripheral is the source of the trigger.

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



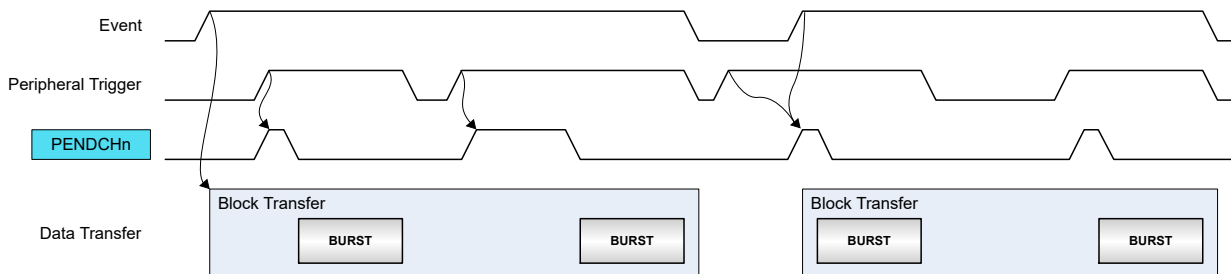
### Conditional Block Transfer

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.

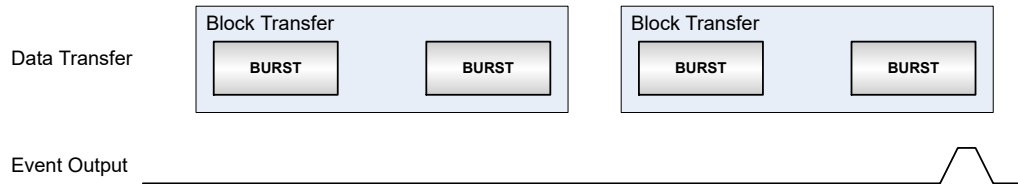
The transfer events (CHEVCTRL.EVOMODE = DEFAULT) are strobe events and their duration is one CLK\_DMACH\_AHB clock period. The transfer event type selection is available in each Descriptor Block Control location (BTCTRL.EVOSEL). Block or burst event output generation is supported.

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.

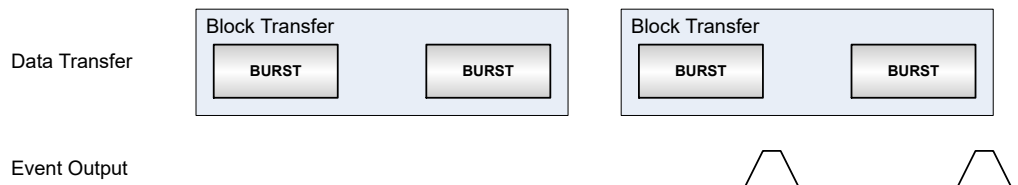
**Figure 22-16. Block Event Output Generation**



**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**

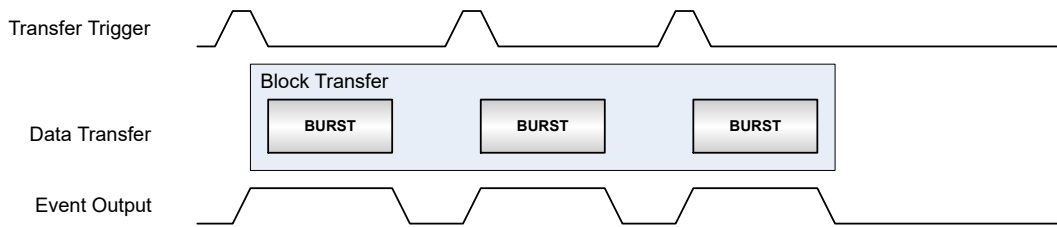


**Trigger Action Event Output**

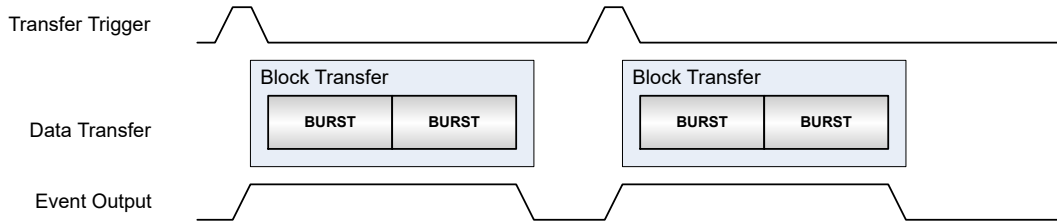
When the trigger action event output is selected, an event level is generated. The event output is set when the transfer trigger occurred, and cleared when the corresponding trigger action is completed. The following figure shows an example for each trigger action type.

**Figure 22-18.** Trigger Action Event Output Generation

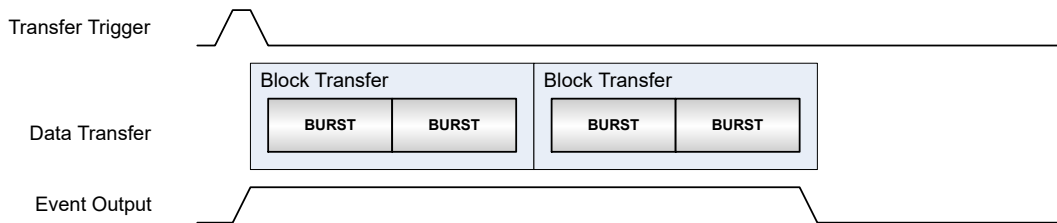
**Burst Trigger Action Event Output**



**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.

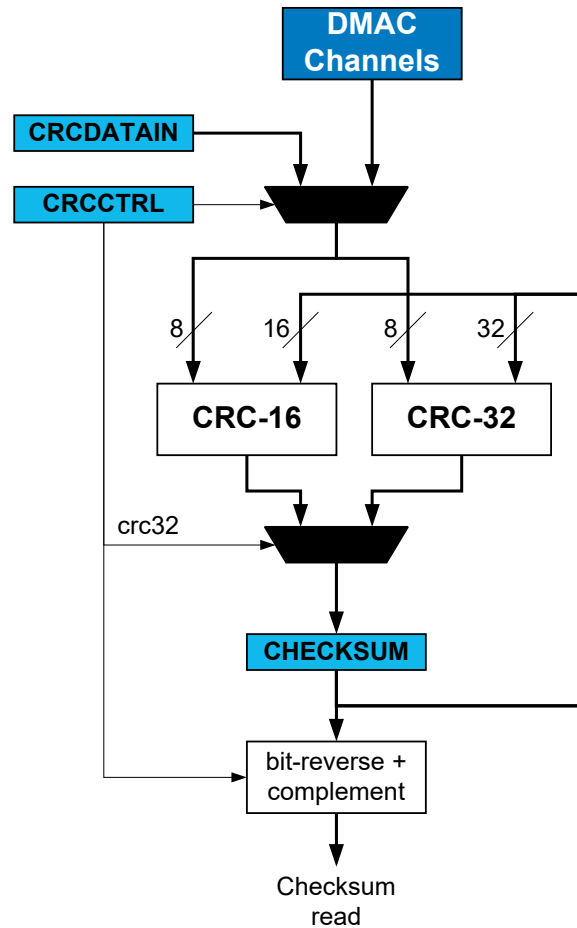
The CRC engine in DMAC supports two commonly used CRC polynomials: CRC-16 (CRC-CCITT) and CRC-32 (IEEE 802.3). Typically, applying CRC-n (CRC-16 or CRC-32) to a data block of arbitrary length will detect any single alteration that is  $\leq n$  bits in length, and will detect the fraction  $1-2^{-n}$  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.

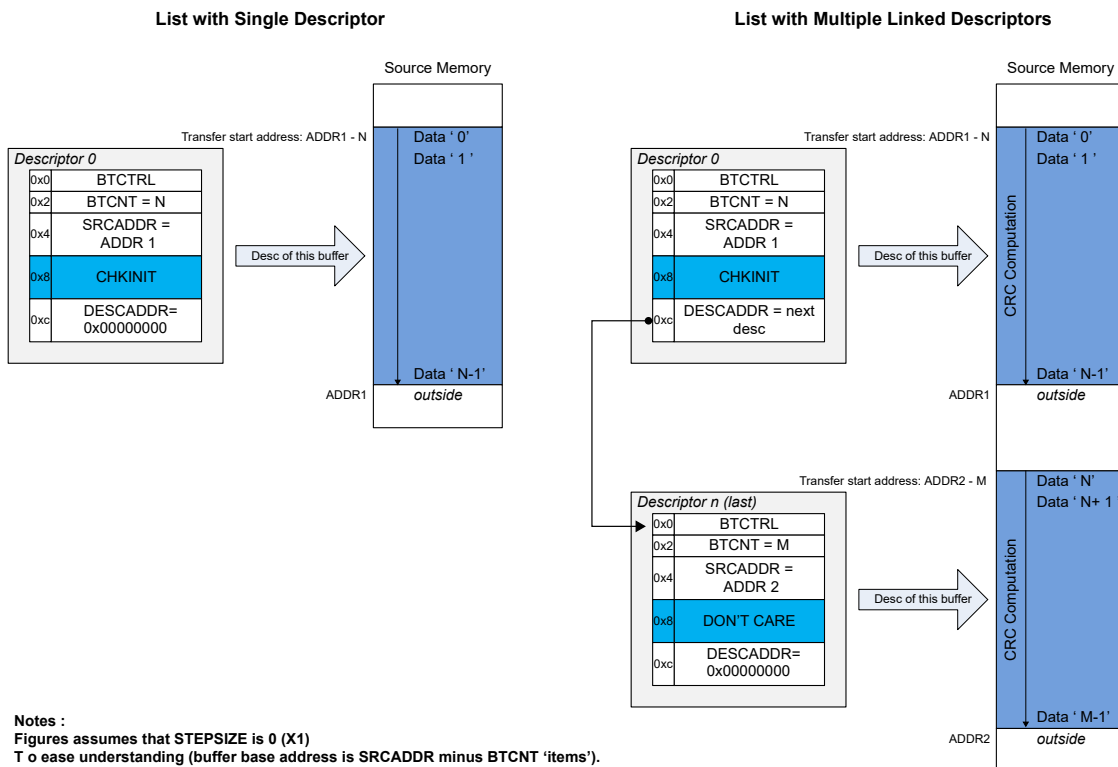
If the linked descriptor is in the list (DESCADDR !=0), the DMA will fetch the next descriptor and CRC calculation continues as described above. When the last list descriptor is executed, the channel is automatically disabled.

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 descriptor 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



### 22.6.3.10 Memory CRC Monitor

When enabled, it is possible to continuously check 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 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.TCMLP) is set, optional interrupt is generated and the DMA will perform the following actions, depending on the descriptor list settings:

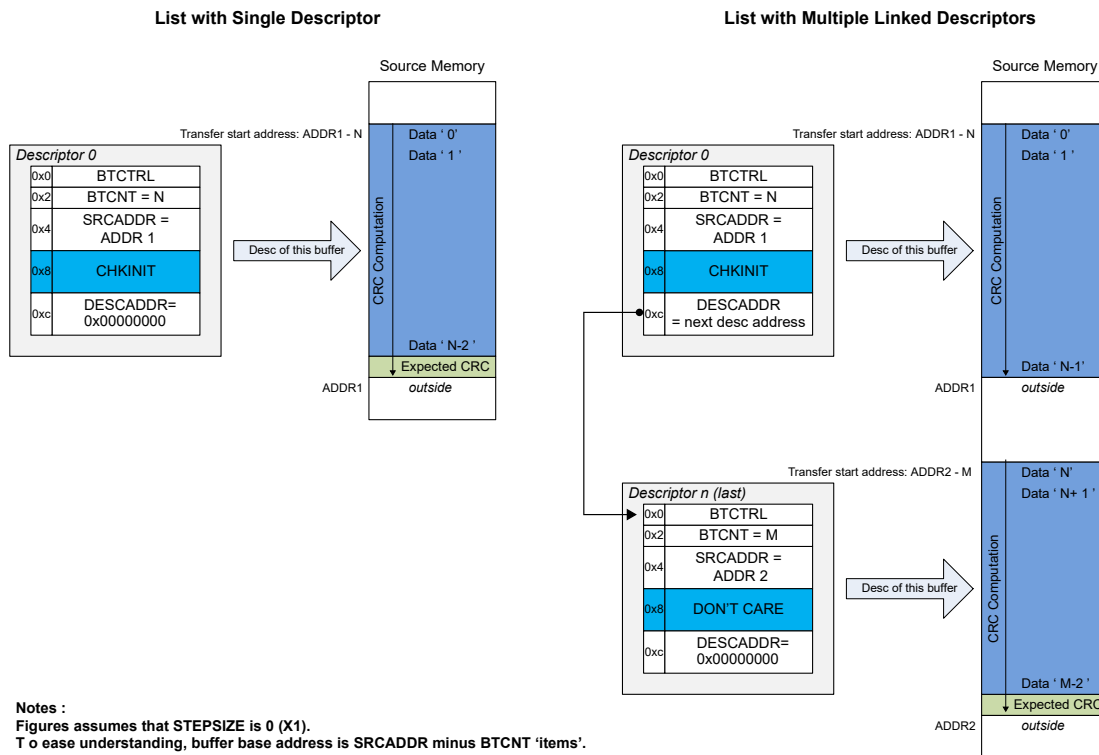
- If the list has only one descriptor, the DMA will re-fetch the descriptor
- If the current descriptor is the last descriptor from the list, the DMA will fetch the first descriptor from the list

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



## 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]			LVLPRIO[4:0]				
		15:8	RRLVLEN1	QOS01[1:0]			LVLPRIO[4:0]				
		23:16	RRLVLEN2	QOS02[1:0]			LVLPRIO[4:0]				
		31:24	RRLVLEN3	QOS03[1:0]			LVLPRIO[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	CHINTn[7:0]								
		15:8	CHINTn[15:8]								
		23:16									
		31:24									
0x28	BUSYCH	7:0	BUSYCHn[7:0]								
		15:8	BUSYCHn[15:8]								
		23:16									
		31:24									
0x2C	PENDCH	7:0	PENDCH7	PENDCH6	PENDCH5	PENDCH4	PENDCH3	PENDCH2	PENDCH1	PENDCH0	
		15:8	PENDCH15	PENDCH14	PENDCH13	PENDCH12	PENDCH11	PENDCH10	PENDCH9	PENDCH8	
		23:16									
		31:24									
0x30	ACTIVE	7:0					LVLEXx3	LVLEXx2	LVLEXx1	LVLEXx0	
		15:8	ABUSY			ID[4:0]					
		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]								
0x38	WRBADDR	7:0	WRBADDR[7:0]								
		15:8	WRBADDR[15:8]								
		23:16	WRBADDR[23:16]								
		31:24	WRBADDR[31:24]								

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
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							PRILVL[1:0]		
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							PRILVL[1:0]		
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							PRILVL[1:0]		
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							PRILVL[1:0]		
0x76	CHEVCTRL3	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]			
0x77 ... 0x7B	Reserved										
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	

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
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							PRILVL[1:0]		
0x86	CHEVCTRL4	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACTION[2:0]			
0x87	...	Reserved									
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							PRILVL[1:0]		
0x96	CHEVCTRL5	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACTION[2:0]			
0x97	...	Reserved									
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							PRILVL[1:0]		
0xA6	CHEVCTRL6	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACTION[2:0]			
0xA7	...	Reserved									
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							PRILVL[1:0]		
0xB6	CHEVCTRL7	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACTION[2:0]			
0xB7	...	Reserved									
0xBB	...	Reserved									
0xBC	CHINTENCLR7	7:0						SUSP	TCMPL	TERR	
0xBD	CHINTENSET7	7:0						SUSP	TCMPL	TERR	
0xBE	CHINTFLAG7	7:0						SUSP	TCMPL	TERR	
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]		

.....continued												
Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0		
0xC5	CHPRILVL8	7:0							PRILVL[1:0]			
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							PRILVL[1:0]			
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							PRILVL[1:0]			
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							PRILVL[1:0]			
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		
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							PRILVL[1:0]			
0x0106	CHEVCTRL12	7:0	EVOE	EVIE	EVOMODE[1:0]			EVACT[2:0]				
0x0107	...											
0x010B	Reserved											

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
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							PRILVL[1:0]	
0x0116	CHEVCTRL13	7:0	EVOE	EVIE	EVOMODE[1:0]				EVACT[2:0]	
0x0117	Reserved									
...										
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							PRILVL[1:0]	
0x0126	CHEVCTRL14	7:0	EVOE	EVIE	EVOMODE[1:0]				EVACT[2:0]	
0x0127	Reserved									
...										
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							PRILVL[1:0]	
0x0136	CHEVCTRL15	7:0	EVOE	EVIE	EVOMODE[1:0]				EVACT[2:0]	
0x0137	Reserved									
...										
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	NOACT	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

Value	Name	Description
0x32	CH18	DMA channel 18
0x33	CH19	DMA channel 19
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	HWWORD	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	CHINTn[15:8]							
Reset	CHINTn[15:8]							
Bit	7	6	5	4	3	2	1	0
Access	CHINTn[7:0]							
Reset	CHINTn[7:0]							

### Bits 15:0 – CHINTn[15:0] Channel n Pending Interrupt [n=15..0]

This bit is set when Channel n has a pending interrupt/the interrupt request is received.  
This bit is cleared when the corresponding Channel n interrupts are disabled or the interrupts sources are cleared.

## 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	BUSYCHn[15:8]							
Reset	BUSYCHn[15:8]							
Bit	7	6	5	4	3	2	1	0
Access	BUSYCHn[7:0]							
Reset	BUSYCHn[7:0]							

### Bits 15:0 – BUSYCHn[15:0] Busy Channel n [n=15..0]

This bit is cleared when the channel trigger action for DMA channel n is complete, when a bus error for DMA channel n is detected, or when DMA channel n is disabled.

This bit is set when DMA channel n starts a DMA transfer.

## 22.8.12 Pending Channels

**Name:** PENDCH  
**Offset:** 0x2C  
**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	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
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, 12, 13, 14, 15 – PENDCHn Pending Channel n [n=0..15]

This bit is cleared when trigger execution defined by channel trigger action settings for DMA channel n is started, when a bus error for DMA channel n is detected or when DMA channel n is disabled. For details on trigger action settings, refer to CHCTRLB.TRIGACT.

This bit is set when a transfer is pending on DMA channel n.

## 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			ID[4:0]					
Access	R			R	R	R	R	R	
Reset	0			0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
					LVLEXx3	LVLEXx2	LVLEXx1	LVLEXx0	
Access					R	R	R	R	
Reset					0	0	0	0	

### 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.

### Bits 12:8 – ID[4:0] Active Channel ID

These bits hold the channel index currently stored in the active channel registers. The value is updated each time the arbiter grants a new channel transfer access request.

### Bits 0, 1, 2, 3 – LVLEXxx Level x Channel Trigger Request Executing [x=3..0]

This bit is set when a level-x channel trigger request is executing or pending.  
This bit is cleared when no request is pending or being executed.

## 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 address 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

Value	Name	Description
0xD	14BEAT	14-beats burst length
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_DMAMC_ID_TIMESTAMP
2	DSU_DMAMC_ID_DCC0
3	DSU_DMAMC_ID_DCC1
4	SERCOM0_DMAMC_ID_RX
5	SERCOM0_DMAMC_ID_TX
6	SERCOM1_DMAMC_ID_RX
7	SERCOM1_DMAMC_ID_TX
8	SERCOM2_DMAMC_ID_RX
9	SERCOM2_DMAMC_ID_TX
10	SERCOM3_DMAMC_ID_RX
11	SERCOM3_DMAMC_ID_TX
12	TCC0_DMAMC_ID_OVF
13	TCC0_DMAMC_ID_MC_0
14	TCC0_DMAMC_ID_MC_1
15	TCC0_DMAMC_ID_MC_2
16	TCC0_DMAMC_ID_MC_3
17	TCC0_DMAMC_ID_MC_4
18	TCC0_DMAMC_ID_MC_5
19	TCC1_DMAMC_ID_OVF
20	TCC1_DMAMC_ID_MC_0
21	TCC1_DMAMC_ID_MC_1
22	TCC1_DMAMC_ID_MC_2
23	TCC1_DMAMC_ID_MC_3
24	TCC1_DMAMC_ID_MC_4
25	TCC1_DMAMC_ID_MC_5
26	TCC2_DMAMC_ID_OVF
27	TCC2_DMAMC_ID_MC_0
28	TCC2_DMAMC_ID_MC_1
29	TC0_DMAMC_ID_OVF
30	TC0_DMAMC_ID_MC_0
31	TC0_DMAMC_ID_MC_1
32	TC1_DMAMC_ID_OVF
33	TC1_DMAMC_ID_MC_0
34	TC1_DMAMC_ID_MC_1
35	TC2_DMAMC_ID_OVF

.....continued	
Number	Name
36	TC2_DMACH_ID_MC_0
37	TC2_DMACH_ID_MC_1
38	TC3_DMACH_ID_OVF
39	TC3_DMACH_ID_MC_0
40	TC3_DMACH_ID_MC_1
41	AES_DMACH_ID_WR
42	AES_DMACH_ID_RD
43	QSPI_DMACH_ID_RX
44	QSPI_DMACH_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

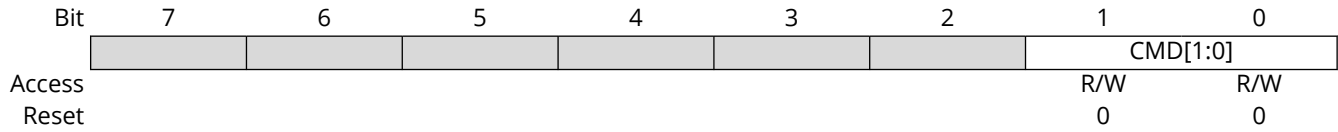
Writing a '0' to this bit has no effect.

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



### 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
							PRILVL[1:0]	
Access							R/W	R/W
Reset							0	0

### Bits 1:0 – PRILVL[1:0] Channel Priority Level

These bits define the priority level used for the DMA channel. The available levels are shown below, where a high level has priority over a low level. These bits are not enable-protected.

Value	Name	Description
0x0	LVL0	Channel Priority Level 0 (Lowest Level)
0x1	LVL1	Channel Priority Level 1
0x2	LVL2	Channel Priority Level 2
0x3	LVL3	Channel Priority Level 3 (Highest Level)

## 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	WORD	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	-	-	-	-	-	-	-	-
Bit	23	22	21	20	19	18	17	16
	SRCADDR[23:16]							
Access	-	-	-	-	-	-	-	-
Reset	-	-	-	-	-	-	-	-
Bit	15	14	13	12	11	10	9	8
	SRCADDR[15:8]							
Access	-	-	-	-	-	-	-	-
Reset	-	-	-	-	-	-	-	-
Bit	7	6	5	4	3	2	1	0
	SRCADDR[7:0]							
Access	-	-	-	-	-	-	-	-
Reset	-	-	-	-	-	-	-	-

#### 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:

$$SRCADDR = SRCADDR_{START} + BTCNT \cdot (BEATSIZE + 1) \cdot 2^{STEPSEL}$$

If BTCTRL.STEPSEL = 0:

$$SRCADDR = SRCADDR_{START} + BTCNT \cdot (BEATSIZE + 1)$$

- SRCADDR<sub>START</sub> 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
- STEPSEL 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	-	-	-	-	-	-	-	-
Bit	23	22	21	20	19	18	17	16
	DSTADDR[23:16]							
Access	-	-	-	-	-	-	-	-
Reset	-	-	-	-	-	-	-	-
Bit	15	14	13	12	11	10	9	8
	DSTADDR[15:8]							
Access	-	-	-	-	-	-	-	-
Reset	-	-	-	-	-	-	-	-
Bit	7	6	5	4	3	2	1	0
	DSTADDR[7:0]							
Access	-	-	-	-	-	-	-	-
Reset	-	-	-	-	-	-	-	-

### 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:

$$DSTADDR = DSTADDR_{START} + BTCNT \cdot (BEATSIZE + 1)$$

If BTCTRL.STEPSEL = 0:

$$DSTADDR = DSTADDR_{START} + BTCNT \cdot (BEATSIZE + 1) \cdot 2^{STEPsize}$$

- DSTADDR<sub>START</sub> 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.

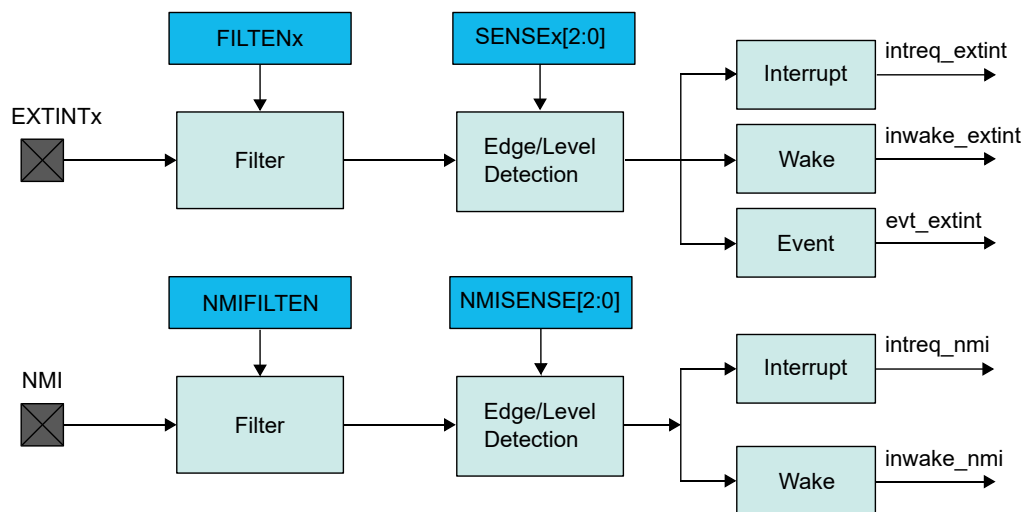
A separate Non-Maskable Interrupt (NMI) is supported. It has properties similar to the other external interrupts, but is connected to the NMI request of the CPU, enabling it to interrupt any other Interrupt mode.

### 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
- Synchronous or Asynchronous Edge Detection mode
- Interrupt pin Debouncing
- 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
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 synchronous 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. Optionally, enable the asynchronous mode.
5. Optionally, enable the debouncer mode.
6. 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.

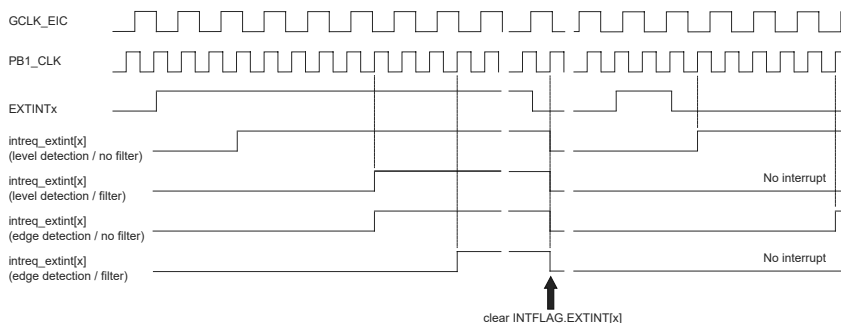
**Table 23-1.** Majority Vote Filter

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 and asynchronous edge detection does not require GCLK\_EIC or CLK\_ULP32K, but interrupt and events can still be generated.

If filtering or synchronous edge detection or debouncing 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 (No Debouncing)

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.

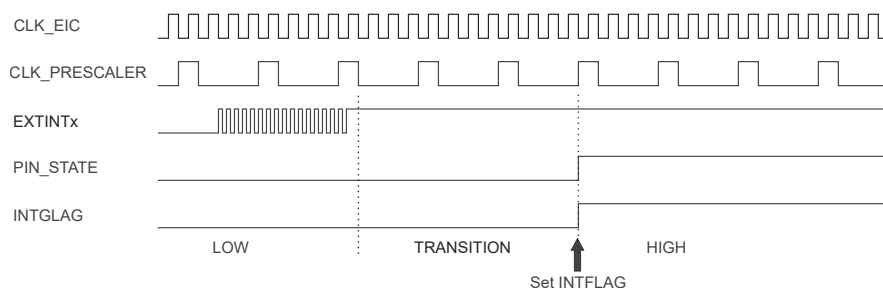
The debouncer manages an internal “valid pin state” that depends on the external interrupt (EXTINT) pin transitions, the debouncing mode and the debouncer prescaler frequency. The valid pin state reflects the pin value after debouncing. The external interrupt pin (EXTINT) is sampled continuously on EIC 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 continuously 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)

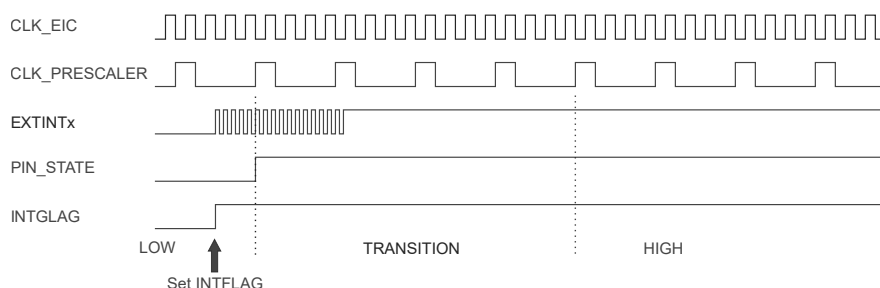


In the synchronous edge detection mode, the EIC clock is required. The synchronous edge detection mode can be used in Idle and Standby sleep modes.

**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.

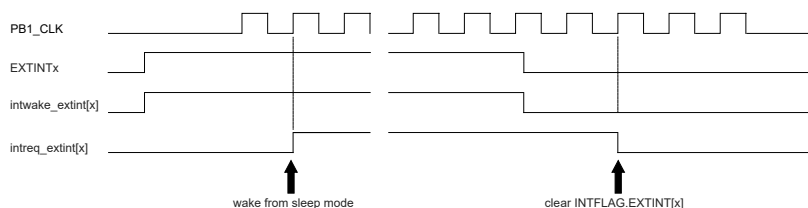
### Related Links

[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					EXTINTEO[3:0]			
		15:8								
		23:16								
		31:24								
0x0C	INTENCLR	7:0					EXTINT[3:0]			
		15:8								
		23:16								
		31:24								
0x10	INTENSET	7:0					EXTINT[3:0]			
		15:8								
		23:16								
		31:24								
0x14	INTFLAG	7:0					EXTINT[3:0]			
		15:8								
		23:16								
		31:24								
0x18	ASYNCH	7:0					ASYNCH[3:0]			
		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					DEBOUNCEN[3:0]			
		15:8								
		23:16								
		31:24								
0x34	DPRESCALER	7:0	STATES1		PRESCALER1[2:0]		STATES0		PRESCALER0[2:0]	
		15:8								
		23:16								TICKON
		31:24								
0x38	PINSTATE	7:0					PINSTATE[3:0]			
		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

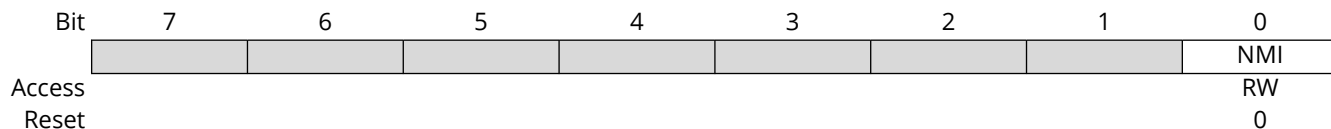
These bits define on which edge or level the NMI triggers.

**Note:** NMI cannot be triggered based on level but it is always based on edge.

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 0 - NMI Non-Maskable Interrupt

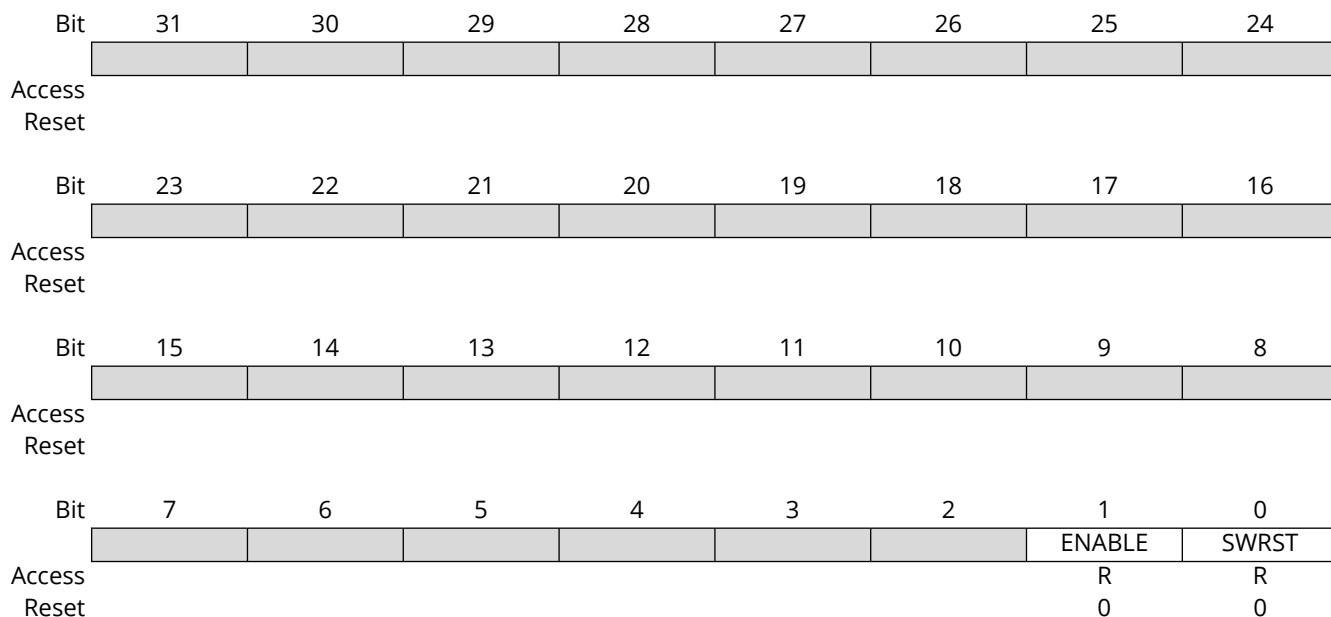
This flag is cleared by writing a '1' to it.

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 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.

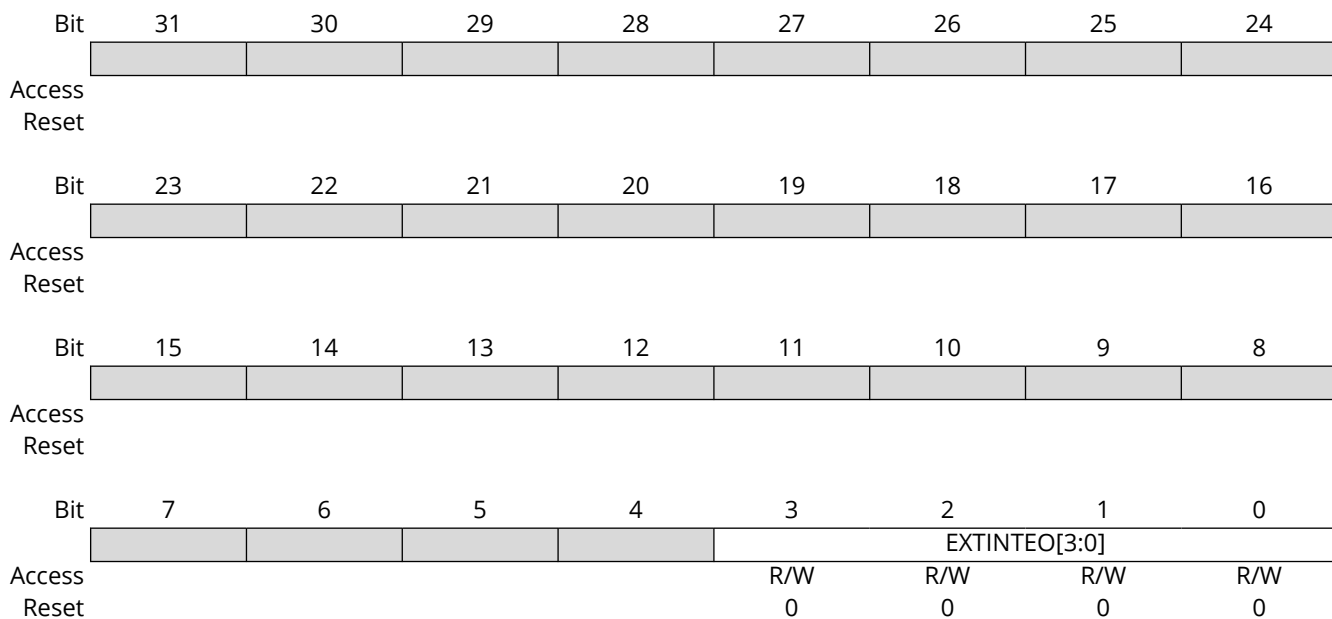
### Bit 0 - SWRST Software Reset Synchronization Busy Status

**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



#### Bits 3:0 – EXTINTEO[3:0] External Interrupt Event Output Enable

The bit x of EXTINTEO enables the event associated with the EXTINTx pin.

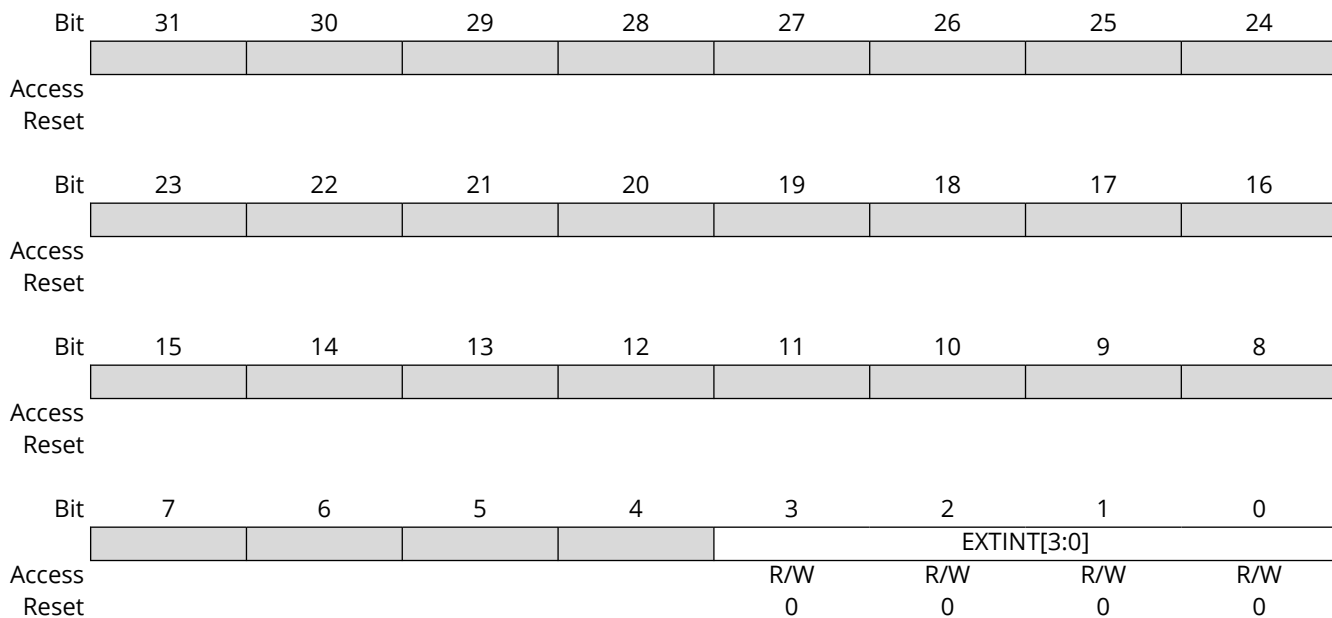
Value	Description
0	Event from pin EXTINTx is disabled.
1	Event from pin EXTINTx is enabled and will be generated when EXTINTx pin matches the external interrupt sensing configuration.



### 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).



#### Bits 3:0 – EXTINT[3:0] External Interrupt Enable

The bit x of EXTINT disables the interrupt associated with the EXTINTx pin.

Writing a '0' to bit x has no effect.

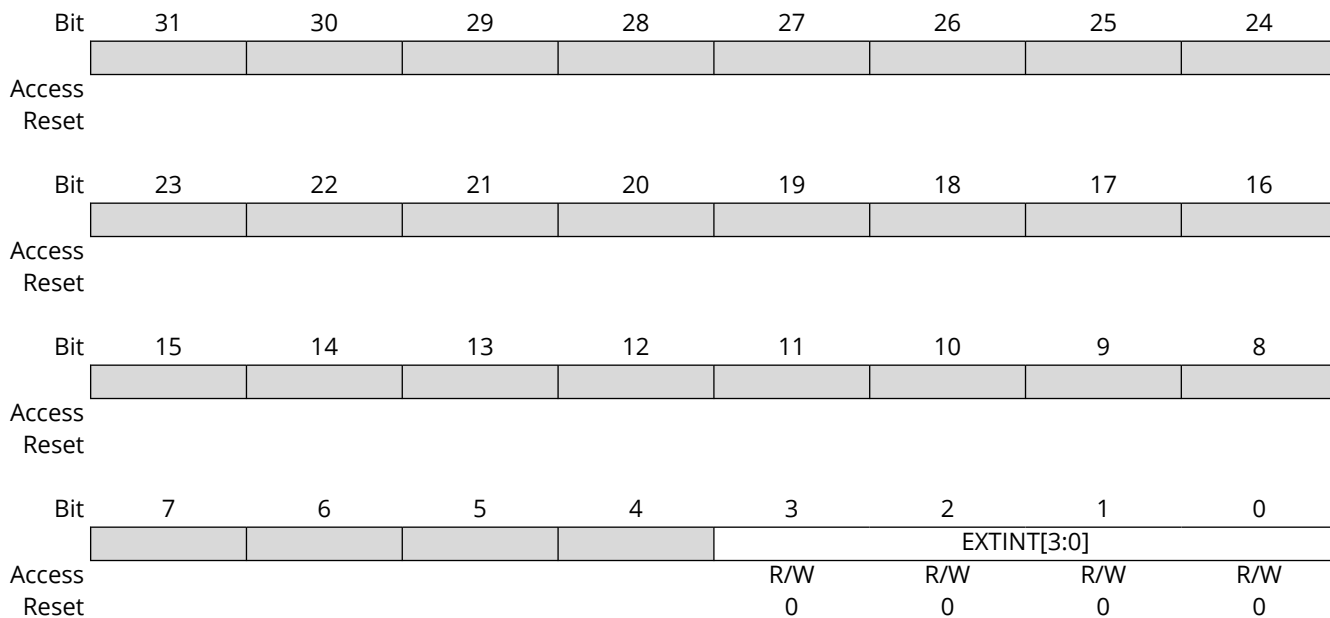
Writing a '1' to bit x will clear the External Interrupt Enable bit x, which disables the external interrupt EXTINTx.

Value	Description
0	The external interrupt x is disabled.
1	The external interrupt x is enabled.

### 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.



#### Bits 3:0 – EXTINT[3:0] External Interrupt Enable

The bit x of EXTINT enables the interrupt associated with the EXTINTx pin.

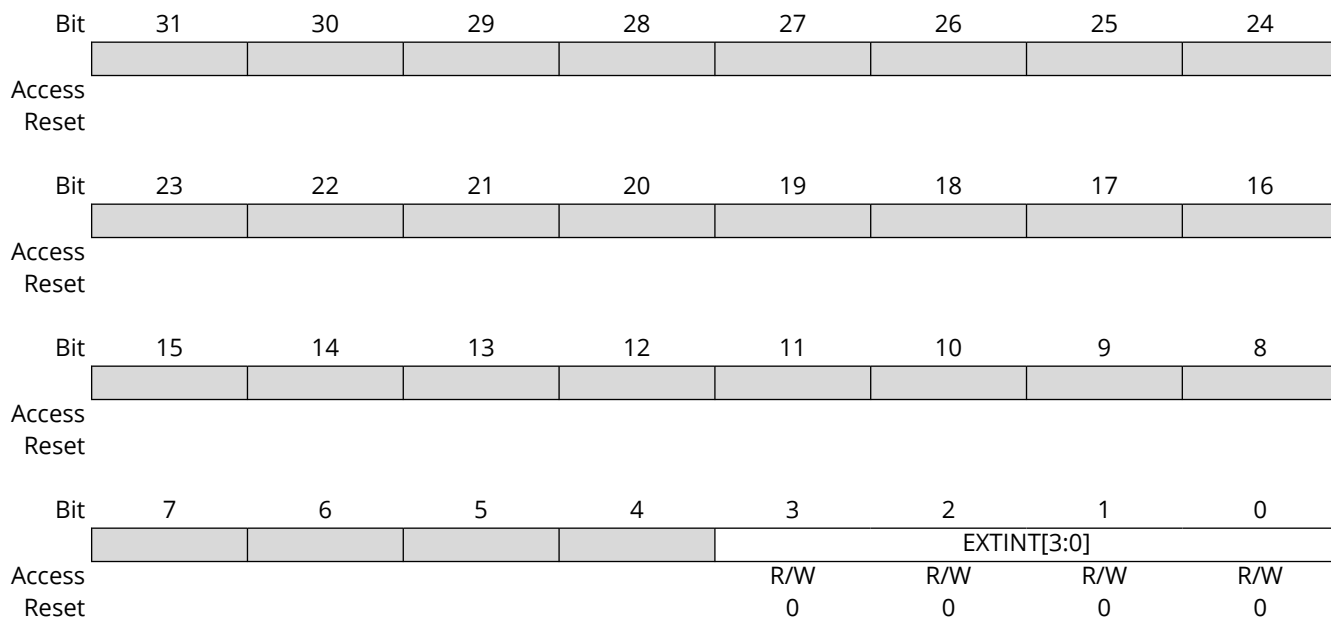
Writing a '0' to bit x has no effect.

Writing a '1' to bit x will set the External Interrupt Enable bit x, which enables the external interrupt EXTINTx.

Value	Description
0	The external interrupt x is disabled.
1	The external interrupt x is enabled.

### 23.8.8 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x14  
**Reset:** 0x00000000  
**Property:** -

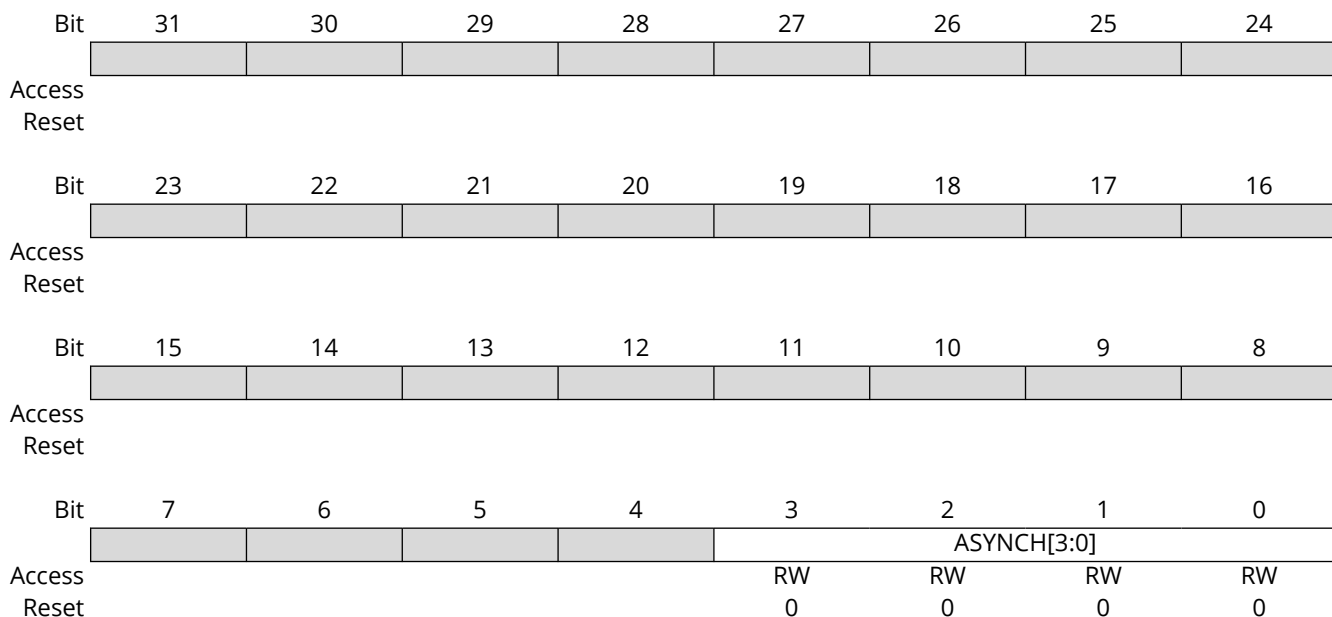


#### Bits 3:0 – EXTINT[3:0] External Interrupt

The flag bit x is cleared by writing a '1' to it.  
 This flag is set when EXTINTx pin matches the external interrupt sense configuration and will generate an interrupt request if INTENCLR/SET.EXTINT[x] is '1'.  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit clears the External Interrupt x flag.

### 23.8.9 External Interrupt Asynchronous Mode

**Name:** ASYNCH  
**Offset:** 0x18  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected



#### Bits 3:0 – ASYNCH[3:0] Asynchronous Edge Detection Mode

The bit x of ASYNCH set the Asynchronous Edge Detection Mode for the interrupt associated with the EXTINTx pin.

Value	Description
0	The EXTINT x edge detection is synchronously operated.
1	The EXTINT x edge detection is asynchronously operated.

### 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	
Access	FILTEN3		SENSE3[2:0]			FILTEN2		SENSE2[2:0]	
Reset	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	
Access	FILTEN1		SENSE1[2:0]			FILTEN0		SENSE0[2:0]	
Reset	RW	RW	RW	RW	RW	RW	RW	RW	
Reset	0	0	0	0	0	0	0	0	

#### Bits 3, 7, 11, 15 – FILTEN<sub>x</sub> 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.

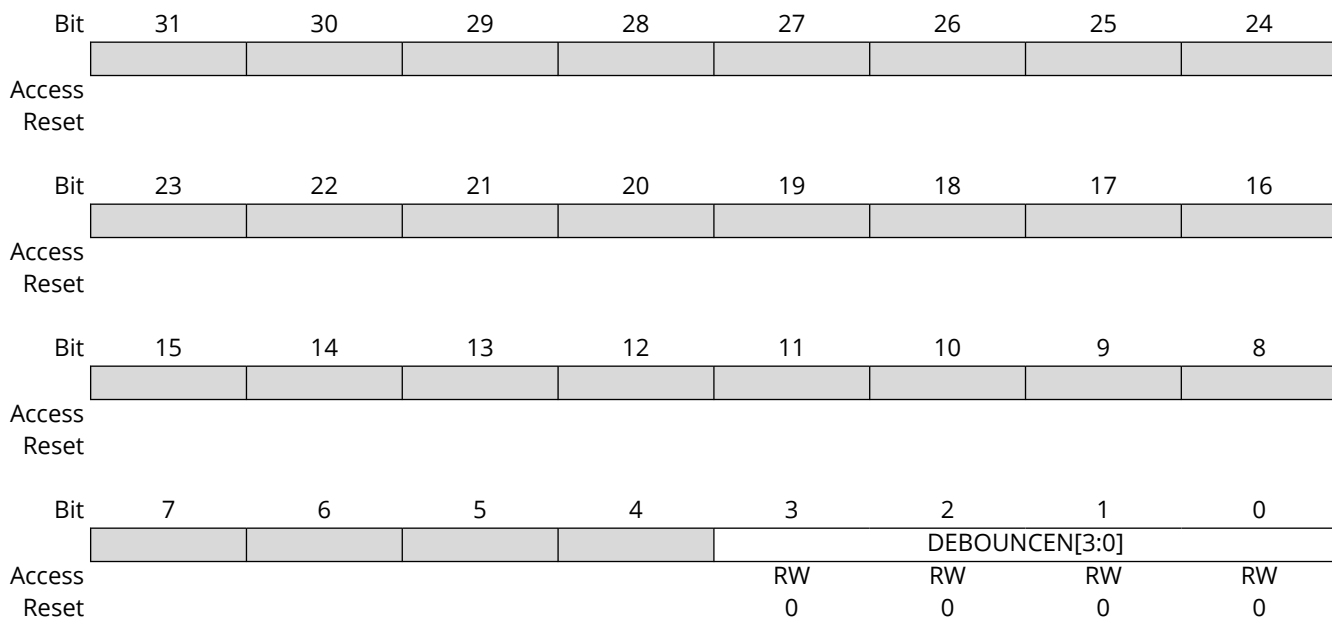
#### Bits 0:2, 4:6, 8:10, 12:14 – SENSE<sub>x</sub> Input Sense Configuration x [x=3..0]

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



#### Bits 3:0 – DEBOUNCEN[3:0] Debouncer Enable

The bit x of DEBOUNCEN set the Debounce mode for the interrupt associated with the EXTINTx pin.

Value	Description
0	The EXTINT x edge input is not debounced.
1	The EXTINT x edge input is debounced.

## 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
Access								TICKON
Reset								RW 0
Bit	15	14	13	12	11	10	9	8
Access								
Reset								
Bit	7	6	5	4	3	2	1	0
Access	STATES1	PRESCALER1[2:0]			STATES0	PRESCALER0[2:0]		
Reset	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0	RW 0

### 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.

### Bits 3, 7 – STATESx Debouncer number of states x

This bit selects the number of samples by the debouncer low frequency clock needed to validate a transition from current pin state to next pin state in synchronous debouncing mode for pins EXTINT[7+(8x):8x].

Value	Description
0	The number of low frequency samples is 3.
1	The number of low frequency samples is 7.

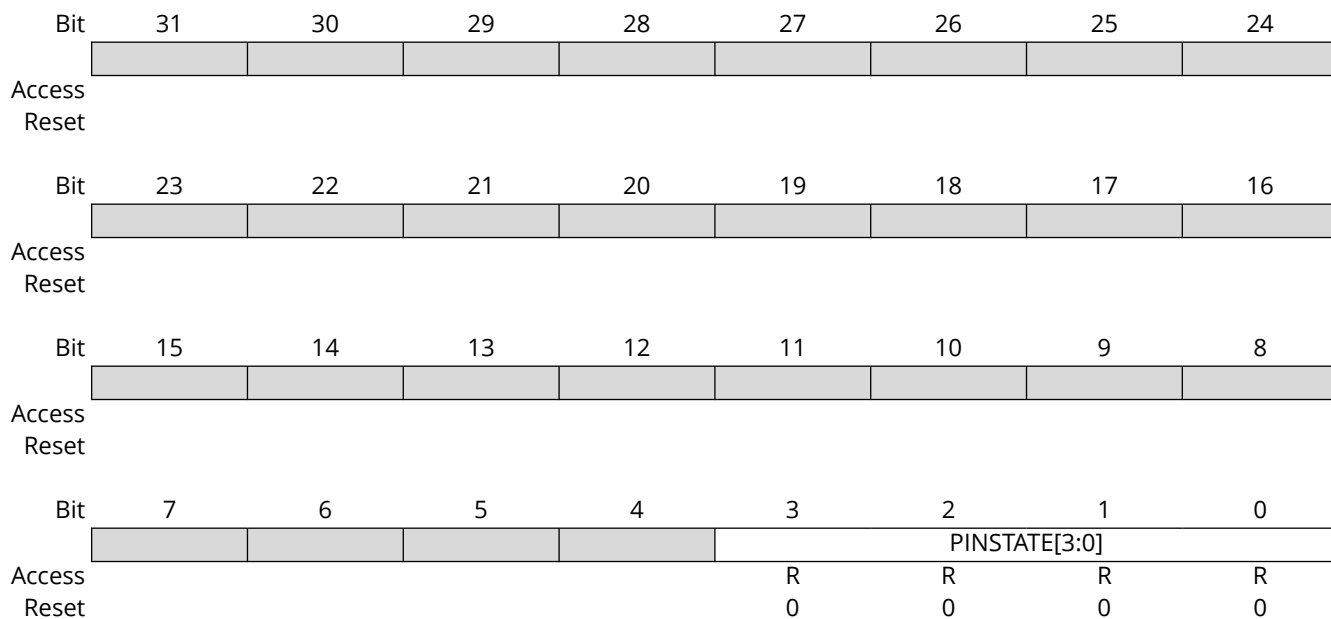
### Bits 0:2, 4:6 – PRESCALERx Debouncer Prescaler x

These bits select the debouncer low frequency clock for pins EXTINT[7+(8x):8x].

Value	Name	Description
0x0	F/2	EIC clock divided by 2
0x1	F/4	EIC clock divided by 4
0x2	F/8	EIC clock divided by 8
0x3	F/16	EIC clock divided by 16
0x4	F/32	EIC clock divided by 32
0x5	F/64	EIC clock divided by 64
0x6	F/128	EIC clock divided by 128
0x7	F/256	EIC clock divided by 256

### 23.8.13 Pin State

**Name:** PINSTATE  
**Offset:** 0x38  
**Reset:** 0x00000000



#### Bits 3:0 – PINSTATE[3:0] Pin State

These bits return the valid pin state of the debounced external interrupt pin EXTINTx.



## 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:

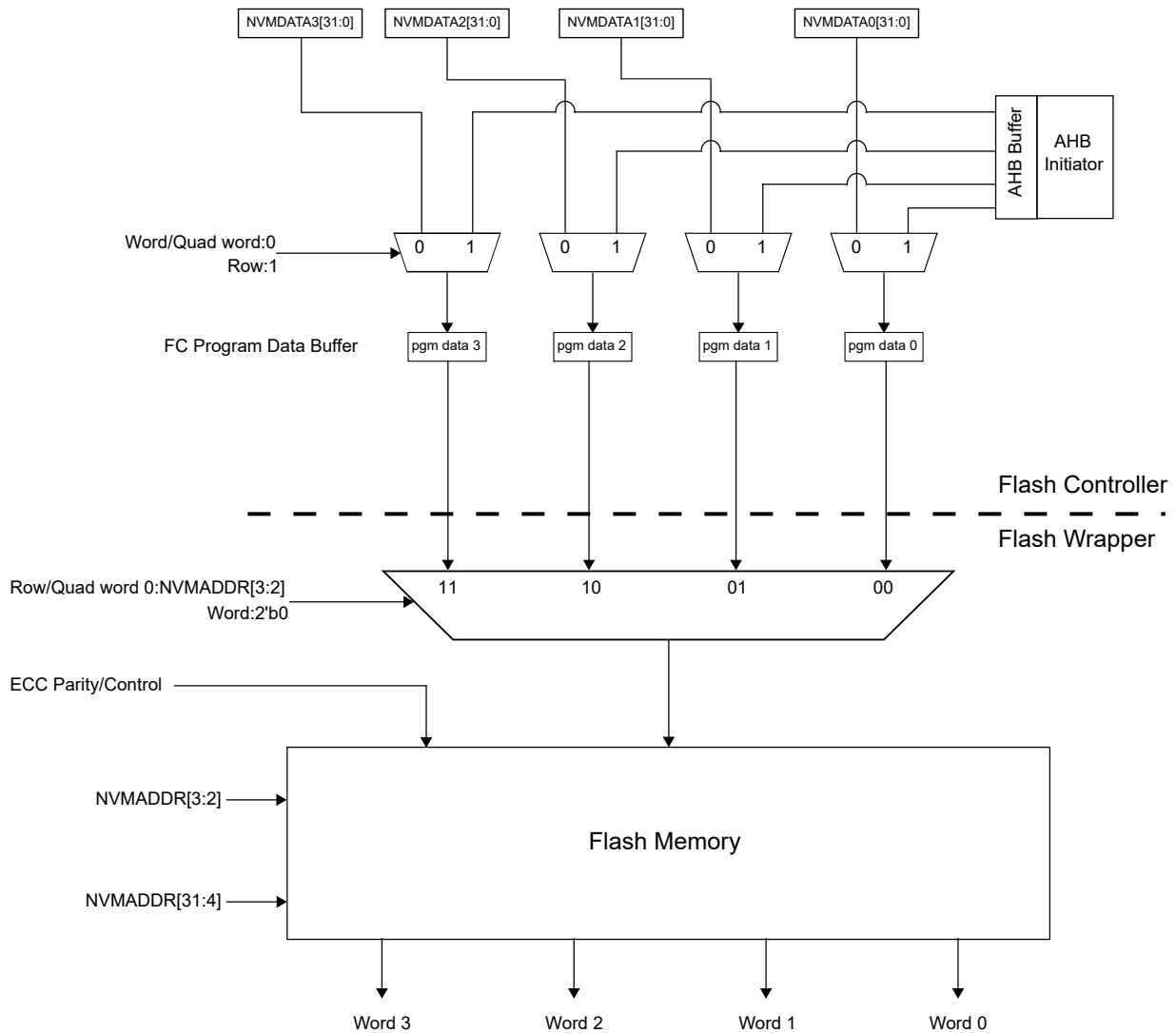
1. Run-Time Self-Programming (RTSP)
2. Serial Wire Debug (SWD) programming using DSU (See *Device Service Unit (DSU)* from Related Links and *PIC32CX-BZ2 Programming Specification*.)

#### 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

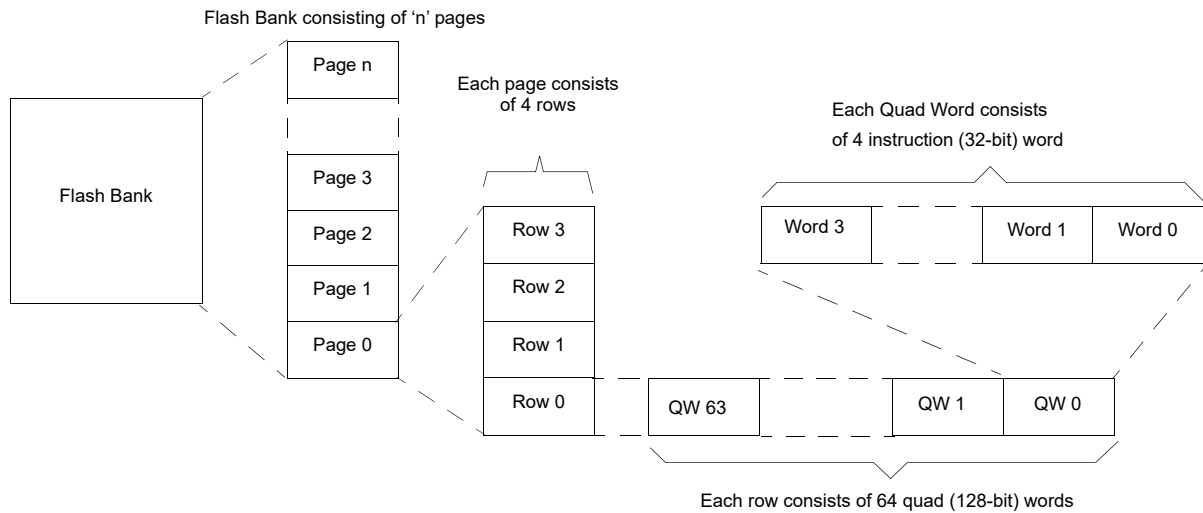
[8. Product Memory Mapping Overview](#)

## 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 0s in the NVMPWPLT register and all 1s 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 Single 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
NVMMADDR= 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 NVMMADDR, NVMDATA1 at NVMMADDR + 0x4, NVMDATA2 at NVMMADDR + 0x8, and NVMDATA3 at address NVMDATA + 0xC.

Quad Word programming is always performed on a Quad Word boundary; therefore, NVMMADDR 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
NVMMADDR = 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 & 0x1FFFFFFF);

// 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_t saveNVMCON2;
uint32_t *cmpPtr;
uint8_t erased;
uint8_t tryCount;

// 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.NVMPPREPG = 1` and, secondly, with `NVMCON2.NVMPPREPG = 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.NVMPPREPG` 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					HTDPM
		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	PWPGE[7:0]							
		15:8	PWPGE[15:8]							
		23:16	PWPGE[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
Access	WR	WREN	WRERR	LVDERR				HTDPGM
Reset	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
Access					NVMOP[3:0]			
Reset					R/W	R/W	R/W	R/W
Reset					0	0	0	0

#### Bit 15 – WR Write Control Bit<sup>(1)</sup>

**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<sup>(1)</sup>

Value	Description
1	Enables writes to WR
0	Disables writes to WR

#### Bit 13 – WRERR Write Error Bit<sup>(1)</sup>

**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<sup>(1)</sup>

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 <sup>(2)</sup> .
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:

- When VREAD1 = 1, WS[4:0] only affects the memory containing NVMADDR[31:0].
- 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	<ul style="list-style-type: none"> <li>Address identifies the page to erase</li> <li>Any address within a 4 Kbytes page boundary will cause the page to be erased</li> </ul>
Row program	<ul style="list-style-type: none"> <li>Address identifies the row to program</li> <li>The value of the address must be aligned to a row boundary</li> </ul>
Word program	<ul style="list-style-type: none"> <li>Address identifies the 32-bit Word to program</li> <li>NVMADDR[1:0] bits are ignored</li> <li>Must be aligned to a Word boundary</li> </ul>
Quad Word program	<ul style="list-style-type: none"> <li>Address identifies the 128-bit Quad Word to program</li> <li>NVMADDR[3:0] bits are ignored</li> <li>Must be aligned to a Quad Word boundary</li> </ul>

### Notes:

- Hardware prevents writes to this register when NVMCON.WR = 1.
- 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.
- 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
	PWPGE[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
	PWPGE[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
	PWPGE[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 PWPGE[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 – PWPGE[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.

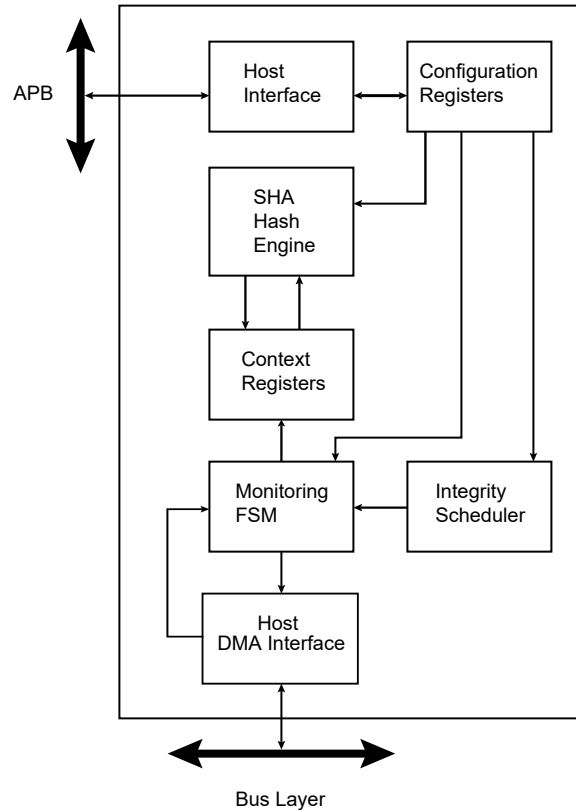
### 25.2 Features

- 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

The ICM will run only when the source clocks are running, i.e. when the CPU is in Active mode.

#### 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

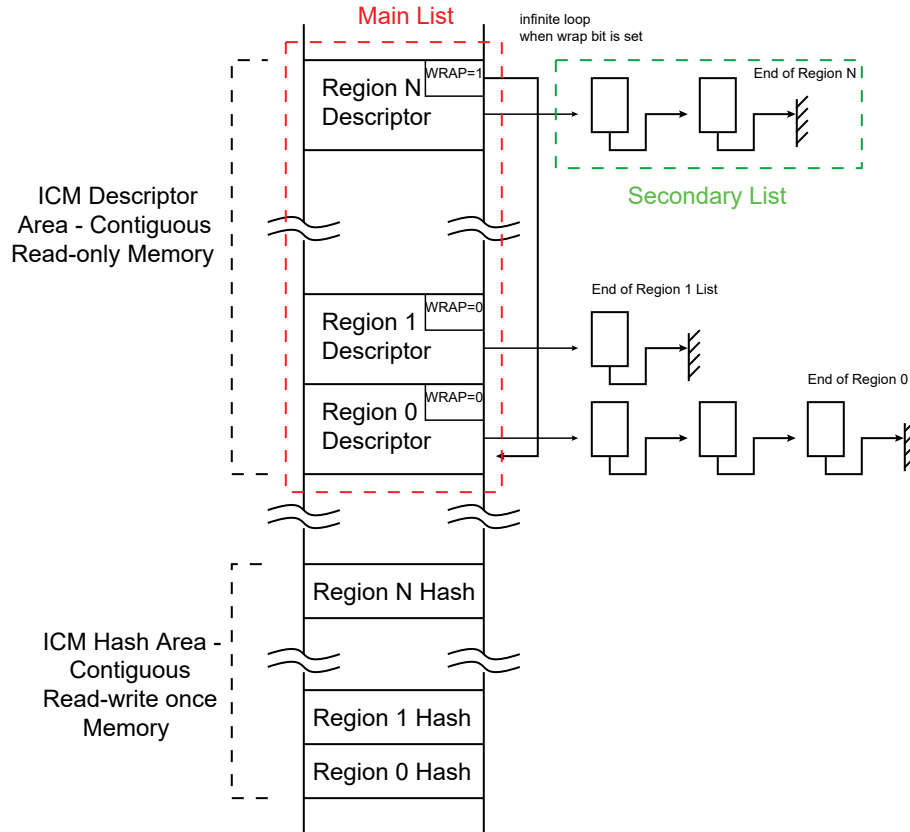
### 25.6.1 Overview

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.

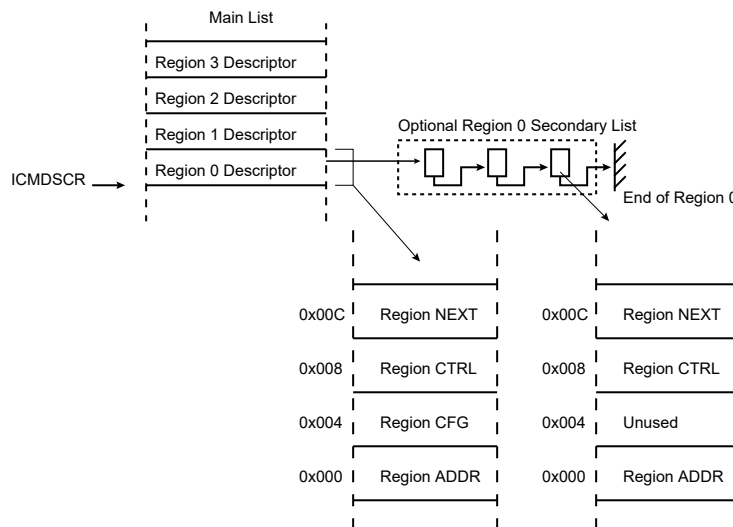
When the ICM module is enabled, it sequentially retrieves a circular list of region descriptors from the memory (Main List described in the following figure). Up to four regions may be monitored. Each region descriptor is composed of four words indicating the layout of the memory region (see *Region 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.

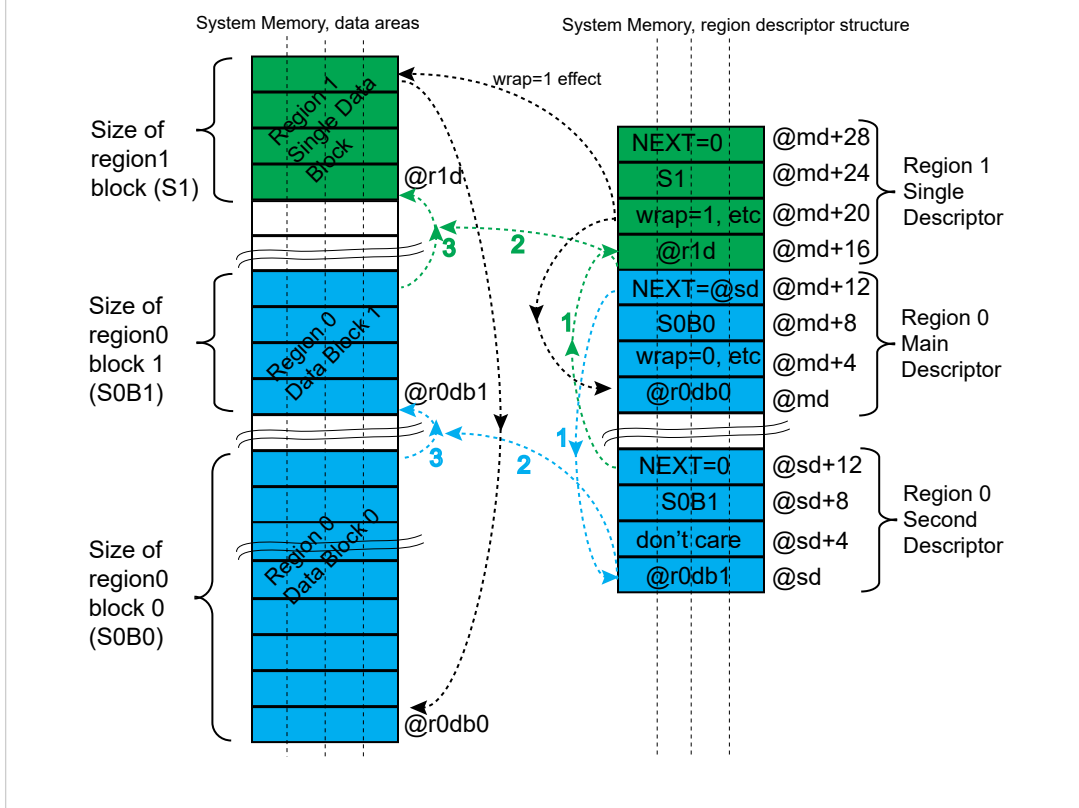
Figure 25-3. Region Descriptor







**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
Access		ALGO[2:0]				PROCDLY	SUIEN	ECIEN
Reset		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
Access	WCIEEN	BEIEN	DMIEN	RHIEN		EOM	WRAP	CDWBN
Reset	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 differs 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.

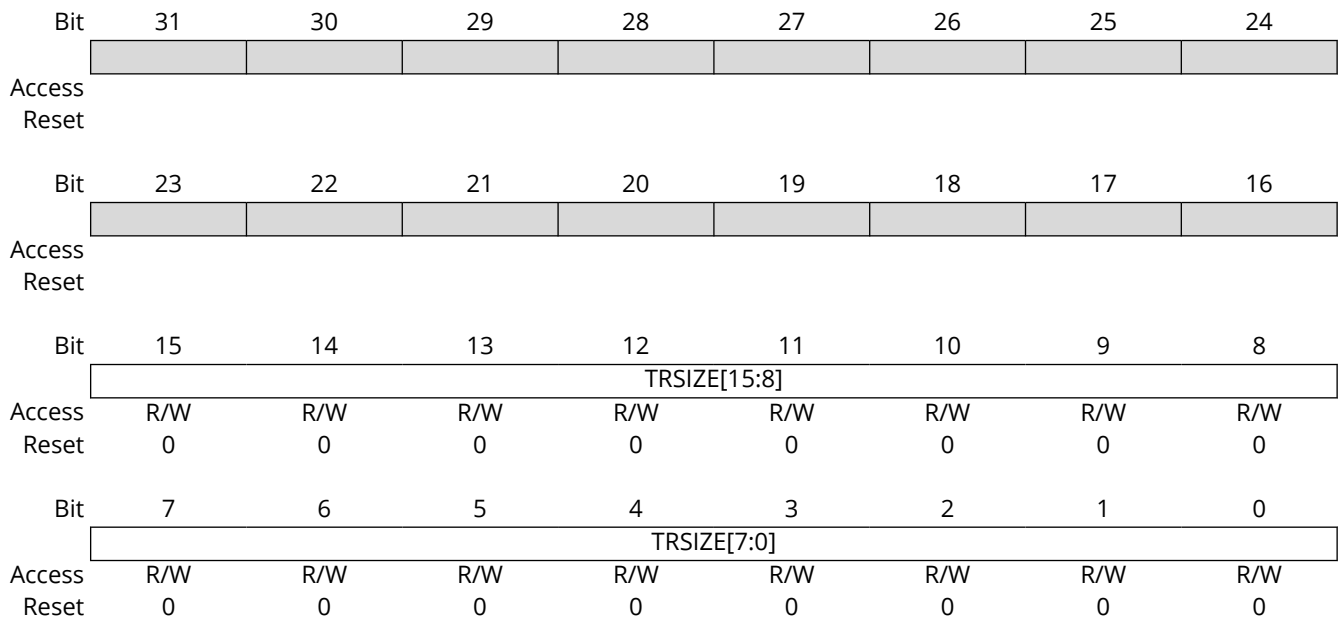
**Bit 0 – CDWBN** Compare Digest or Write Back Digest

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

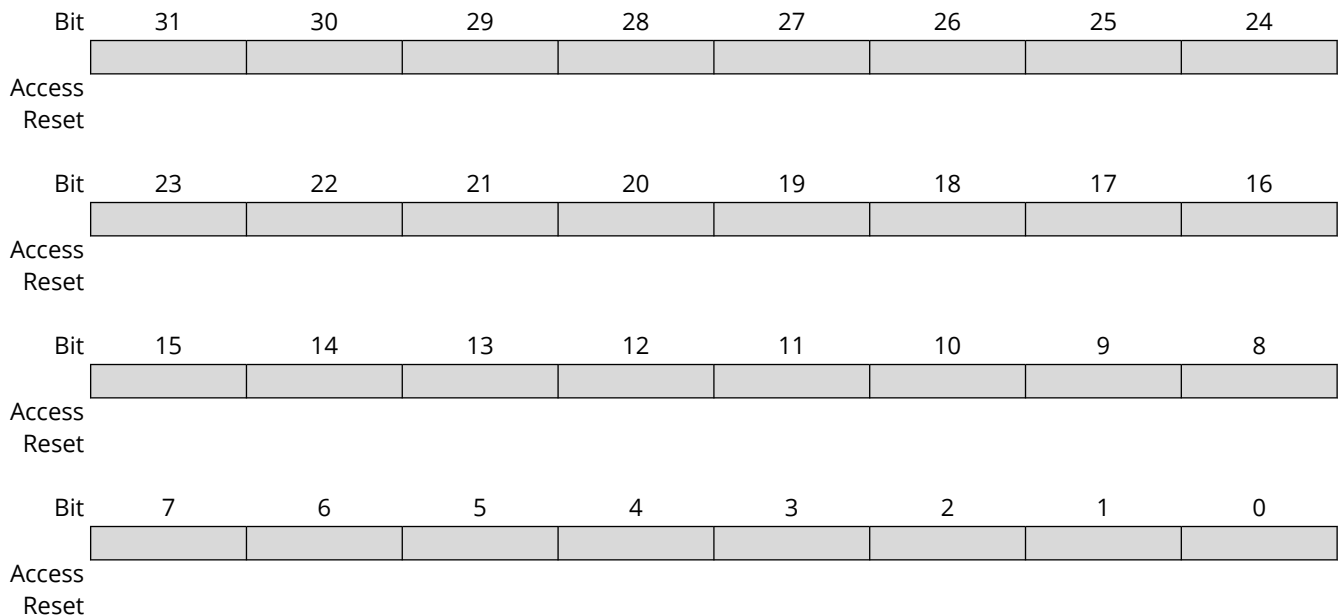
**Name:** RCTRL  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** R/W



**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



### 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 RHIE and ECIE in the Region Configuration Structure Member (RCFGn) to '0':

- The flag RHC[i], 'i' being the region index, is set (if RHIE 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 ECIE 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]					
0x40	UIHVALx2	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	
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
Access	UALGO[2:0]			UIHASH			DUALBUFF	ASCD
Reset	-	-	-	-			-	R/W
Reset	0	0	0	0			0	0
Bit	7	6	5	4	3	2	1	0
Access	BBC[3:0]					SLBDIS	EOMDIS	WBDIS
Reset	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
Access	RMEN[3:0]				RMDIS[3:0]			
Reset	W	W	W	W	W	W	W	W
Bit	7	6	5	4	3	2	1	0
Access	REHASH[3:0]					SWRST	DISABLE	ENABLE
Reset	W	W	W	W		W	W	W
							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.
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 <a href="#">RMEN[i]</a> of <a href="#">CTRL</a>
1	Region i monitoring has been deactivated by writing a 1 in <a href="#">RMDIS[i]</a> of <a href="#">CTRL</a>

#### 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.
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
Access								URAD
Reset								W
Bit	23	22	21	20	19	18	17	16
Access	RSU[3:0]				REC[3:0]			
Reset	W	W	W	W	W	W	W	W
Bit	15	14	13	12	11	10	9	8
Access	RWC[3:0]				RBE[3:0]			
Reset	W	W	W	W	W	W	W	W
Bit	7	6	5	4	3	2	1	0
Access	RDM[3:0]				RHC[3:0]			
Reset	W	W	W	W	W	W	W	W

### 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.



Value	Description
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.
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

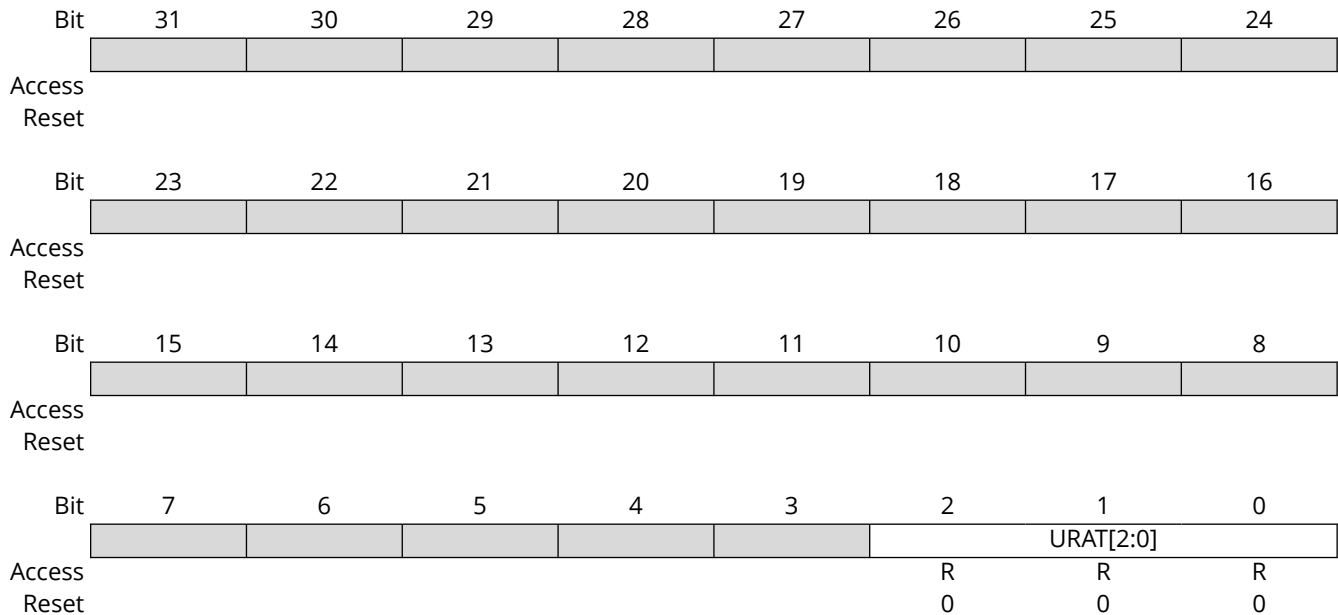
RDM[i] is set when there is a digest comparison mismatch between the hash value of region i and the reference value located in the Hash Area.

### 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:** -

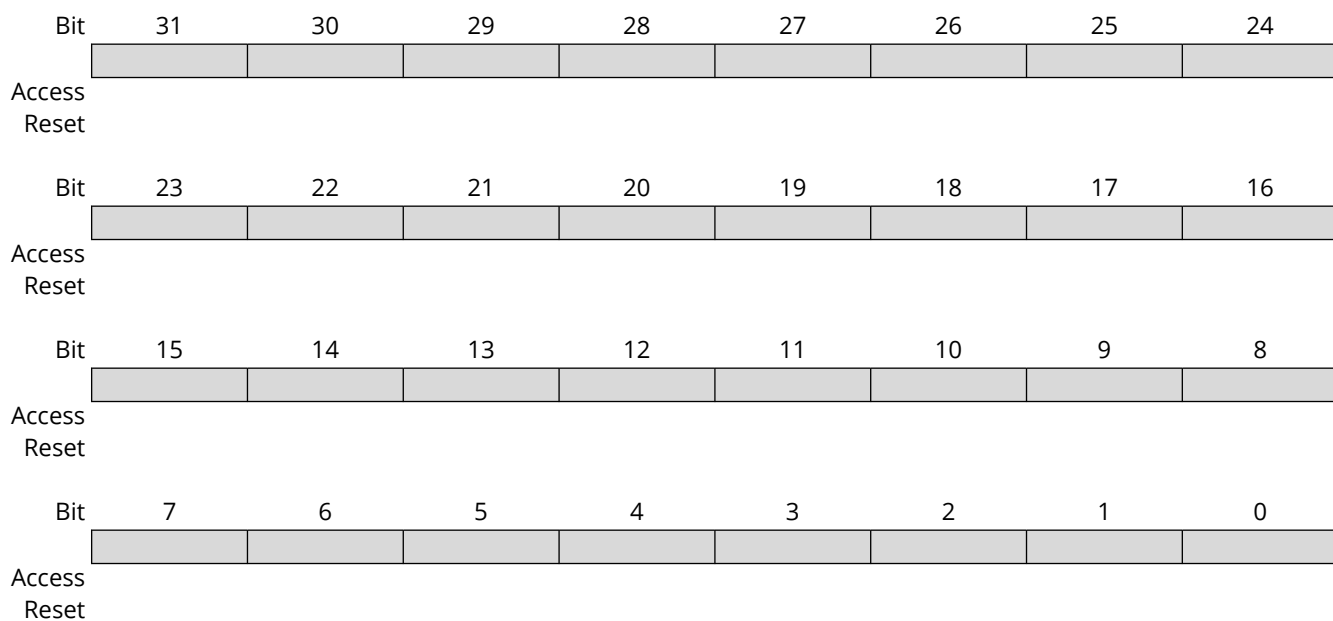
Bit	31	30	29	28	27	26	25	24
	DASA[25:18]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	DASA[17:10]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	DASA[9:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
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:** -



## 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

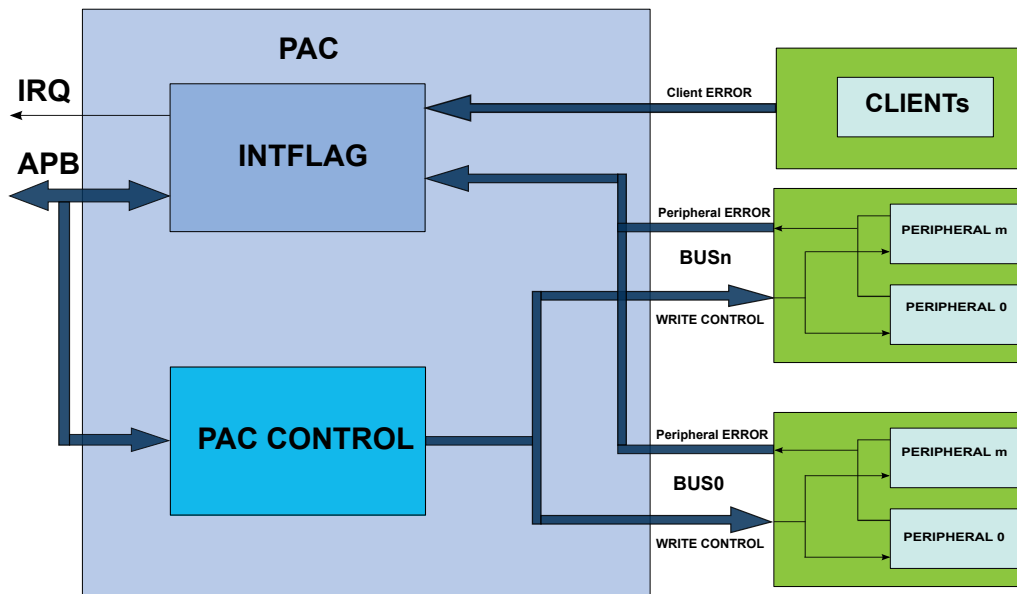
[18. System Configuration and Register Locking \(CFG\)](#)

### 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. 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 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	Reserved									
...										
0x07										
0x08	INTENCLR	7:0								ERR
0x09	INTENSET	7:0								ERR
0x0A	Reserved									
...										
0x0F										
0x10	INTFLAGAHB	7:0	PBBB	PBAB	PFLASH	CFLASH	SRAM3	SRAM2	SRAM1	SRAM0
		15:8					PUKCC	QSPI	PBPICB	PBCB
		23:16								
		31:24								
0x14	INTFLGA	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		SERCOM3	SERCOM2		AES	QSPI
		15:8						TRNG		
		23:16								
		31:24								
0x20	Reserved									
...										
0x33										
0x34	STATUSA	7:0	TC2	TC1	TC0	SERCOM1	SERCOM0	EIC	FREQM	PAC
		15:8					TCC2	TCC1	TCC0	TC3
		23:16								
		31:24								
0x38	STATUSB	7:0				RAMECC	EVSYS	DMAC		DSU
		15:8								
		23:16								
		31:24								
0x3C	STATUSC	7:0	AC	CCL		SERCOM3	SERCOM2		AES	QSPI
		15:8						TRNG		
		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:

$$PERID = 32 * BridgeNumber + N$$

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, which can be found in the tables of Section 12 Peripheral Configuration Summary in the "PAC, Index" column.

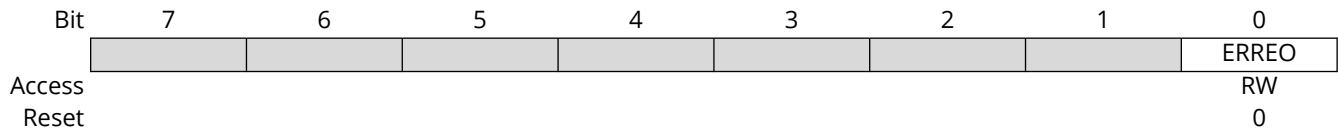
**Table 26-2.** PERID Values

Periph. Bridge Name	BridgeNumber	PERID Values
A	0	0+N
B	1	32+N
C	2	64+N
D	3	96+N



## 26.7.2 Event Control

**Name:** EVCTRL  
**Offset:** 0x04  
**Reset:** 0x00



### 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
Access					PUKCC	QSPI	PBPICB	PBCB
Reset					RW	RW	RW	RW
					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	PBBB	PBAB	PFLASH	CFLASH	SRAM3	SRAM2	SRAM1	SRAM0
Reset	RW	RW	RW	RW	RW	U	U	RW
	0	0	0	0	0	0	0	0

### Bit 11 – PUKCC Interrupt Flag for PUKCC

This flag is set when an access error is detected by the PUKCC 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 PUKCC interrupt flag.

### 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
Access					TCC2	TCC1	TCC0	TC3
Reset					RW	RW	RW	RW
					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	TC2	TC1	TC0	SERCOM1	SERCOM0	EIC	FREQM	PAC
Reset	RW	RW	RW	RW	RW	RW	RW	RW
	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.

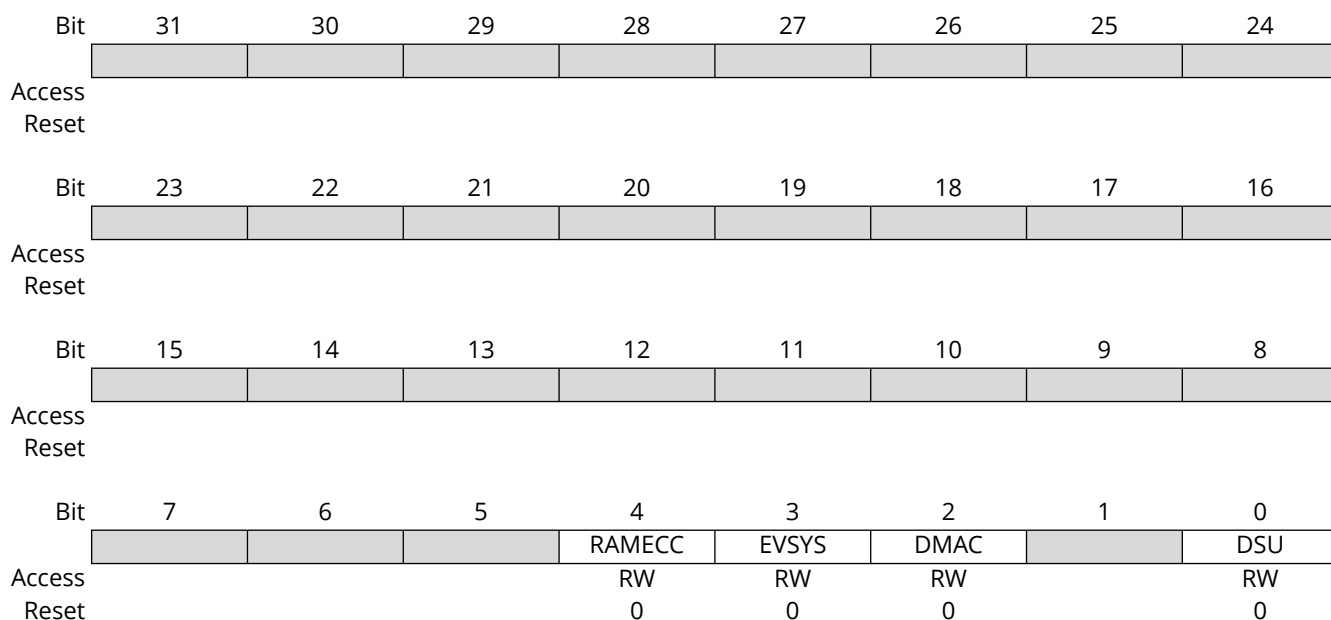
## 26.7.7 Peripheral Interrupt Flag Status – Bridge B

**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 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						TRNG		
Reset						RW		
						0		
Bit	7	6	5	4	3	2	1	0
Access	AC	CCL		SERCOM3	SERCOM2		AES	QSPI
Reset	RW	RW		RW	RW		RW	RW
	0	0		0	0		0	0

### Bit 10 – TRNG Interrupt Flag for TRNG

This flag is set when a Peripheral Access Error occurs while accessing the peripheral associated with the TRNG 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 TRNG interrupt flag.

### 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.

### Bit 4 – SERCOM3 Interrupt Flag for SERCOM3

This flag is set when a Peripheral Access Error occurs while accessing the peripheral associated with the SERCOM3 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 SERCOM3 interrupt flag.

**Bit 3 – SERCOM2** Interrupt Flag for SERCOM2

This flag is set when a Peripheral Access Error occurs while accessing the peripheral associated with the SERCOM2 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 SERCOM2 interrupt flag.

**Bit 1 – AES** Interrupt Flag for AES

This flag is set when a Peripheral Access Error occurs while accessing the peripheral associated with the AES 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 AES interrupt flag.

**Bit 0 – QSPI** Interrupt Flag for QSPI

This flag is set when a Peripheral Access Error occurs while accessing the peripheral associated with the QSPI 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 QSPI 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
Access					TCC2	TCC1	TCC0	TC3
Reset					R	R	R	R
					0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access	TC2	TC1	TC0	SERCOM1	SERCOM0	EIC	FREQM	PAC
Reset	R	R	R	R	R	R	R	R
	0	0	0	0	0	0	0	0

### Bit 11 – TCC2 TCC2 APB Protect Enable

Value	Description
0	TCC2 is not write protected
1	TCC2 is write protected

### Bit 10 – TCC1 TCC1 APB Protect Enable

Value	Description
0	TCC1 is not write protected
1	TCC1 is write protected

### Bit 9 – TCC0 TCC0 APB Protect Enable

Value	Description
0	TCC0 is not write protected
1	TCC0 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

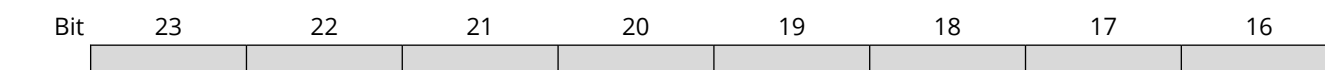
Writing to this register has no effect.

Reading STATUS register returns peripheral write protection status:

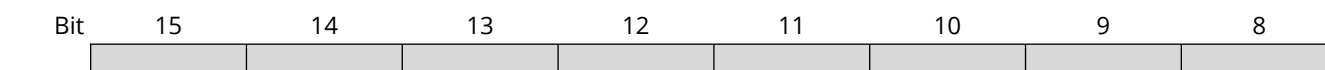
Value	Description
0	Peripheral is not write protected.
1	Peripheral is write protected.



Access  
Reset



Access  
Reset



Access  
Reset



Access  
Reset

			R	R	R		R
			0	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

### Bit 0 – DSU DSU APB Protect Enable

Value	Description
0	DSU peripheral is not write protected
1	DSU 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						TRNG		
Reset						R		
						0		
Bit	7	6	5	4	3	2	1	0
Access	AC	CCL		SERCOM3	SERCOM2		AES	QSPI
Reset	R	R		R	R		R	R
	0	0		0	0		0	0

### Bit 10 – TRNG TRNG APB Protection Enable

Value	Description
0	Peripheral is not write protected
1	Peripheral is write protected

### 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

**Bit 1 - AES** AES APB Protection Enable

Value	Description
0	Peripheral is not write protected
1	Peripheral is write protected

**Bit 0 - QSPI** QSPI APB Protection Enable

Value	Description
0	Peripheral is not write protected
1	Peripheral is write protected



## 27. Frequency Meter (FREQM)

### 27.1 Overview

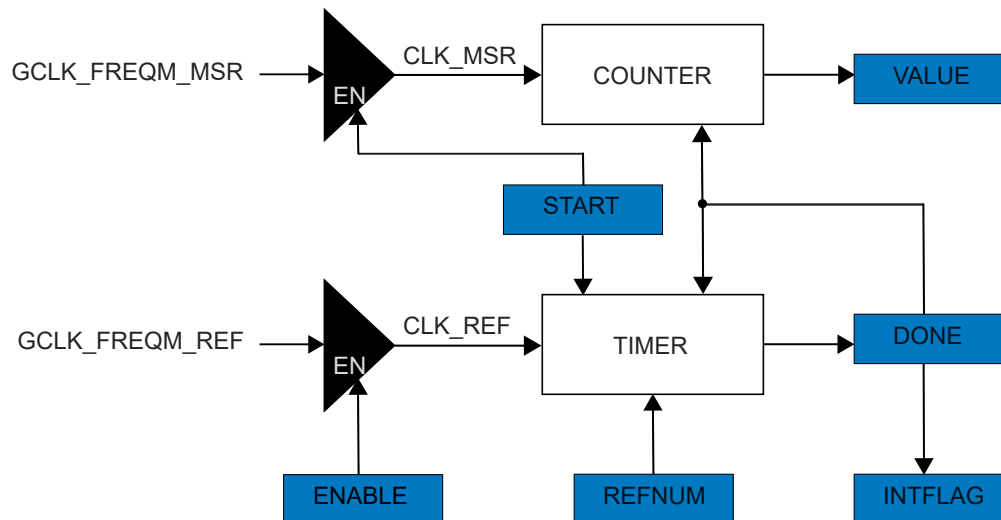
The Frequency Meter (FREQM) can be used to accurately measure the frequency of a clock by comparing it to a known reference clock.

### 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  $\text{REFNUM}/f_{\text{CLK\_REF}}$  and stored in the Value register (VALUE.VALUE). REFNUM is the number of Reference clock cycles selected in the Configuration A register (CFG.A.REFNUM).

The frequency of the measured clock,  $f_{\text{CLK\_MSR}}$ , 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:

- Each of the generic clocks (GCLK\_FREQM\_REF and GCLK\_FREQM\_MSR) must be configured and enabled.
- 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.

The following register is enable-protected, meaning that it can only be written when the FREQM is disabled (CTRLA.ENABLE=0):

- 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:

$$f_{CLK\_MSR} = \left( \frac{VALUE}{REFNUM} \right) f_{CLK\_REF}$$

#### 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 *CFG*, *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

The FREQM has one interrupt source:

- 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

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.

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	Reserved									
...										
0x07										
0x08	INTENCLR	7:0								DONE
0x09	INTENSET	7:0								DONE
0x0A	INTFLAG	7:0								DONE
0x0B	STATUS	7:0							OVF	BUSY
									ENABLE	SWRST
0x0C	SYNCBUSY	7:0								
		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  
**Reset:** 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:** -

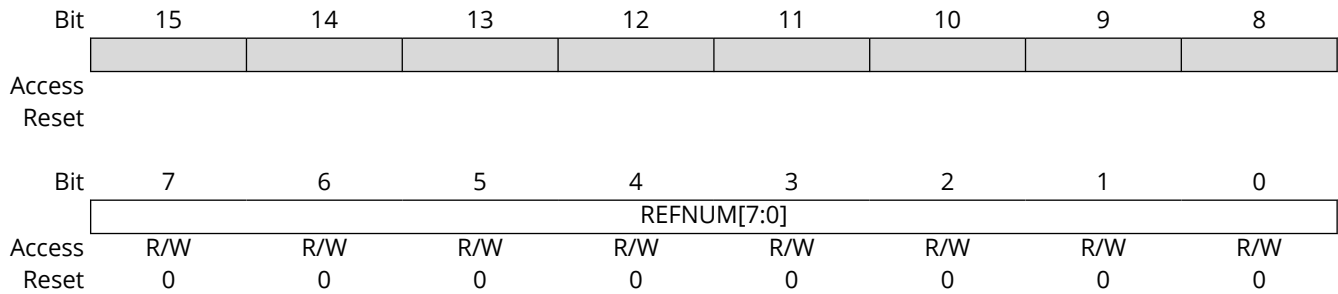
Bit	7	6	5	4	3	2	1	0
								START
Access								W
Reset								0

### 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



**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

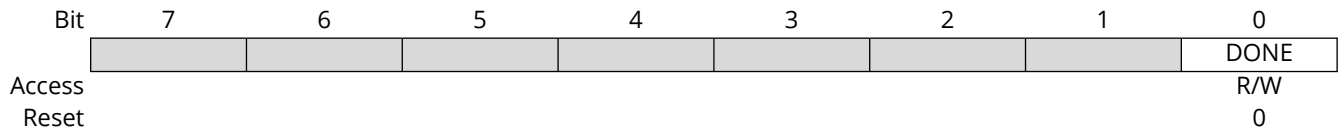
Writing a '0' to this bit has no effect.

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 0 – DONE Measurement Done Interrupt Enable

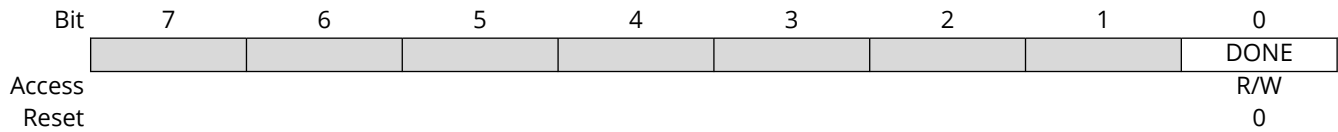
Writing a '0' to this bit has no effect.

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 0 - DONE Measurement 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.

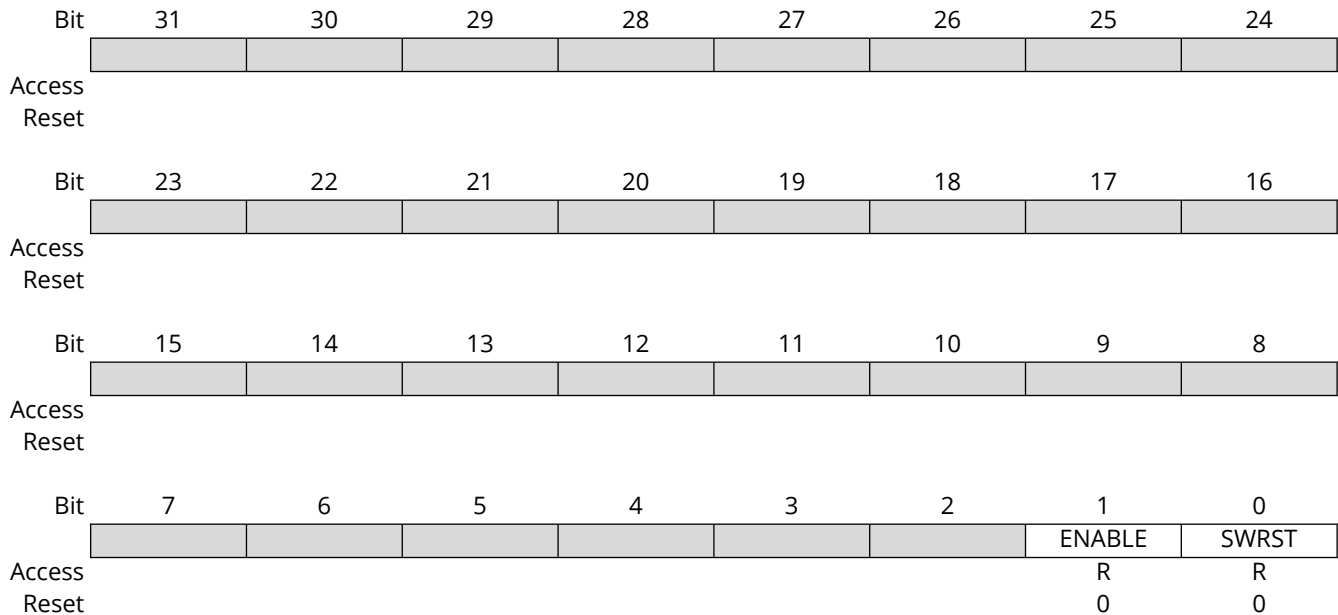
Writing a '1' to this bit will clear the OVF status.

### 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 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.

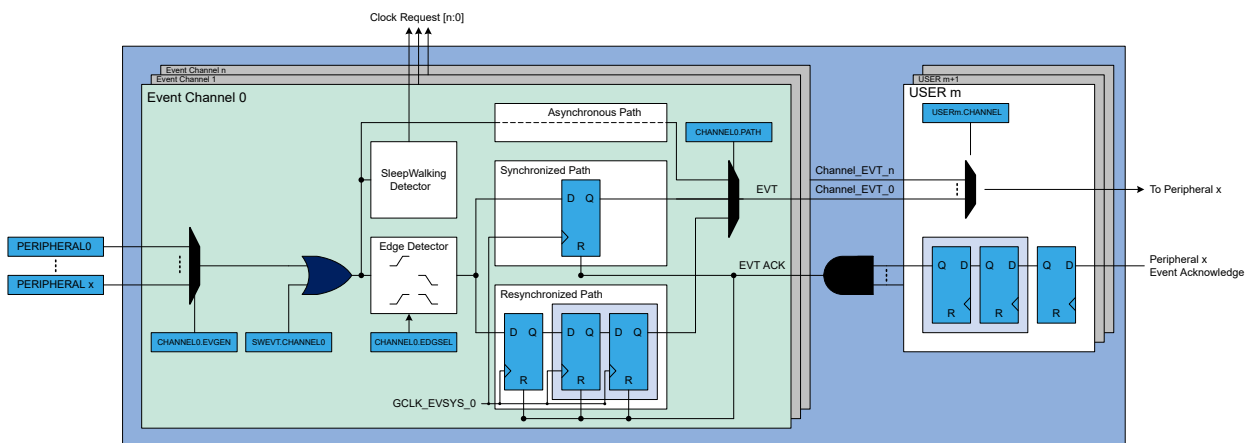
Communication is made without CPU intervention and without consuming system resources such as bus or RAM bandwidth. This reduces the load on the CPU and other system resources, compared to a traditional interrupt-based system.

### 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 peripheral generating the event, 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 or 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 (EVACTION) in the respective Event control register, for example, TC.EVCTRL.EVACTION, PDEC.EVCTRL.EVACTION.
 

**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 user multiplexers 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 from 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
- Synchronous 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.

### Synchronous Path

The synchronous path must be used when the event generator and the event channel share the same generator for the generic clock. If they do not share the same clock, a logic change from the event generator to the event channel might not be detected in the channel, which means that the event will not be propagated to the event user.

When using the synchronous path, the channel is able to generate interrupts. The channel status bits in the Channel Status register (CHSTATUS) are also updated and available for use.

### 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.
- Synchronous Path: The maximum routing latency of an external event is one GCLK\_EVSYS\_CHANNEL\_n clock cycle.
- 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 synchronous or 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 synchronous or 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.

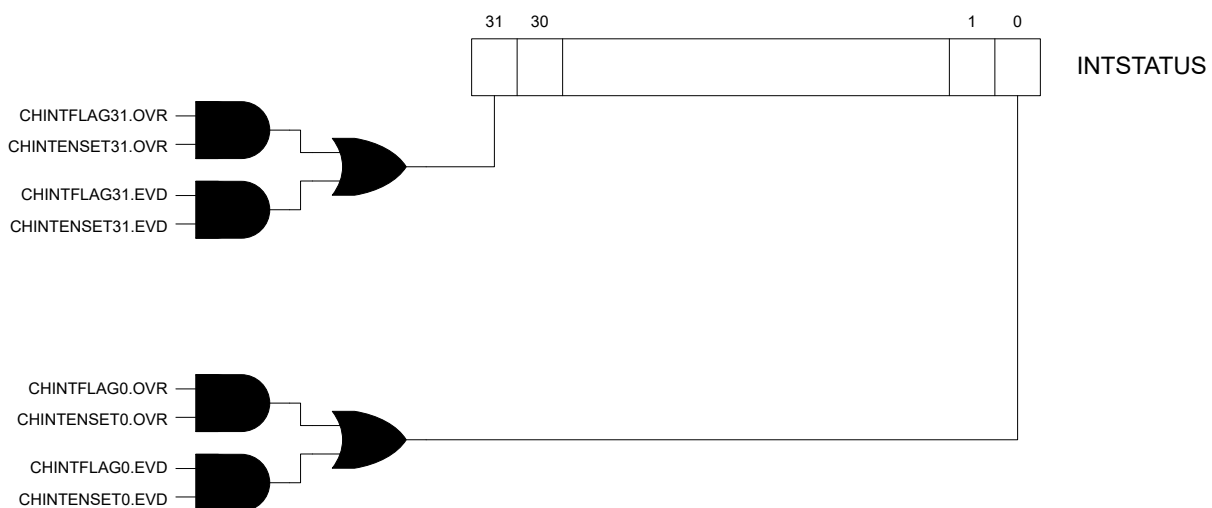
### 28.5.2.12 Software Event

A software event can be initiated on a channel by writing a '1' to the Software Event bit in the Channel register (CHANNELm.SWEVT). Then the software event can be serviced as any event generator; i.e., when the bit is set to '1', an 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.

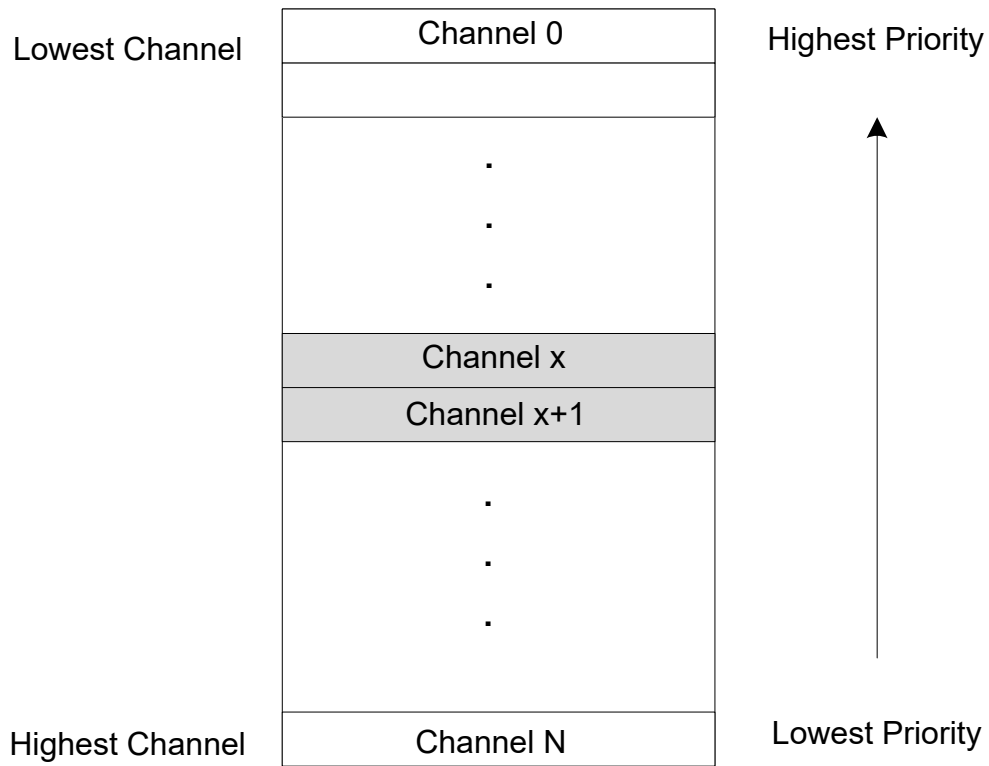
**Figure 28-2.** Interrupt Status Register



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.

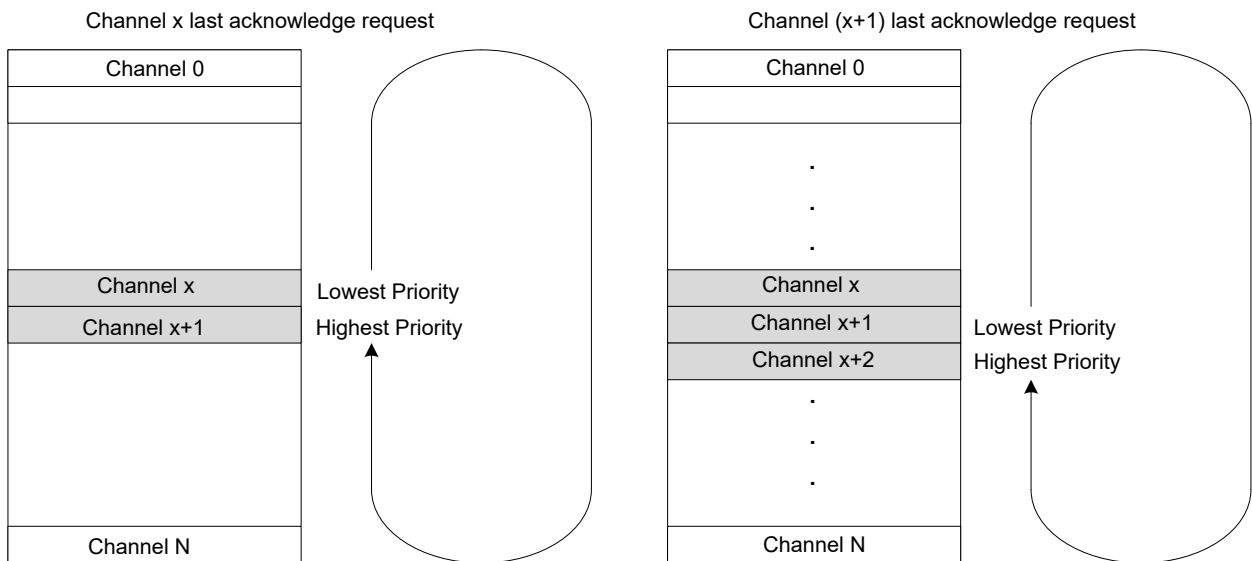
By default, static arbitration is enabled (PRICTRL.RRENx is '0'), the arbiter will prioritize a low channel number over a high channel number as shown below. When using the status scheme, there is a risk of high channel numbers never being granted access by the arbiter. This can be avoided using a dynamic 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 sleep 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 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 (i.e., 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
ASYN	0	0	Only run in IDLE sleep modes if an event must be propagated. Disabled in STANDBY sleep mode.
SYNC/RESYNC	0	1	Run in both IDLE and STANDBY sleep modes.
SYNC/RESYNC	1	0	Only run in IDLE sleep modes if an event must be propagated. Disabled in STANDBY sleep 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 sleep modes. Two GCLK_EVSYS_n latency added in RESYNC path before the event is propagated internally.

## 28.6 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
0x00	CTRLA	7:0								SWRST	
0x01	Reserved										
...											
0x03											
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	Reserved										
...											
0x0F											
0x10	INTPEND	7:0						ID[4:0]			
		15:8	BUSY	READY					EVD	OVR	
0x12	Reserved										
...											
0x13											
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		
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		

.....continued											
Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
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		
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		



.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
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									
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								

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
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									
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									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
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.

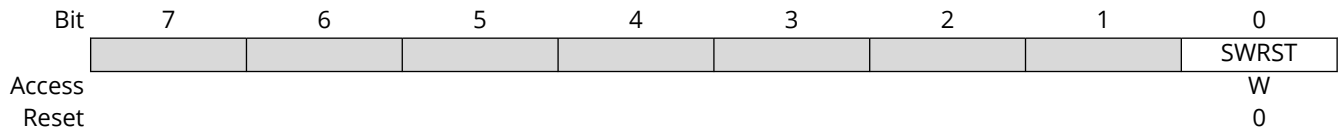
### Related Links

[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 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
Access					CHINT11	CHINT10	CHINT9	CHINT8
Reset					R	R	R	R
					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	CHINT7	CHINT6	CHINT5	CHINT4	CHINT3	CHINT2	CHINT1	CHINT0
Reset	R	R	R	R	R	R	R	R
	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
Access					BUSYCH11	BUSYCH10	BUSYCH9	BUSYCH8
Reset					R	R	R	R
					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	BUSYCH7	BUSYCH6	BUSYCH5	BUSYCH4	BUSYCH3	BUSYCH2	BUSYCH1	BUSYCH0
Reset	R	R	R	R	R	R	R	R
	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
Access					R	R	R	R
Reset					1	1	1	1

Bit	7	6	5	4	3	2	1	0
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
Access	ONDEMAND	RUNSTDBY			EDGSEL[1:0]		PATH[1:0]	
Reset	RW	RW			RW	RW	RW	RW
Reset	1	0			0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	EVGEN[7:0]							
Reset	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
0x0	SYNCHRONOUS	Synchronous path
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.

USERm	UserMultiplexer	Description	PathType <sup>(1)</sup>
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 four instances of the Serial Communication interface (SERCOM) peripheral.

A SERCOM can be configured to support a number of modes: I<sup>2</sup>C, 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).

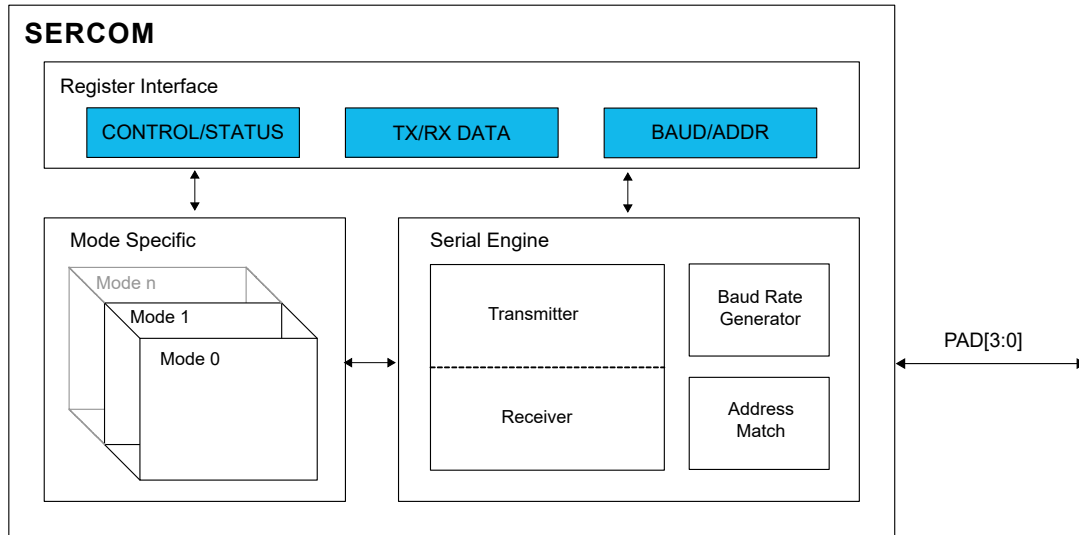
### 29.2 Features

- Interface for Configuring into one of the Following:
  - Inter-Integrated Circuit (I<sup>2</sup>C) 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
- 32-bit Extension for Better System Bus Utilization

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<sup>2</sup>C, 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\_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

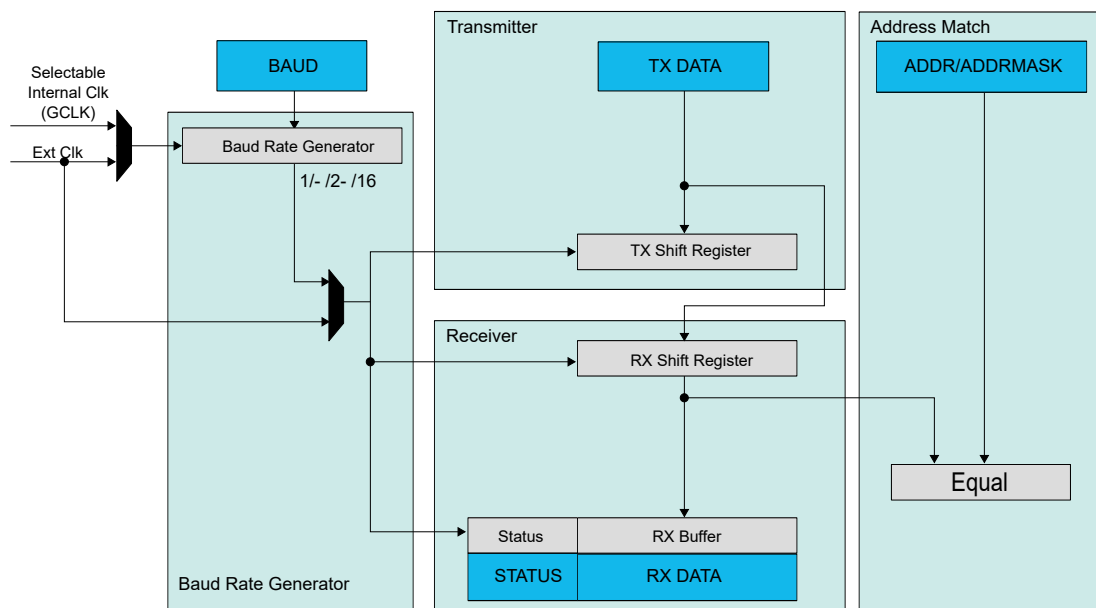
Not applicable.

## 29.6 Functional Description

### 29.6.1 Principle of Operation

The basic structure of the SERCOM serial engine is shown in [Figure 29-2](#). 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<sup>2</sup>C), 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<sup>2</sup>C 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 <sup>2</sup> C client operation
0x5	I <sup>2</sup> C 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

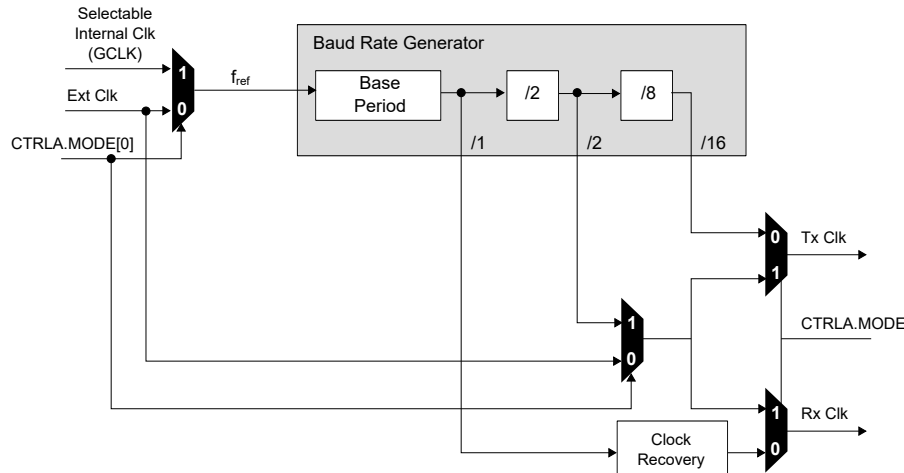
The baud-rate generator, as shown in the following figure, generates internal clocks for asynchronous and synchronous communication. The output frequency ( $f_{\text{BAUD}}$ ) is determined by the Baud register (BAUD) setting and the baud reference frequency ( $f_{\text{ref}}$ ). The baud reference clock 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.

*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

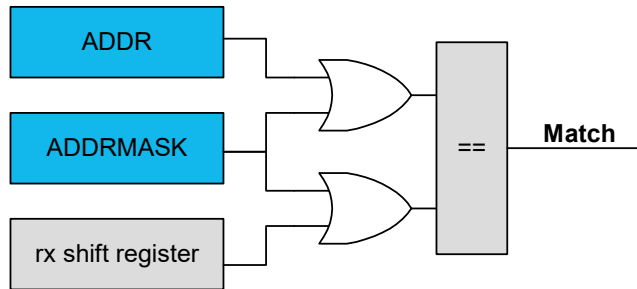
#### 29.6.3.1 Address Match and Mask

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.

### 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.

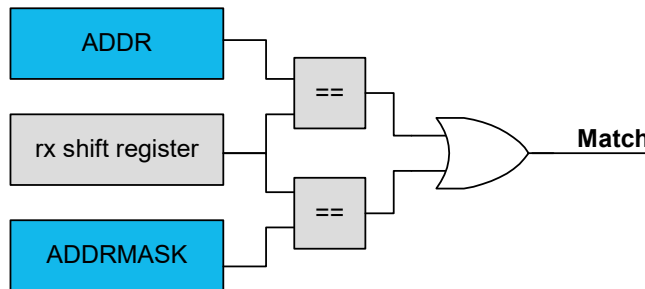
Figure 29-4. Address With Mask



### Two Unique Addresses

The two addresses written to ADDR and ADDRMASK will cause a match.

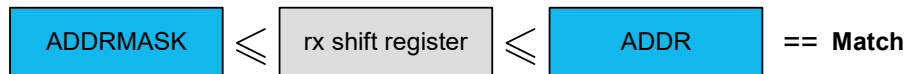
Figure 29-5. Two Unique Addresses



### 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

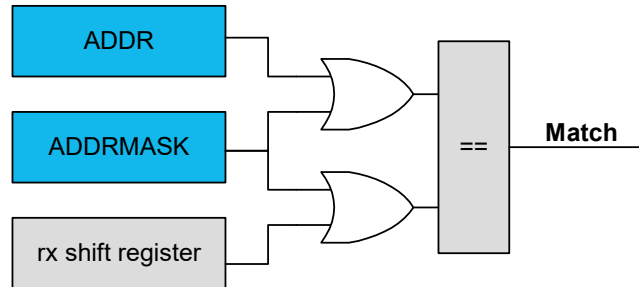


#### 29.6.3.1.1 Address With Mask

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.



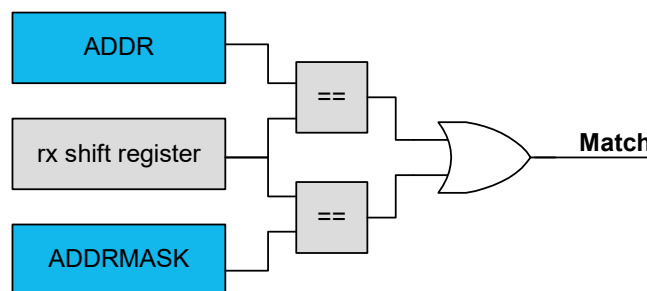
Figure 29-7. Address With Mask



### 29.6.3.1.2 Two Unique Addresses

The two addresses written to ADDR and ADDRMASK will cause a match.

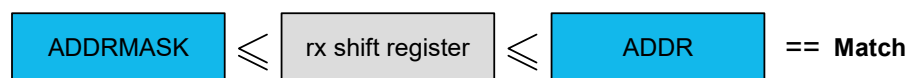
Figure 29-8. Two Unique Addresses



### 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 29-9. 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.

#### **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](#)

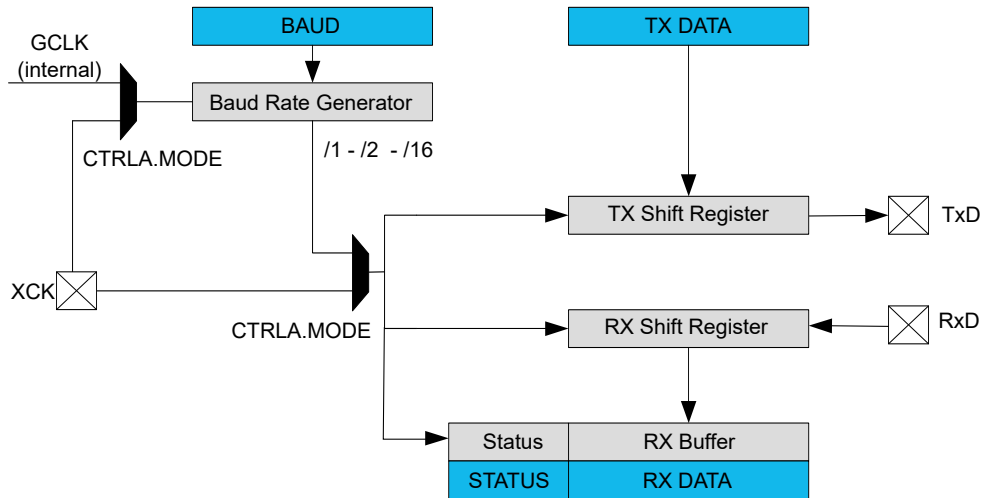
### 30.2 USART Features

- 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
- ISO 7816 T=0 or T=1 protocols for Smart Card Interfacing
- RS485 Support
- Start-of-frame detection

- Two-Level Receive Buffer
- Can work with DMA
- 32-bit Extension for Better System Bus Utilization

### 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

[30.8.14. 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:

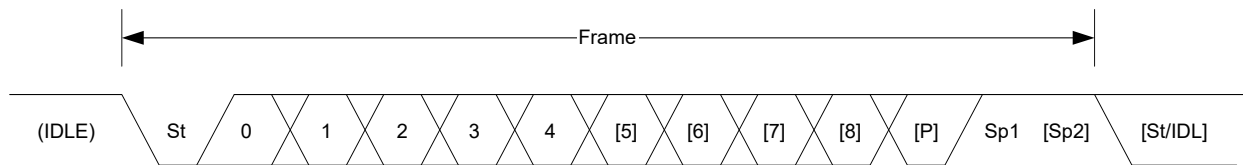
- RXD for receiving
- TXD for transmitting
- XCK for the transmission clock in synchronous operation

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.

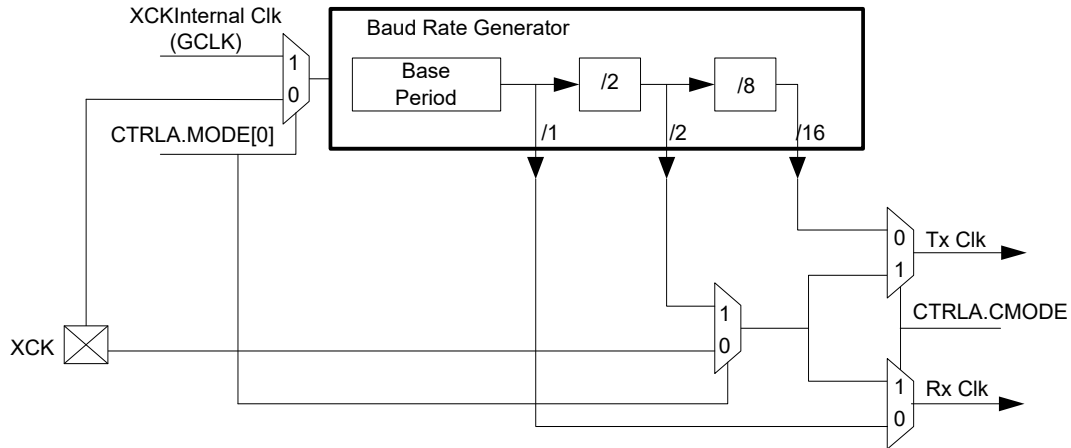
The internal clock source is selected by writing '1' to the Operation Mode bit field in the Control A register (CTRLA.MODE), the external clock source is selected by writing '0' to CTRLA.MODE.

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**

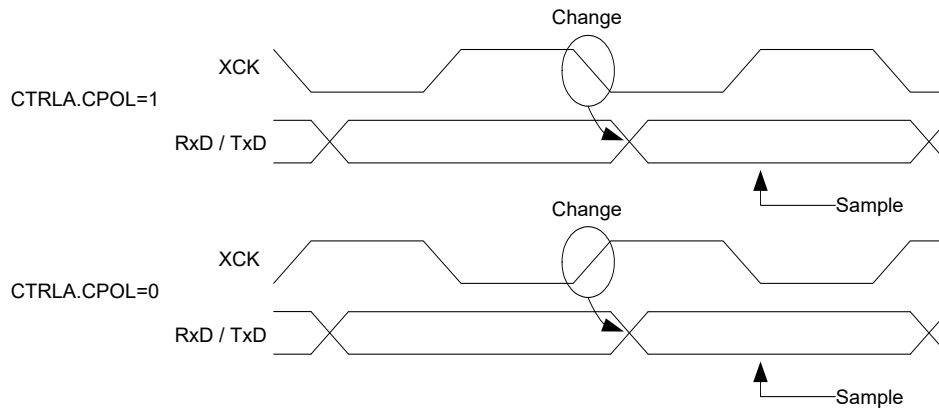
In Synchronous mode, the CTRLA.MODE bit field determines whether the transmission clock line (XCK) serves either as input or output. The dependency between clock edges, data sampling, and data change is the same for internal and external clocks. Data input on the RxD pin is sampled at the opposite XCK 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 <sub>SLOW</sub> [%]	R <sub>FAST</sub> [%]	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

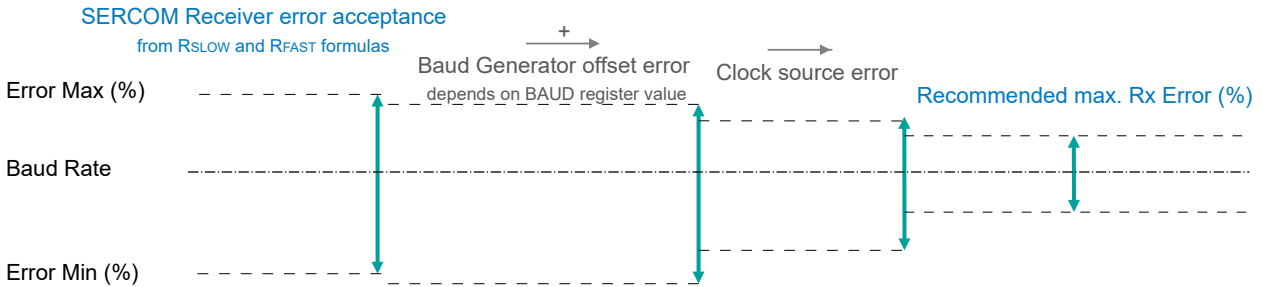
The following equations calculate the ratio of the incoming data rate and internal receiver baud rate:

$$R_{\text{SLOW}} = \frac{16(D + 1)}{16(D + 1) + 6} \quad , \quad R_{\text{FAST}} = \frac{16(D + 2)}{16(D + 1) + 8}$$

- $R_{\text{SLOW}}$  is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate
- $R_{\text{FAST}}$  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)

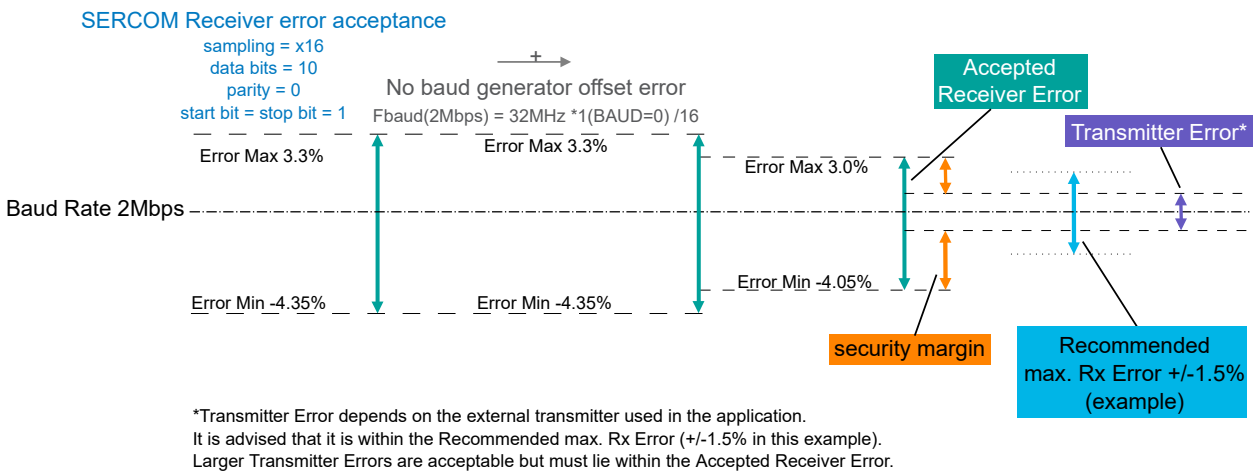
The recommended maximum Rx Error assumes that the receiver and transmitter equally divide the maximum total error. Its connection to the SERCOM Receiver error acceptance is depicted in this figure:

**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 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.

### 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 <sub>SLOW</sub> [%]	R <sub>FAST</sub> [%]	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

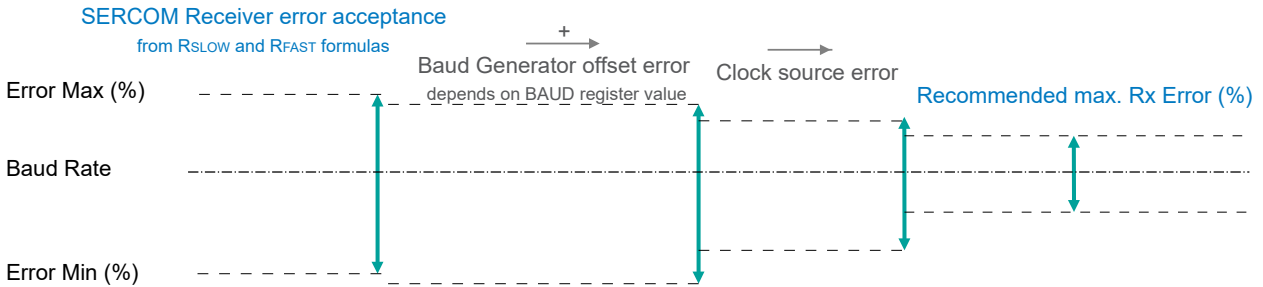
The following equations calculate the ratio of the incoming data rate and internal receiver baud rate:

$$R_{\text{SLOW}} = \frac{(D + 1)S}{S - 1 + D \cdot S + S_F} \quad , \quad R_{\text{FAST}} = \frac{(D + 2)S}{(D + 1)S + S_M}$$

- $R_{\text{SLOW}}$  is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate
- $R_{\text{FAST}}$  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.

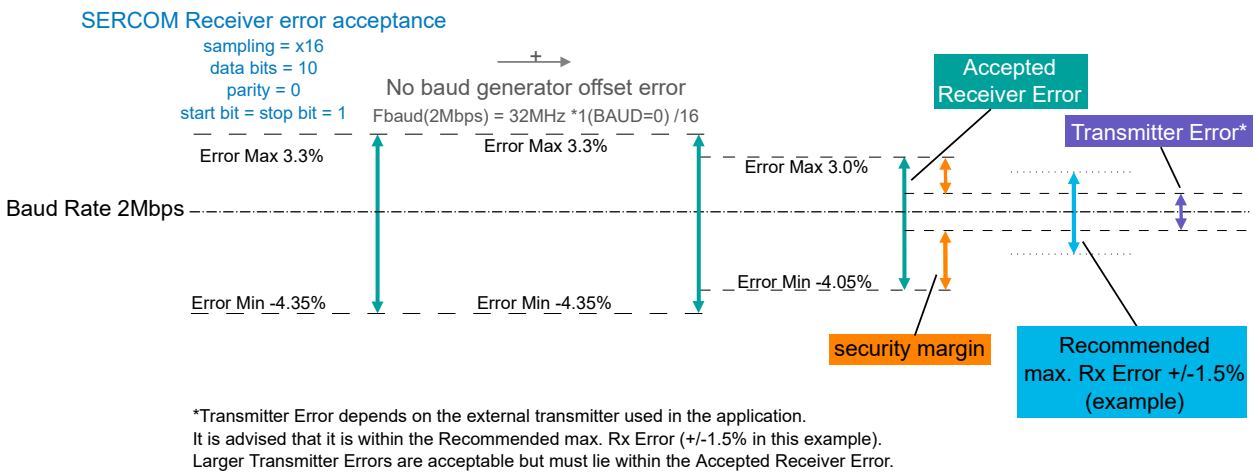
The recommended maximum Rx Error assumes that the receiver and transmitter equally divide the maximum total error. Its connection to the SERCOM Receiver error acceptance is depicted in this figure:

**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**



**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.

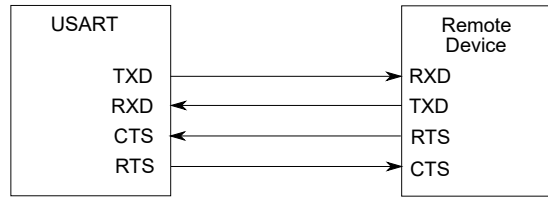
If *odd parity* is selected (CTRLB.PMODE=1), the Parity bit of an outgoing frame is '1' if the data contains an even number of bits that are '0', making the total number of '1' odd.

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

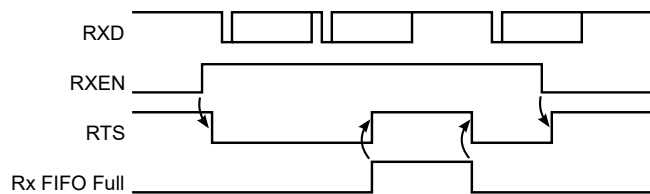


Hardware handshaking is only available in the following configuration:

- USART with internal clock (CTRLA.MODE=1),
- Asynchronous mode (CTRLA.CMODE=0), and
- Flow control pinout (CTRLA.TXPO=2).

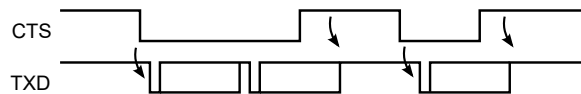
When the receiver is disabled or the receive FIFO is full, the receiver will drive the RTS pin high. This notifies the remote device to stop transfer after the ongoing transmission. Enabling and disabling the receiver by writing to CTRLB.RXEN will set/clear the RTS pin after a synchronization delay. 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

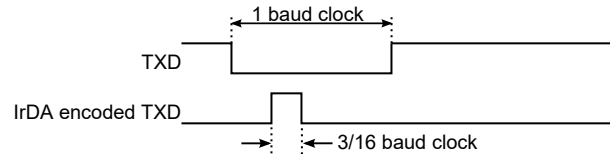


### 30.6.3.3 IrDA Modulation and Demodulation

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.

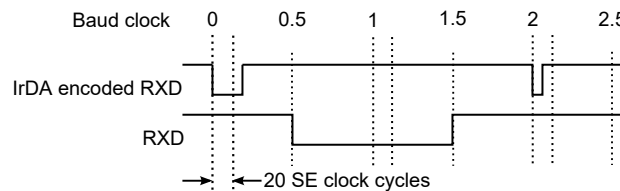
**Figure 30-12.** IrDA Transmit Encoding

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/LIN Responder

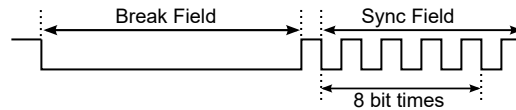
Break character detection and auto-baud are available in this configuration:

- 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.

The auto-baud follows the LIN format. All LIN Frames start with a Break Field followed by a Sync Field.

**Figure 30-14.** LIN Break and Sync Fields



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 30-15.** 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.

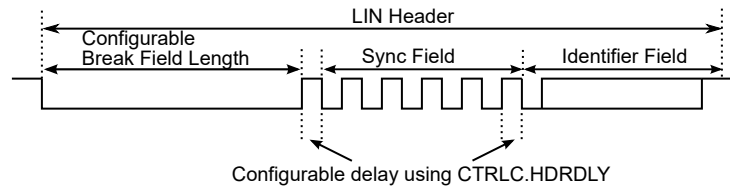
- CTRLB.LINCMD is written to 0x2.
- DATA register written to the identifier. This triggers transmission of the complete header by hardware. First the break field is transmitted. Next, the sync field is transmitted, and finally the identifier is transmitted.

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 30-16.** LIN Header Generation



After header transmission is complete, the responder responds with 1-8 data bytes plus checksum.

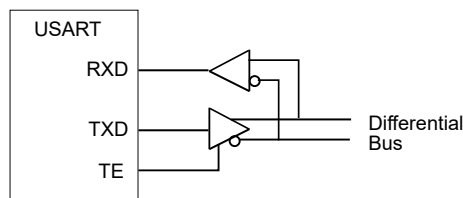
### 30.6.3.6 RS485

RS485 is available with the following configuration:

- USART frame format (CTRLA.FORM = 0x00 or 0x01)
- RS485 pinout (CTRLA.TXPO=0x3).

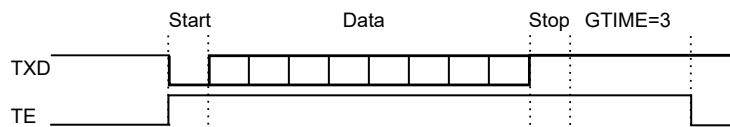
The RS485 feature enables control of an external line driver as shown in the figure below. While operating in RS485 mode, the transmit enable pin (TE) is driven high when the transmitter is active.

**Figure 30-17.** RS485 Bus Connection



The TE pin will remain high for the complete frame including stop bit(s). If a Guard Time is programmed in the Control C register (CTRLC.GTIME), the line will remain driven after the last character completion. The following figure shows a transfer with one stop bit and CTRLC.GTIME=3.

**Figure 30-18.** Example of TE Drive with Guard Time



The Transmit Complete interrupt flag (INTFLAG.TXC) will be raised after the guard time is complete and TE goes low.

### 30.6.3.7 ISO 7816 for Smart Card Interfacing

The SERCOM USART features an ISO/IEC 7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO 7816 link. Both T=0 and T=1 protocols defined by the ISO 7816 specification are supported.

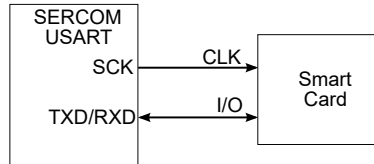
ISO 7816 is available with the following configuration:

- ISO 7816 format (CTRLA.FORM = 0x07)
- Inverse transmission and reception (CTRLA.RXINV=1 and CTRLA.TXINV=1)
- Single bidirectional data line (CTRLA.TXPO and CTRLA.RXPO configured to use the same data pin)
- Even parity (CTRLB.PMODE=0)

- 8-bit character size (CTRLB.CHSIZE=0)
- T=0 (CTRLA.CMODE=1) or T=1 (CTRLA.CMODE=0)

ISO 7816 is a half duplex communication on a single bidirectional line. The USART connects to a smart card as shown below. The output is only driven when the USART is transmitting. The USART is considered as the host of the communication as it generates the clock.

**Figure 30-19.** Connection of a Smart Card to the SERCOM USART



ISO 7816 characters are specified as 8 bits with even parity. The USART must be configured accordingly.

The USART cannot operate concurrently in both receiver and transmitter modes as the communication is unidirectional. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO 7816 mode may lead to unpredictable results.

The ISO 7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value (CTRLA.RXINV=1 and CTRLA.TXINV=1).

### Protocol T=0

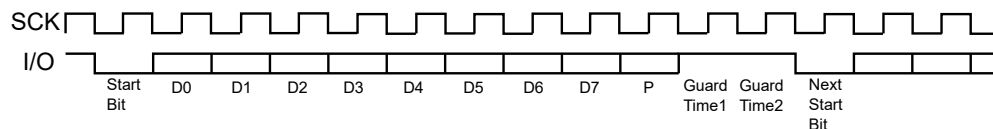
In T=0 protocol, a character is made up of:

- one start bit,
- eight data bits,
- one parity bit
- and one guard time, which lasts two bit times.

The transfer is synchronous (CTRLA.CMODE=1). The transmitter shifts out the bits and does not drive the I/O line during the guard time. Additional guard time can be added by programming the Guard Time (CTRLC.GTIME).

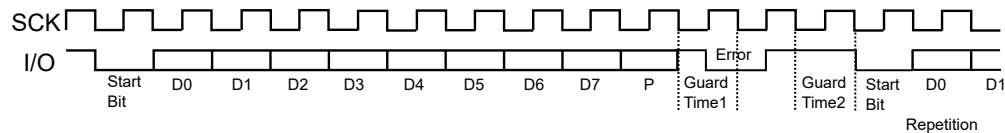
If no parity error is detected, the I/O line remains during the guard time and the transmitter can continue with the transmission of the next character, as shown in the following figure.

**Figure 30-20.** T=0 Protocol without Parity Error



If a parity error is detected by the receiver, it drives the I/O line to 0 during the guard time, as shown in the next figure. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time, which lasts 1 bit time.

**Figure 30-21.** T=0 Protocol with Parity Error



When the USART is the receiver and it detects a parity error, the parity error bit in the Status Register (STATUS.PERR) is set and the character is not written to the receive FIFO.

### Receive Error Counter

The receiver also records the total number of errors (receiver parity errors and NACKs from the remote transmitter) up to a maximum of 255. This can be read in the Receive Error Count (RXERRCNT) register. RXERRCNT is automatically cleared on read.

### Receive NACK Inhibit

The receiver can also be configured to inhibit error generation. This can be achieved by setting the Inhibit Not Acknowledge (CTRLC.INACK) bit. If CTRLC.INACK is 1, no error signal is driven on the I/O line even if a parity error is detected. Moreover, if CTRLC.INACK is set, the erroneous received character is stored in the receive FIFO, and the STATUS.PERR bit is set. Inhibit not acknowledge (CTRLC.INACK) takes priority over disable successive receive NACK (CTRLC.DSNACK).

### Transmit Character Repetition

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next character. Repetition is enabled by writing the Maximum Iterations register (CTRLC.MAXITER) to a non-zero value. The USART repeats the character the number of times specified in CTRLC.MAXITER.

When the USART repetition number reaches the programmed value in CTRLC.MAXITER, the STATUS.ITER bit is set and the internal iteration counter is reset. If the repetition of the character is acknowledged by the receiver before the maximum iteration is reached, the repetitions are stopped and the iteration counter is cleared.

### Disable Successive Receive NACK

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the Disable Successive NACK bit (CTRLC.DSNACK). The maximum number of NACKs transmitted is programmed in the CTRLC.MAXITER field. As soon as the maximum is reached, the character is considered as correct, an acknowledge is sent on the line, the STATUS.ITER bit is set and the internal iteration counter is reset.

### Protocol T=1

When operating in ISO7816 protocol T=1, the transmission is asynchronous (CTRL1.CMODE=0) with one or two stop bits. After the stop bits are sent, the transmitter does not drive the I/O line.

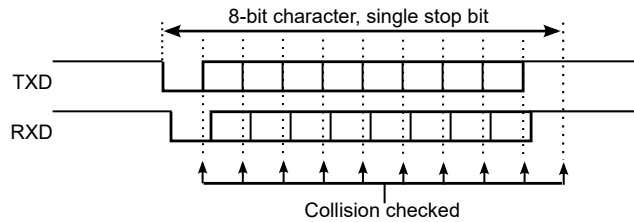
Parity is generated when transmitting and checked when receiving. Parity error detection sets the STATUS.PERR bit, and the erroneous character is written to the receive FIFO. When using T=1 protocol, the receiver does not signal errors on the I/O line and the transmitter does not retransmit.

## 30.6.3.8 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).

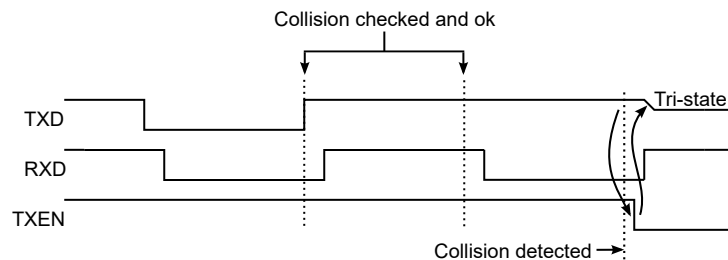
Collision detection is performed for each bit transmitted by comparing the received value with the transmit value, as shown in the figure below. While the transmitter is idle (no transmission in progress), characters can be received on RxD without triggering a collision.

**Figure 30-22.** 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 30-23.** 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.

### 30.6.3.9 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.

### 30.6.3.10 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](#)

#### 30.6.3.11 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.

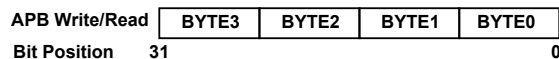
#### 30.6.3.12 32-bit Extension

For better system bus utilization, 32-bit data receive and transmit can be enabled separately by writing to the Data 32-bit bit field in the Control C register (CTRLC.DATA32B). When enabled, writes and/or reads to 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.

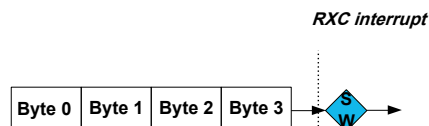
The figure below shows the order of transmit and receive when using 32-bit extension. Bytes are transmitted or received, and stored in order from 0 to 3. Only 8-bit and smaller character sizes are supported. If the character size is less than 8 bits, characters will still be 8-bit aligned within the 32-bit APB write or read. The unused bits within each byte will be zero for received data and unused for transmit data.

**Figure 30-24.** 32-bit Extension Ordering



A receive transaction using 32-bit extension is in the next figure. The Receive Complete flag (INTFLAG.RXC) is raised every four received Bytes. For transmit transactions, the Data Register Empty flag (INTFLAG.DRE) is raised instead of INTFLAG.RXC.

**Figure 30-25.** 32-bit Extension Receive Operation



#### Data Length Configuration

When the Data Length Enable bit field in the Length register (LENGTH.LENEN) is written to 0x1 or 0x2, the Data Length bit (LENGTH.LEN) determines the number of characters to be transmitted or received from 1 to 255.

**Note:** There is one internal length counter that can be used for either transmit (LENGTH.LENEN=0x1) or receive (LENGTH.LENEN=0x2), but not for both simultaneously.

The LENGTH register must be written before the frame begins. If LENGTH.LEN is not a multiple of 4 Bytes, the final INTFLAG.RXC/DRE interrupt will be raised when the last byte is received/sent. The internal length counter is reset when LENGTH.LEN is reached or when LENGTH.LENEN is written to 0x0.

Writing 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 may be lost. Attempting to use the length counter for transmit and receive at the same time will produce unpredictable results.

### 30.6.4 DMA, Interrupts and Events

**Table 30-5.** Module Request for SERCOM USART

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	
Receive Start (RXS)	NA	Yes	
Clear to Send Input Change (CTSIC)	NA	Yes	
Receive Break (RXBRK)	NA	Yes	
Error (ERROR)	NA	Yes	

#### 30.6.4.1 DMA Operation

The USART generates the following DMA requests:

- Data received (RX): The request is set when data is available in the receive FIFO. The request is cleared when DATA is read.
- Data transmit (TX): The request is set when the transmit buffer (TX DATA) is empty. The request is cleared when DATA is written.

#### 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]					RXINV	TXINV	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	SFDE	COLDEN	
		23:16							RXEN	TXEN	
		31:24							LINCMD[1:0]		
0x08	CTRLC	7:0							GTIME[2:0]		
		15:8					HDRDLY[1:0]		BRKLEN[1:0]		
		23:16	MAXITER[2:0]						DSNACK	INACK	
		31:24							DATA32B[1:0]		
0x0C	BAUD	7:0	BAUD[7:0]								
		15:8	BAUD[15:8]								
0x0E	RXPL	7:0	RXPL[7:0]								
0x0F	...	Reserved									
0x13	...	Reserved									
0x14	INTENCLR	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE	
0x15	...	Reserved									
0x16	INTENSET	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE	
0x17	...	Reserved									
0x18	INTFLAG	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE	
0x19	...	Reserved									
0x1A	STATUS	7:0	ITER	TXE	COLL	ISF	CTS	BUFOVF	FERR	PERR	
		15:8									
0x1C	SYNCBUSY	7:0				LENGTH	RXERRCNT	CTRLB	ENABLE	SWRST	
		15:8									
		23:16									
		31:24									
0x20	RXERRCNT	7:0	RXERRCNT[7:0]								
0x21	...	Reserved									
0x22	LENGTH	7:0	LEN[7:0]								
		15:8							LENEN[1:0]		
0x24	...	Reserved									
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	...	Reserved									
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]					RXINV	TXINV	IBON
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
	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
0x2	LIN Commander - Break and sync generation. See LIN Command (CTRLB.LINCMD).
0x3	Reserved
0x4	Auto-baud (LIN Responder) - break detection and auto-baud.
0x5	Auto-baud - break detection and auto-baud with parity
0x6	Reserved
0x7	ISO 7816
0x8-0xF	Reserved

**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.

TXPO	TxD Pin Location	XCK Pin Location (When Applicable)	RTS/TE	CTS
0x0	SERCOM PAD[0]	SERCOM PAD[1]	N/A	N/A
0x1	Reserved			
0x2	SERCOM PAD[0]	N/A	SERCOM PAD[2]	SERCOM PAD[3]
0x3	SERCOM_PAD[0]	SERCOM_PAD[1]	SERCOM_PAD[2]	N/A

**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 10 – RXINV** Receive Data Invert

This bit controls whether the receive data (RxD) is inverted or not.

**Note:** Start, parity and stop bit(s) are unchanged. When enabled, parity is calculated on the inverted data.

This bit is not synchronized.

Value	Description
0	RxD is not inverted.
1	RxD is inverted.

#### Bit 9 – TXINV Transmit Data Invert

This bit controls whether the transmit data (TxD) is inverted or not.

**Note:** Start, parity and stop bit(s) are unchanged. When enabled, parity is calculated on the inverted data.

This bit is not synchronized.

Value	Description
0	TxD is not inverted.
1	TxD is inverted.

#### 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.
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
							LINCMD[1:0]	
Access							R/W	R/W
Reset							0	0
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	SFDE	COLDEN
Access			R/W			R/W	R/W	R/W
Reset			0			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

#### Bits 25:24 – LINCMD[1:0] LIN Command

These bits define the LIN header transmission control. This field is only valid in LIN commander mode (CTRLA.FORM=LIN Commander).

These are strobe bits and will always read back as zero.

These bits are not enable-protected.

Value	Description
0x0	Normal USART transmission.
0x1	Break field is transmitted when DATA is written.
0x2	Break, sync and identifier are automatically transmitted when DATA is written with the identifier.
0x3	Reserved

#### 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 9 – SFDE** Start of Frame Detection Enable

This bit controls whether the start-of-frame detector will wake up the device when a start bit is detected on the RxD line.

This bit is not synchronized.

SFDE	INTENSET.RXS	INTENSET.RXC	Description
0	X	X	Start-of-frame detection disabled.
1	0	0	Reserved
1	0	1	Start-of-frame detection enabled. RXC wakes up the device from all sleep modes.
1	1	0	Start-of-frame detection enabled. RXS wakes up the device from all sleep modes.
1	1	1	Start-of-frame detection enabled. Both RXC and RXS wake up the device from all sleep modes.

**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
							DATA32B[1:0]	
Access							R/W	R/W
Reset							0	0
Bit	23	22	21	20	19	18	17	16
	MAXITER[2:0]						DSNACK	INACK
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
					HDRDLY[1:0]		BRKLEN[1:0]	
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
							GTIME[2:0]	
Access							R/W	R/W
Reset							0	0

#### Bits 25:24 – DATA32B[1:0] Data 32 Bit

These bits configure 32-bit Extension for read and write transactions to the DATA register. When disabled, access is according to CTRLB.CHSIZE.

Value	Description
0x0	DATA reads (for received data) and writes (for transmit data) according to CTRLB.CHSIZE.
0x1	DATA reads according to CTRLB.CHSIZE. DATA writes using 32-bit Extension.
0x2	DATA reads using 32-bit Extension. DATA writes according to CTRLB.CHSIZE.
0x3	DATA reads and writes using 32-bit Extension.

#### Bits 22:20 – MAXITER[2:0] Maximum Iterations

These bits define the maximum number of retransmit iterations.

These bits also define the successive NACKs sent to the remote transmitter when CTRLC.DSNACK is set.

This field is only valid when using ISO7816 T=0 mode (CTRLA.MODE=0x7).

#### Bit 17 – DSNACK Disable Successive Not Acknowledge

This bit controls how many times NACK will be sent on parity error reception.

This bit is only valid in ISO7816 T=0 mode and when CTRLC.INACK=0.

Value	Description
0	NACK is sent on the ISO line for every parity error received.
1	Successive parity errors are counted up to the value specified in CTRLC.MAXITER. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line.

**Bit 16 – INACK** Inhibit Not Acknowledge

This bit controls whether a NACK is transmitted when a parity error is received.

This bit is only valid in ISO7816 T=0 mode.

Value	Description
0	NACK is transmitted when a parity error is received.
1	NACK is not transmitted when a parity error is received.

**Bits 11:10 – HDRDLY[1:0]** LIN Commander Header Delay

These bits define the delay between break and sync transmission in addition to the delay between the sync and identifier (ID) fields when in LIN commander mode (CTRLA.FORM=0x2).

This field is only valid when using the LIN header command (CTRLB.LINCMD=0x2).

Value	Description
0x0	Delay between break and sync transmission is 1 bit time. Delay between sync and ID transmission is 1 bit time.
0x1	Delay between break and sync transmission is 4 bit time. Delay between sync and ID transmission is 4 bit time.
0x2	Delay between break and sync transmission is 8 bit time. Delay between sync and ID transmission is 4 bit time.
0x3	Delay between break and sync transmission is 14 bit time. Delay between sync and ID transmission is 4 bit time.

**Bits 9:8 – BRKLEN[1:0]** LIN Commander Break Length

These bits define the length of the break field transmitted when in LIN commander mode (CTRLA.FORM=0x2).

Value	Description
0x0	Break field transmission is 13 bit times
0x1	Break field transmission is 17 bit times
0x2	Break field transmission is 21 bit times
0x3	Break field transmission is 26 bit times

**Bits 2:0 – GTIME[2:0]** Guard Time

These bits define the guard time when using RS485 mode (CTRLA.FORM=0x0 or CTRLA.FORM=0x1, and CTRLA.TXPO=0x3) or ISO7816 mode (CTRLA.FORM=0x7).

For RS485 mode, the guard time is programmable from 0-7 bit times and defines the time that the transmit enable pin (TE) remains high after the last stop bit is transmitted and there is no remaining data to be transmitted.

For ISO7816 T=0 mode, the guard time is programmable from 2-9 bit times and defines the guard time between each transmitted byte.

### 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  $SE_{per}$ .

$$PULSE \geq (RXPL + 1) \cdot SE_{per}$$

### 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	RXS	RXC	TXC	DRE
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 – 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 3 – RXS Receive Start Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Receive Start Interrupt Enable bit, which disables the Receive Start interrupt.

Value	Description
0	Receive Start interrupt is disabled.
1	Receive Start 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	RXS	RXC	TXC	DRE
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 – 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 3 – RXS Receive Start Interrupt Enable

Writing '0' to this bit has no effect.

Writing '1' to this bit will set the Receive Start Interrupt Enable bit, which enables the Receive Start interrupt.

Value	Description
0	Receive Start interrupt is disabled.
1	Receive Start 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	RXS	RXC	TXC	DRE
Access	R/W		R/W	R/W	R/W	R	R/W	R
Reset	0		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 3 – RXS Receive Start

This flag is cleared by writing '1' to it.

This flag is set when a Start condition is detected on the RxD line and start-of-frame detection is enabled (CTRLB.SFDE is '1').

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Receive Start Interrupt 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
Access	ITER	TXE	COLL	ISF	CTS	BUFOVF	FERR	PERR
Reset	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

#### Bit 7 – ITER Maximum Number of Repetitions Reached

This bit is set when the maximum number of NACK repetitions or retransmissions is met in ISO7816 T=0 mode.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.

#### Bit 6 – TXE Transmitter Empty

When CTRLA.FORM is set to LIN Commander mode, this bit is set when any ongoing transmission is complete and TxDATA is empty.

When CTRLA.FORM is not set to LIN Commander mode, this bit will always read back as zero.

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.

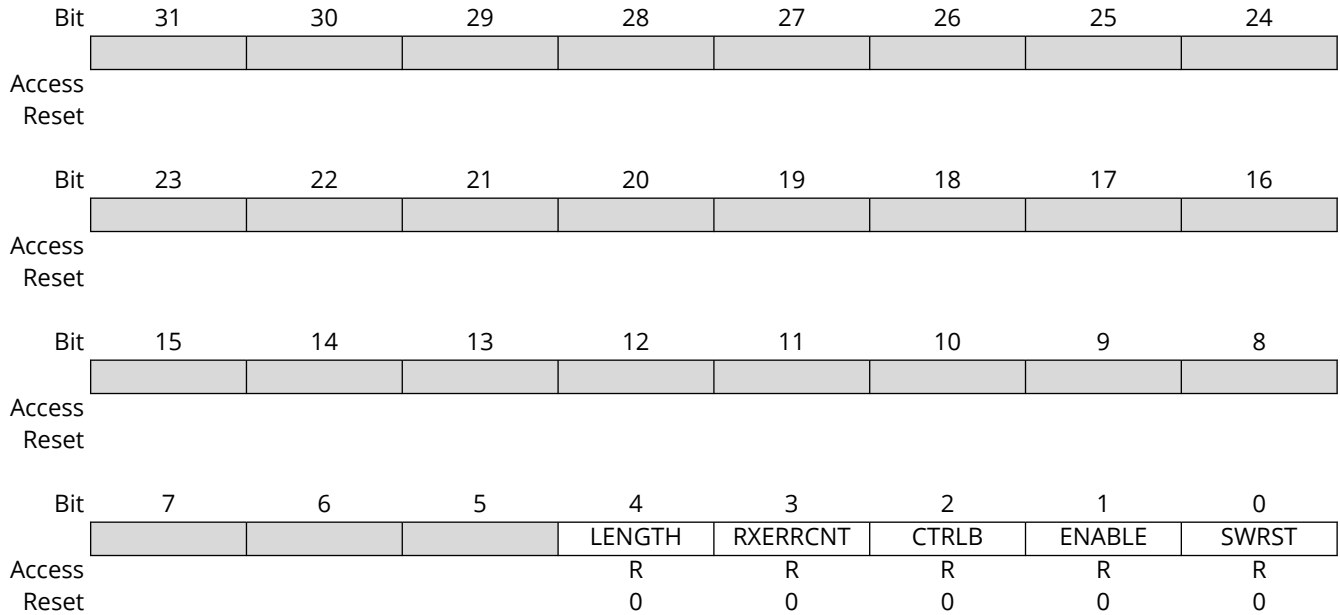
This bit is set if parity checking is enabled (CTRLA.FORM is 0x1, 0x5, or 0x7) and a parity error is detected.

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 4 - LENGTH LENGTH Synchronization Busy

Writing to the LENGTH register requires synchronization. When writing to LENGTH, SYNCBUSY.LENGTH will be set until synchronization is complete. If the LENGTH register is written to while SYNCBUSY.LENGTH is asserted, an APB error is generated.

Value	Description
0	LENGTH synchronization is not busy.
1	LENGTH synchronization is busy.

#### Bit 3 - RXERRCNT Receive Error Count Synchronization Busy

The RXERRCNT register is automatically synchronized to the APB domain upon error. When returning from sleep, this bit will be raised until the new value is available to be read.

Value	Description
0	RXERRCNT synchronization is not busy.
1	RXERRCNT synchronization is busy.

#### 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 Receive Error Count

**Name:** RXERRCNT  
**Offset:** 0x20  
**Reset:** 0x00  
**Property:** Read-Synchronized

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

**Bits 7:0 – RXERRCNT[7:0]** Receive Error Count

This register records the total number of parity errors and NACK errors combined in ISO7816 mode (CTRLA.FORM=0x7).

This register is automatically cleared on read.

### 30.8.12 Length

**Name:** LENGTH  
**Offset:** 0x22  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
							LENEN[1:0]	
Access							R/W	R/W
Reset							0	0
Bit	7	6	5	4	3	2	1	0
	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

#### Bits 9:8 – LENEN[1:0] Data Length Enable

In 32-bit Extension mode, this bit field configures the length counter either for transmit or receive transactions.

Value	Description
0x0	Length counter disabled
0x1	Length counter enabled for transmit
0x2	Length counter enabled for receive
0x3	Reserved

#### Bits 7:0 – LEN[7:0] Data Length

In 32-bit Extension mode, this bit field configures the data length after which the flags INTFLAG.RXC or INTFLAG.DRE are raised.

Value	Description
0x00	Reserved if LENEN=0x1 or LENEN=0x2
0x01-0xFF	Data Length



### 30.8.13 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

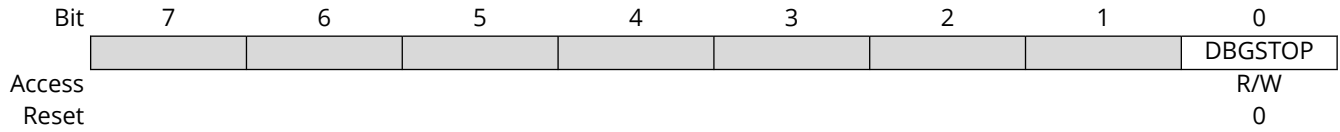
Reading these bits will return the contents of the Receive Data register. The register must be read only when the Receive Complete Interrupt Flag bit in the Interrupt Flag Status and Clear register (INTFLAG.RXC) is set. The status bits in STATUS must be read before reading the DATA value in order to 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.

### 30.8.14 Debug Control

**Name:** DBGCTRL  
**Offset:** 0x30  
**Reset:** 0x00  
**Property:** PAC Write-Protection



#### 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](#)

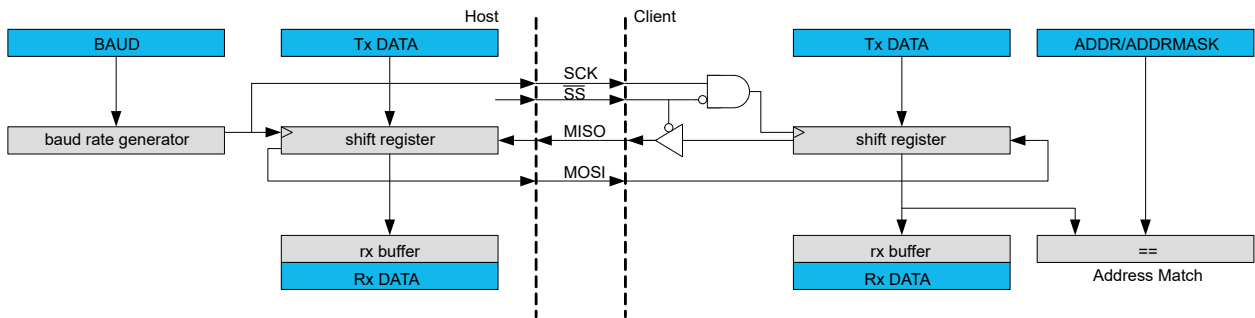
### 31.2 Features

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
- 32-bit extension for better system bus utilization
- Host operation:
  - Serial clock speed up to half the system clock
  - 8-bit clock generator
  - Hardware controlled  $\overline{SS}$
  - Optional inter-character spacing
- 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.MSEN) is '1'.

Table 31-2. SPI Pin Configuration

Pin	Host SPI	Client SPI
MOSI	Output	Input
MISO	Input	Output
SCK	Output	Input
$\overline{SS}$	Output (CTRLB.MSEN = 1)	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

[31.8.13. 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.

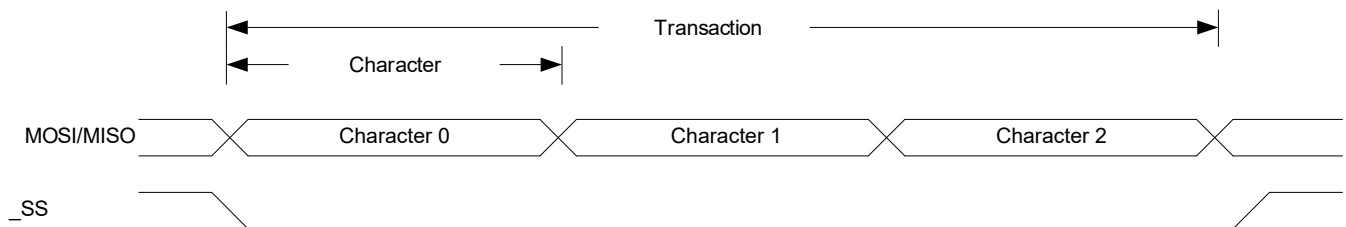
The SPI can operate as host or client. As host, the SPI initiates and controls all data transactions. The SPI is single buffered for transmitting and double buffered for receiving.

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 if multiple clients are connected to the bus. The SPI select line can be wired low if there is only one SPI client on the bus. 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 is 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 (CTRLA.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**

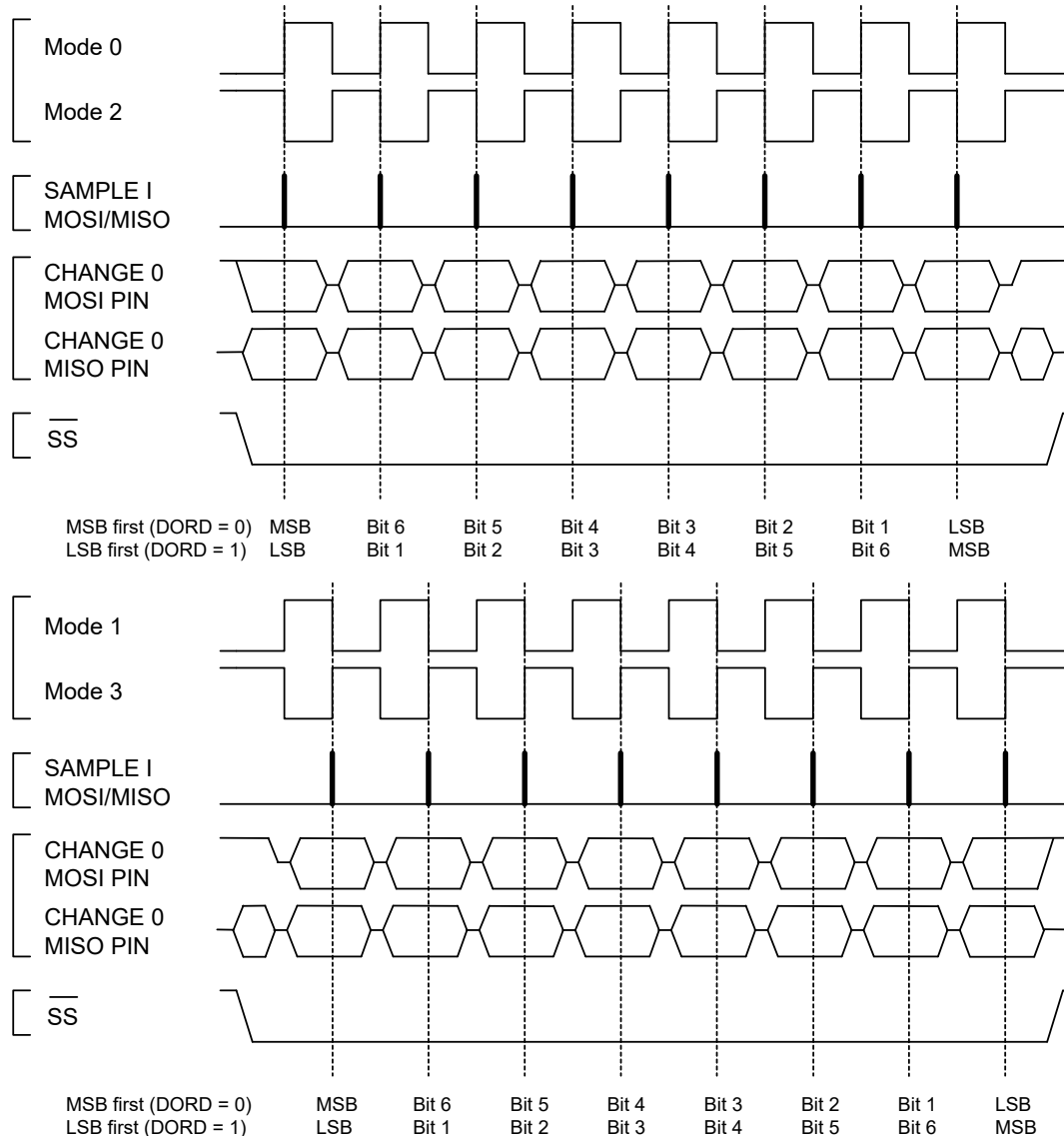
Mode	CPOL	CPHA	Leading Edge	Trailing Edge
0	0	0	Rising, sample	Falling, setup
1	0	1	Rising, setup	Falling, sample
2	1	0	Falling, sample	Rising, setup
3	1	1	Falling, setup	Rising, sample

**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 Client Enable Select (CTRLB.MSSEN) is '1', hardware will control the  $\overline{SS}$  line.



When Host Client 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. Then 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 Client 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.

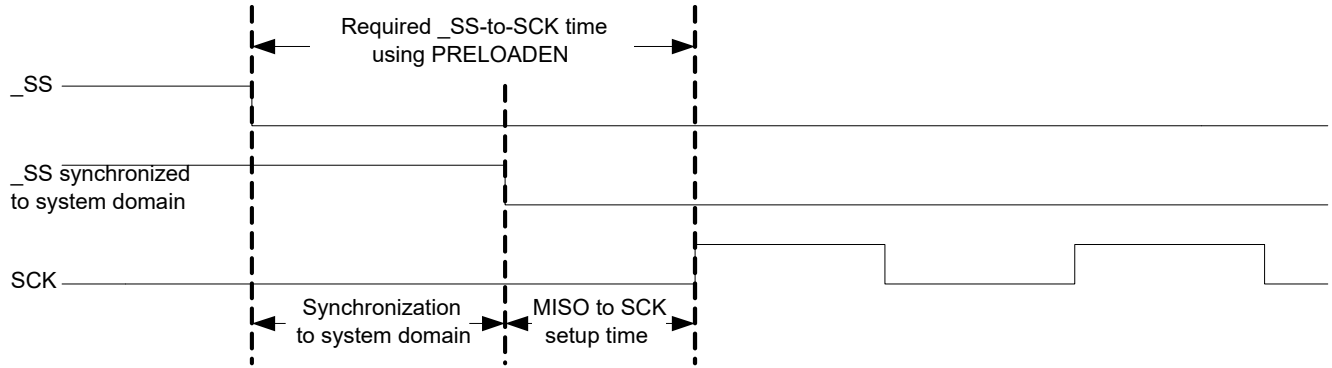
Preloading can be used to preload data into the shift register while  $\overline{SS}$  is high: this eliminates sending a dummy character when starting a transaction. If the shift register is not preloaded, the current contents of the shift register will be shifted out.

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



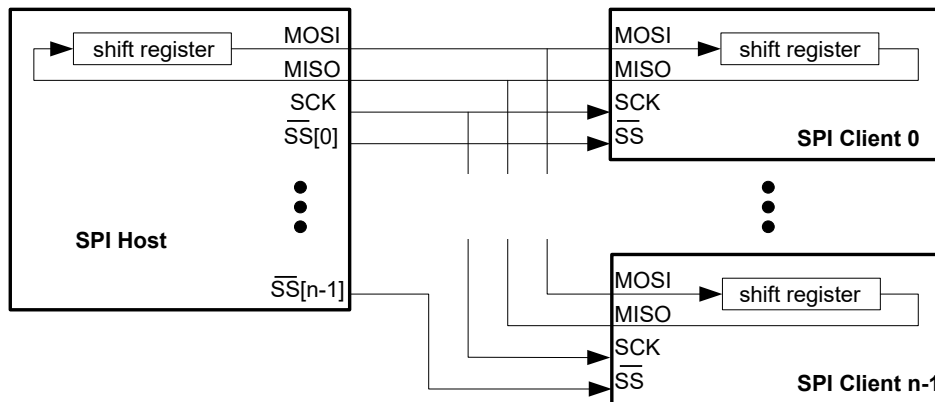
**Related Links**

[43. Electrical Characteristics](#)

**31.6.3.3 Host with Several Clients**

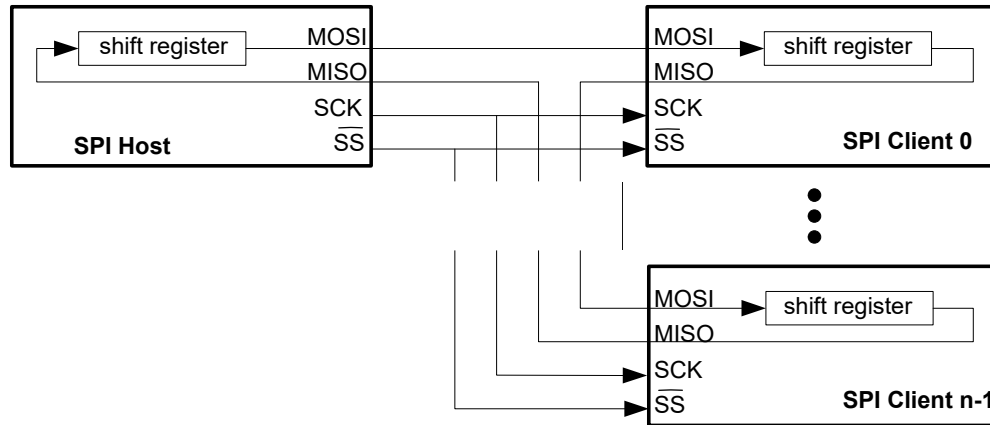
Host with multiple clients in parallel is only available when Host SPI Select Enable (CTRLB.MSEN) is set to zero and hardware  $\overline{SS}$  control is disabled. If the bus consists of several SPI clients, a SPI host can use general purpose I/O pins to control the  $\overline{SS}$  line to each of 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. Depending on the Host SPI Select Enable bit (CTRLB.MSEN), the  $\overline{SS}$  line can be controlled either by hardware or user software and normal GPIO.

Figure 31-6. Multiple Clients in Series



### 31.6.3.4 Loop-Back Mode

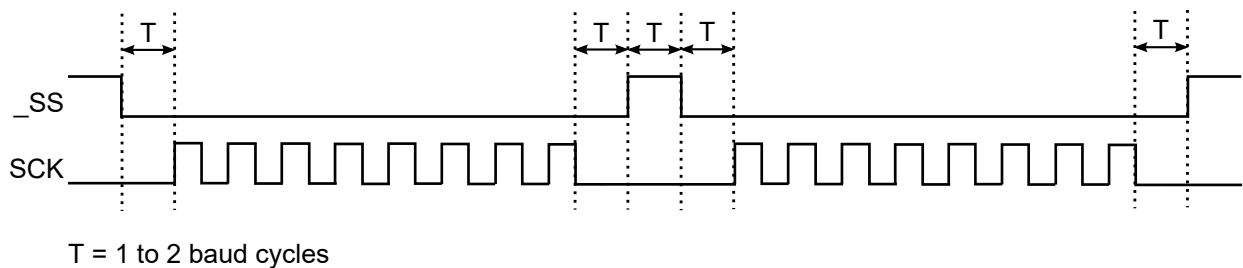
For Loop-back mode, configure the Data In Pinout (CTRLA.DIPO) and Data Out Pinout (CTRLA.DOPO) to use the same data pins for transmit and receive. The loop-back is through the pad, so the signal is also available externally.

### 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  $\overline{SS}$



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

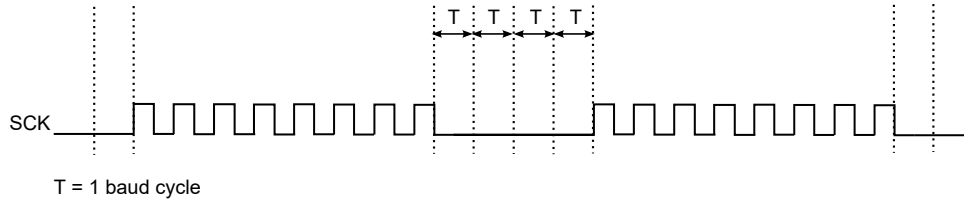
In Client mode, the SPI can wake the CPU when the SPI Select ( $\overline{SS}$ ) goes low. When the SPI Select Low Detect is enabled (CTRLB.SSDE=1), a high-to-low transition will set the SPI Select Low Interrupt flag (INTFLAG.SSL) and the device will wake-up if applicable.

### 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

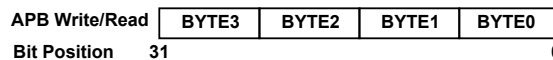
For better system bus utilization, 32-bit data receive and transmit can be enabled by writing to the Data 32-bit bit field in the Control C register (CTRLC.DATA32B=1). When enabled, write and read transaction to/from the DATA register are 32 bit in size.

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.

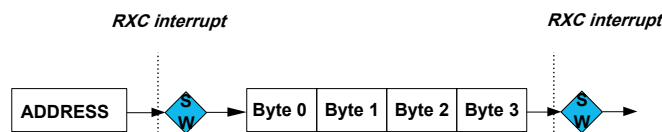
**Figure 31-9.** 32-bit Extension Byte Ordering



### 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:

- Data received (RX): The request is set when data is available in the receive FIFO. The request is cleared when DATA is read.
- Data transmit (TX): The request is set when the transmit buffer (TX DATA) is empty. The request is cleared when DATA is written.

### 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\_SERCOMx\_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]		MSEN			SSDE		
		23:16						RXEN		
		31:24								
0x08	CTRLC	7:0			ICSPACE[5:0]					
		15:8								
		23:16								
		31:24								DATA32B
0x0C	BAUD	7:0	BAUD[7:0]							
0x0D	Reserved									
...										
0x13										
0x14		INTENCLR	7:0	ERROR				SSL	RXC	TXC
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					LENERR			
0x1C	SYNCBUSY	7:0				LENGTH		CTRLB	ENABLE	SWRST
		15:8								
		23:16								
		31:24								
0x20	Reserved									
...										
0x21										
0x22	LENGTH	7:0	LEN[7:0]							
		15:8								LENEN
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	Reserved									
...										
0x2F										
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.MSEN=1, or by a GPIO driven by the application when CTRLB.MSEN=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 $\overline{SS}$ (MSEN = 1)	Host $\overline{SS}$ (MSEN = 0)
0x0	PAD[0]	PAD[1]	PAD[2]	PAD[2]	Any GPIO configured by the application
0x1				Reserved	
0x2	PAD[3]	PAD[1]	PAD[2]	PAD[2]	Any GPIO configured by the application
0x3				Reserved	

**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 CTRLA.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 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.

### 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
Access							R/W	
Reset							0	
Bit	15	14	13	12	11	10	9	8
Access	R/W		R/W	R/W			R/W	
Reset	0		0	0			0	
Bit	7	6	5	4	3	2	1	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.

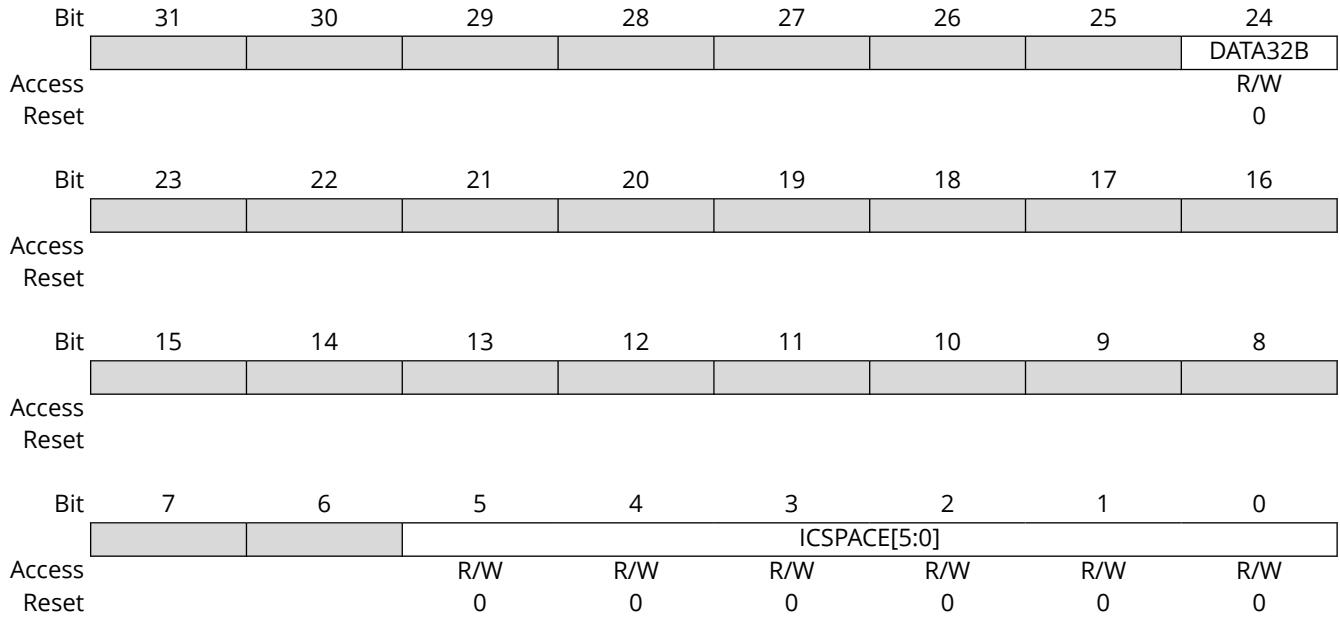
**Bits 2:0 – CHSIZE[2:0]** Character Size

These bits are not synchronized.

CHSIZE[2:0]	Name	Description
0x0	8BIT	8 bits
0x1	9BIT	9 bits
0x2-0x7	—	Reserved

### 31.8.3 Control C

**Name:** CTRLC  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected



#### Bit 24 – DATA32B Data 32 Bit

This bit enables 32-bit Extension for read and write transactions to the DATA register. When disabled, access is according to CTRLB.CHSIZE.

Value	Description
0	Transactions from and to DATA register are 8-bit
1	Transactions from and to DATA register are 32-bit

#### Bits 5:0 – ICSPACE[5:0] Inter-Character Spacing

When non-zero, CTRLC.ICSPACE selects the minimum number of baud cycles the SCK line will not toggle between characters.

Value	Description
0x00	Inter-Character Spacing is disabled
0x01–0x3F	The minimum Inter-Character Spacing

### 31.8.4 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*.



### 31.8.5 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.

### 31.8.6 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.

### 31.8.7 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 and the LENERR 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.

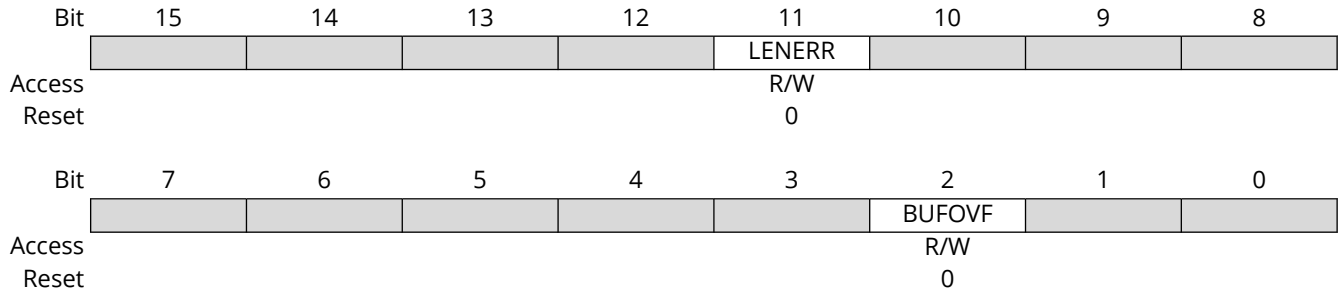
This flag is set when DATA is empty and ready for new data to transmit.

Writing '0' to this bit has no effect.

Writing '1' to this bit has no effect.

### 31.8.8 Status

**Name:** STATUS  
**Offset:** 0x1A  
**Reset:** 0x0000  
**Property:** -



#### Bit 11 – LENERR Transaction Length Error

This bit is set in client mode when the length counter is enabled (LENGTH.LENEN=1) and the transfer length while  $\overline{SS}$  is low is not a multiple of LENGTH.LEN.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.

Value	Description
0	No Length Error has occurred.
1	A Length Error has occurred.

#### 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.

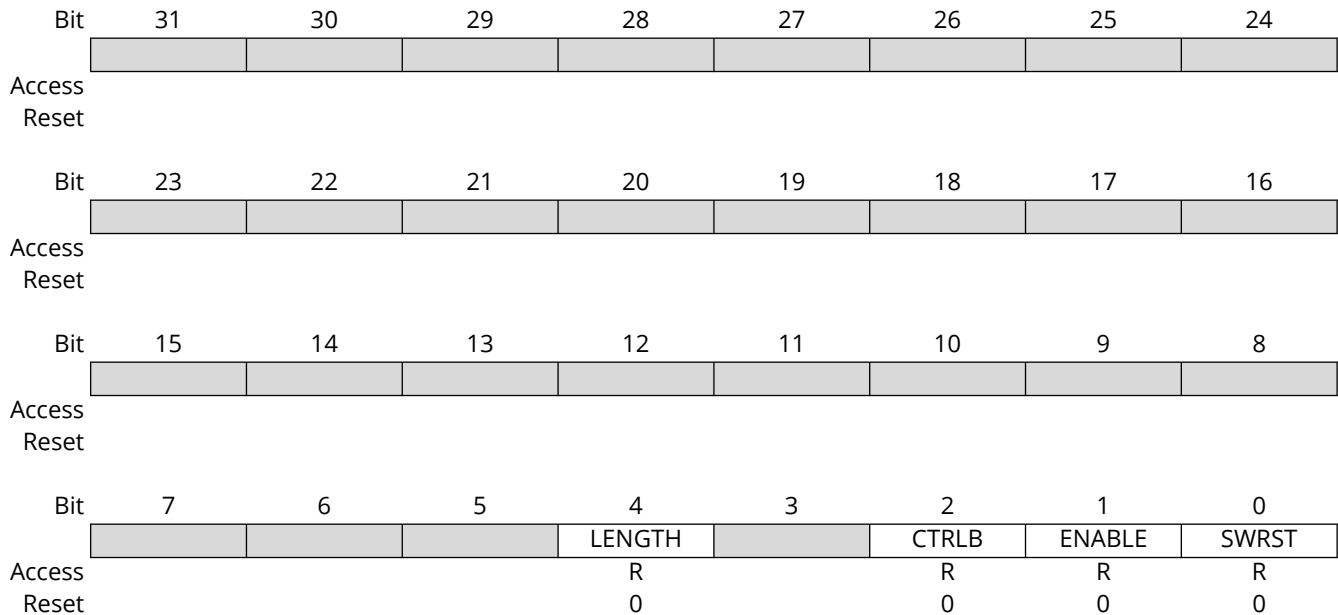
Value	Description
0	No Buffer Overflow has occurred.
1	A Buffer Overflow has occurred.

#### Related Links

[31.8.1. CTRLA](#)

### 31.8.9 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x1C  
**Reset:** 0x00000000  
**Property:** -



#### Bit 4 - LENGTH LENGTH Synchronization Busy

Writing to the LENGTH register requires synchronization. When writing to LENGTH, SYNCBUSY.LENGTH will be set until synchronization is complete. If the LENGTH register is written to while SYNCBUSY.LENGTH is asserted, an APB error is generated.

**Note:** In client mode, the clock is only running during data transfer, so SYNCBUSY.LENGTH will remain asserted until the next data transfer begins.

Value	Description
0	LENGTH synchronization is not busy.
1	LENGTH synchronization is busy.

#### 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

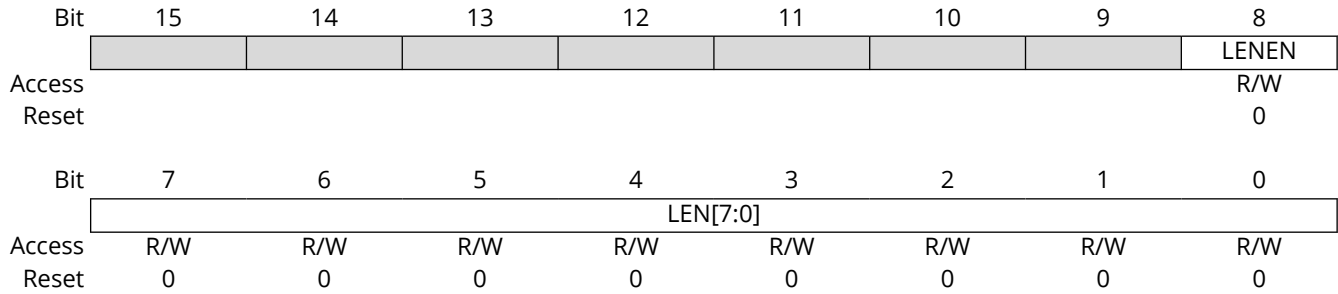
Resetting the SERCOM (CTRLA.SWRST) requires synchronization. Ongoing synchronization is indicated by SYNCBUSY.SWRST=1 until synchronization is complete.

**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.

### 31.8.10 Length

**Name:** LENGTH  
**Offset:** 0x22  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Write-Synchronized



#### Bit 8 – LENEN Data Length Enable

In 32-bit Extension mode, this bit field enables the length counter.

Value	Description
0	Length counter disabled
1	Length counter enabled

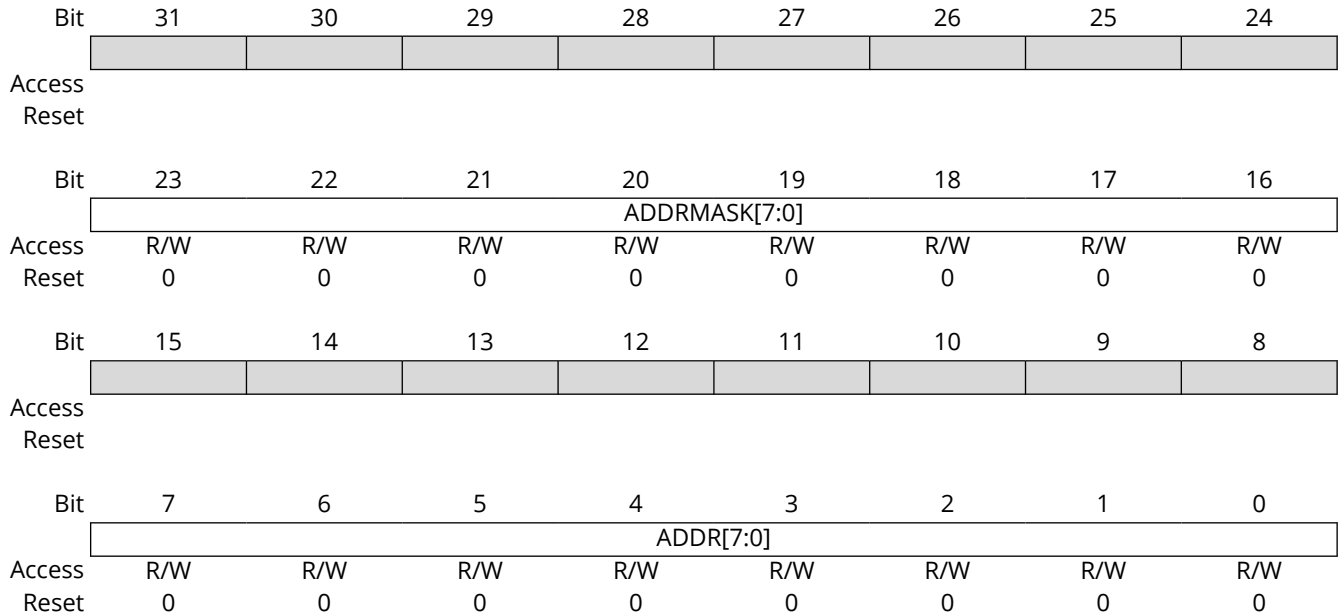
#### Bits 7:0 – LEN[7:0] Data Length

In 32-bit Extension mode, this bit field configures the data length after which the flags INTFLAG.RCX or INTFLAG.DRE are raised.

Value	Description
0x00	Reserved if LENEN=0x1
0x01-0xFF	Data Length

### 31.8.11 Address

**Name:** ADDR  
**Offset:** 0x24  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected



#### 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).



### 31.8.12 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

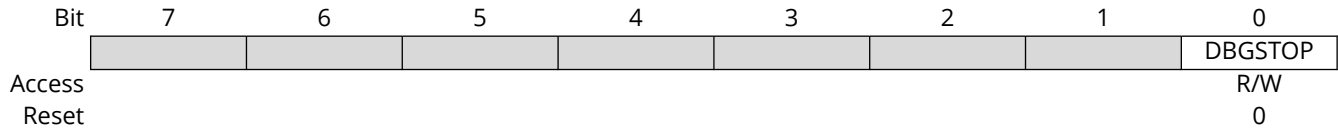
Reading these bits will return the contents of the receive data buffer. The register must be read only when the Receive Complete Interrupt Flag bit in the Interrupt Flag Status and Clear register (INTFLAG.RXC) is set.

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 CTRLB.CHSIZE based on the CTRLC.DATA32B setting.

### 31.8.13 Debug Control

**Name:** DBGCTRL  
**Offset:** 0x30  
**Reset:** 0x00  
**Property:** PAC Write-Protection



#### 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 I<sup>2</sup>C)

### 32.1 Overview

The Inter-Integrated Circuit (I<sup>2</sup>C) interface is one of the available modes in the serial communication interface (SERCOM).

The I<sup>2</sup>C 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 I<sup>2</sup>C host or an I<sup>2</sup>C client. Both host and client have an interface containing a shift register, a transmit buffer and a receive buffer. In addition, the I<sup>2</sup>C host uses the SERCOM baud-rate generator, while the I<sup>2</sup>C client uses the SERCOM address match logic.

**Note:** Traditional Inter-Integrated Circuit (I<sup>2</sup>C) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

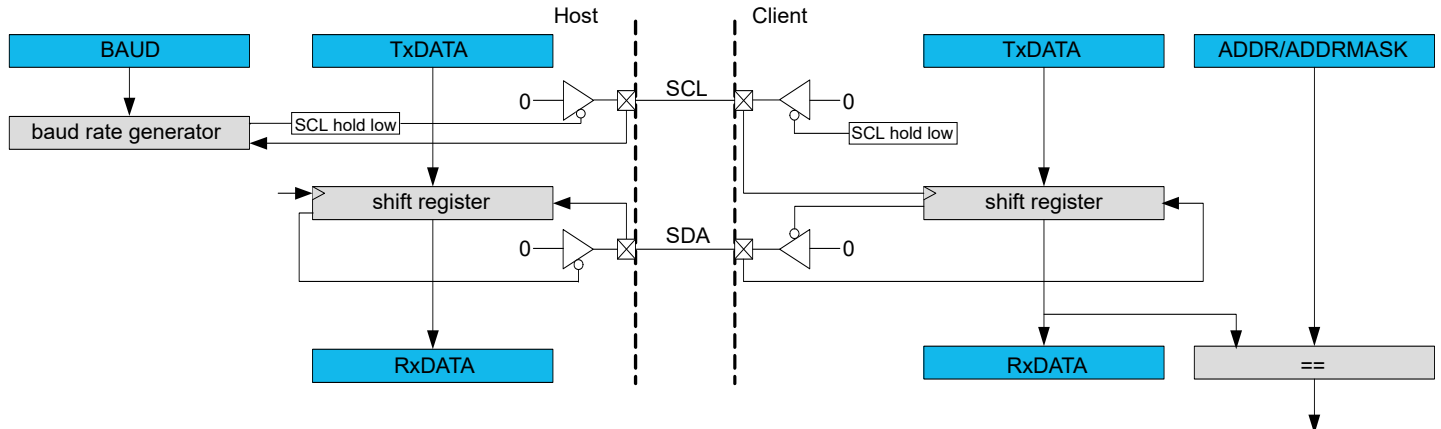
### 32.2 Features

SERCOM I<sup>2</sup>C includes the following features:

- Host or Client Operation
- Can be used with DMA
- Philips I<sup>2</sup>C Compatible
- SMBus Compatible
- PMBus™ Compatible
- Support of 100 kHz and 400 kHz, 1 MHz I<sup>2</sup>C mode
- 32-bit Data Extension for better system bus utilization
- 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. I<sup>2</sup>C 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 I<sup>2</sup>C pins.

### 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 I<sup>2</sup>C mode, the SERCOM controls the direction and value of the I/O pins. In I<sup>2</sup>C 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<sup>2</sup>C 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<sup>2</sup>C.

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

[32.10.12. 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)

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<sup>2</sup>C 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<sup>2</sup>C 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<sup>2</sup>C 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<sup>2</sup>C client or host, the I<sup>2</sup>C 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

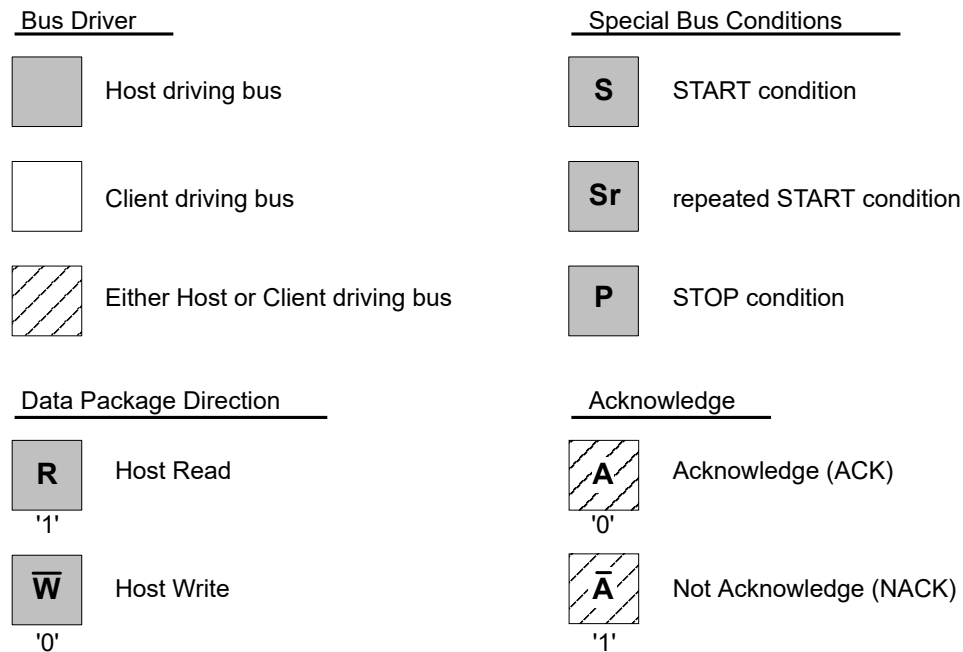
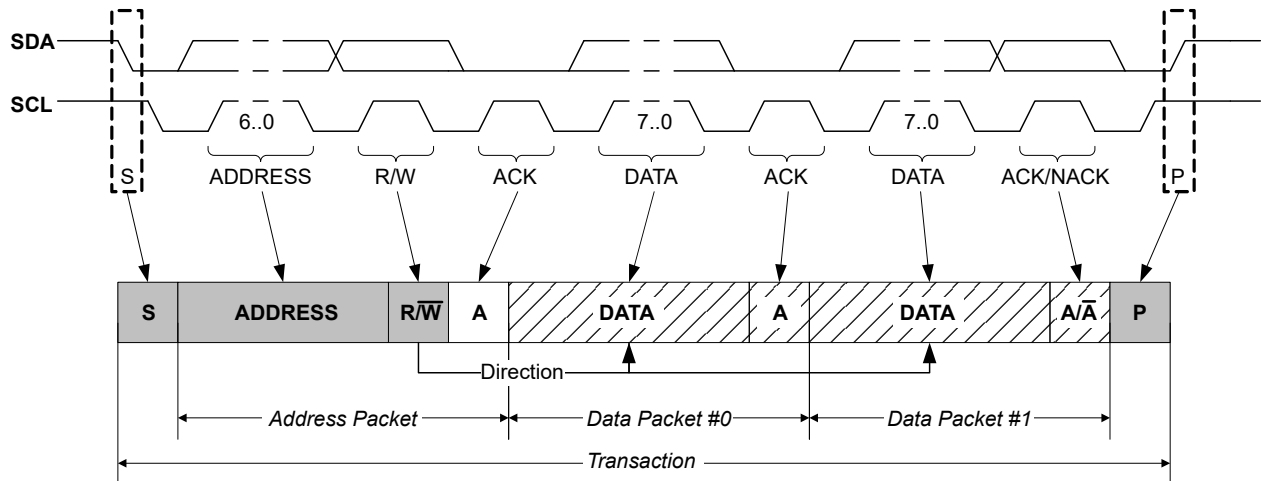


Figure 32-3. Basic I<sup>2</sup>C Transaction Diagram



## 32.6.2 Basic Operation

### 32.6.2.1 Initialization

The following registers are enable-protected, meaning they can be written only when the I<sup>2</sup>C 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 I<sup>2</sup>C is enabled or is being enabled (CTRLA.ENABLE=1), writing to these registers will be discarded. If the I<sup>2</sup>C 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 I<sup>2</sup>C is enabled it must be configured as outlined by the following steps:

1. Select I<sup>2</sup>C 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. In Client mode, the minimum client setup time for the SDA can be selected in the SDA Setup Time bit group in the Control C register (CTRLC.SDASETUP).
4. If desired, enable smart operation by setting the Smart Mode Enable bit in the CTRLB register (CTRLB.SMEN).
5. If desired, enable SCL low time-out by setting the SCL Low Time-Out bit in the Control A register (CTRLA.LOWTOUT).
6. 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

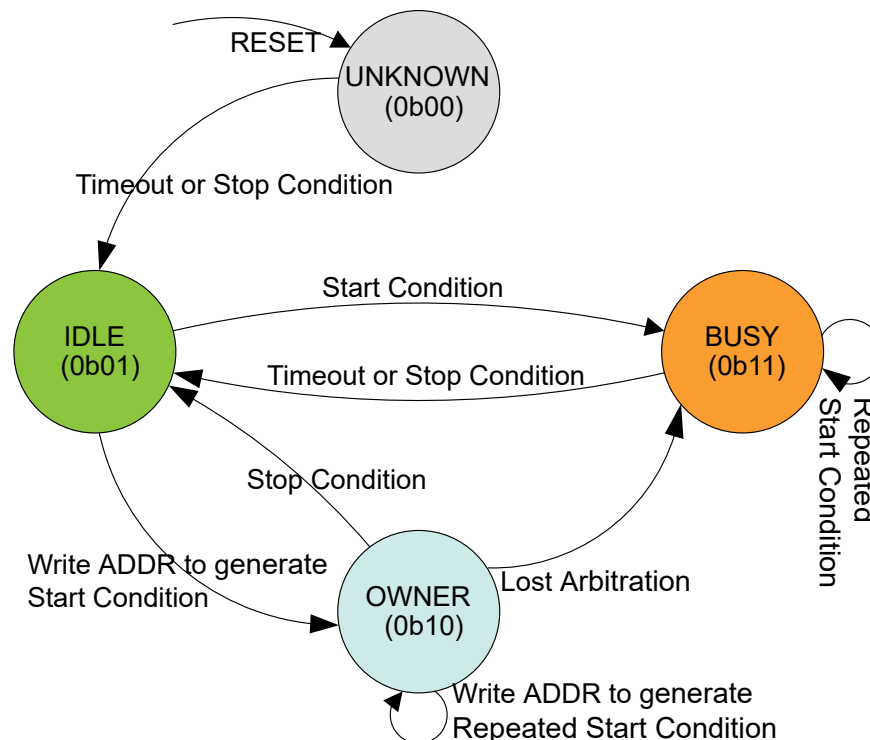
This peripheral is enabled by writing '1' to the Enable bit in the Control A register (CTRLA.ENABLE), and disabled by writing '0' to it.

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 I<sup>2</sup>C Bus State Logic

The Bus state logic includes several logic blocks that continuously monitor the activity on the I<sup>2</sup>C 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 I<sup>2</sup>C host is enabled.

After the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C to hang. If this happens it is possible to recover from this state by a software Reset (CTRLA.SWRST='1').

#### 32.6.2.4 I<sup>2</sup>C Host Operation

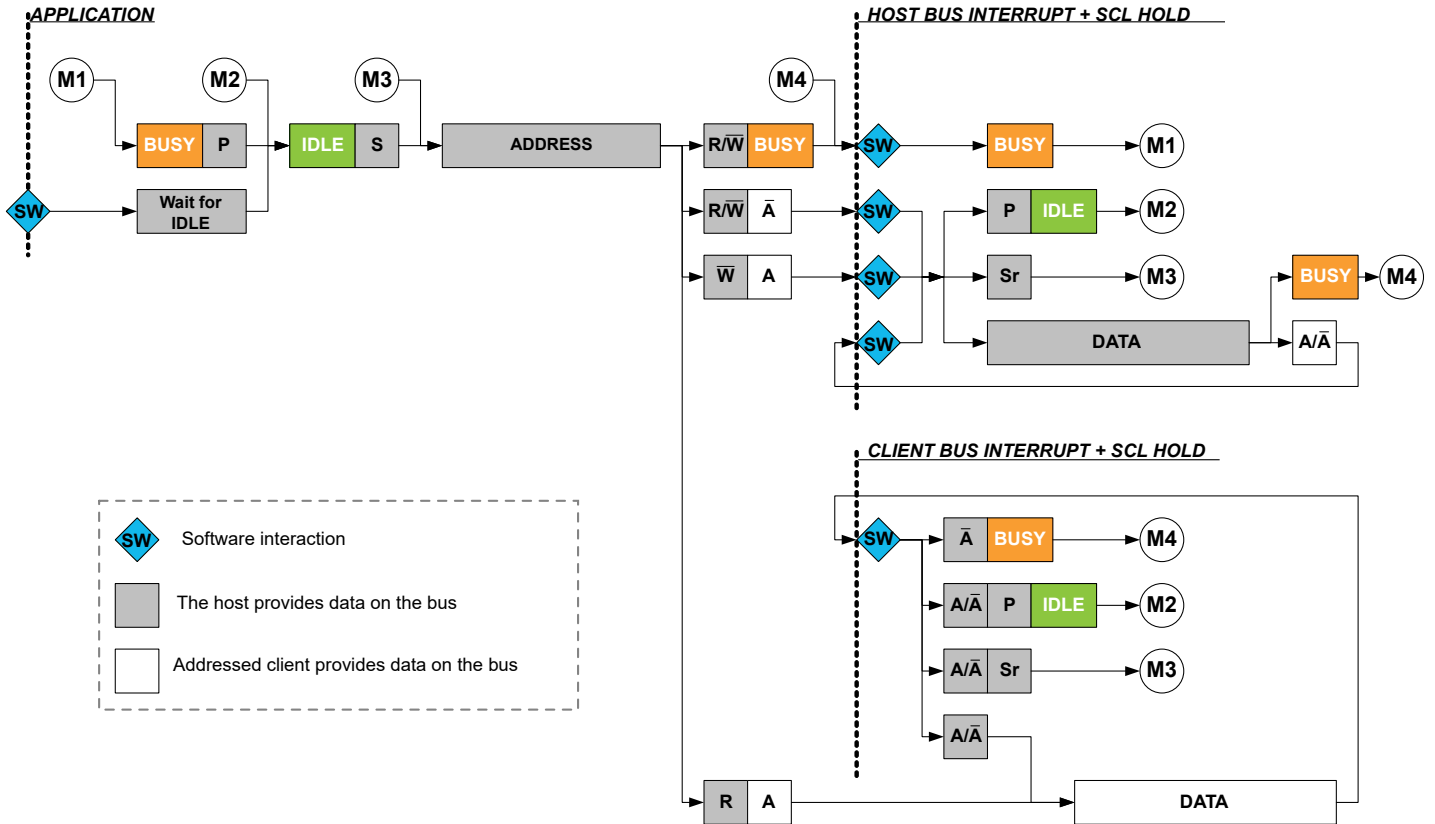
The I<sup>2</sup>C host is byte-oriented and interrupt based. The number of interrupts generated is kept at a minimum by automatic handling of most incidents. The software driver complexity and code size are reduced by auto-triggering of operations, and a Special Smart mode, which can be enabled by the Smart Mode Enable bit in the Control A register (CTRLA.SMEN).

The I<sup>2</sup>C 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 I<sup>2</sup>C host operates according to *I<sup>2</sup>C Host Behavioral Diagram (SCLSM=0)* as shown in the following figure. The circles labeled "Mn" (M1, M2..) indicate the nodes the bus logic can jump to, based on software or hardware interaction.

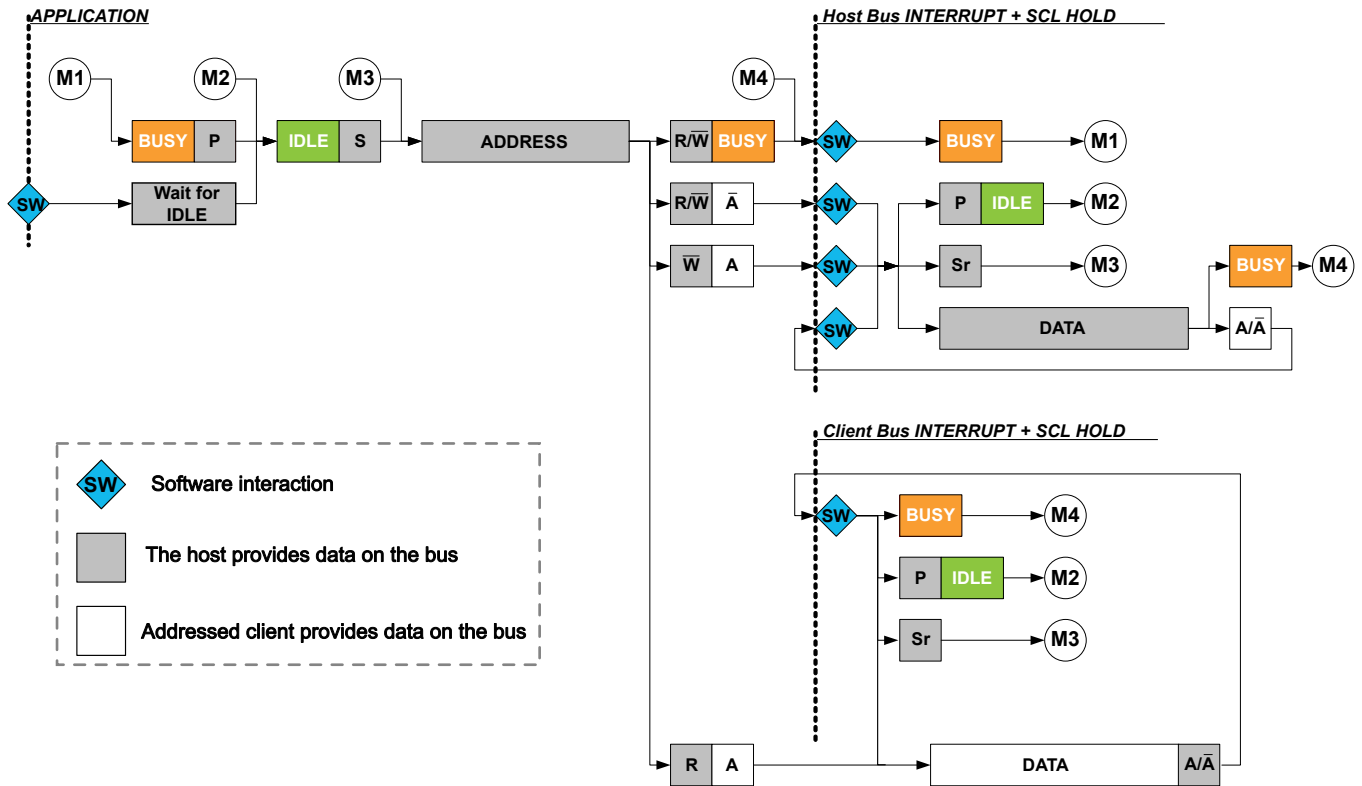
This diagram is used as reference for the description of the I<sup>2</sup>C host operation throughout the document.

Figure 32-5. I<sup>2</sup>C 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. I<sup>2</sup>C Host Behavioral Diagram (SCLSM=1)



### 32.6.2.4.1 Host Clock Generation

The SERCOM peripheral supports several I<sup>2</sup>C bidirectional modes:

- Standard mode (*Sm*) up to 100 kHz
- Fast mode (*Fm*) up to 400 kHz
- Fast mode Plus (*Fm+*) up to 1 MHz

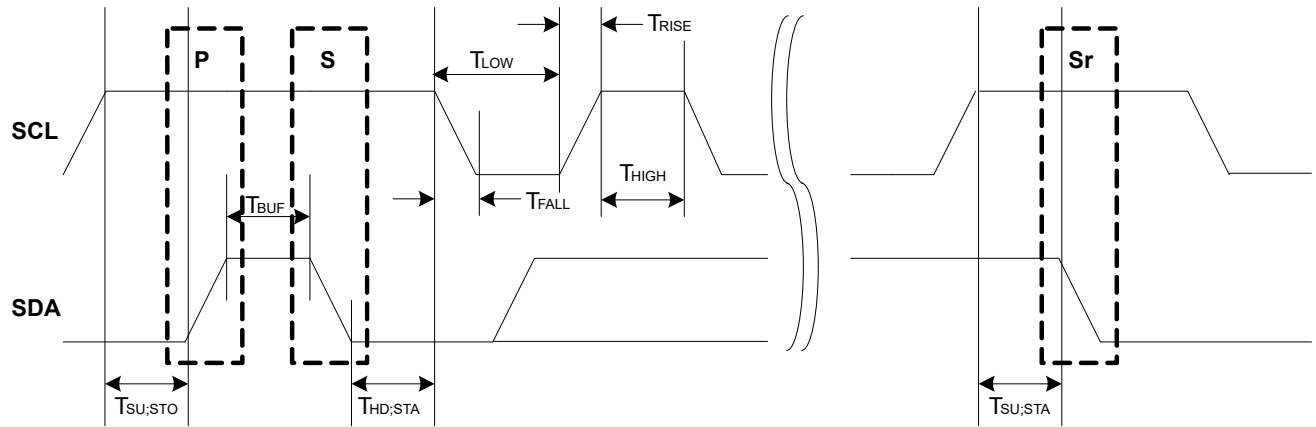
The Host clock configuration for *Sm*, *Fm* and *Fm+* are described in *Clock Generation (Standard-Mode, Fast-Mode and Fast-Mode Plus)* as follows.

#### Clock Generation (Standard-Mode, Fast-Mode, and Fast-Mode Plus)

In I<sup>2</sup>C *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:

$$f_{SCL} = \frac{1}{T_{LOW} + T_{HIGH} + T_{RISE}}$$

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:

$$f_{SCL} = \frac{f_{GCLK}}{10 + 2BAUD + f_{GCLK} \cdot T_{RISE}}$$

When BAUD.BAUDLOW is non-zero, the following formula determines the SCL frequency:

$$f_{SCL} = \frac{f_{GCLK}}{10 + BAUD + BAUDLOW + f_{GCLK} \cdot T_{RISE}}$$

The following formulas can determine the SCL  $T_{LOW}$  and  $T_{HIGH}$  times:

$$T_{LOW} = \frac{BAUDLOW + 5}{f_{GCLK}}$$

$$T_{HIGH} = \frac{BAUD + 5}{f_{GCLK}}$$

**Note:** The I<sup>2</sup>C 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 I<sup>2</sup>C host starts a bus transaction by writing the I<sup>2</sup>C 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 I<sup>2</sup>C host will wait until the bus becomes idle before continuing the operation. When the bus is idle, the I<sup>2</sup>C host will issue a start condition on the bus. The I<sup>2</sup>C host will then transmit an address packet using the address written to ADDR.ADDR. After the address packet has been transmitted by the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C host receives an acknowledge response from the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C host receives an ACK from the I<sup>2</sup>C client, the I<sup>2</sup>C host proceeds to receive the next byte of data from the I<sup>2</sup>C 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 I<sup>2</sup>C operation to continue:

- Let the I<sup>2</sup>C 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 I<sup>2</sup>C host will start transmitting data via the I<sup>2</sup>C bus by writing to DATA.DATA, and monitor continuously for packet collisions.

If a collision is detected, the I<sup>2</sup>C host will lose arbitration and STATUS.ARBLOST will be set. If the transmit was successful, the I<sup>2</sup>C host will receive an ACK bit from the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C host is not allowed to continue transmitting data packets if a NACK is received from the I<sup>2</sup>C client.

#### 32.6.2.4.4 Receiving Data Packets (SCLSM=0)

When INTFLAG.SB is set, the I<sup>2</sup>C host will already have received one data packet. The I<sup>2</sup>C 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 I<sup>2</sup>C 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.

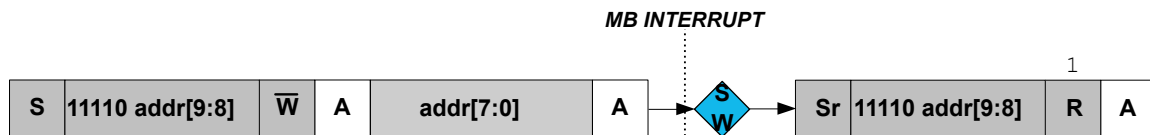
#### 32.6.2.4.6 10-Bit Addressing

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 I<sup>2</sup>C Client Operation

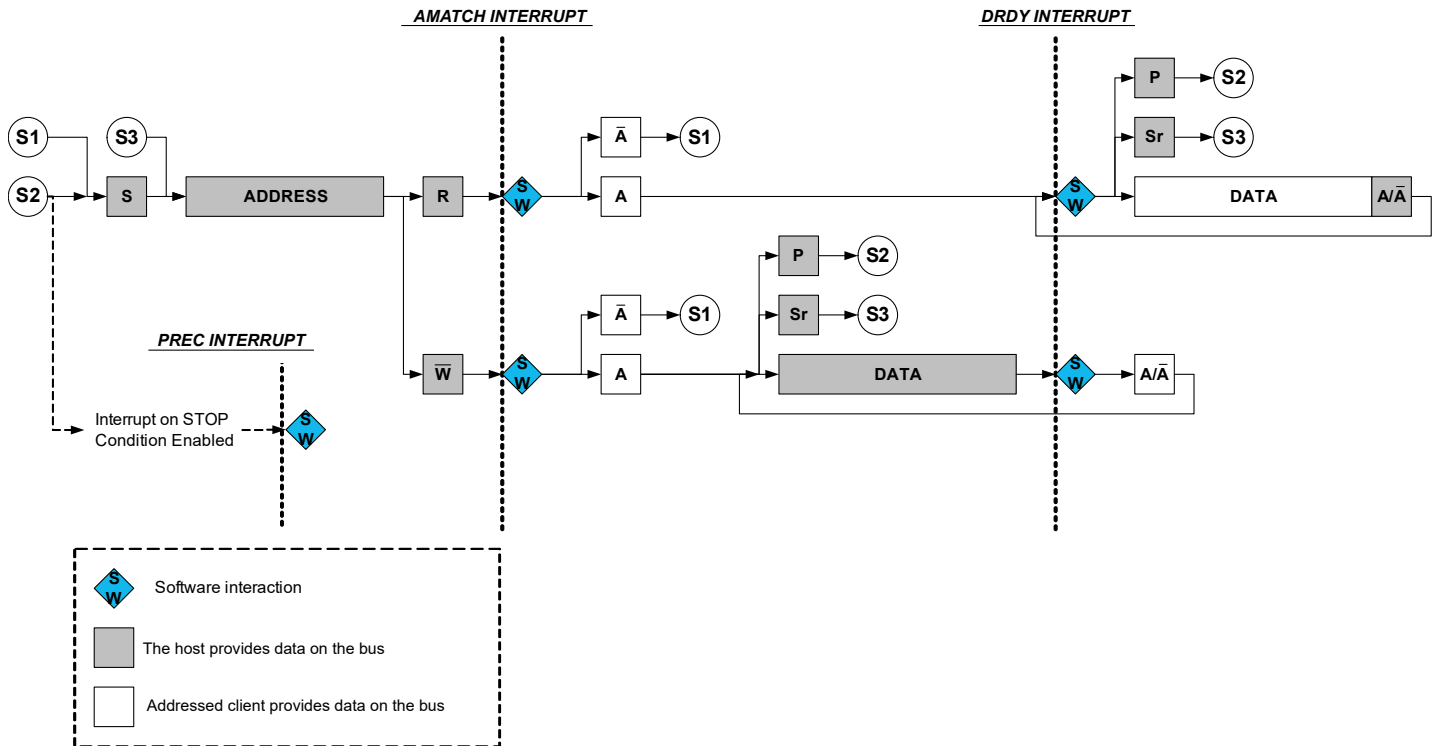
The I<sup>2</sup>C client is byte-oriented and interrupt-based. The number of interrupts generated is kept at a minimum by automatic handling of most events. The software driver complexity and code size are reduced by auto-triggering of operations, and a special smart mode, which can be enabled by the Smart Mode Enable bit in the Control A register (CTRLA.SMEN).

The I<sup>2</sup>C 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 I<sup>2</sup>C client operates according to *I<sup>2</sup>C Client Behavioral Diagram (SCLSM=0)* as shown in the following figure. The circles labelled "Sn" (S1, S2..) indicate the nodes the bus logic can jump to, based on software or hardware interaction.

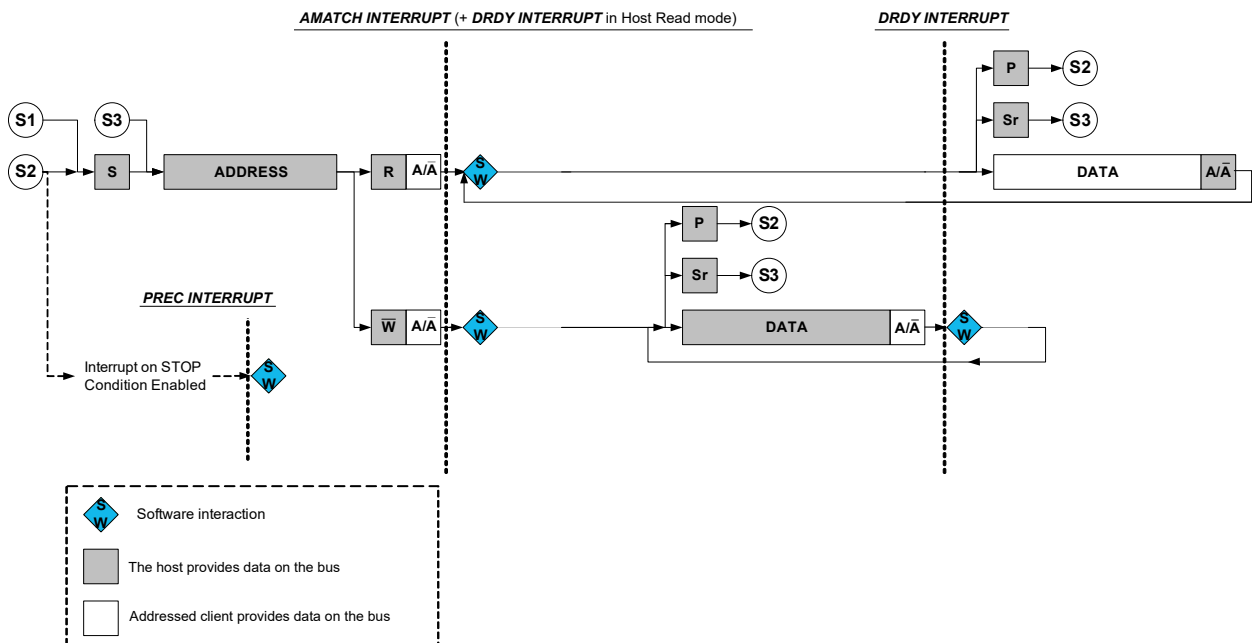
This diagram is used as reference for the description of the I<sup>2</sup>C client operation throughout the document.

Figure 32-9. I<sup>2</sup>C 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 *I<sup>2</sup>C 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. I<sup>2</sup>C Client Behavioral Diagram (SCLSM=1)





### 32.6.2.5.1 Receiving Address Packets (SCLSM=0)

When CTRLA.SCLSM=0, the I<sup>2</sup>C client stretches the SCL line according to [Figure 32-9](#). When the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C client clears INTFLAG.AMATCH. As the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C host read operation. The SCL line is forced low, stretching the bus clock. If an ACK is sent, I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C host write operation. The SCL line is forced low, stretching the bus clock. If an ACK is sent, the I<sup>2</sup>C client will wait for data to be received. Data, repeated start or stop can be received.

If a NACK is sent, the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C client will stretch the SCL line only after an ACK (see [Figure 32-10](#)). When the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C client clears INTFLAG.AMATCH. As the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C host, INTFLAG.AMATCH can be set to '1' to clear it.

### 32.6.2.5.3 Receiving and Transmitting Data Packets

After the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C client must expect a stop or a repeated start to be received. The I<sup>2</sup>C client must release the data line to allow the I<sup>2</sup>C 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 I<sup>2</sup>C client will return to IDLE state.

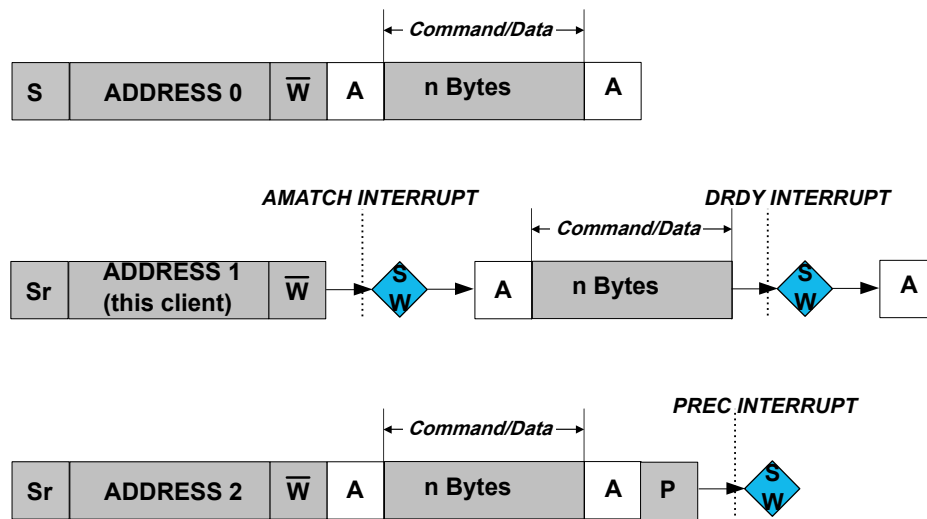
### 32.6.2.5.4 PMBus Group Command

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<sup>2</sup>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<sup>2</sup>C interface also allows for a SMBus compatible SDA hold time.

- $T_{\text{TIMEOUT}}$ : SCL low time of 25..35ms – Measured for a single SCL low period. It is enabled by CTRLA.LOWTOUTEN.
- $T_{\text{LOW:SEXT}}$ : 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.
- $T_{\text{LOW:MEXT}}$ : 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.

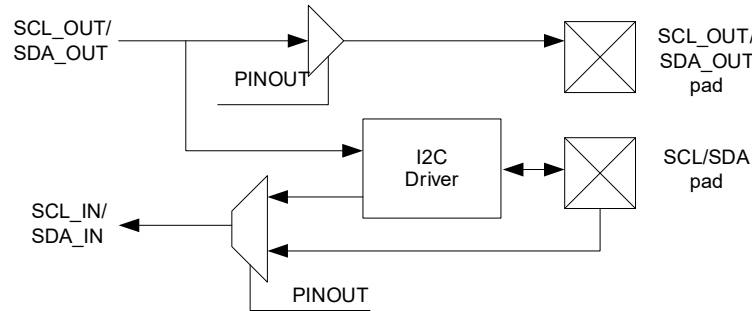
#### 32.6.3.2 Smart Mode

The I<sup>2</sup>C interface has a Smart mode that simplifies application code and minimizes the user interaction needed to adhere to the I<sup>2</sup>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<sup>2</sup>C tri-state drivers are bypassed, and an external I<sup>2</sup>C compliant tri-state driver is needed when connecting to an I<sup>2</sup>C bus.

**Figure 32-12.** I<sup>2</sup>C 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.

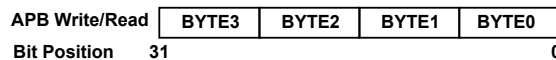
### 32.6.3.5 32-bit Extension

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.

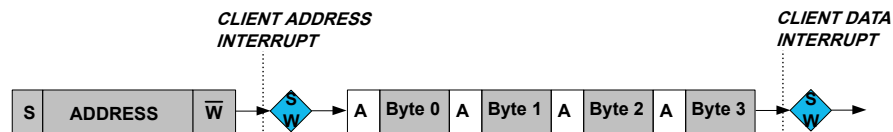
**Figure 32-13.** 32-bit Extension Byte Ordering



### 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<sup>2</sup>C 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 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 is active until the Interrupt flag is cleared, the interrupt is disabled or the I<sup>2</sup>C 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<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C client generates the following requests:

- Write data received (RX): The request is set when host write data is received. The request is cleared when DATA is read.
- Read data needed for transmit (TX): The request is set when data is needed for a host read operation. The request is cleared when DATA is written.

#### 32.6.4.1.2 Host DMA

When using the I<sup>2</sup>C 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 I<sup>2</sup>C host generates the following requests:

- Read data received (RX): The request is set when host read data is received. The request is cleared when DATA is read.
- Write data needed for transmit (TX): The request is set when data is needed for a host write operation. The request is cleared when DATA is written.

### 32.6.4.2 Interrupts

The I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 \(NVIC\)](#)

[32.10.7. INTFLAG](#)

#### 32.6.4.3 Events

Not applicable.

#### 32.6.5 Sleep Mode Operation

##### I<sup>2</sup>C Host Operation

The generic clock (GLK\_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.

##### I<sup>2</sup>C 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					SDASETUP[3:0]				
		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				LENGTH			ENABLE	SWRST	
		15:8									
		23:16									
		31:24									
0x20 ... 0x21	Reserved										
0x22	LENGTH	7:0	LEN[7:0]								
		15:8								LENEN	
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 - I<sup>2</sup>C 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 <a href="#">Figure 32-9</a>
1	SCL stretch only after ACK bit according to <a href="#">Figure 32-10</a>

#### 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<sup>2</sup>C 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
Access						ACKACT	CMD[1:0]	
Reset						R/W	W	W
						0	0	0
Bit	15	14	13	12	11	10	9	8
Access	AMODE[1:0]					AACKEN	GCMD	SMEN
Reset	R/W	R/W				R/W	R/W	R/W
	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.

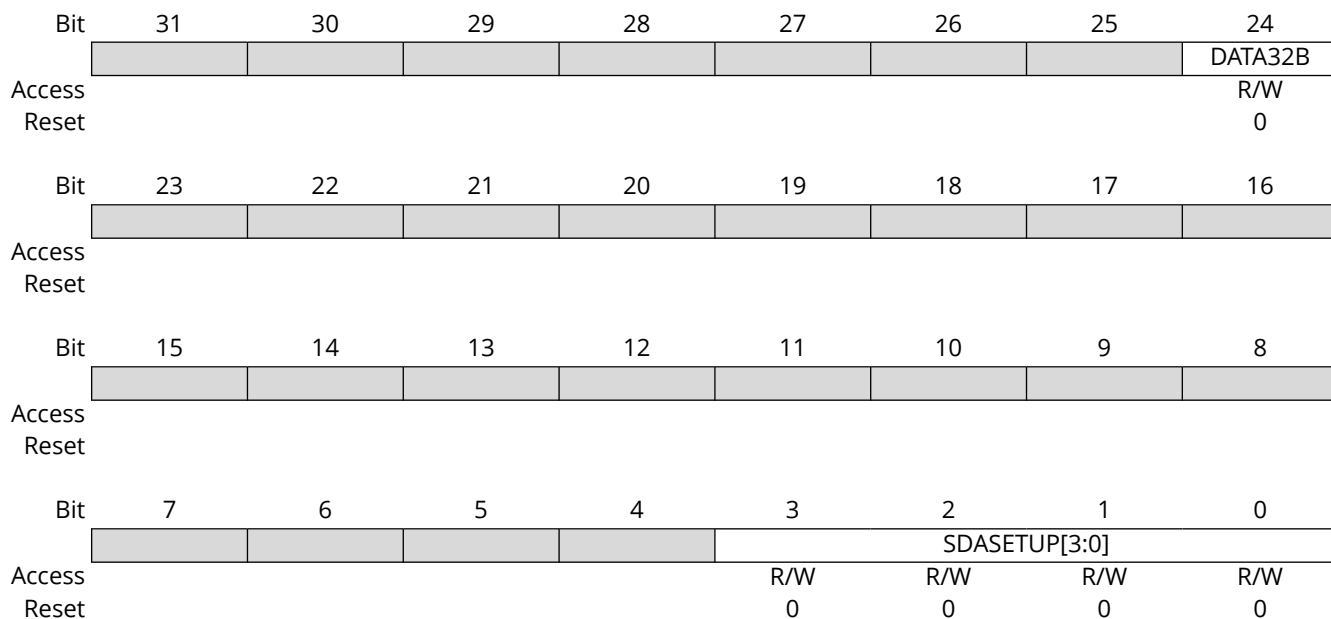
**Bit 8 – SMEN Smart Mode Enable**

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 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

#### Bits 3:0 – SDASETUP[3:0] SDA Setup Time

These bits select the minimum SDA-to-SCL setup time, measured from the release of SDA to the release of SCL:

$$t_{SU:DAT} = (GCLK\_SERCOMx \times APB \text{ period } (PBx\_CLK)) \times (6 + 16 \times SDASETUP)$$

### 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 LENERR, 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<sup>2</sup>C 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<sup>2</sup>C 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.

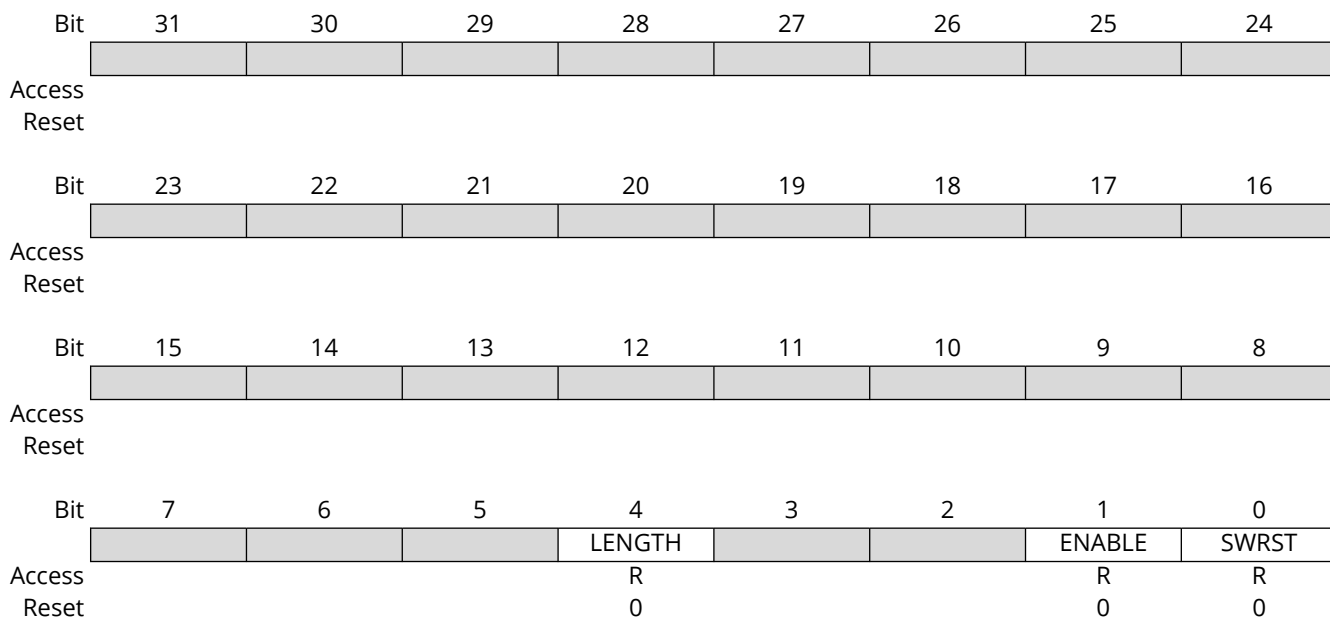
Writing a '0' to this bit has no effect.

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 4 - LENGTH Synchronization Busy

Writing LENGTH requires synchronization. When written, this bit will be set until synchronization is complete. If LENGTH is written while SYNCBUSY.LENGTH is asserted, an APB error will be generated.

**Note:** In client mode, the clock is only running during data transfer, so SYNCBUSY.LENGTH will remain asserted until the next data transfer begins.

Value	Description
0	LENGTH synchronization is not busy.
1	LENGTH 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

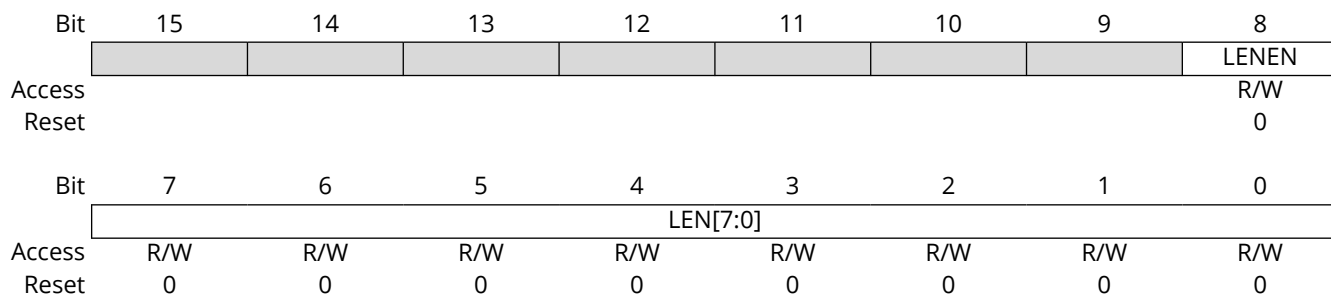
Resetting the SERCOM (CTRLA.SWRST) requires synchronization. Ongoing synchronization is indicated by SYNCBUSY.SWRST = 1 until synchronization is complete.

**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.

### 32.8.9 Length

**Name:** LENGTH  
**Offset:** 0x22  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Write-Synchronized



#### Bit 8 – LENEN Data Length Enable

In 32-bit Extension mode (CTRLC.DATA32B=1), this bit field enables the length counter.

Value	Description
0	Length counter is disabled.
1	Length counter is enabled.

#### Bits 7:0 – LEN[7:0] Data Length

In 32-bit Extension mode (CTRLC.DATA32B=1) with Data Length counting enabled (LENGTH.LENEN), this bit field configures the data length from 0 to 255 Bytes after which the flag INTFLAG.DRDY is raised.

### 32.8.10 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<sup>2</sup>C 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.

### 32.8.11 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<sup>2</sup>C 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							QCEN	SMEN	
		15:8									
		23:16						ACKACT	CMD[1:0]		
		31:24									
0x08	CTRLC	7:0									
		15:8									
		23:16									
		31:24									DATA32B
0x0C	BAUD	7:0	BAUD[7:0]								
		15:8	BAUDLOW[7:0]								
		23:16									
		31:24									
0x10 ...	Reserved										
0x13											
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	SYNDBUSY	7:0						SYSOP	ENABLE	SWRST	
		15:8									
		23:16									
		31:24									
0x20 ...	Reserved										
0x23											
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 ...	Reserved										
0x2F											
0x30	DBGCTRL	7:0								DBGSTOP	

## 32.10 Register Description – I<sup>2</sup>C 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<sup>2</sup>C 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 <a href="#">Figure 32-5</a>
1	SCL stretch only after ACK bit, <a href="#">Figure 32-6</a>

**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 <sup>2</sup> C 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<sup>2</sup>C 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.

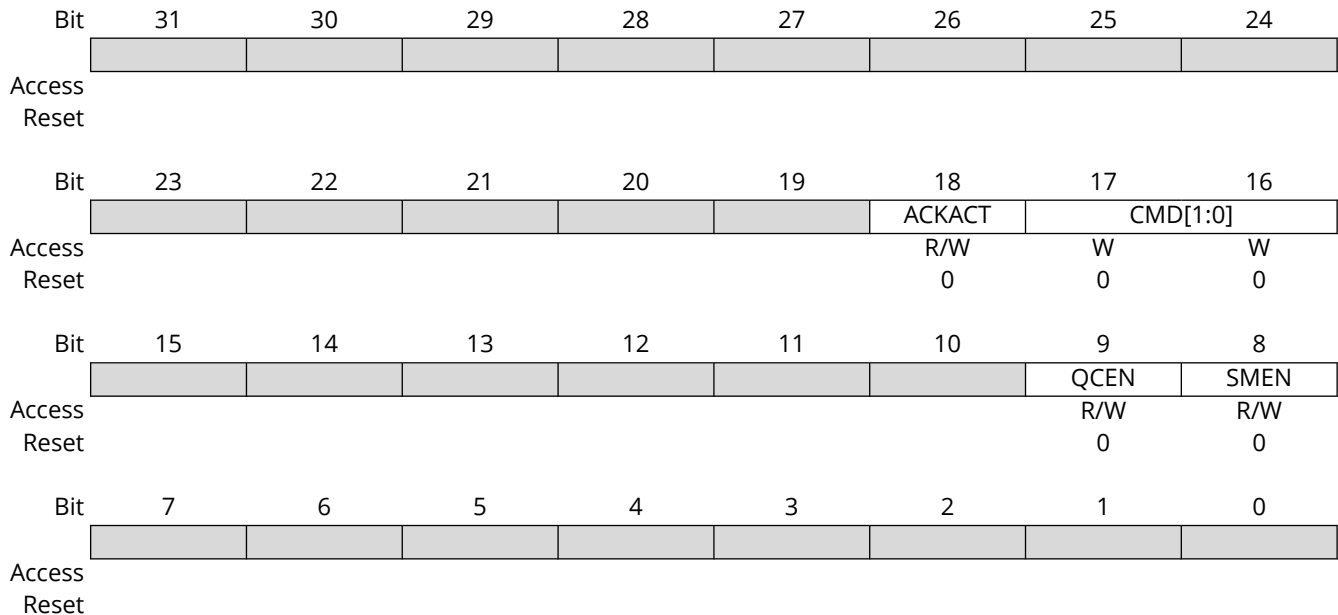
This bit is not enable-protected.

**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 18 – ACKACT Acknowledge Action

This bit defines the I<sup>2</sup>C host's acknowledge behavior after a data byte is received from the I<sup>2</sup>C 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)

.....continued

CMD[1:0]	Direction	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

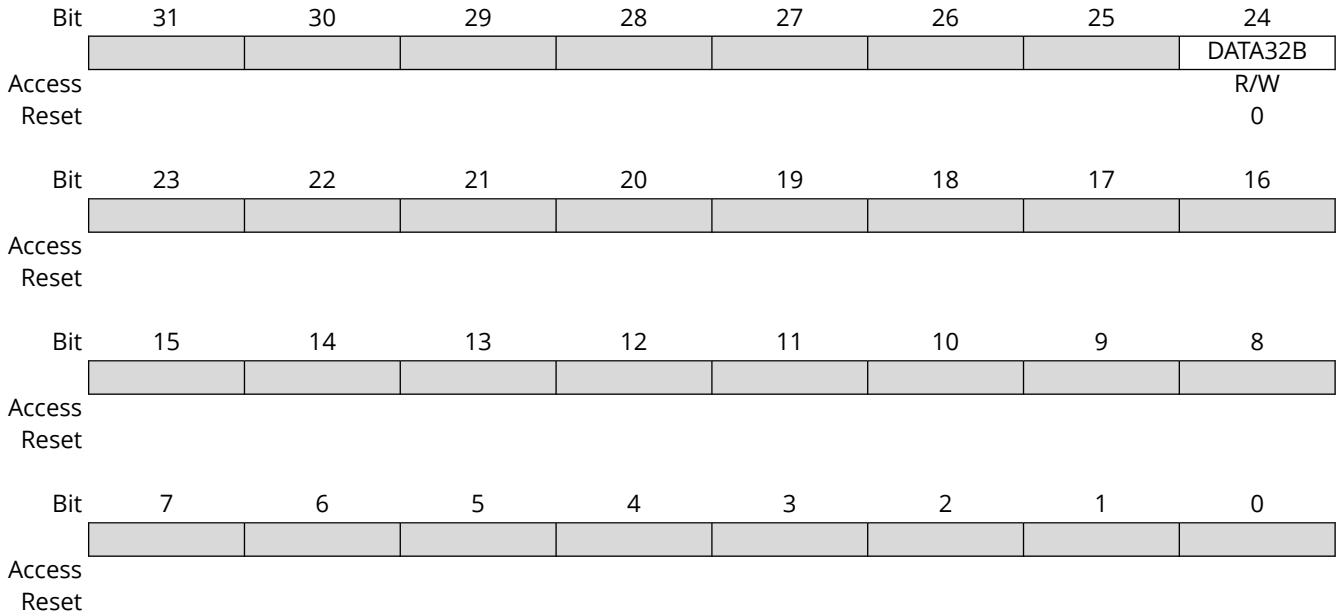
When Smart mode is enabled, acknowledge action is sent when DATA.DATA is read.

This bit is not write-synchronized.

Value	Description
0	Smart mode is disabled.
1	Smart mode is enabled.

### 32.10.3 Control C

**Name:** CTRLC  
**Offset:** 0x08  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected



#### Bit 24 - DATA32B Data 32 Bit

This bit enables 32-bit data writes and reads to/from the DATA register.

Value	Description
0	Data transactions to/from DATA are 8-bit in size
1	Data transactions to/from DATA are 32-bit in size

### 32.10.4 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](#)



### 32.10.5 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.

### 32.10.6 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.

### 32.10.7 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<sup>2</sup>C 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<sup>2</sup>C 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.

### 32.10.8 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 and/or 32-bit 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<sup>2</sup>C 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<sup>2</sup>C 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<sup>2</sup>C 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 <sup>2</sup> C 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 <sup>2</sup> C host is the current owner of the bus
0x3	BUSY	Some other I <sup>2</sup> C 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<sup>2</sup>C 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<sup>2</sup>C 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.

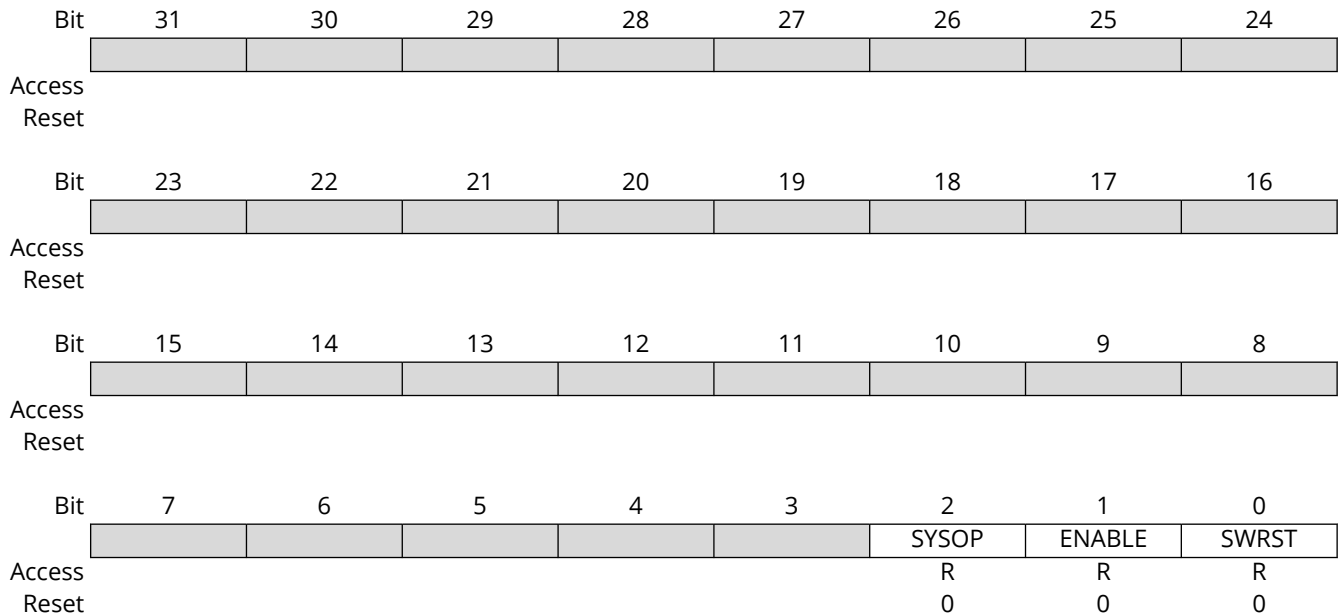
Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.

This bit is not write-synchronized.

### 32.10.9 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x1C  
**Reset:** 0x00000000



#### Bit 2 – SYSOP System Operation Synchronization Busy

Writing CTRLB.COMD, STATUS.BUSSTATE, ADDR or DATA when the SERCOM is enabled requires synchronization. When written, the SYNCBUSY.SYSOP bit will be set until synchronization is complete.

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.

### 32.10.10 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 and/or 32-bit 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<sup>2</sup>C host will await further operation until the bus becomes IDLE.

IDLE: The I<sup>2</sup>C 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<sup>2</sup>C 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.



### 32.10.11 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

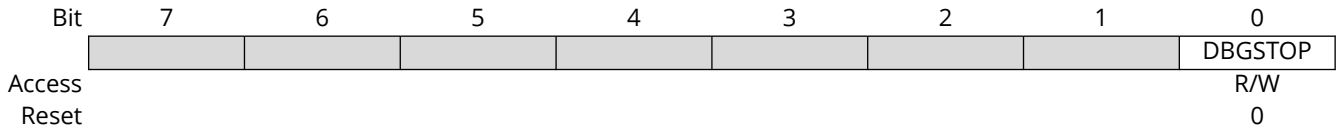
The host data register I/O location (DATA) provides access to the host transmit and receive data buffers. Reading valid data or writing data to be transmitted can be successfully done only when SCL is held low by the host (STATUS.CLKHOLD is set). An exception is reading the last data byte after the stop condition has been sent.

Accessing DATA.DATA auto-triggers I<sup>2</sup>C 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.

### 32.10.12 Debug Control

**Name:** DBGCTRL  
**Offset:** 0x30  
**Reset:** 0x00  
**Property:** PAC Write-Protection



#### 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.

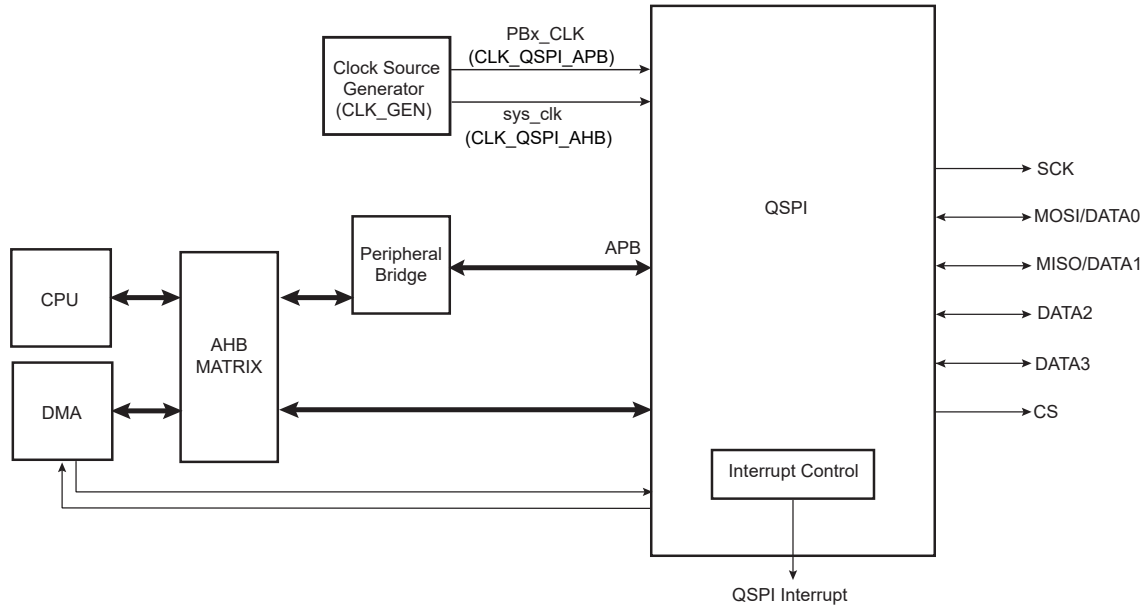
**Note:** Traditional Quad SPI Interface (QSPI) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client” respectively.

### 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, CAN 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

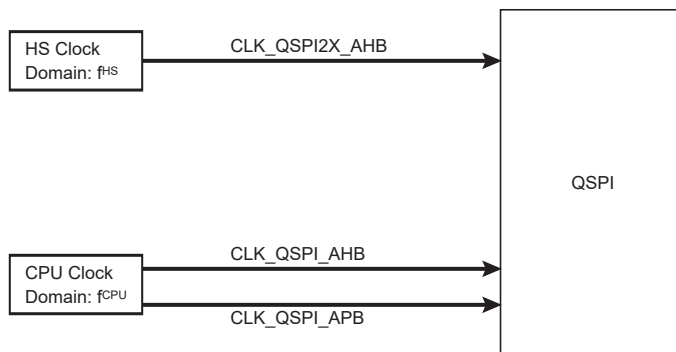
[15. Power Management Unit \(PMU\)](#)

### 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  $\leq 48$  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

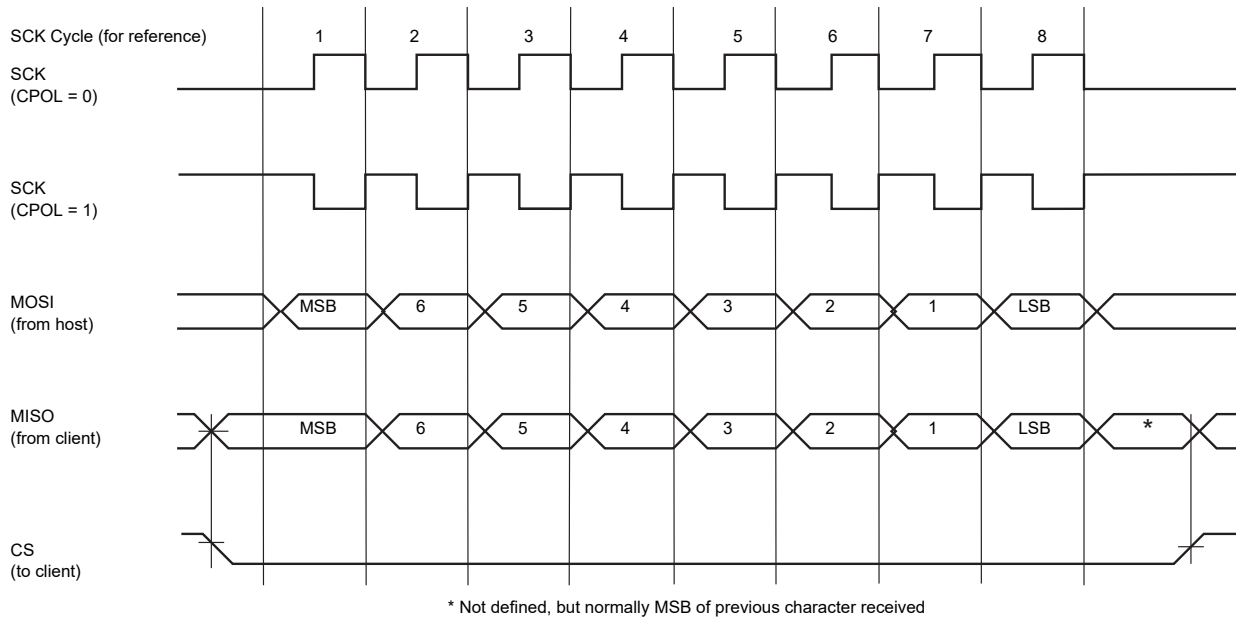
Four combinations of polarity and phase are available for data transfers. Writing the Clock Polarity bit in the QSPI Baud register (BAUD.CPOL) selects the polarity. The Clock Phase bit in the BAUD register programs the clock phase (BAUD.CPHA). These two parameters determine the edges of the clock signal 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

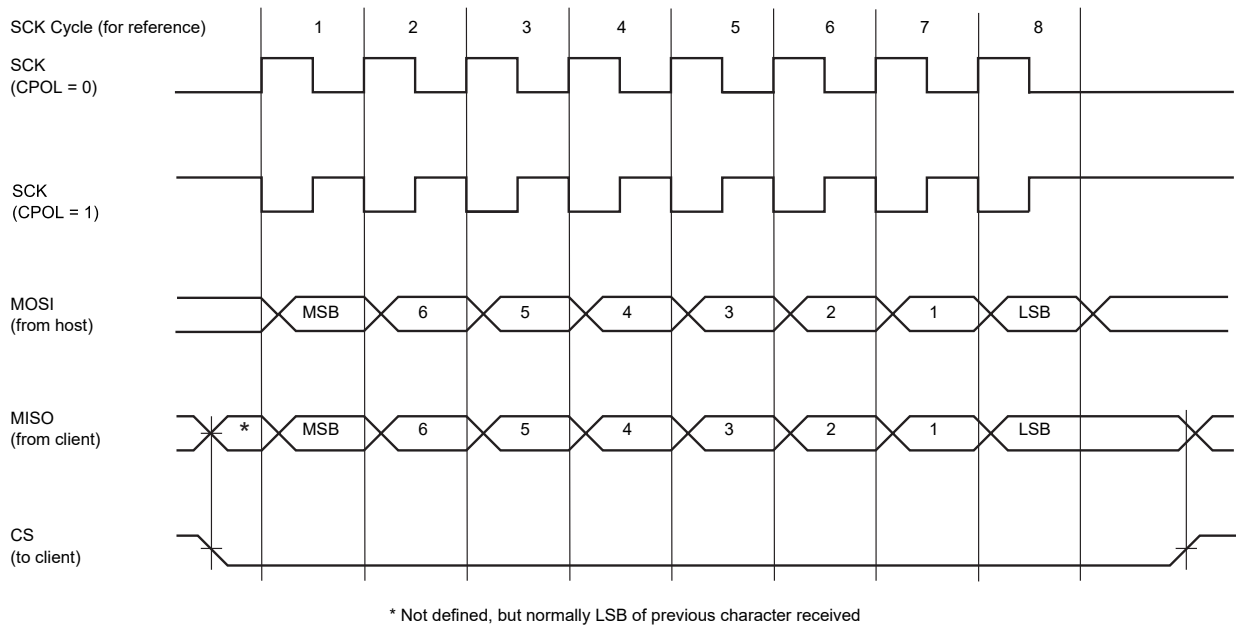
Clock Mode	BAUD.CPOL	BAUD.CPHA	Shift SCK Edge	Capture SCK Edge	SCK Inactive Level
0	0	0	Falling	Rising	Low
1	0	1	Rising	Falling	Low
2	1	0	Rising	Falling	High
3	1	1	Falling	Rising	High

**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)



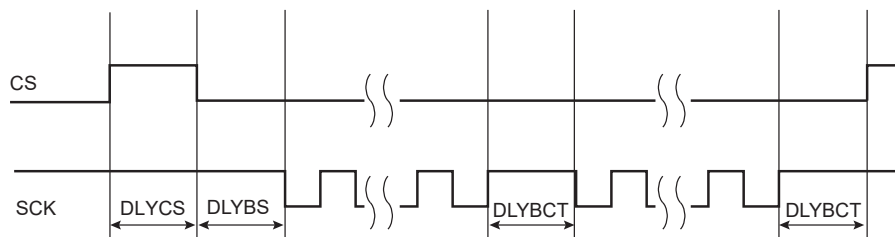
### 33.6.6 Transfer Delays

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.

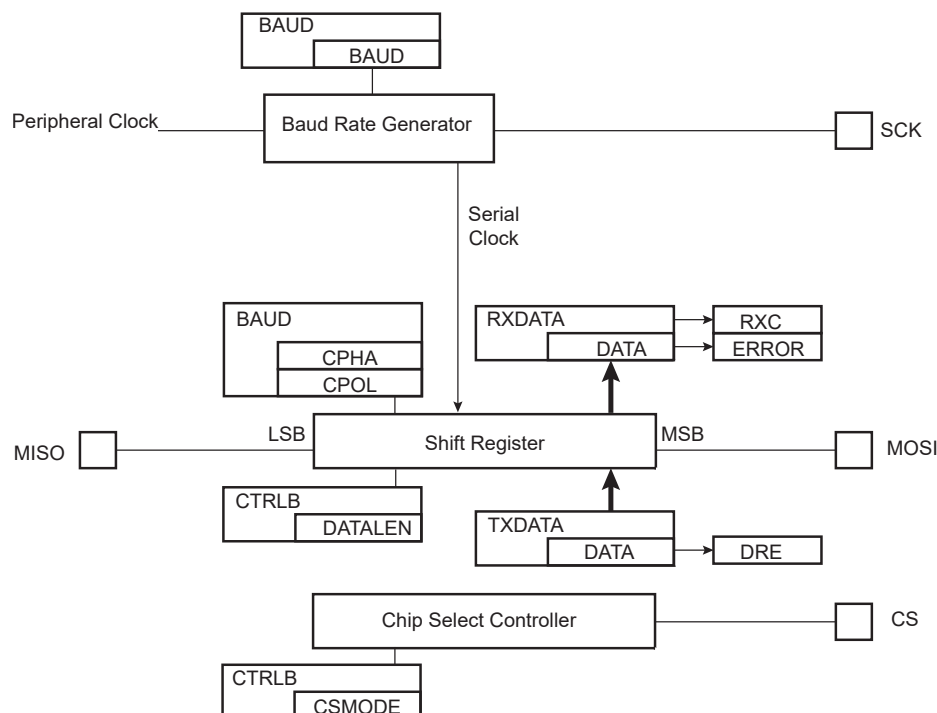
Ongoing transfer of received data from the internal shift register into RXDATA is indicated by the Receive Data Register Full flag (INTFLAG.RXC). When the received data is read, the RXC bit is cleared.

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.

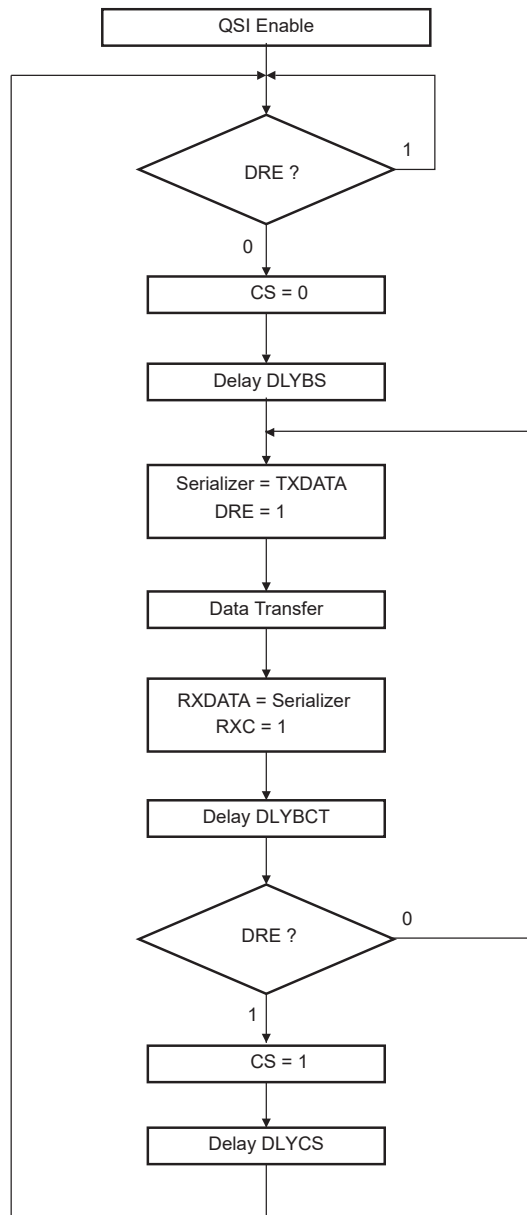
### 33.6.7.2 SPI Mode Block Diagram

Figure 33-6. SPI Mode Block Diagram

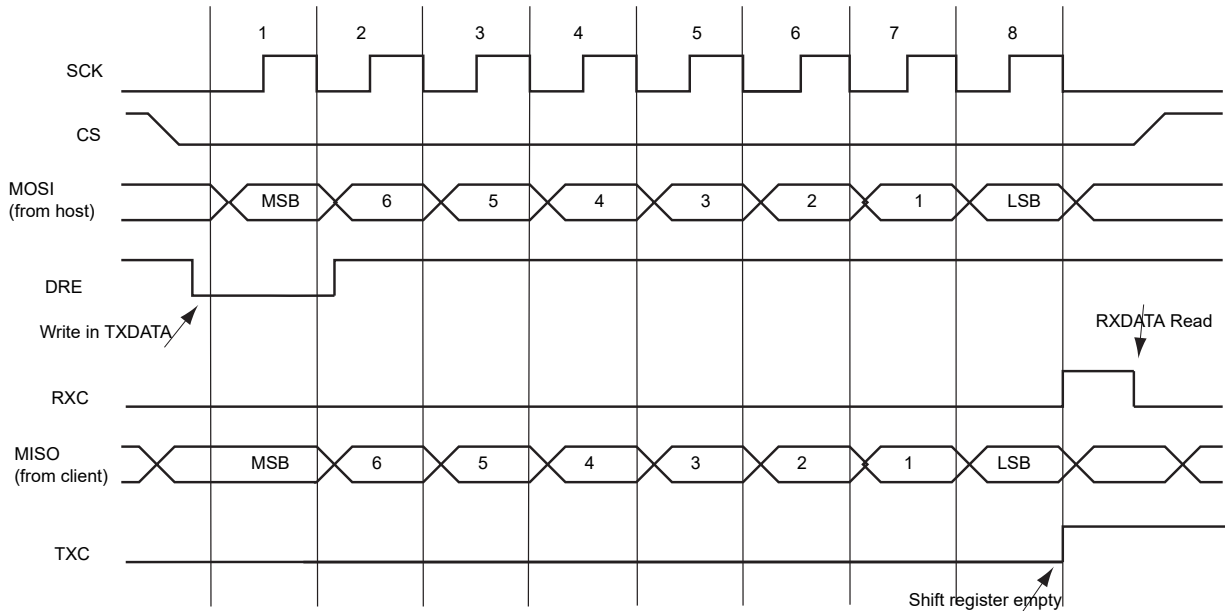


### 33.6.7.3 SPI Mode Flow Diagram

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, etc.) 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).

#### 33.6.8.1 Instruction Frame

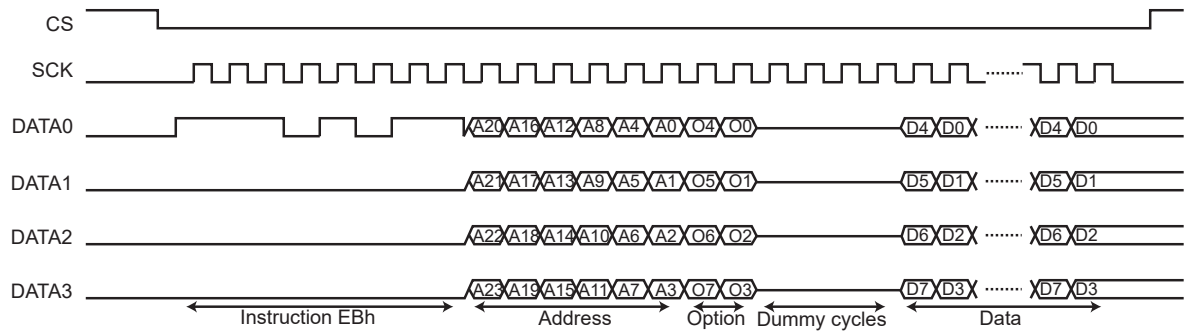
In order to control serial flash memories, the QSPI is able to sent instructions by the SPI bus (ex: READ, PROGRAM, ERASE, LOCK, etc.). Because instruction set implemented in serial flash memories is memory vendor dependant, the QSPI includes a complete instruction registers, which makes it very flexible 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.
- ADDRLEN 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.

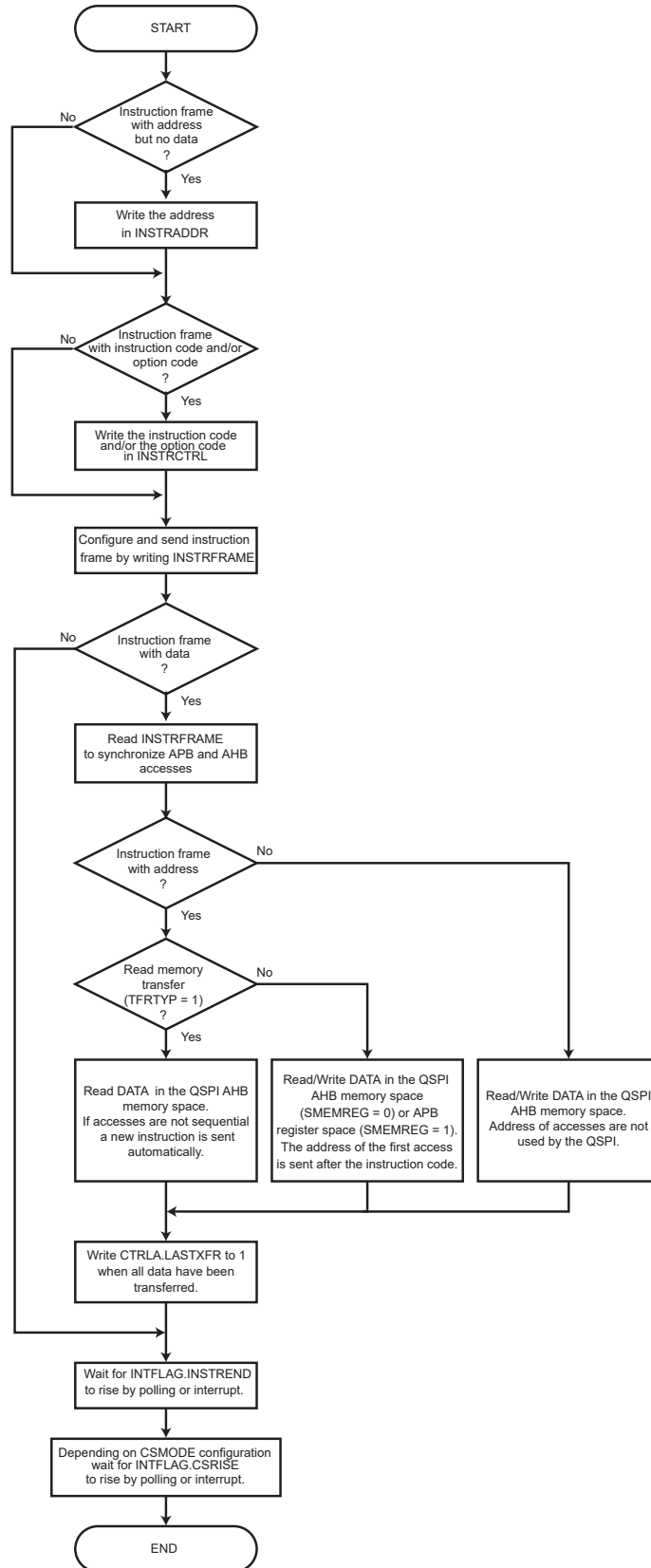
If TFRTYPE has a value other than 0x1 and CTRLB.SMEMREG=0, the address sent in the instruction frame is the address of the first system bus accesses. The addresses of the subsequent access actions are not used by the QSPI. At each system bus access, an SPI transfer is performed with the same size. For 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 TFRTYPE=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.

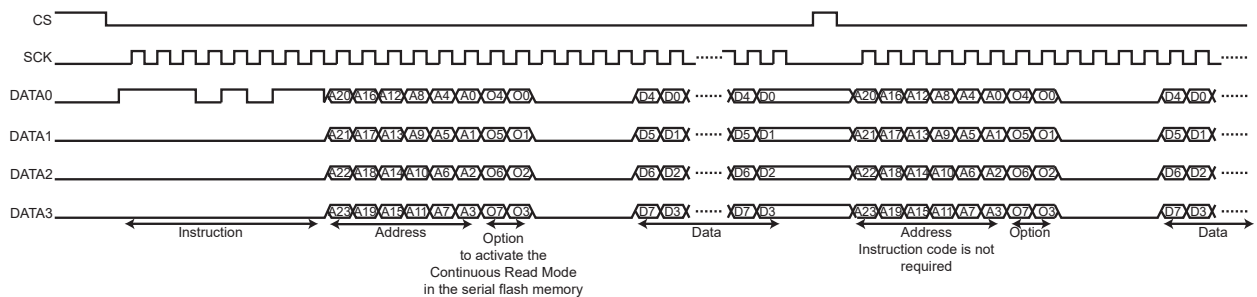
In the QSPI, CRM is used when reading data from the memory (INSTRFRAME.TFRTYPE=0x1). The addresses of the system bus read accesses are often non-sequential, this leads to many instruction frames with always the same instruction code. By disabling the sending of the instruction code, the CRM reduces the 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 (INSTRFRAME.CRMODE=1, INSTRFRAME.TFRTYPE must be 0x1). The CRM is enabled in the Serial Flash memory by sending a specific option code.



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

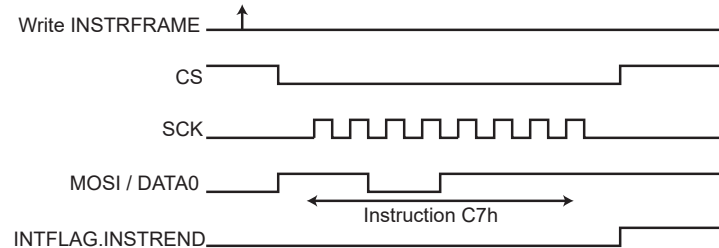
Instruction in Single-bit SPI, without address, without option, without data.



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



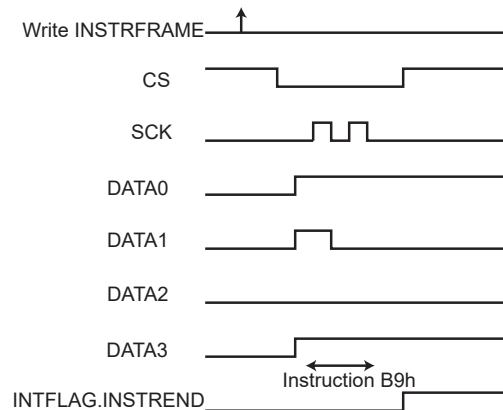
**Example 33-2.** Example 2

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



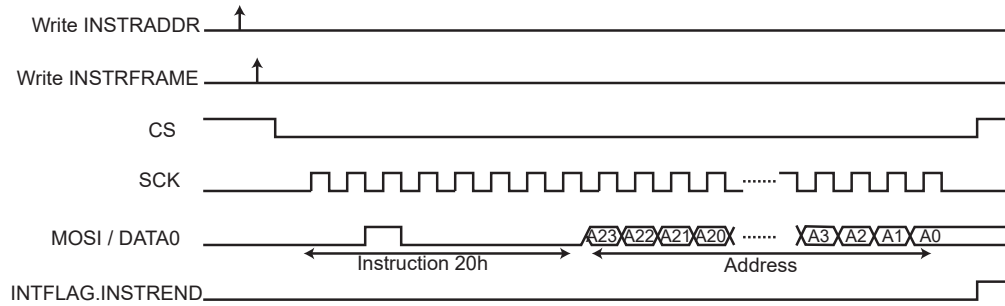
**Example 33-3.** Example 3

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 to INSTRFRAME register.
- Wait for INTFLAG.INSTREND to rise.

**Figure 33-14.** Instruction Transmission Waveform 3



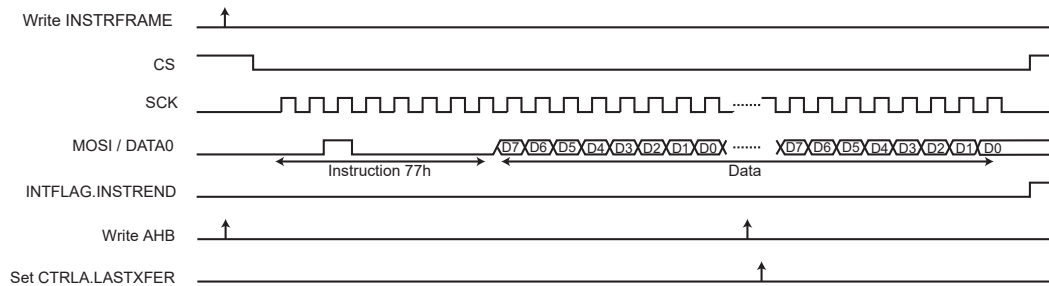
**Example 33-4.** Example 4

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

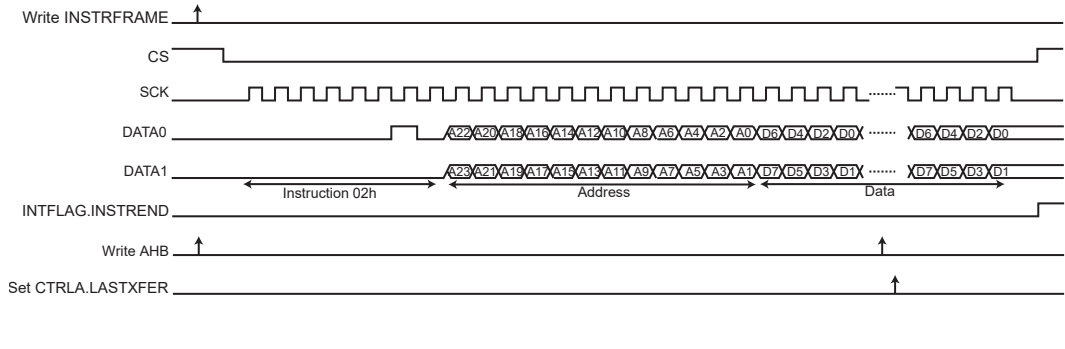
Instruction in Single-bit SPI, with address in Dual SPI, without option, with data write in Dual SPI.

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



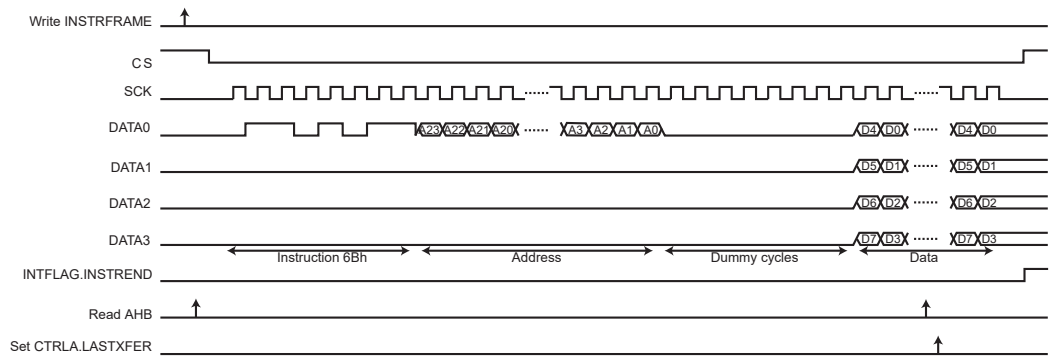
**Example 33-6.** Example 6

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 to 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

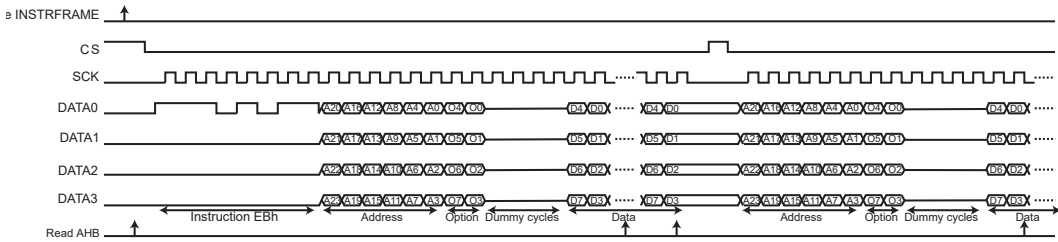
Instruction in Single-bit SPI, with address and option in Quad SPI, with data read from Quad SPI, with four dummy cycles, with fetch and continuous read.

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



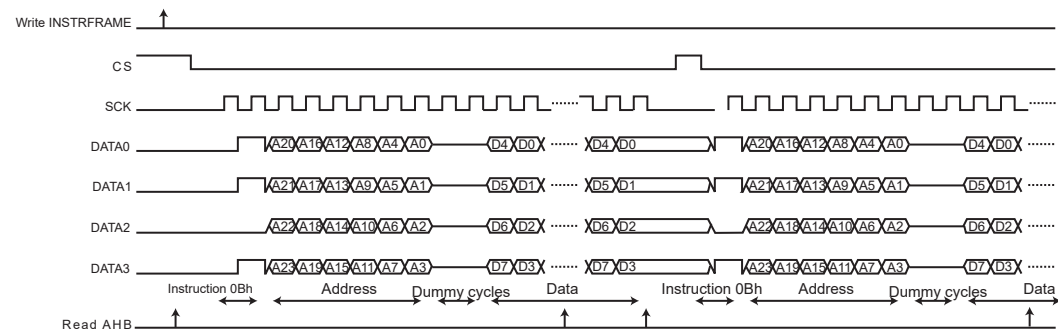
**Example 33-8.** Example 8

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]			ADDRLN	OPTCODELEN[1:0]	
		23:16								DUMMYLEN[4:0]	
		31:24									
0x3C ... 0x3F	Reserved										
0x40	SCRAMBCTRL	7:0							RANDOMDIS	ENABLE	
		15:8									
		23:16									
		31:24									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
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							R/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:

$$DLYCS = \text{Minimum inactive} \times f_{\text{peripheral clock}}$$

#### 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:

$$DLYBCT = (\text{Delay Between Consecutive Transfers} \times f_{\text{peripheral clock}}) / 32$$

#### 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

Value	Name	Description
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
Access	DLYBS[7:0]							
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	BAUD[7:0]							
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access							CPHA	CPOL
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

$$\text{Delay Before SCK} = \frac{DLYBS}{MCK}$$

#### 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

$$\text{SCK Baud Rate} = \frac{MCK}{(BAUD + 1)}$$

#### 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.
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
Access	DATA[15:8]							
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	DATA[7:0]							
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
Access	DATA[15:8]							
Reset	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
Access	DATA[7:0]							
Reset	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
Access						INSTREND		CSRISE
Reset						R/W 0		R/W 0
Bit	7	6	5	4	3	2	1	0
Access					ERROR	TXC	DRE	RXC
Reset					R/W 0	R/W 0	R/W 0	R/W 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
Access						INSTREND		CSRISE
Reset						R/W 0		R/W 0
Bit	7	6	5	4	3	2	1	0
Access					ERROR	TXC	DRE	RXC
Reset					R/W 0	R/W 0	R/W 0	R/W 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
Access						INSTREND		CSRISE
Reset						R/W 0		R/W 0
Bit	7	6	5	4	3	2	1	0
Access					ERROR	TXC	DRE	RXC
Reset					R/W 0	R/W 0	R/W 0	R/W 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.

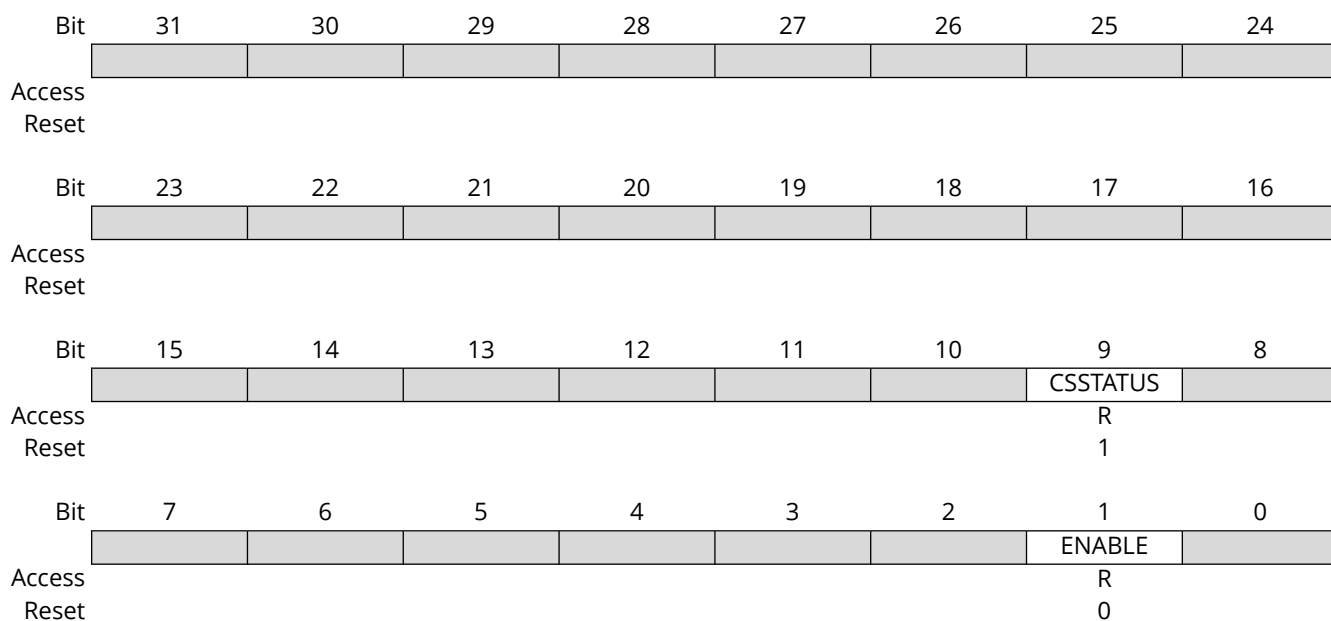
**Bit 0 – RXC** Receive Data Register Full

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 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	
Access				DUMMYLEN[4:0]					
Reset				R/W	R/W	R/W	R/W	R/W	
				0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8	
Access	DDREN	CRMODE	TFRTYPE[1:0]			ADDRLEN	OPTCODELEN[1:0]		
Reset	R/W	R/W	R/W	R/W		R/W	R/W	R/W	
	0	0	0	0		0	0	0	
Bit	7	6	5	4	3	2	1	0	
Access	DATAEN	OPTCODEEN	ADDRLEN	INSTREN		WIDTH[2:0]			
Reset	R/W	R/W	R/W	R/W		R/W	R/W	R/W	
	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

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.

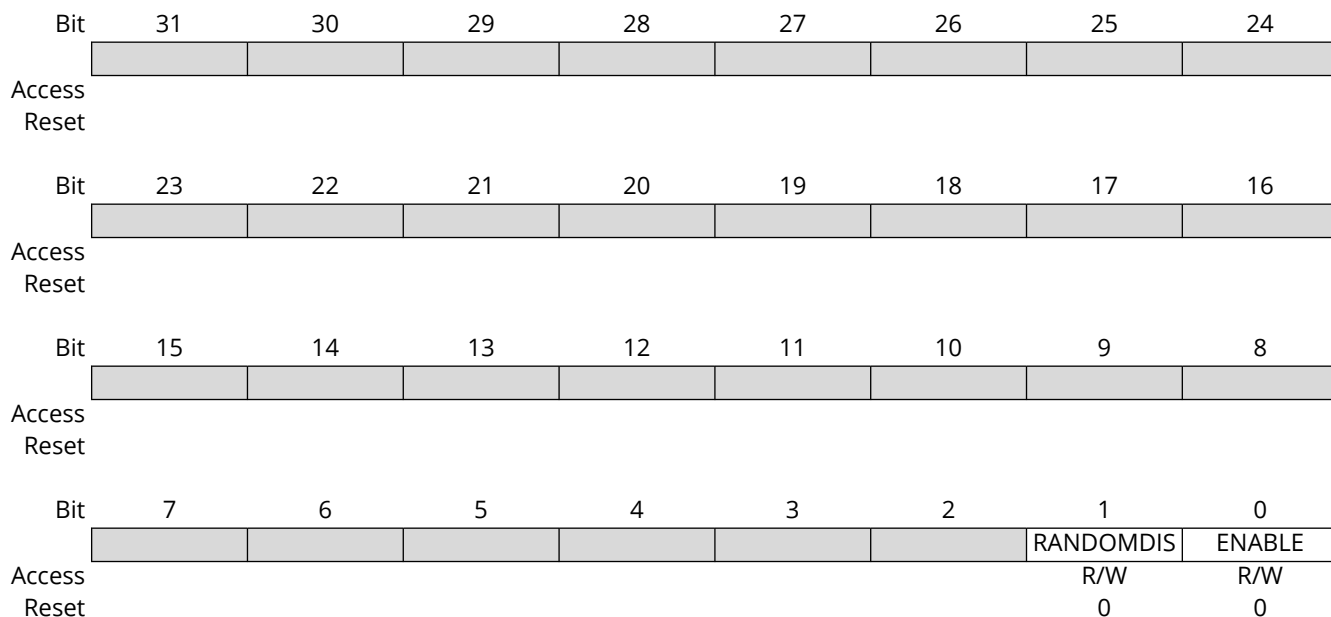
#### Bits 2:0 – WIDTH[2:0] Instruction Code, Address, Option Code and Data Width

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 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.

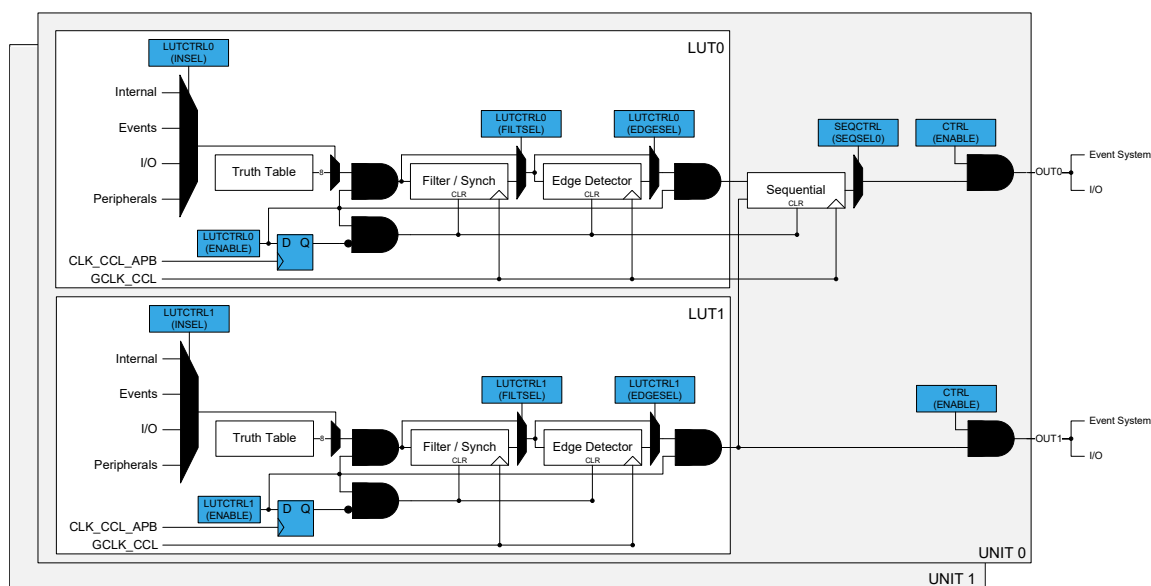
The output can be combinatorially generated from the inputs, and can be filtered to remove spikes. Optional sequential logic can be used. The inputs of the sequential module are individually controlled by two independent, adjacent LUT (LUT0/LUT1) outputs, enabling complex waveform generation.

### 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 constraints 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.

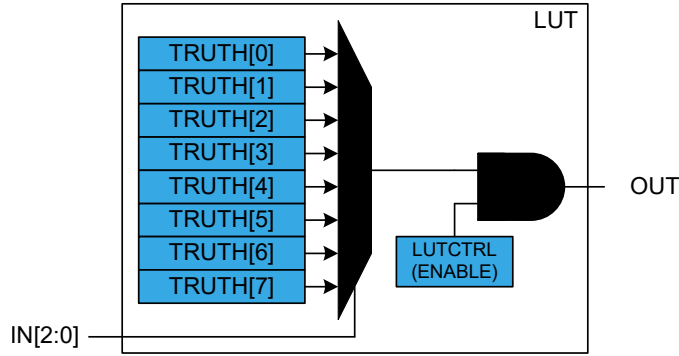
Each LUT is enabled by writing a '1' to the Enable bit in the LUT Control x register (LUTCTRLx.ENABLE). Each LUT is disabled by writing a '0' to LUTCTRLx.ENABLE.

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:

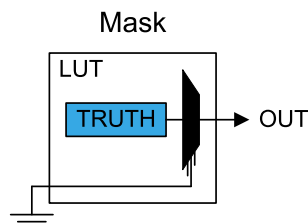
- Masked
- Driven by peripherals:
  - Analog comparator output (AC)
  - Timer/Counters waveform outputs (TC)
  - Serial Communication output transmit interface (SERCOM)
- Driven by internal events from Event System
- Driven by other CCL sub-modules

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:

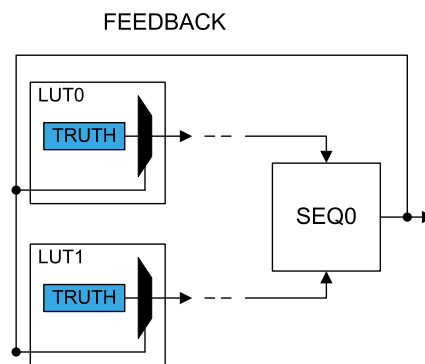
$$IN[2N][i] = SEQ[N]$$

$$IN[2N+1][i] = SEQ[N]$$

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

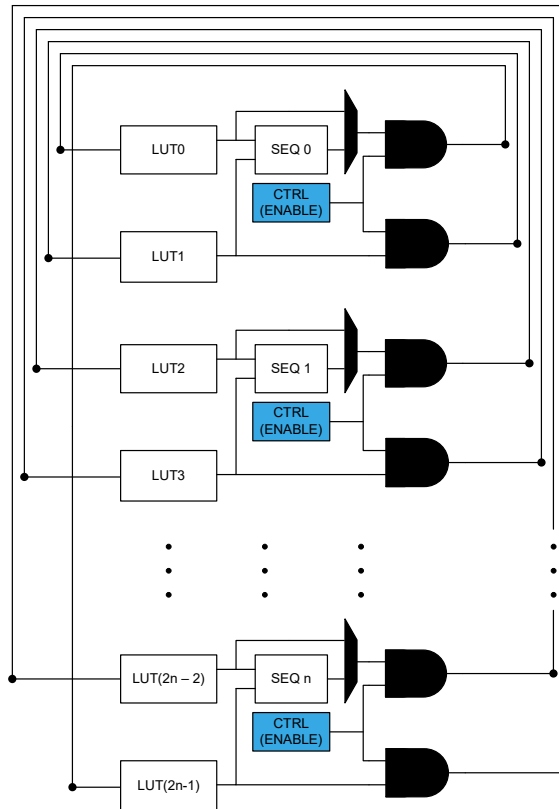


### Linked LUT (LINK)

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**



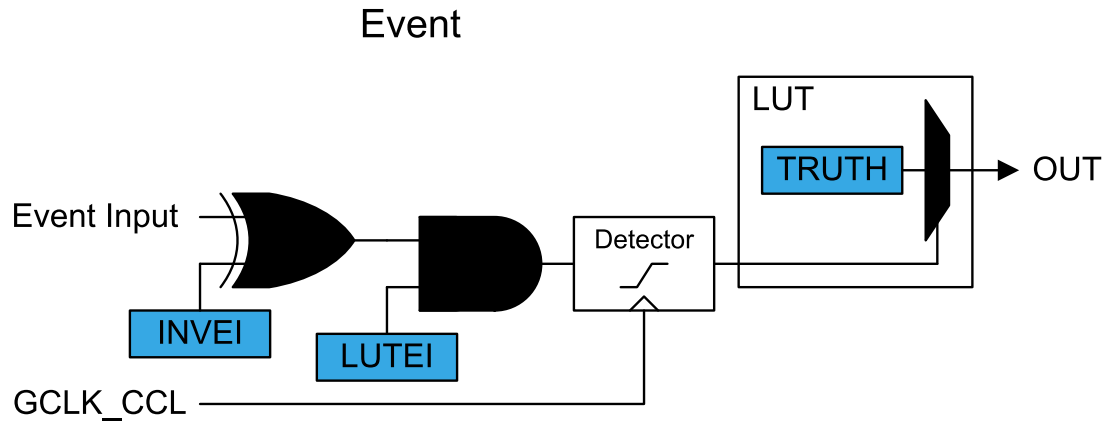
### Internal Events Inputs Selection (EVENT)

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'.

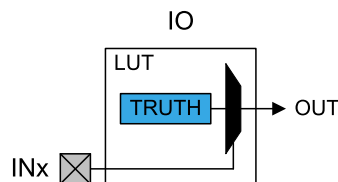
Figure 34-6. Event Input Selection



### 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:

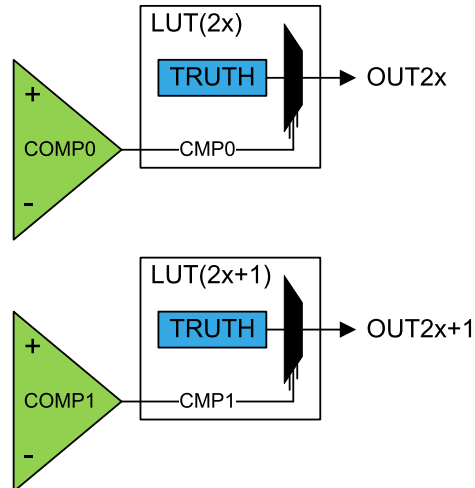
$$IN[N][i] = AC[N \% ComparatorOutput\_Number]$$

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.

The output of comparator 0 is available on even LUTs ("LUT(2x)": LUT0) and the comparator 1 output is available on odd LUTs ("LUT(2x+1)": LUT1), as shown in the figure below.

**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:

$$IN[N][i] = \text{DefaultTC}[N \% \text{TC\_Instance\_Number}]$$

$$IN[N][i] = \text{AlternativeTC}[(N + 1) \% \text{TC\_Instance\_Number}]$$

Where  $N$  represents the LUT number and  $i$  represents the LUT input index ( $i=0,1,2$ ).

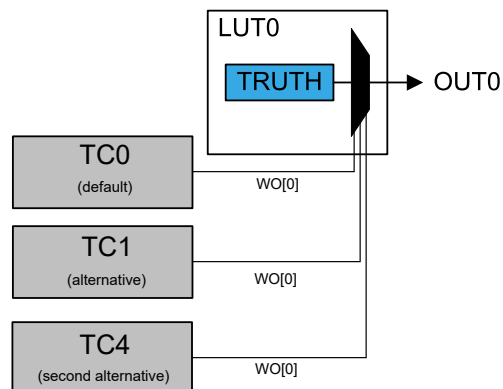
It is also possible to enable a second alternative option (LUTCTRLx.INSEL= ALT2TC). This option is intended to relax the alternative pin function or PCB design constraints when the default or the alternative TC instances are used for other purposes. When enabled, the Timer/Counter selection for each LUT follows the formula:

$$IN[N][i] = \text{SecondAlternativeTC}[(N + 4) \% \text{TC\_Instance\_Number}]$$

Note that for not implemented TC\_Instance\_Number, the corresponding input is tied to ground.

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.

**Note:**

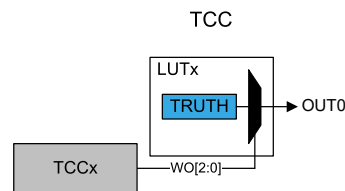
The TCC selection for each LUT follows the formula:

$$IN[N][i] = TCC[N \% TCC\_Instance\_Number].WO[i]$$

Where  $N$  represents the LUT number and  $i$  represents the LUT input index ( $i=0,1,2$ ).

Before selecting the waveform outputs, the TCC must be configured first.

**Figure 34-10.** TCC Input Selection



### Serial Communication Output Transmit Inputs (SERCOM)

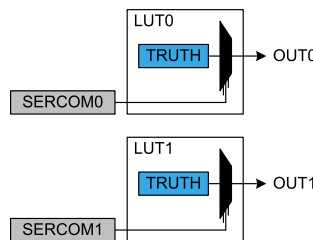
The serial engine transmitter output from Serial Communication Interface (SERCOM TX, TXd for USART, MOSI for SPI) can be used as input source for the LUT. The figure below shows an example for LUT0 and LUT1. The SERCOM selection for each LUT follows the formula:

$$IN[N][i] = SERCOM[N \% SERCOM\_Instance\_Number]$$

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](#)

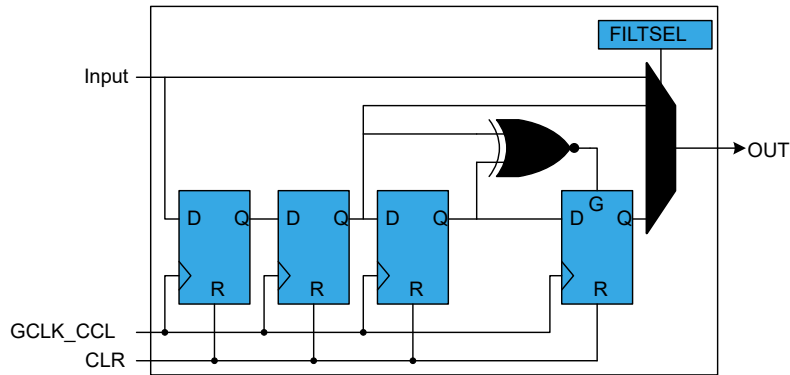
#### 34.6.2.5 Filter

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



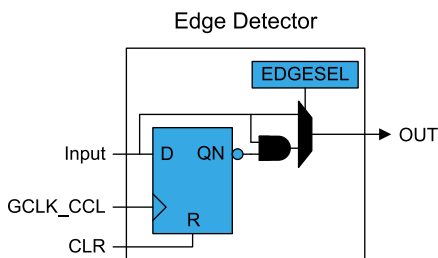
### 34.6.2.6 Edge Detector

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

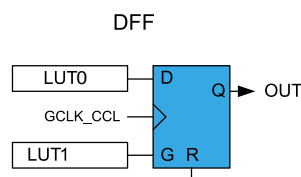
Each LUT pair can be connected to the internal sequential logic, which can be configured to work as D flip flop, JK flip flop, gated D-latch or RS-latch by writing the Sequential Selection bits on the corresponding Sequential Control x register (SEQCTRLx.SEQSEL). Before using sequential logic, the GCLK\_CCL 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.

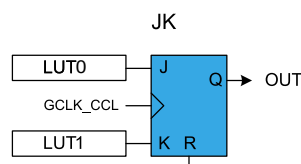
**Table 34-2.** DFF Characteristics

R	G	D	OUT
1	X	X	Clear
0	1	1	Set
		0	Clear
0	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.

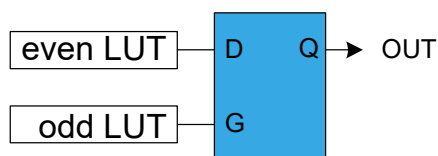
**Table 34-3.** JK Characteristics

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)

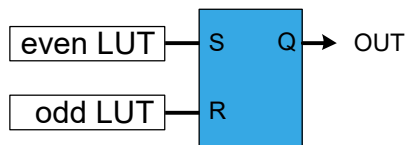
.....continued

G	D	OUT
1	0	Clear
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.LUTE0) 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	<a href="#">CTRL</a>	7:0		RUNSTDBY					ENABLE	SWRST
0x01	Reserved									
...										
0x03										
0x04	<a href="#">SEQCTRLX</a>	7:0					SEQSEL[3:0]			
0x05	Reserved									
...										
0x07										
0x08	<a href="#">LUTCTRL0</a>	7:0	EDGESEL		FILTSEL[1:0]				ENABLE	
		15:8	INSEL1[3:0]			INSEL0[3:0]				
		23:16		LUTEO	LUTEI	INVEI	INSEL2[3:0]			
		31:24	TRUTH[7:0]							
0x0C	<a href="#">LUTCTRL1</a>	7:0	EDGESEL		FILTSEL[1:0]				ENABLE	
		15:8	INSEL1[3:0]			INSEL0[3:0]				
		23:16		LUTEO	LUTEI	INVEI	INSEL2[3:0]			
		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
		LUTE0	LUTE1	INVE1	INSEL2[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	15	14	13	12	11	10	9	8
	INSEL1[3:0]				INSEL0[3:0]			
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
	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 – LUTE0 LUT Event Output Enable

Value	Description
0	LUT event output is disabled.
1	LUT event output is enabled.

#### Bit 21 – LUTE1 LUT Event Input Enable

Value	Description
0	LUT incoming event is disabled.
1	LUT incoming event is enabled.

#### Bit 20 – INVE1 Inverted Event Input Enable

Value	Description
0	Incoming event is not inverted.
1	Incoming event is inverted.

#### Bits 8:11, 12:15, 16:19 – 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

Value	Name	Description
0x5	AC	AC input source: CMP[0] (LUT0) / CMP[1] (LUT1)
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.

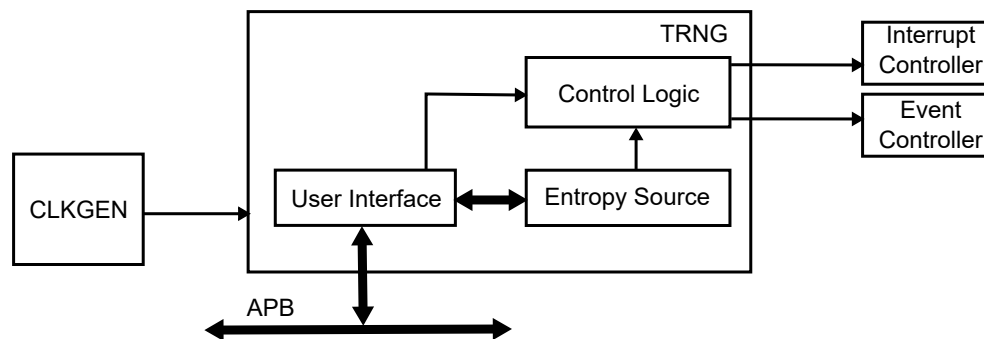
The TRNG may be used as an entropy source for seeding an NIST approved DRNG (Deterministic RNG) as required by FIPS PUB 140-2 and 140-3.

### 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.

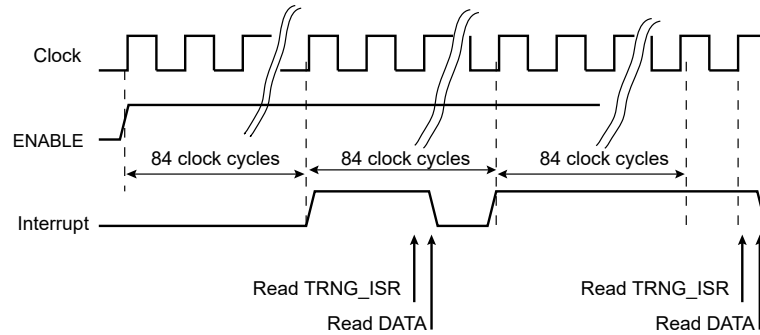
## 35.6 Functional Description

### 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.

The following register is enable-protected, meaning that it can only be written when the TRNG is disabled (CTRLA.ENABLE is zero):

- Event Control register (EVCTRL)

Enable-protection is denoted by the Enable-Protected property in the register description.

### 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	Reserved										
...											
0x03											
0x04	EVCTRL	7:0								DATARDYEO	
0x05	Reserved										
...											
0x07											
0x08	INTENCLR	7:0								DATARDY	
0x09	INTENSET	7:0								DATARDY	
0x0A	INTFLAG	7:0								DATARDY	
0x0B	Reserved										
...											
0x1F											
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:

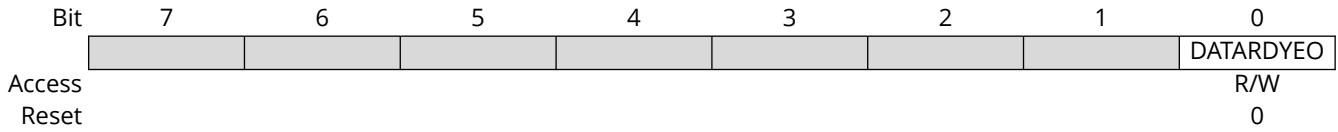
Value	Description
0	The TRNG is halted during standby sleep mode.
1	The TRNG is not stopped in standby sleep mode.

#### 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, Enable-Protected



#### 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

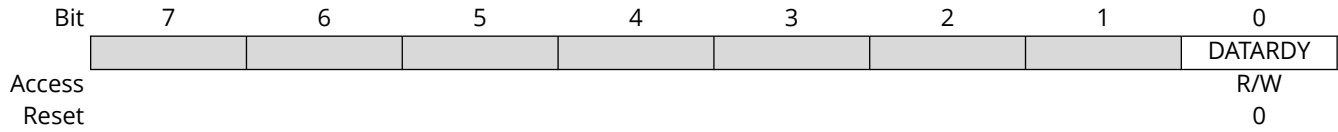
#### Bit 0 – DATARDY Data Ready Interrupt Enable

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 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

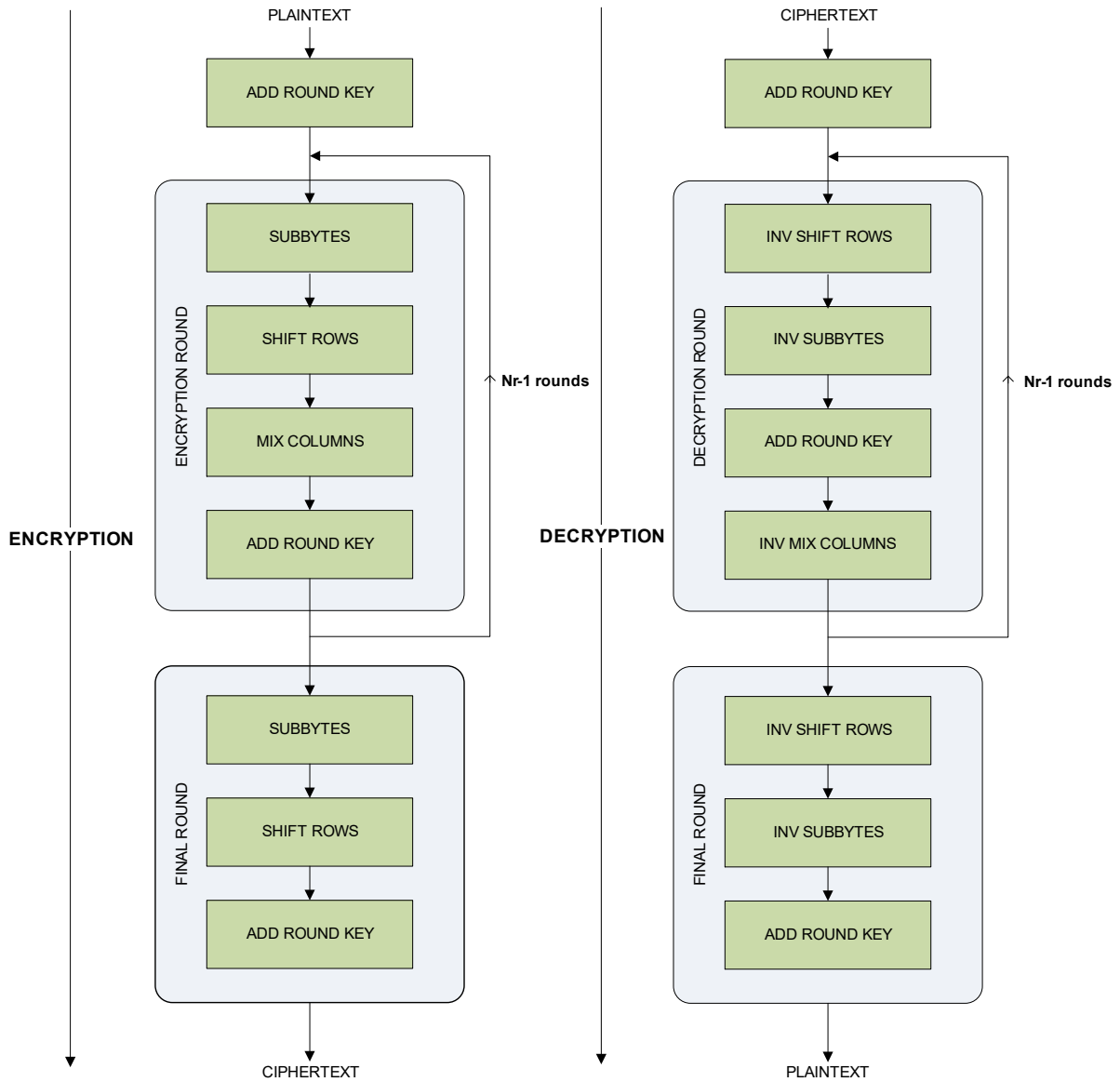
### 36.2 Features

- 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 its 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

.....continued

Confidentiality Mode	Relevant Input / Output Data Registers
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

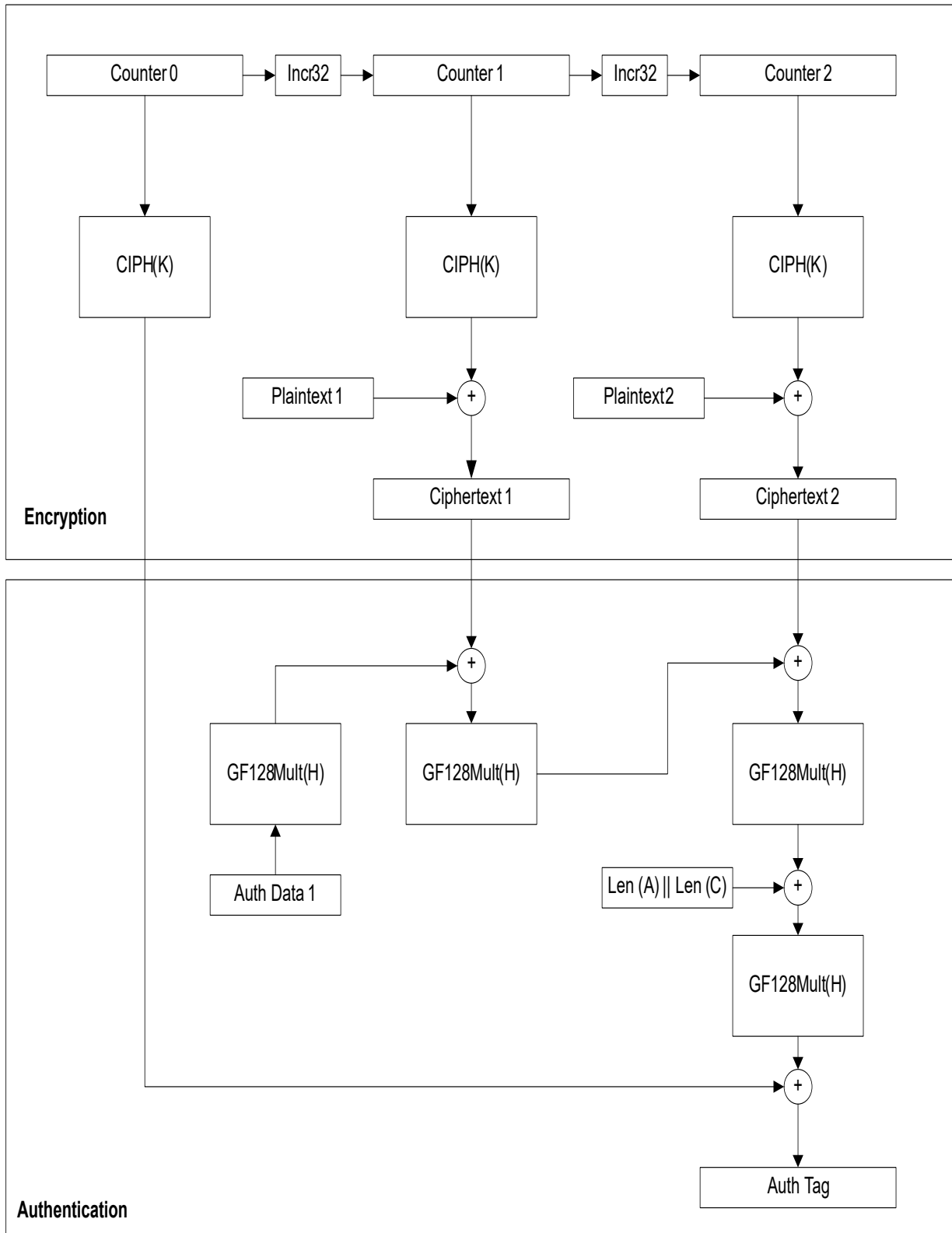
The AES module features four types of hardware countermeasures that are useful for protecting data against differential power analysis attacks:

- 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 key size. 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.



### 36.6.3.1 GCM Operation

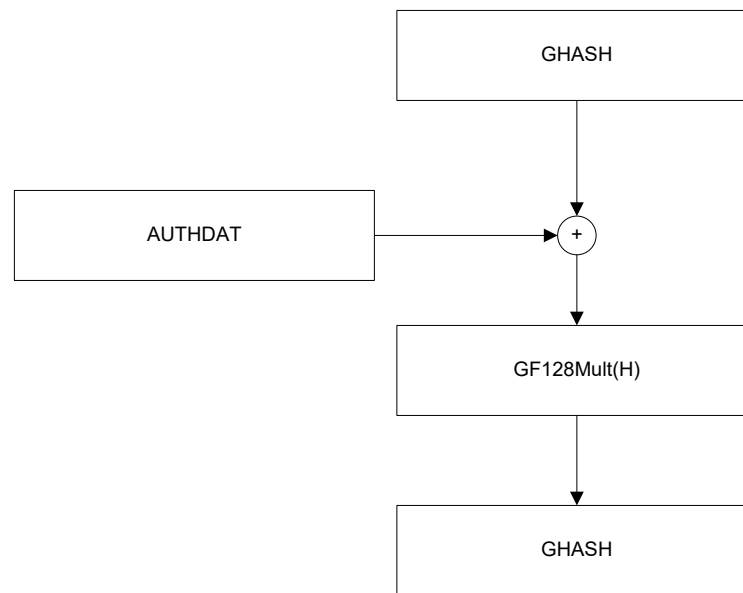
#### 36.6.3.1.1 Hashkey Generation

- 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 CIPLLEN 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 as 1
- 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.NEWMMSG for the new set of plain text processing.
- Load CIPLN reg.
- Load (J0+1) in INTVECT register.
- As described in NIST documentation  $J = IV \parallel 0 \ 31 \parallel 1$  when  $\text{len}(IV)=96$  and  $J_0 = \text{GHASH}_H (IV \parallel 0 \ s+64 \parallel [\text{len}(IV)] \ 64)$  (s is the minimum number of zeroes that must be padded with the Initialization Vector to make it a multiple of 128) if  $\text{len}(IV) \neq 96$ .
- 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  $[\text{LEN}(A)]64 \parallel [\text{LEN}(C)]64$  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.NEWMMSG for the new set of plain text processing.
- Load CIPLN 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  $[\text{LEN(A)}]64 \mid [\text{LEN(C)}]64$  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.

#### 36.6.4 Synchronization

Not applicable.

## 36.7 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0		CFBS[2:0]			AESMODE[2:0]		ENABLE	SWRST
		15:8		XORKEY	KEYGEN	LOD	STARTMODE	CIPHER	KEYSIZE[1:0]	
		23:16							CTYPE[3:0]	
		31:24								
0x04	CTRLB	7:0					GFMUL	EOM	NEWMSG	START
0x05	INTENCLR	7:0							GFMCMP	ENCCMP
0x06	INTENSET	7:0							GFMCMP	ENCCMP
0x07	INTFLAG	7:0							GFMCMP	ENCCMP
0x08	DATABUFPTR	7:0							INDATAPTR[1:0]	
0x09	DBGCTRL	7:0								DBGRUN
0x0A ... 0x0B	Reserved									
0C	KEYWORD0	7:0					KEYWORD[7:0]			
		15:8					KEYWORD[15:8]			
		23:16					KEYWORD[23:16]			
		31:24					KEYWORD[31:24]			
10	KEYWORD1	7:0					KEYWORD[7:0]			
		15:8					KEYWORD[15:8]			
		23:16					KEYWORD[23:16]			
		31:24					KEYWORD[31:24]			
14	KEYWORD2	7:0					KEYWORD[7:0]			
		15:8					KEYWORD[15:8]			
		23:16					KEYWORD[23:16]			
		31:24					KEYWORD[31:24]			
18	KEYWORD3	7:0					KEYWORD[7:0]			
		15:8					KEYWORD[15:8]			
		23:16					KEYWORD[23:16]			
		31:24					KEYWORD[31:24]			
1C	KEYWORD4	7:0					KEYWORD[7:0]			
		15:8					KEYWORD[15:8]			
		23:16					KEYWORD[23:16]			
		31:24					KEYWORD[31:24]			
20	KEYWORD5	7:0					KEYWORD[7:0]			
		15:8					KEYWORD[15:8]			
		23:16					KEYWORD[23:16]			
		31:24					KEYWORD[31:24]			
24	KEYWORD6	7:0					KEYWORD[7:0]			
		15:8					KEYWORD[15:8]			
		23:16					KEYWORD[23:16]			
		31:24					KEYWORD[31:24]			
28	KEYWORD7	7:0					KEYWORD[7:0]			
		15:8					KEYWORD[15:8]			
		23:16					KEYWORD[23:16]			
		31:24					KEYWORD[31:24]			
0x2C ... 0x37	Reserved									
0x38	INDATA	7:0					INDATA[7:0]			
		15:8					INDATA[15:8]			
		23:16					INDATA[23:16]			
		31:24					INDATA[31:24]			
3C	INTVECTV0	7:0					INTVECTV[7:0]			
		15:8					INTVECTV[15:8]			
		23:16					INTVECTV[23:16]			
		31:24					INTVECTV[31:24]			

.....continued											
Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
40	INTVECTV1	7:0								INTVECTV[7:0]	
		15:8								INTVECTV[15:8]	
		23:16									INTVECTV[23:16]
		31:24									INTVECTV[31:24]
44	INTVECTV2	7:0								INTVECTV[7:0]	
		15:8								INTVECTV[15:8]	
		23:16									INTVECTV[23:16]
		31:24									INTVECTV[31:24]
48	INTVECTV3	7:0								INTVECTV[7:0]	
		15:8								INTVECTV[15:8]	
		23:16									INTVECTV[23:16]
		31:24									INTVECTV[31:24]
0x4C ... 0x5B	Reserved										
0x5C	HASHKEY0	7:0								HASHKEY[7:0]	
		15:8								HASHKEY[15:8]	
		23:16									HASHKEY[23:16]
		31:24									HASHKEY[31:24]
0x60	HASHKEY1	7:0								HASHKEY[7:0]	
		15:8								HASHKEY[15:8]	
		23:16									HASHKEY[23:16]
		31:24									HASHKEY[31:24]
0x64	HASHKEY2	7:0								HASHKEY[7:0]	
		15:8								HASHKEY[15:8]	
		23:16									HASHKEY[23:16]
		31:24									HASHKEY[31:24]
0x68	HASHKEY3	7:0								HASHKEY[7:0]	
		15:8								HASHKEY[15:8]	
		23:16									HASHKEY[23:16]
		31:24									HASHKEY[31:24]
0x6C	GHASH0	7:0								GHASH[7:0]	
		15:8								GHASH[15:8]	
		23:16									GHASH[23:16]
		31:24									GHASH[31:24]
0x70	GHASH1	7:0								GHASH[7:0]	
		15:8								GHASH[15:8]	
		23:16									GHASH[23:16]
		31:24									GHASH[31:24]
0x74	GHASH2	7:0								GHASH[7:0]	
		15:8								GHASH[15:8]	
		23:16									GHASH[23:16]
		31:24									GHASH[31:24]
0x78	GHASH3	7:0								GHASH[7:0]	
		15:8								GHASH[15:8]	
		23:16									GHASH[23:16]
		31:24									GHASH[31:24]
0x7C ... 0x7F	Reserved										
80	CIPLN	7:0								CIPLN[7:0]	
		15:8								CIPLN[15:8]	
		23:16									CIPLN[23:16]
		31:24									CIPLN[31:24]
0x84	RANDSEED	7:0								RANDSEED[7:0]	
		15:8								RANDSEED[15:8]	
		23:16									RANDSEED[23:16]
		31:24									RANDSEED[31:24]

## 36.8 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

[36.5.8. Register Access Protection](#)

### 36.8.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**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					CTYPE[3:0]			
Reset					R/W	R/W	R/W	R/W
					0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access		XORKEY	KEYGEN	LOD	STARTMODE	CIPHER	KEYSIZE[1:0]	
Reset		R/W	R/W	R/W	R/W	R/W	R/W	R/W
		0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	CFBS[2:0]			AESMODE[2:0]			ENABLE	SWRST
Reset	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	0	0	0	0	0	0	0	0

#### Bits 19:16 – CTYPE[3:0] Counter Measure Type

Value	Name	Description
xxx0	CTYPE1 disabled	Countermeasure1 disabled
xxx1	CTYPE1 enabled	Countermeasure1 enabled
xx0x	CTYPE2 disabled	Countermeasure2 disabled
xx1x	CTYPE2 enabled	Countermeasure2 enabled
x0xx	CTYPE3 disabled	Countermeasure3 disabled
x1xx	CTYPE3 enabled	Countermeasure3 enabled
0xxx	CTYPE4 disabled	Countermeasure4 disabled
1xxx	CTYPE4 enabled	Countermeasure4 enabled

#### Bit 14 – XORKEY XOR Key Operation

Value	Description
0	No effect
1	The user keyword gets XORed with the previous keyword register content.

#### Bit 13 – KEYGEN Last Key Generation

Value	Description
0	No effect
1	Start Computation of the last NK words of the expanded key

#### Bit 12 – LOD Last Output Data Mode

Value	Description
0	No effect
1	Start encryption in Last Output Data mode

#### Bit 11 – STARTMODE Start Mode Select

Value	Name	Description
0	Manual Mode	Start Encryption / Decryption in Manual mode
1	Auto Mode	Start Encryption / Decryption in Auto mode

#### Bit 10 – CIPHER Cipher Mode Select

Value	Description
0	Decryption
1	Encryption

#### Bits 9:8 – KEYSIZE[1:0] Encryption Key Size

Value	Name	Description
0	128-bit Key	128-bit Key for Encryption / Decryption
1	192-bit Key	192-bit Key for Encryption / Decryption
2	256-bit Key	256-bit Key for Encryption / Decryption
3	Reserved	Reserved

#### Bits 7:5 – CFBS[2:0] Cipher Feedback Block Size

Value	Name	Description
0	128-bit data block	128-bit Input data block for Encryption/Decryption in Cipher Feedback mode
1	64-bit data block	64-bit Input data block for Encryption/Decryption in Cipher Feedback mode
2	32-bit data block	32-bit Input data block for Encryption/Decryption in Cipher Feedback mode
3	16-bit data block	16-bit Input data block for Encryption/Decryption in Cipher Feedback mode
4	8-bit data block	8-bit Input data block for Encryption/Decryption in Cipher Feedback mode
5–7	Reserved	Reserved

#### Bits 4:2 – AESMODE[2:0] AES Modes of Operation

Value	Name	Description
0	ECB	Electronic code book mode
1	CBC	Cipher block chaining mode
2	OFB	Output feedback mode
3	CFB	Cipher feedback mode
4	Counter	Counter mode
5	CCM	CCM mode
6	GCM	Galois counter mode
7	Reserved	Reserved

#### 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 AES module to their initial state, and the module will be disabled.

Writing a '1' to *SWRST* will always take precedence, meaning that all other writes in the same write operation will be discarded.

Value	Description
0	There is no reset operation ongoing
1	The reset operation is ongoing

## 36.8.2 Control B

**Name:** CTRLB  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
					GFMUL	EOM	NEWMSG	START
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

### Bit 3 – GFMUL GF Multiplication

This bit is applicable only to GCM mode.

Value	Description
0	No action
1	Setting this bit calculates GF multiplication with data buffer content and hashkey register content.

### Bit 2 – EOM End of Message

This bit is applicable only to GCM mode.

Value	Description
0	No action
1	Setting this bit generates final GHASH value for the message.

### Bit 1 – NEWMSG New Message

This bit is used in cipher block chaining (CBC), cipher feedback (CFB) and output feedback (OFB), counter (CTR) modes to indicate the hardware to use Initialization vector for encrypting the first block of message.

Value	Description
0	No action
1	Setting this bit indicates start of new message to the module.

### Bit 0 – START Start Encryption/Decryption

Value	Description
0	No action
1	Start encryption / decryption in manual mode.



### 36.8.3 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x05  
**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
							GFMCMP	ENCCMP
Access							R/W	R/W
Reset							0	0

#### Bit 1 – GFMCMP GF Multiplication Complete Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the GF Multiplication Complete Interrupt Enable bit, which disables the GF Multiplication Complete interrupt.

Value	Description
0	The GF Multiplication Complete interrupt is disabled.
1	The GF Multiplication Complete Complete interrupt is enabled.

#### Bit 0 – ENCCMP Encryption Complete Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Encryption Complete Interrupt Enable bit, which disables the Encryption Complete interrupt.

Value	Description
0	The Encryption Complete interrupt is disabled.
1	The Encryption Complete Complete interrupt is enabled.

### 36.8.4 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x06  
**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
							GFMCMP	ENCCMP
Access							R/W	R/W
Reset							0	0

#### Bit 1 - GFMCMP GF Multiplication Complete Interrupt Enable

Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the GF Multiplication Complete Interrupt Enable bit, which enables the GF Multiplication Complete interrupt.

Value	Description
0	The GF Multiplication Complete interrupt is disabled.
1	The GF Multiplication Complete interrupt is enabled.

#### Bit 0 - ENCCMP Encryption Complete Interrupt Enable

Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Encryption Complete Interrupt Enable bit, which enables the Encryption Complete interrupt.

Value	Description
0	The Encryption Complete interrupt is disabled.
1	The Encryption Complete interrupt is enabled.

### 36.8.5 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x07  
**Reset:** 0x00

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

#### Bit 1 – GFMCMP GF Multiplication Complete

This flag is cleared by writing a '1' to it.

This flag is set when GHASH value is available on the Galois Hash Registers (GHASH<sub>x</sub>) in GCM mode.

Writing a '0' to this bit has no effect.

This flag is also automatically cleared in the following cases.

1. Manual encryption/decryption occurs (START in CTRLB register).
2. Reading from the GHASH<sub>x</sub> register.

#### Bit 0 – ENCCMP Encryption Complete

This flag is cleared by writing a '1' to it.

This flag is set when encryption/decryption is complete and valid data is available on the Data Register.

Writing a '0' to this bit has no effect.

This flag is also automatically cleared in the following cases:

1. Manual encryption/decryption occurs (START in CTRLA register). (This feature is needed only if we do not support double buffering of INDATA registers).
2. Reading from the data register (INDATA<sub>x</sub>) when LOD = 0.
3. Writing into the data register (INDATA<sub>x</sub>) when LOD = 1.
4. Reading from the Hash Key register (HASHKEY<sub>x</sub>).

### 36.8.6 Data Buffer Pointer

**Name:** DATABUFPTR  
**Offset:** 0x08  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

#### Bits 1:0 – INDATAPTR[1:0] Input Data Pointer

Writing to this field changes the value of the input data pointer, which determines which of the four data registers is written to/read from when the next write/read to the `INDATA` register address is performed.

### 36.8.7 Debug

**Name:** DBGCTRL  
**Offset:** 0x09  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

#### Bit 0 - DBGRUN Debug Run

Writing a '0' to this bit causes the AES to halt during debug mode.

Writing a '1' to this bit allows the AES to continue normal operation during debug mode. This bit can only be changed while the AES is disabled.

### 36.8.8 Keyword

**Name:** KEYWORD  
**Offset:** 0x0C + n\*0x04 [n=0..7]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	KEYWORD[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
	KEYWORD[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
	KEYWORD[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
	KEYWORD[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

#### Bits 31:0 – KEYWORD[31:0] Key Word Value

The four/six/eight 32-bit Key Word registers set the 128-bit/192-bit/256-bit cryptographic key used for encryption/decryption. KEYWORD0 .KEYWORD corresponds to the first word of the key and KEYWORD3/KEYWORD5/KEYWORD7 .KEYWORD to the last one.

**Note:** By setting the XORKEY bit of CTRLA register, keyword will update with the resulting XOR value of user keyword and previous keyword content.

### 36.8.9 Data

**Name:** INDATA  
**Offset:** 0x38  
**Reset:** 0x00000000

Bit	31	30	29	28	27	26	25	24
	INDATA[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
	INDATA[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
	INDATA[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
	INDATA[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 – INDATA[31:0] Data Value

A write to or read from this register corresponds to a write to or read from one of the four data registers. The four 32-bit Data registers set the 128-bit data block used for encryption/decryption. The data register that is written to or read from is given by the `DATABUFPTR.INDATPTR` field.

**Note:** Both input and output shares the same data buffer. Reading INDATA register will return 0's when AES is performing encryption or decryption operation.

### 36.8.10 Initialization Vector Register

**Name:** INTVECTV  
**Offset:** 0x3C + n\*0x04 [n=0..3]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	INTVECTV[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
	INTVECTV[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
	INTVECTV[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
	INTVECTV[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

#### Bits 31:0 – INTVECTV[31:0] Initialization Vector Value

The four 32-bit Initialization Vector registers  $INTVECTV_n$  set the 128-bit Initialization Vector data block that is used by some modes of operation as an additional initial input.  $INTVECTV_0$  .  $INTVECTV$  corresponds to the first word of the Initialization Vector,  $INTVECTV_3$  .  $INTVECTV$  to the last one. These registers are write-only to prevent the Initialization Vector from being read by another application. For CBC, OFB, and CFB modes, the Initialization Vector corresponds to the initialization vector. For CTR mode, it corresponds to the counter value.



### 36.8.11 Hash Key (GCM mode only)

**Name:** HASHKEY  
**Offset:** 0x5C + n\*0x04 [n=0..3]  
**Reset:** 0x00000000  
**Property:** PAC Write-protection

Bit	31	30	29	28	27	26	25	24
	HASHKEY[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
	HASHKEY[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
	HASHKEY[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
	HASHKEY[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 – HASHKEY[31:0] Hash Key Value

The four 32-bit `HASHKEY` registers contain the 128-bit Hash Key value computed from the AES KEY. The Hash Key value can also be programmed offering single GF128 multiplication possibilities.

### 36.8.12 Galois Hash (GCM mode only)

**Name:** GHASH  
**Offset:** 0x6C + n\*0x04 [n=0..3]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	GHASH[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
	GHASH[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
	GHASH[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
	GHASH[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 – GHASH[31:0] Galois Hash Value

The four 32-bit Hash Word registers `GHASH` contain the `GHASH` value after GF128 multiplication in GCM mode. Writing a new key to `KEYWORD` registers causes `GHASH` to be initialized with zeroes. These registers can also be programmed.

### 36.8.13 Galois Hash x (GCM mode only)

**Name:** CIPLN  
**Offset:** 0X80  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	CIPLN[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
	CIPLN[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
	CIPLN[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
	CIPLN[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 – CIPLN[31:0] Cipher Length

This register contains the length in bytes of the Cipher text that is to be processed. This is programmed by the user in GCM mode for Tag generation.

### 36.8.14 Random Seed

**Name:** RANDSEED  
**Offset:** 0x84  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	RANDSEED[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
	RANDSEED[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
	RANDSEED[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
	RANDSEED[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 – RANDSEED[31:0] Random Seed

A write to this register corresponds to loading a new seed into the Random number generator.

## 37. Public Key Cryptography Controller (PUKCC)

### 37.1 Overview

The Public Key Cryptography Controller (PUKCC) processes public key cryptography algorithm calculus in both  $GF(p)$  and  $GF(2n)$  fields.

The Public Key Cryptography Library (PUKCL) is stored in ROM inside the device. The library can be used in applications to access features of PUKCC, and includes the complete implementation of the following public key cryptography algorithms:

- RSA (Rivest-Shamir-Adleman public key cryptosystem), DSA (Digital Signature Algorithm):
  - Modular Exponentiation with CRT up to 7168 bits
  - Modular Exponentiation without CRT up to 5376 bits
  - Prime generation
  - Utilities: GCD/modular Inverse, Divide, Modular reduction, Multiply, ...
- Elliptic Curves:
  - ECDSA  $GF(p)$  up to 521 bits for common curves (up to 1120 bits for future use)
  - ECDSA  $GF(2n)$  up to 571 bits for common curves (up to 1440 bits for future use)
  - Choice of the curve parameters for compatibility with NIST Curves or other curves in Weierstrass equation
  - Point Multiply
  - Point Add/Doubling
  - Other high level elliptic curve algorithms (ECDH, ...) can be implemented by user using library functions
- Deterministic Random Number Generation (DRNG ANSI X9.31) for DSA

### 37.2 Product Dependencies

#### 37.2.1 I/O Lines

Not applicable.

#### 37.2.2 Power Management

The PUKCC will continue to operate in any sleep mode, as long as its source clock is running.

#### 37.2.3 Clocks

The bus clock (PB2\_CLK) can be enabled and disabled by the CRU.

#### 37.2.4 DMA

Not applicable.

#### 37.2.5 Interrupts

Not applicable.

#### 37.2.6 Events

Not applicable.

## 37.3 Functional Description

### 37.3.1 Public Key Cryptography Library (PUKCL) Application Programming Interface (API)

The Public Key Cryptography Controller (PUKCC) is a peripheral that can be used to accelerate public key cryptography, and processes public key cryptography algorithm calculus in both Prime field ( $GF(p)$ ) and Binary field ( $GF(2^n)$ ). Different functionalities of the PUKCC are accessed with the help of the Public Key Cryptography Library (PUKCL), which is embedded into a dedicated ROM inside the microcontroller.

The PUKCL provides access to many algorithms and functions. The features provided, start from basic addition or comparison, up to the RSA or ECDSA complete computation. The library can be utilized by including the PUKCL Driver in the application and passing parameters through a common Application Programming Interface (API). The PUKCC Driver is available in Harmony 3. This library can be used in conjunction with a SSL software stack to improve performance and helps to reduce the RAM usage and time taken to perform different cryptographic functions.

### 37.3.2 PUKCL Features

PUKCL features include:

- [37.3.4. Basic Arithmetic and Cryptographic Services](#) - PUKCL self-test, GCD, integral division, etc.
- [37.3.5. Modular Arithmetic Services](#) - Modular reduction, modular exponentiation, probable prime generation and modular exponentiation
- [37.3.6. Elliptic Curves Over  \$GF\(p\)\$  Services](#) - Point addition and doubling on an elliptic curve in a prime field, ECDSA signature generation and verification on an elliptic curve over  $GF(p)$
- [37.3.7. Elliptic Curves Over  \$GF\(2^n\)\$  Services](#) - Point addition and doubling on an elliptic curve in a prime field, ECDSA signature generation and verification on an elliptic curve over  $GF(2^n)$

### 37.3.3 PUKCL Usage

The following sections provide details on accessing the PUKCL and its features.

#### 37.3.3.1 Initializing the PUKCC and PUKCL

For a project created with Harmony 3, the clock initialization is handled by the initialization function `CLK_Initialize()`. After a power-on reset, and when the PUKCC Clock is enabled, a Crypto RAM clear process is launched. It is mandatory to wait until the end of this process before using the Crypto Library.

The following code shows how to wait for the Crypto RAM clear process.

```
while ((PUKCCSR & BIT_PUKCCSR_CLRRAM_BUSY) != 0);
```

The next task to be done is self-test. From the generated project in Harmony 3, copy the example for the PUKCC Driver SelfTest and add it to the main source file. This is a mandatory step before using the library. The return values from the SelfTest service must be compared against known values mentioned in the service description (see the **Description** section in [37.3.4.1. SelfTest](#)).

#### Example 37-1. PUKCC Initialization

```
void PUKCC_self_test(void)
{
    // Clear contents of PUKCLParam
    memset(&PUKCLParam, 0, sizeof(PUKCL_PARAM));

    pvPUKCLParam = &PUKCLParam;
    vPUKCL_Process(SelfTest, pvPUKCLParam);

    // In case of error, loop here
    while (PUKCL(u2Status) != PUKCL_OK) {
        ;
    }
}
```

```

    }
    while (pvPUKCLParam->P.PUKCL_SelfTest.u4Version != PUKCL_VERSION) {
    ;
    }
    while (pvPUKCLParam->P.PUKCL_SelfTest.u4CheckNum1 != 0x6E70DDD2) {
    ;
    }
    while (pvPUKCLParam->P.PUKCL_SelfTest.u4CheckNum2 != 0x25C8D64F) {
    ;
    }
}

int main(void)
{
    /* Initializes MCU, drivers and middleware */
    SYS_Initialize();

    // Wait for Crypto RAM clear process
    while ((PUKCCSR & BIT_PUKCCSR_CLRRAM_BUSY) != 0);

    // Initialize PUKCC and perform self test
    PUKCC_self_test();
    while(1)
    {
    }
}

```

**Note:** It may also be necessary to initialize the Random Number Generator (RNG) on the microcontroller, as some services in the library use the peripheral. Before calling such services, be sure to follow the directives given for random number generation on the selected microcontroller (particularly initialization and seeding) and compulsorily start the RNG. For details refer to each service.

### 37.3.3.2 Accessing Different Library Services

All cryptographic services in the library are accessed by the macro `vPUKCL_Process`. All of these services use the same process for receiving and returning parameters. PUKCL receives two arguments: the requested service and a pointer to a structure called the parameter block. The parameter block contains two structures, a common parameter structure for all commands and specific parameter structure for each service. A specific service is accessed with `vPUKCL_Process` by passing the service name as the first argument. For example, to perform `SelfTest`, use `vPUKCL_Process(SelfTest, pvPUKCLParam)`.

#### Example 37-2. PUKCL Parameter Block

```

typedef struct _PUKCL_param {
    PUKCL_HEADER PUKCL_Header;
    union {
        _PUKCL_CLEARFLAGS PUKCL_ClearFlags;
        _PUKCL_COMP PUKCL_Comp;
        _PUKCL_CONDCOPY PUKCL_CondCopy;
        _PUKCL_CRT PUKCL_CRT;
        _PUKCL_DIV PUKCL_Div;
        _PUKCL_EXPMOD PUKCL_ExpMod;
        _PUKCL_FASTCOPY PUKCL_FastCopy;
        _PUKCL_FILL PUKCL_Fill;
        _PUKCL_FMULT PUKCL_Fmult;
        _PUKCL_GCD PUKCL_GCD;
        _PUKCL_PRIMEGEN PUKCL_PrimeGen;
        _PUKCL_REDMOD PUKCL_RedMod;
        _PUKCL_RNG PUKCL_Rng;
        _PUKCL_SELFTEST PUKCL_SelfTest;
        _PUKCL_SMULT PUKCL_Smult;
        _PUKCL_SQUARE PUKCL_Square;
        _PUKCL_SWAP PUKCL_Swap;

        // ECC
        _PUKCL_ZPECCADD PUKCL_ZpEccAdd;
        _PUKCL_ZPECCDBL PUKCL_ZpEccDb1;
        _PUKCL_ZPECCADDSUB PUKCL_ZpEccAddSub;
        _PUKCL_ZPECCMUL PUKCL_ZpEccMul;
        _PUKCL_ZPECCSAGENERATE PUKCL_ZpEcDsaGenerate;
    }
}

```

```

_PUKCL_ZPECDSAVERIFY          PUKCL_ZpEcDsaVerify;
_PUKCL_ZPECDSAQUICKVERIFY     PUKCL_ZpEcDsaQuickVerify;
_PUKCL_ZPECCQUICKDUALMUL      PUKCL_ZpEccQuickDualMul;
_PUKCL_ZPECCONVPROJTOAFFINE   PUKCL_ZpEcConvProjToAffine;
_PUKCL_ZPECCONVAFFINETOPROJECTIVE PUKCL_ZpEcConvAffineToProjective;
_PUKCL_ZPECRANDOMIZECOORDINATE PUKCL_ZpEcRandomiseCoordinate;
_PUKCL_ZPECPOINTISONCURVE     PUKCL_ZpEcPointIsOnCurve;

// ECC
_PUKCL_GF2NECCADD             PUKCL_GF2NEccAdd;
_PUKCL_GF2NECCDBL            PUKCL_GF2NEccDb1;
_PUKCL_GF2NECCMUL            PUKCL_GF2NEccMul;
_PUKCL_GF2NECDSAGENERATE     PUKCL_GF2NEcDsaGenerate;
_PUKCL_GF2NECDSAVERIFY       PUKCL_GF2NEcDsaVerify;
_PUKCL_GF2NECCONVPROJTOAFFINE PUKCL_GF2NEcConvProjToAffine;
_PUKCL_GF2NECCONVAFFINETOPROJECTIVE PUKCL_GF2NEcConvAffineToProjective;
_PUKCL_GF2NECRANDOMIZECOORDINATE PUKCL_GF2NEcRandomiseCoordinate;
_PUKCL_GF2NECPOINTISONCURVE   PUKCL_GF2NEcPointIsOnCurve;
} P;
} PUKCL_PARAM,

```

### 37.3.3.2.1 PUKCL\_HEADER Structure

The PUKCL\_HEADER is common for all services of the library. This header includes standard fields to indicate the requested service, sub-service, options, return status, and so on, as shown in the following tables.

Different terms used in the below description to be understood, are as follows:

- Parameter – Represents a variable used by the PUKCL. Every parameter belongs to either PUKCL\_HEADER or PUKCL Service Specific Header
- Type – Indicates the data type. For details on data type, please refer to CryptoLib\_typedef\_pb.h file in the library
- Dir – Direction. Indicates whether PUKCL considers the variable as input or output. Input means that the application passes data to the PUKCL using the variable. Output means that the PUKCL uses the variable to pass data to the application.
- Location – Suggests whether the parameter need to be stored in Crypto RAM or device SRAM. The PUKCL driver has macros for placing parameters into Crypto RAM, so that the user does not have to worry about the addresses
- Data Length – If a parameter is a pointer variable, the Data Length column shows the size of the data pointed by the pointer

**Table 37-1.** PUKCL\_HEADER Structure

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u1Service	u1	I	-	-	Required service	Executed service
u1SubService	u1	I	-	-	Required sub-service	Executed sub-service
u2Option	u2	I	-	-	Required option	Executed option
Specific	PUKCL_STATUS	I/O	-	-	See the following table PUKCL_STATUS Structure	See the following table PUKCL_STATUS Structure
u2Status	u2	I/O	-	-	-	Output Status
Reserved	u2	-	-	-	-	-
Reserved	u4	-	-	-	-	-

The Specific field in the PUKCL\_HEADER structure is another structure named PUKCL\_STATUS. The following table describes this structure. The details of the use of these bits are provided in the individual service descriptions.

### 37.3.3.2.2 PUKCL\_STATUS Structure

Members of the PUKCL\_STATUS structure are shown in the following table.



**Table 37-2. PUKCL\_STATUS Structure**

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
CarryIn (see <b>Note 1</b> )	bit	I	-	-	CarryIn	-
CarryOut	bit	O	-	-	-	CarryOut
Zero	bit	O	-	-	-	1: Result is zero 0: Result is not zero
Gf2n (see <b>Note 1</b> )	bit	I	-	-	Mathematical field 0: Integers ( $Z_p$ ) 1: Field $GF(2^n)$	-
Violation	bit	O	-	-	-	Indicates a violation

**Note:**

- Two of these fields must be filled in to avoid problems during computations. If the Gf2n and CarryIn fields are not reset or initialized properly, problems may be encountered during computations. For instance, not initializing the Gf2n field may result in getting a correct mathematical result, but computed over  $GF(2^n)$  instead of  $Z_p$ .

### 37.3.3.2.3 PUKCL Service Specific Header

Details about each service specific header are provided with service descriptions in a subsequent section. Such structures may contain input or output parameters. A parameter is considered as an input parameter when it used for passing information to the PUKCL, and it is considered as an output parameter when the PUKCL uses it to pass a result back to the application code.

The following code provides the service specific header example for the SelfTest service.

```
typedef struct _PUKCL_selftest {
    u4 u4Version;
    u4 u4PUKCCVersion;
    u4 u4CheckNum1;
    u4 u4CheckNum2;
    u1 u1Step;
} _PUKCL_SELFTEST;
```

After the SelfTest service is invoked (with `vPUKCL_Process(SelfTest, pvPUKCLParam)`), the service specific return values can be checked using `pvPUKCLParam`.

To check whether the version returned by the PUKCL is correct, the following code can be used.

```
while (pvPUKCLParam->P.PUKCL_SelfTest.u4Version != PUKCL_VERSION);
```

In a similar way, other returns can also be accessed.

### 37.3.3.3 Parameter Passing (Special Considerations)

Most of the PUKCL services work with memory area and accept pointers and lengths as parameters to define input and output areas. Most of the time, the pointers and lengths are untouched by the services, while the defined areas are read, filled, or overwritten. These memory areas are defined with an initial pointer and a byte length. For most of the commands, the memory area location must be in the PUKCC Cryptographic RAM. The Cryptographic RAM is the memory area for parameter exchange with the PUKCL and is 4 Kbytes large. Sometimes memory areas can be located in Embedded SRAM, which is detailed in the Location column of the parameters description tables.

When working with binary fields, polynomials in  $GF(2^n)$  need no transformation to be written in an area:

- Each bit represents a polynomial coefficient 0 or 1
- The polynomials must be written Low Significant Byte First
- A zero padding on the Most Significant Bytes may be added if the area is larger than the real size of the polynomial



**Table 37-4.** Return Codes

Value for Bits 00–13	Severity Code	Reason Code
0x0000	—	PUKCL_OK
0x4001	Informative	PUKCL_NUMBER_IS_NOT_PRIME
0x4002	Informative	PUKCL_NUMBER_IS_PRIME
0xC001	Severe	PUKCL_COMPUTATION_NOT_STARTED
0xC002	Severe	PUKCL_UNKNOWN_SERVICE
0xC003	Severe	PUKCL_UNEXPLOITABLE_OPTIONS
0xC004	Severe	PUKCL_HARDWARE_ISSUE
0xC005	Severe	PUKCL_WRONG_HARDWARE
0xC006	Severe	PUKCL_LIBRARY_MALFORMED
0xC007	Severe	PUKCL_ERROR
0xC008	Severe	PUKCL_UNKNOWN_SUBSERVICE
0xC101	Severe	PUKCL_DIVISION_BY_ZERO
0xC102	Severe	PUKCL_MALFORMED_MODULUS
0xC103	Severe	PUKCL_FAULT_DETECTED
0xC104	Severe	PUKCL_MALFORMED_KEY

Please note the following rules about return codes:

- A status value indicating a severe error, means that an expected operation has not been executed or has been corrupted. Therefore, the result of such an operation must not be used.
- A status value indicating a warning must be looked at precisely, as the expected correctness of the result cannot be guaranteed.
- A status value indicating an information always means that the result is correct with no possible misinterpretation of the values.
- A status value zero indicates that there is no error or no severity.

In the following sections, for each service, the constraints on the parameters placement are detailed. For reduced code size and higher execution speed, tests are processed on these constraints. It is important that PUKCL users take these placement constraints into consideration at the development and test stages to ensure the correct functioning of the library.

### 37.3.4 Basic Arithmetic and Cryptographic Services

#### 37.3.4.1 SelfTest

##### 37.3.4.1.1 Purpose

This service is used to initialize the PUKCL. It resets the PUKCC, clears the Crypto RAM, and returns the library and PUKCC version numbers.

It must be called before using any other services in the library and the user must verify the return status at the end of the service execution.

##### 37.3.4.1.2 How to Use the Service

##### 37.3.4.1.3 Description

This service processes internal tests and returns information and status codes as described in [37.3.4.1.7. Status Returned Values](#). The service name for this operation is `SelfTest`.

##### 37.3.4.1.4 Parameters Definition

It is possible to directly address this service through the `PUKCL_SelfTest()` macro.

**Table 37-5.** SelfTest Service Parameters

Parameter	Type	Dir.	Location	Data Length	Before Executing the Service	After Executing the Service
u4Version	u4	O	-	-	-	PUKCL version
u4PUKCCVersion	u4	O	-	-	-	PUKCC Version
u4CheckNum1	u4	O	-	-	-	Test result value 1
u4CheckNum2	u4	O	-	-	-	Test result value 2
u1Step	u1	O	-	-	-	Latest correctly executed step

### 37.3.4.1.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library
vPUKCL_Process(SelfTest,pvPUKCLParam);

if (PUKCL(u2Status) == PUKCL_OK)
{
    // The Library version is available
    // in PUKCL_SelfTest(u4Version)
    // The PUKCL version is available
    // in PUKCL_SelfTest(u4PUKCCVersion)
}

```

### 37.3.4.1.6 Returned Values

The expected u4Version value depends on the version of PUKCL being used, and the u4PUKCCVersion value depends on the version of PUKCC being used.

The expected u4CheckNum1 value is 0x6e70ddd2 and the expected one for u4CheckNum2 is 0x25c8d64f. The expected final u1Step value is 3.

### 37.3.4.1.7 Status Returned Values

**Table 37-6.** SelfTest Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly.
PUKCL_ERROR	Severe	An issue has been encountered.

### 37.3.4.2 Clear Flags

#### 37.3.4.2.1 Purpose

This service can be used to clear parameter structure flags.

#### 37.3.4.2.2 How to Use the Service

#### 37.3.4.2.3 Description

This service clears CarryOut, CarryIn, Zero and Violation flags in the Specific bit field. The Gf2n flag is untouched.

The service name for this operation is ClearFlags.

#### 37.3.4.2.4 Parameters Definition

It is possible to directly address this service through the PUKCL\_ClearFlags() macro.

**Table 37-7.** Clear Flags Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
Specific/CarryOut	Bit	O	-	-	-	Cleared
Specific/CarryIn	Bit	O	-	-	-	Cleared
Specific/Zero	Bit	O	-	-	-	Cleared

.....continued

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
Specific/Violation	Bit	O	-	-	-	Cleared

### 37.3.4.2.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ClearFlags,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    // Success
}
else // Manage the error

```

### 37.3.4.2.6 Status Returned Values

Table 37-8. ClearFlags Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly.

### 37.3.4.3 Swap

#### 37.3.4.3.1 Purpose

This service performs swapping of two buffers.

#### 37.3.4.3.2 How to Use the Service

#### 37.3.4.3.3 Description

This service swaps two buffers, X and Y, of the same size in memory.

The service name for this operation is `Swap`.

#### 37.3.4.3.4 Parameters Definition

This service can easily be accessed through the use of the `PUKCL_Swap()` macro.

Table 37-9. Swap Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1XBase	nu1	I	Crypto RAM	u2Length	Base of the number X	Base of X filled with Y
nu1YBase	nu1	I	Crypto RAM	u2Length	Base of the number Y	Base of Y filled with X
u2XLength	u2	I	-	-	Length of X and Y	Length of X and Y

### 37.3.4.3.5 Code Example

```

_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// Initialize parameters
PUKCL_Swap(nu1XBase) = <Base of the X number>;
PUKCL_Swap(nu1YBase) = <Base of the Y number>;
PUKCL_Swap(u2XLength) = <Length of the numbers>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Swap,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.4.3.6 Constraints

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase or nu1YBase are not aligned on 32-bit boundaries
- u2XLength is either <4, >0xffc, or not a 32-bit length
- {nu1XBase, u2XLength} or {nu1YBase, u2XLength} do not entirely lie in PUKCCRAM
- {nu1XBase, u2XLength} overlaps {nu1YBase, u2YLength}

### 37.3.4.3.7 Status Returned Values

**Table 37-10.** Swap Service Return Codes

Returned status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly

### 37.3.4.4 Fill

#### 37.3.4.4.1 Purpose

This service performs a memory fill operation, with a given 32-bit constant.

#### 37.3.4.4.2 How to Use the Service

#### 37.3.4.4.3 Description

This service fills a Crypto RAM space with a provided 32-bit constant: Fill (R, FillValue)

The service name for this operation is Fill.

#### 37.3.4.4.4 Parameters Definition

This service can easily be accessed through the use of the PUKCL\_Fill() macro.

**Table 37-11.** Fill Service Parameters

Parameter	Type	Direction.	Location	Data Length	Before Executing the Service	After Executing the Service
nu1RBase	nu1	I	Crypto RAM	u2RLength	Base of R	Base of R value filled repetitively with u4FillValue
u2RLength	u2	I	Crypto RAM	-	Length of R	Length of R
u4FillValue	u4	I	-	-	Filling value	Filling value

### 37.3.4.4.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PUPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// Initialize parameters
PUKCL_Fill(nu1RBase) = <Base of the R number>;
PUKCL_Fill(u2RLength) = <Length of the R number>;
PUKCL_Fill(u4FillValue) = <32-bits value to fill with>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Fill, pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.4.4.6 Constraints

The following conditions must be avoided to ensure that the service works correctly:

- nu1RBase are not aligned on 32-bit boundaries
- u2RLength is either: <4, >0xffc or not a 32-bit length

- {nu1RBase, u2RLength} do not entirely lie in Crypto RAM

### 37.3.4.4.7 Status Returned Values

**Table 37-12.** Fill Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly.

### 37.3.4.5 Fast Copy/Clear

#### 37.3.4.5.1 Purpose

This service performs a copy from a memory area to another or a memory area clear.

#### 37.3.4.5.2 How to Use the Service

#### 37.3.4.5.3 Description

This service copies a number X into another number R, padding with zero on the MSB side up to the length specified for R.

$$R = X$$

If the lengths of R and X are equal, a complete fast copy is processed.

If the length of R is strictly greater than the length of X, X is first copied in the Low Significant Bytes side of R, and R is padded with zeros on the Most Significant Bytes side.

If the pointer on the X area equals zero, R is filled with zeros. This operation can also be made by using the Fill service (see 37.3.4.4. Fill).

The service name for this operation is `FastCopy`.



**Important:** The length of R must be greater or equal to the length of X.

#### 37.3.4.5.4 Parameters Definition

This service can easily be accessed through the use of the `PUKCL_FastCopy()` macro.

**Table 37-13.** FastCopy Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1XBase	nu1	I	Crypto RAM	u2XLength	Base of X	Base of X number untouched
nu1RBase	nu1	I	Crypto RAM	u2RLength	Base of R	Base of R filled with X
u2RLength	u2	I	-	-	Length of R	Length of R
u2XLength	u2	I	-	-	Length of X	Length of X

#### 37.3.4.5.5 Code Example

```

PUKCL_PARAM PUKCLParam;
P_PUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// Initialize parameters
PUKCL_FastCopy(nu1XBase) = <Base of the X number>;
PUKCL_FastCopy(nu1RBase) = <Base of the R number>;
PUKCL_FastCopy(u2XLength) = <Length of the X number>;
PUKCL_FastCopy(u2RLength) = <Length of the R number>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(FastCopy, pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    ...
}

```

```
    }  
else // Manage the error
```

### 37.3.4.5.6 Constraints

The parameter placements that are not allowed are as follows.

If nu1XBase equals zero, no checks are made on nu1XBase (fixed) and u2XLength (unused).

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase or nu1RBase are not aligned on 32-bit boundaries
- u2XLength or u2RLength is either: <4, >0xffc or not a 32-bit length or u2XLength >u2RLength
- {nu1XBase, u2XLength} or {nu1RBase, u2RLength} do not entirely lie in Crypto RAM
- {nu1XBase, u2XLength} overlaps {nu1RBase, u2RLength}

### 37.3.4.5.7 Status Returned Values

**Table 37-14.** FastCopy Service Return Codes

Returned status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly

### 37.3.4.6 Conditional Copy/Clear

#### 37.3.4.6.1 Purpose

This service conditionally performs a copy from a memory area to another or a memory area clear.

#### 37.3.4.6.2 How to Use the Service

#### 37.3.4.6.3 Description

This service copies a number X into another number R, padding with zero on the MSB side up to the length specified for R. This copy operation is performed under the conditions specified in the options.

If the condition is verified,  $R = X$ .

The copy or clear action is made under condition.

The four possible options for the condition are described in the following table. Two of the conditions check the Specific.CarryIn bit.

The processing is done as follows:

- If the condition is not verified, nothing is processed.
- If the condition is verified the copy or clear follows the rules:
  - If the lengths of R and X are equal, a complete fast copy is processed
  - If the length of R is strictly greater than the length of X, X is first copied in the Low Significant Bytes side of R, and R is padded with zeros on the Most Significant Bytes side.
  - If the pointer on the X area equals zero, R is filled with zeros.

The service name for this operation is CondCopy.



**Important:** If the condition is verified, the length of R must be greater or equal to the length of X.

#### 37.3.4.6.4 Parameters Definition

This service can easily be accessed through the use of the PUKCL\_CondCopy () and PUKCL () macros.



**Table 37-15.** CondCopy Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	-	-	Option for condition (see the following table)	Option for condition (see the following table)
Specific/CarryIn	Bit	I	-	-	Bit CarryIn	Bit CarryIn
nu1XBase	nu1	I	Crypto RAM	u2XLength	Base of X	Base of X number untouched
nu1RBase	nu1	I	Crypto RAM	u2RLength	Base of R	Base of R filled with X if condition holds
u2RLength	u2	I	-	-	Length of R	Length of R
u2XLength	u2	I	-	-	Length of X	Length of X

### 37.3.4.6.5 Available Options

The option for the condition is set by the u2Options input parameter that must take one of the values listed in the following table.

**Table 37-16.** CondCopy Service Options

Option	Purpose	Needed parameters
PUKCL_CONDCOPY_ALWAYS	Always perform the copy	nu1XBase,u2XLength,nu1RBase, u2RLength
PUKCL_CONDCOPY_NEVER	Never perform the copy	None
PUKCL_CONDCOPY_IF_CARRY	Perform the copy if CarryIn is 1	Specific/CarryIn nu1XBase,u2XLength,nu1RBase, u2RLength
PUKCL_CONDCOPY_IF_NOT_CARRY	Perform the copy if CarryIn is zero	Specific/CarryIn nu1XBase,u2XLength,nu1RBase, u2RLength

### 37.3.4.6.6 Code Example

```

PUKCL_PARAM PUKCLParam;
PvPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// CarryIn shall be beforehand filled (with zero or one) PUKCL(Specific).CarryIn = ...;

// Condition Option PUKCL(u2Options) = ...;

// Initialize parameters
PUKCL_CondCopy(nu1XBase) = <Base of the X number>;
PUKCL_CondCopy(nu1RBase) = <Base of the R number>;
PUKCL_CondCopy(u2XLength) = <Length of the X number>;
PUKCL_CondCopy(u2RLength) = <Length of the R number>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(CondCopy,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.4.6.7 Constraints

The parameters placement that are not allowed are listed below.

If the conditional option and the CarryIn do not lead to execute the copy, no checks are made on the constraints to be respected.

If nu1XBase equals zero, no checks are made on nu1XBase (fixed) and u2XLength (unused).

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase or nu1RBase are not aligned on 32-bit boundaries
- u2XLength or u2RLength is either: <4, >0xffc or not a 32-bit length or u2XLength >u2RLength
- {nu1XBase, u2XLength} or {nu1RBase, u2RLength} do not entirely lie in Crypto RAM

- {nu1XBase, u2XLength} overlaps {nu1RBase, u2RLength}

### 37.3.4.6.8 Status Returned Values

**Table 37-17.** CondCopy Service Return Codes

Returned status	Importance	Meaning
PUKCL_WRONG_SERVICE	Severe	An inconsistency has been detected between the called service and the provided service number.
PUKCL_OK	-	Service functioned correctly

### 37.3.4.7 Small Multiply, Add, Subtract, Exclusive OR

#### Related Links

[37.3.4.5. Fast Copy/Clear](#)

[37.3.5.1. Modular Reduction](#)

#### 37.3.4.7.1 Purpose

This purpose of this service is to multiply a large number X by a single-word number, MulValue, and perform an optional accumulation/subtract with a large number Z, returning the result R.

The following options are available:

- Work in the GF(2<sup>n</sup>) or in the standard GF(p) arithmetic integer field
- Add of a supplemental CarryOperand
- Overlap of the operands is possible, taking into account some constraints
- Modulo-reduction of the computation result (see *Modular Reduction* from Related Links)

In addition to a multiply, possible uses of this service can include:

- Copy a block of data from one place to another (if u4MulValue is 1). This operation can alternatively be made by using the Fast Copy service (see *Fast Copy/Clear* from Related Links)
- Adding/Subtracting two numbers (if u4MulValue is 1)
- Xoring two blocks of data (if u4MulValue is 1 and the selected mathematical field is GF(2<sup>n</sup>))

#### 37.3.4.7.2 How to Use the Service

#### 37.3.4.7.3 Description

This service processes the following operation (if not computing a modular reduction of the result):

$$R = [Z] \pm (MulValue \times X + CarryOperand)$$

Or (if computing a modular reduction of the result):

$$R = ([Z] \pm (MulValue \times X + CarryOperand)) \bmod N$$

The service name for this operation is `Smult`.

The result of the Small Multiply Operation is stored on u2RLength bytes, so the choice of this length compared to u2XLength may lead to:

- A truncation if the result is too big to be stored on u2RLengthbytes.
- A padding on the MSB side if the result does not take all the u2RLengthbytes.  
However, in all cases this rule must be followed:



**Important:** The length of R must be greater than or equal to the length of X.

In these computations, the following parameters need to be provided:

- R the result (pointed by {nu1RBase,u2Rlength})
- X one input number or  $GF(2^n)$  polynomial (pointed by {nu1XBase,u2XLength})
- Z one optional input number or  $GF(2^n)$  polynomial (pointed by {nu1ZBase,u2Rlength}).
- MulValue one input number or  $GF(2^n)$  polynomial on one word (provided in u4MulValue)
- CarryOperand (provided through the CarryOptions and Carry values).



**Important:** Even if neither accumulation nor subtraction is specified, the nu1ZBase must always be filled and point to a Crypto RAM space. In this case, nu1ZBase can point to the same space as the nu1RBase.

If using the modular reduction option, the Multiply operation is followed by a reduction (see *Modular Reduction* from Related Links) and the following parameters must be additionally provided:

- N—the modulus (pointed by {nu1ModBase,u2Modlength +4})
- Cns—the reduction constant
  - In case of Big reduction, Cns is pointed by {nu1CnsBase,64bytes}.
  - In case of Fast or Normalized reduction, Cns is pointed by {nu1CnsBase,u2ModLength +8}



**Important:**

The result buffer R must first be padded with zero bytes until its length is sufficient to perform the reduction ( $2 * u2ModLength + 8$ ) to be used by the Modular Reduction service as an input parameter.

The result of the reduction is written in the area X pointed by {nu1XBase, u2ModLength + 4}.

- For example, if relevant u2ModLength is 0x80 bytes and u2XLength is 0x80 too, the length of the Rspace may be  $2 * (u2ModLength + 4) = 0x108$  bytes.  
In case of fast or normalized reduction, the length of the result may be  $u2ModLength + 4$  so 0x84 bytes. Therefore, the zone X may lengths 0x84 bytes (at least). The multiplication of X by 1 word provide a result in the zone R which MSB bytes will be padded with zero bytes.  
In that example, the length of the zone R will be  $2 * u2ModLength + 8 = 0x108$  bytes.

### 37.3.4.7.4 Parameters Definition

**Table 37-18.** Smult Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	—	—	Options (see below)	Options (see below)
Specific/Gf2n CarryIn	Bits	I	—	—	$GF(2^n)$ Bit and Carry In	—
Specific/CarryOut Zero Violation	Bits	I	—	—	—	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase	nu1	I	Crypto RAM	$u2ModLength + 4$	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	$u2ModLength + 8$	Base of Cns	Base of Cns untouched
u2ModLength	u2	I	—	—	Length of N	Length of N
nu1XBase	nu1	I	Crypto RAM	$u2XLength$ or $u2ModLength + 4^{(1)}$	Base of X	Base of $X^{(2)}$
u2XLength	u2	I	—	—	Length of X	Length of X

.....continued

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ZBase	nu1	I	Crypto RAM	u2RLength	Base of Z	Base of Z untouched
nu1RBase	nu1	I	Crypto RAM	u2RLength	Base of R	Base of R (see <b>Note 3</b> )
u2RLength	u2	I	—	—	Length of R	Length of R
u4MulValue	u4	I	—	—	Value of MulValue	Value of MulValue untouched

**Notes:**

1. If a reduction option is specified, the area X will be, if necessary, extended to u2ModLength + 4 bytes.
2. If Smult is without reduction, X is untouched. If Smult is with reduction, X is filled with the final result.
3. If Smult is without reduction, R is filled with the final result. If Smult is with reduction, R is corrupted.

**37.3.4.7.5 Available Options**

The options are set by the u2Options input parameter, which is composed of:

- The mandatory Small Multiplication operation option described in the following table.
- The mandatory CarryOperand option described in Smult Service (with Accumulate/Subtract From) Carry Settings and Smult Service Carry Settings tables.
- The facultative Modular Reduction option (see Modular Reduction). If the Modular Reduction is not requested, this option is absent.

The u2Options number is calculated by an “Inclusive OR” of the options. Some examples in C language are:

- Operation: Small Multiply only without carry and without Modular Reduction  
`PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_SMULT_ONLY) | SET_CARRYOPTION(CARRY_NONE);`
- Operation: Small Multiply with addition with Specific/CarryIn addition and with Fast Modular Reduction  
`PUKCL(u2Options) =SET_MULTIPLIEROPTION(PUKCL_SMULT_ADD) | SET_CARRYOPTION(ADD_CARRY) | PUKCL_REDMOD_REDUCTION | PUKCL_REDMOD_USING_FASTRED;`

The following table lists all of the necessary parameters for the Small Multiply option. When the Addition or Subtraction option is not chosen, it is not necessary to fill in the nu1ZBase parameter.

**Table 37-19. Smult Service Operation Options**

Option	Purpose	Required Parameters
SET_MULTIPLIEROPTION(PUKCL_SMULT_ONLY)	Perform $R = \text{MulValue} * X + \text{CarryOperand}$	nu1RBase, u2RLength, nu1XBase, u2XLength, u4MulValue
SET_MULTIPLIEROPTION(PUKCL_SMULT_ADD)	Perform $R = Z + \text{MulValue} * X + \text{CarryOperand}$	nu1RBase, u2RLength, nu1ZBase, nu1XBase, u2XLength, u4MulValue
SET_MULTIPLIEROPTION(PUKCL_SMULT_SUB)	Perform $R = Z - (\text{MulValue} * X + \text{CarryOperand})$	nu1RBase, u2RLength, nu1ZBase, nu1XBase, u2XLength, u4MulValue

**37.3.4.7.6 Code Example**

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// Gf2n and CarryIn shall be beforehand filled (with zero or one)
PUKCL(Specific).Gf2n = ...;

```

```

PUKCL(Specific).CarryIn = ...; PUKCL(u2Options) =...;

// Depending on the option specified, all fields must not be filled
PUKCL_Smult(nu1XBase) = <Base of the X number>;
PUKCL_Smult(u2XLength) = <Length of the X number>;
PUKCL_Smult(nu1RBase) = <Base of the R number>;
PUKCL_Smult(u2RLength) = <Length of the R number>;
PUKCL_Smult(nu1ZBase) = <Base of the Z number>;
PUKCL_Smult(u4MulValue) = <Value to be multiplied with>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Smult,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
    {
        // The Small multiplication has been executed correctly
        ...
    }
else // Manage the error

```

**Note:**

The length of R must be greater or equal to the length of X. Additional options are available through the use of a modular reduction to be executed at the end of this operation. Some important considerations have to be taken into account concerning the length of resulting operands to get a mathematically correct result.

The output of this operation is not obviously compatible with the modular reduction, as it may be either smaller or bigger. In the case (most of the time) where the result (pointed by nu1RBase) is smaller in size than twice the modulus plus one word, it is mandatory to add padding bytes to zero. Otherwise, the reduced value will be taken considering the high order words (potentially uninitialized) as part of the number, thus resulting in a mathematically correct but unexpected result.

In the case that the result is bigger than twice the modulus plus one word, the modular reduction feature has to be executed as a separate operation, using an Euclidean division.

**37.3.4.7.7 Constraints**

For the case of a small multiplication with an option indicating either subtraction or accumulation, the following conditions must be avoided to ensure the service works correctly:

- nu1XBase, nu1RBase or nu1ZBase are not aligned on 32-bit boundaries
- {nu1XBase, u2XLength}, {nu1ZLength, u2RLength} or {nu1RBase, u2RLength} do not entirely lie in Crypto RAM
- u2XLength or u2RLength is either: < 4, > 0xffc or not a 32-bit length or u2XLength >u2RLength
- {nu1RBase, u2RLength} overlaps {nu1XBase, u2XLength} or nu1R < nu1Z and {nu1RBase,u2RLength} overlaps {nu1ZBase, u2RLength}

If the nu1R value is greater or equals to the nu1Z one, the overlapping between R and Z is allowed.

If a modular reduction is specified, the relevant parameters must be defined according to the chosen reduction and follow the description in Modular Reduction. Additional constraints to be respected and error codes are described in this section and in Smult Service Return Codes.

**Multiplication with Accumulation or Subtraction**

When the options bits specify that either an Accumulation or a Subtraction must be performed, this service performs the following operation:

$$R = (Z \pm (MulValue \times X + CarryIn)) \bmod B^{RLength}$$

**Table 37-20.** Smult Service (with Accumulate/Subtract From) Carry Settings

Carry Options	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	R = Z ± (MulValue*X + CarryIn)
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	R = Z ± (MulValue*X - CarryIn)

.....continued

Carry Options	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	$R = Z \pm (\text{MulValue} * X + 1 + \text{CarryIn})$
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	$R = Z \pm (\text{MulValue} * X + 1 - \text{CarryIn})$
SET_CARRYOPTION(CARRY_NONE)	0	$R = Z \pm (\text{MulValue} * X)$
SET_CARRYOPTION(ADD_1)	1	$R = Z \pm (\text{MulValue} * X + 1)$
SET_CARRYOPTION(SUB_1)	- 1	$R = Z \pm (\text{MulValue} * X - 1)$
SET_CARRYOPTION(ADD_2)	2	$R = Z \pm (\text{MulValue} * X + 2)$

### Multiplication without Accumulation or Subtraction

When the case the options bits specify that neither an Accumulation nor a Subtraction must be performed, this service performs the following operation:

$$R = (\text{MulValue} \times X + \text{CarryOperand}) \bmod B^{R\text{Length}}$$

**Table 37-21.** Smult Service Carry Settings

Carry Options	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	$R = \text{MulValue} * X + \text{CarryIn}$
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	$R = \text{MulValue} * X - \text{CarryIn}$
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	$R = \text{MulValue} * X + 1 + \text{CarryIn}$
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	$R = \text{MulValue} * X + 1 - \text{CarryIn}$
SET_CARRYOPTION(CARRY_NONE)	0	$R = \text{MulValue} * X$
SET_CARRYOPTION(ADD_1)	1	$R = \text{MulValue} * X + 1$
SET_CARRYOPTION(SUB_1)	-1	$R = \text{MulValue} * X - 1$
SET_CARRYOPTION(ADD_2)	2	$R = \text{MulValue} * X + 2$

### 37.3.4.7.8 Status Returned Values

**Table 37-22.** Smult Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	—	Service functioned correctly

### 37.3.4.8 Compare

#### 37.3.4.8.1 Purpose

The purpose of this service is to compare two numbers in classical arithmetic GF(p).



**Important:** This service works only with integers.

#### 37.3.4.8.2 How to Use the Service

#### 37.3.4.8.3 Description

This service accepts two numbers in classical arithmetic in input and performs a comparison, virtually subtracting  $(X + \text{CarryIn})$  from Y:

CompareGetFlags  $(Y - (X + \text{CarryIn}))$

The numbers X and Y are untouched but the resulting flags CarryOut and the Zero Bit are filled. If the lengths of Y and X are equal, a comparison is processed.

If the length of Y is strictly greater than the length of X, X is first virtually padded with zeros on the Most Significant Bytes side, then a comparison is processed.

**Note:** The length of Y must be greater or equal to the length of X.

In this computation, the following data need to be provided:

- X (pointed by {nu1XBase,u2XLength})
- Y (pointed by {nu1YBase,u2YLength})

The service name for this operation is Comp.

### 37.3.4.8.4 Parameters Definition

**Table 37-23.** Comp Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
Specific/Gf2n CarryIn	Bits	I	-	-	GF(2n) Bit and Carry In	-
Specific/CarryOut Zero Violation	Bits	I	-	-	-	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1XBase	nu1	I	Crypto RAM	u2XLength	Base of X	Base of X
u2XLength	u2	I	-	-	Length of X	Length of X
nu1YBase	nu1	I	Crypto RAM	u2YLength	Base of Y	Base of Y
u2YLength	u2	I	-	-	Length of Y	Length of Y

### 37.3.4.8.5 Code Example

```

PUKCL_PARAM PUKCLParam;
P_PUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// CarryIn shall be beforehand filled (with zero or one) PUKCL(Specific).CarryIn = ...;

// Initializing parameters
PUKCL_Comp(nu1XBase) = <Base of the ram location of X>;
PUKCL_Comp(u2XLength) = <Length of X>;
PUKCL_Comp(nu1YBase) = <Base of the ram location of Y>;
PUKCL_Comp(u2YLength) = <Length of Y>;

// vPUKCL_Process() is a macro command,
// and then calls the library...
vPUKCL_Process(Comp,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    // The COMPARE has been executed correctly
    // CarryOut, Zero ... are available
    ... = PUKCL(Specific).CarryOut;
    ... = PUKCL(Specific).Zero;
}
else // Manage the error

```

### 37.3.4.8.6 Constraints

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase or nu1YBase are not aligned on 32-bit boundaries
- {nu1XBase, u2XLength} or {nu1YBase, u2YLength} are not in Crypto RAM
- u2XLength or u2YLength is either: < 4, > 0xffc or not a 32-bit length or u2XLength > u2YLength

### 37.3.4.8.7 Status Returned Values

**Table 37-24.** Comp Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly

### 37.3.4.9 Full Multiply

#### Related Links

[37.3.5.1. Modular Reduction](#)

#### 37.3.4.9.1 Purpose

The purpose of this service is to multiply two large numbers, X and Y, and optionally accumulate/subtract from a third large number, Z, returning the result, R.

The available options are as follows:

- Work in the  $GF(2^n)$  field or in the standard arithmetic field
- Add of a supplemental CarryOperand
- Overlap of the operands is possible, taking into account some constraints
- Modular Reduction of the computation result (see *Modular Reduction* from Related Links)

#### 37.3.4.9.2 How to Use the Service

#### 37.3.4.9.3 Description

This service provides the following (if not computing a modular reduction of the result):

$$R = [Z] \pm (X \times Y + \text{CarryOperand})$$

Or (if computing a modular reduction of the result):

$$R = ([Z] \pm (X \times Y + \text{CarryOperand})) \bmod N$$

The service name for this operation is `Fmult`.

In these computations, the following data has to be provided:

- R the result (pointed by {nu1RBase,u2Xlength +u2YLength})
- X one input number or  $GF(2^n)$  polynomial (pointed by {nu1XBase,u2XLength})
- Y one input number or  $GF(2^n)$  polynomial (pointed by {nu1YBase,u2YLength})
- Z one optional input number or  $GF(2n)$  polynomial (pointed by {nu1ZBase,u2Xlength +u2YLength})
- CarryOperand (provided through the Carry Options and Carry values)



**Important:** Even if neither accumulation nor subtraction is specified, the nu1ZBase must always be filled and point to a Crypto RAM space. In this case, nu1ZBase can point to the same space as the nu1RBase.

If using the big modular reduction option, the Multiply operation is followed by a reduction (see *Modular Reduction* from Related Links). In this case, the length of Cns is 64 bytes.

If using the modular reduction option, the Multiply operation is followed by a reduction (see *Modular Reduction* from Related Links). In this case the following parameters must be additionally provided:

- N—the modulus (pointed by {nu1ModBase,u2Modlength +4})
- Cns—the reduction constant
  - In case of Big reduction, Cns is pointed by {nu1CnsBase,64bytes}.
  - In case of Fast or Normalized reduction, Cns is pointed by (pointed by {nu1CnsBase,u2ModLength+ 8})



**Note:**

The result buffer R must first be padded with zero bytes until its length is sufficient to perform the reduction ( $2 \cdot u2ModLength + 8$ ) to be used by the Modular Reduction service as an input parameter.

The result of the reduction is written in the area X pointed by  $\{nu1XBase, u2ModLength + 4\}$ .

For example, if  $u2ModLength$ ,  $u2XLength$  and  $u2YLength$  are 0x80 bytes, the length of the R space is  $2 \cdot (u2ModLength + 4) = 0x108$  bytes because of the constraints of modular reduction.

In case of Fast or Normalized Reduction, the length of the result is  $u2ModLength + 4$  so 0x84 bytes. Thus, the zone X has a length of 0x84 bytes (at least). The multiplication of X by Y provides a result of length 0x100 bytes in the zone R so the 8 MSB bytes must be previously padded with zero bytes (in offsets 0x100 to 0x107).

### 37.3.4.9.4 Parameters Definition

**Table 37-25.** Fmult Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	-	-	Options (see below)	Options (see below)
Specific/Gf2n CarryIn	Bits	I	-	-	GF(2n) Bit and Carry In	-
Specific/CarryOut Zero Violation	Bits	I	-	-	-	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase	nu1	I	Crypto RAM	$u2ModLength + 4$	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	$u2ModLength + 8$ or 64 bytes	Base of Cns	Base of Cns untouched
u2ModLength	u2	I	-	-	Length of N	Length of N
nu1XBase	nu1	I	Crypto RAM	$u2XLength$ or $u2ModLength + 4^{(1)}$	Base of X	Base of $X^{(2)}$
u2XLength	u2	I	-	-	Length of X	Length of X
nu1YBase	nu1	I	Crypto RAM	$u2YLength$	Base of Y	Base of Y
u2YLength	u2	I	-	-	Length of Y	Length of Y
nu1ZBase	nu1	I	Crypto RAM	$u2XLength + u2YLength$	Base of Z	Base of Z untouched
nu1RBase	nu1	I	Crypto RAM	$u2XLength + u2YLength$	Base of R	Base of $R^{(3)}$

**Notes:**

1. In case of a reduction option is specified, if necessary, the area X will be extended to  $u2ModLength + 4$  bytes.
2. If FMult is without reduction, X is untouched. If FMult is with reduction, X is filled with the final result.
3. If FMult is without reduction, R is filled with the final result. If FMult is with reduction, R is corrupted.

### 37.3.4.9.5 Available Options

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Full Multiplication operation option described in [Table 37-26](#)
- the mandatory CarryOperand option described in [Table 37-27](#) and [Table 37-28](#)
- the facultative Modular Reduction option(see *Modular Reduction* from Related Links). If the Modular Reduction is not requested, this option is absent.

The u2Options number is calculated by an Inclusive OR of the options.

Some Examples in C language are:

- Operation: Full Multiply only without carry and without Modular Reduction  

```
PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_FMUL_ONLY) |
SET_CARRYOPTION(CARRY_NONE);
```
- Operation: Full Multiply with addition with Specific/CarryIn addition and with Fast Modular Reduction  

```
PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_FMUL_ADD) |
SET_CARRYOPTION(ADD_CARRY) |
PUKCL_REDMOD_REDUCTION |
PUKCL_REDMOD_USING_FASTRED;
```

The following table shows all of the necessary parameters for the Full Multiply option. When the Addition or Subtraction option is not chosen, it is not necessary to fill in the nu1ZBase parameter.

**Table 37-26.** Fmult Service Options

Option	Purpose	Required Parameters
SET_MULTIPLIEROPTION(PUKCL_FMUL_ONLY)	Perform $R = X*Y + \text{CarryOperand}$	nu1RBase, nu1YBase, u2YLength, nu1XBase, u2XLength
SET_MULTIPLIEROPTION(PUKCL_FMUL_ADD)	Perform $R = Z + X*Y + \text{CarryOperand}$	nu1RBase, nu1ZBase, nu1YBase, u2YLength, nu1XBase, u2XLength
SET_MULTIPLIEROPTION(PUKCL_FMUL_SUB)	Perform $R = Z - (X*Y + \text{CarryOperand})$	nu1RBase, nu1ZBase, nu1YBase, u2YLength, nu1Xlength, u2XLength

### 37.3.4.9.6 Code Example

```
PUKCL_PARAM PUKCLParam;
P_PUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// Gf2n and CarryIn shall be beforehand filled (with zero or one)
PUKCL(Specific).Gf2n = ...;
PUKCL(Specific).CarryIn = ...;

PUKCL(u2Option) = ...;
// Depending on the option specified, not all fields must be filled
PUKCL_Fmult(nu1XBase) = <Base of the ram location of X>;
PUKCL_Fmult(u2XLength) = <Length of X>;
PUKCL_Fmult(nu1YBase) = <Base of the ram location of Y>;
PUKCL_Fmult(u2YLength) = <Length of Y>;
PUKCL_Fmult(nu1ZBase) = <Base of the ram location of Z>;
PUKCL_Fmult(nu1RBase) = <Base of the ram location of R>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Fmult,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    // The Full multiply has been executed correctly
    ...
}
else // Manage the error
```

### 37.3.4.9.7 Important Considerations for Modular Reduction of a Fmult Computation Result

**Note:**

Additional options are available through the use of a modular reduction to be executed at the end of this operation. Some important considerations have to be taken into account concerning the length of resulting operands to get a mathematically correct result.

The output of this operation is not always compatible with the modular reduction as it may be either smaller or bigger. In the case (most of the time) the result (pointed by nu1RBase) is smaller in size than “twice the modulus plus one word” by one word, a padding word must be added to zero. Otherwise, the reduced value will be taken considering the high order words (potentially uninitialized) as part of the number, thus resulting in getting a mathematically correct but unexpected result.

In the case that the result is bigger than twice the modulus plus one word, the modular reduction feature has to be executed as a separate operation, using an Euclidean division.

### 37.3.4.9.8 Constraints

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase, nu1YBase, nu1RBase or nu1ZBase are not aligned on 32-bit boundaries
- {nu1XBase, u2XLength}, {nu1YLength, u2YLength}, {nu1ZBase, u2XLength+u2YLength} or {nu1RBase, u2XLength+u2YLength} are not in Crypto RAM
- u2XLength, u2YLength is either: < 4, > 0xffc or not a 32-bit length
- {nu1RBase, u2XLength+u2YLength} overlaps {nu1YBase, u2YLength} or {nu1RBase, u2XLength+u2YLength} overlaps {nu1XBase, u2XLength}
- {nu1RBase, u2XLength+u2YLength} overlaps {nu1ZBase, u2XLength+u2YLength} and nu1RBase > nu1ZBase

If a modular reduction is specified, the relevant parameters must be defined according to the chosen reduction and follow the description in Modular Reduction (see *Modular Reduction* from Related Links). Additional constraints to be respected and error codes are described in this section and in [Table 37-49](#).

### Multiplication with Accumulation or Subtraction

In the case where the options bits specify that either an Accumulation or a subtraction must be performed, this service performs the following operation:

$$R = (Z \pm (X \times Y + CarryOperand)) \bmod B^{XLength + YLength}$$

**Table 37-27.** Fmult Service (with Accumulate/Subtract From) Carry Settings

Option AND CARRYOPTIONS	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	$R = Z \pm (X*Y + CarryIn)$
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	$R = Z \pm (X*Y - CarryIn)$
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	$R = Z \pm (X*Y + 1 + CarryIn)$
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	$R = Z \pm (X*Y + 1 - CarryIn)$
SET_CARRYOPTION(CARRY_NONE)	0	$R = Z \pm (X*Y)$
SET_CARRYOPTION(ADD_1)	1	$R = Z \pm (X*Y + 1)$
SET_CARRYOPTION(SUB_1)	- 1	$R = Z \pm (X*Y - 1)$
SET_CARRYOPTION(ADD_2)	2	$R = Z \pm (X*Y + 2)$

### Multiplication without Accumulation or Subtraction

In the case the options bits specify that either an Accumulation or a subtraction must be performed, this service performs the following operation:

$$R = (X \times Y + CarryOperand) \bmod B^{XLength + YLength}$$

**Table 37-28.** Fmult Service Carry Settings

Option AND CARRYOPTIONS	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	$R = X*Y + \text{CarryIn}$
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	$R = X*Y - \text{CarryIn}$
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	$R = X*Y + 1 + \text{CarryIn}$
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	$R = X*Y + 1 - \text{CarryIn}$
SET_CARRYOPTION(CARRY_NONE)	0	$R = X*Y$
SET_CARRYOPTION(ADD_1)	1	$R = X*Y + 1$
SET_CARRYOPTION(SUB_1)	- 1	$R = X*Y - 1$
SET_CARRYOPTION(ADD_2)	2	$R = X*Y + 2$

### 37.3.4.9.9 Status Returned Values

**Table 37-29.** Fmult Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly

### 37.3.4.10 Square

#### Related Links

[37.3.5.1. Modular Reduction](#)

#### 37.3.4.10.1 Purpose

The purpose of this service is to compute the square of a big number and optionally accumulate/ subtract from a second big number.

Please note that this service uses an optimized implementation of the squaring. It also means that when the GF(2<sup>n</sup>) flag is set, the execution time will be smaller than when not set (in that case, the squaring execution time will still be smaller than for a standard multiplication).

The available options are as follows:

- Work in the GF(2<sup>n</sup>) or in the standard integer arithmetic field
- Add of a supplemental CarryOperand
- Overlapping of the operands is possible, taking into account some constraints
- Modular Reduction of the computation result

#### 37.3.4.10.2 How to Use the Service

#### 37.3.4.10.3 Description

This service provides the following (if not computing a modular reduction of the result):

$$R = [Z] \pm (X^2 + \text{CarryOperand})$$

Or (if computing a modular reduction of the result):

$$R = ([Z] \pm (X^2 + \text{CarryOperand})) \bmod N$$

The service name for this operation is `Square`.

In these computations, the following data has to be provided:

- R the result (pointed by {nu1RBase,2 \*u2Xlength})
- X one input number or GF(2n) polynomial (pointed by{nu1XBase,u2XLength})
- Z one optional input number or GF(2n) polynomial (pointed by {nu1ZBase,2 \*u2Xlength})
- CarryOperand (provided through the CarryOptions and Carry values)



**Important:** Even if neither accumulation nor subtraction is specified, the nu1ZBase must always be filled and point to a Crypto RAM space. In this case, nu1ZBase can point to the same space as the nu1RBase.

If using the big modular reduction option, the Multiply operation is followed by a reduction (see *Modular Reduction* from Related Links). In this case, the length of Cns is 64 bytes.

If using the modular reduction option the Square operation is followed by a reduction (see *Modular Reduction* from Related Links). In this case the following parameters must be additionally provided:

- N—the modulus (pointed by {nu1ModBase,u2Modlength +4}).
- Cns—the reduction constant (pointed by {nu1CnsBase,u2Modlength +8})
  - In case of big reduction option, the length of Cns is 64bytes.

**Note:**

The result buffer R must first be padded with zero bytes until its length is sufficient to perform the reduction ( $2 * u2ModLength + 8$ ) to be used by the Modular Reduction service as an input parameter.

The result of the reduction is written in the area X pointed by {nu1XBase, u2ModLength + 4}.

For example, if u2ModLength, u2XLength is 0x80 bytes, the length of the R space is  $2 * (u2ModLength + 4) = 0x108$  bytes because of the constraints of modular reduction.

In case of Fast or Normalized Reduction, the length of the result is u2ModLength + 4 so 0x84 bytes. Thus, the zoneX has a length of 0x84 bytes (at least). The square of X provides a result of length 0x100 bytes in the zone R so the 8 MSB bytes previously must be previously padded with zero bytes (in offsets 0x100 to 0x107).

### 37.3.4.10.4 Parameters Definition

**Table 37-30.** Square Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	-	-	Options (see below)	Options (see below)
Specific/Gf2n CarryIn	Bits	I	-	-	GF(2n) Bit and Carry In	-
Specific/CarryOut Zero Violation	Bits	I	-	-	-	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8 or 64 bytes	Base of Cns	Base of Cns untouched
u2ModLength	u2	I	-	-	Length of N	Length of N
nu1XBase	nu1	I	Crypto RAM	u2XLength or u2ModLength + 4 <sup>(1)</sup>	Base of X	Base of X <sup>(2)</sup>
u2XLength	u2	I	-	-	Length of X	Length of X
nu1ZBase	nu1	I	Crypto RAM	2 * u2XLength	Base of Z	Base of Z
nu1RBase	nu1	I	Crypto RAM	2 * u2XLength	Base of R	Base of R <sup>(3)</sup>

**Notes:**

1. In case of a reduction option is specified, the area X will be, if necessary, extended to  $u2ModLength + 4$  bytes.
2. If Square is without reduction, X is untouched. If Square is with reduction, X is filled with the final result.
3. If Square is without reduction, R is filled with the final result. If Square is with reduction, R is corrupted.

**37.3.4.10.5 Available Options**

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Square operation option described in [Table 37-31](#)
- the mandatory CarryOperand option described in [Table 37-32](#) and [Table 37-33](#)
- the facultative Modular Reduction option (see *Modular Reduction* from Related Links). If the Modular Reduction is not requested, this option is absent.

The u2Options number is calculated by an Inclusive OR of the options. Some Examples in C language are:

- Operation: Square only without carry and without Modular Reduction  
`PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_SQUARE_ONLY) | SET_CARRYOPTION(CARRY_NONE);`
- Operation: Square with addition with Specific/CarryIn addition and with Fast Modular Reduction  
`PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_SQUARE_ADD) | SET_CARRYOPTION(ADD_CARRY) | PUKCL_REDMOD_REDUCTION | PUKCL_REDMOD_USING_FASTRED;`

The following table lists all of the necessary parameters for the Square option. When the Addition or Subtraction option is not chosen it is not necessary to fill in the nu1ZBase parameter.

**Table 37-31. Square Service Options**

Option	Purpose	Required Parameters
SET_MULTIPLIEROPTION(PUKCL_SQUARE_ONLY)	Perform $R = X^2 + \text{CarryOperand}$	nu1RBase, nu1ZBase, nu1XBase, u2XLength
SET_MULTIPLIEROPTION(PUKCL_SQUARE_ADD)	Perform $R = Z + X^2 + \text{CarryOperand}$	nu1RBase, nu1ZBase, nu1XBase, u2XLength
SET_MULTIPLIEROPTION(PUKCL_SQUARE_SUB)	Perform $R = Z - (X^2 + \text{CarryOperand})$	nu1RBase, nu1ZBase, nu1Xlength, u2XLength

**37.3.4.10.6 Code Example**

```

PUKCL_PARAM PUKCLParam;
P_PUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// Gf2n and CarryIn shall be beforehand filled (with zero or one)
PUKCL(Specific).Gf2n = ...;
PUKCL(Specific).CarryIn = ...;

PUKCL(u2Option) = ...;
// Depending on the option specified, not all fields must be filled
PUKCL_Fmult(nu1XBase) = <Base of the ram location of X>;
PUKCL_Fmult(u2XLength) = <Length of X>;
PUKCL_Fmult(nu1ZBase) = <Base of the ram location of Z>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library..
vPUKCL_Process(Square,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    // The Squaring has been executed correctly
}

```

```

    ...
    }
else // Manage the error

```

### 37.3.4.10.7 Important Considerations for Modular Reduction of a Square Computation

**Note:**

Additional options are available through the use of a modular reduction to be executed at the end of this operation. Some important considerations have to be taken into account concerning the length of resulting operands to get a mathematically correct result.

The output of this operation is not obviously compatible with the modular reduction as it may be either smaller or bigger. In the case (most of the time) the result (pointed by nu1RBase) is smaller in size than “twice the modulus plus one word” by one word, a padding word must be added to zero. Otherwise, the reduced value will be taken considering the high order words (potentially uninitialized) as part of the number, thus resulting in getting a mathematically correct but unexpected result.

In the case that the result is greater than twice the modulus plus one word, the modular reduction feature has to be executed as a separate operation, using an Euclidean division.

### 37.3.4.10.8 Constraints

When the options only indicate a square, the constraints involving nu1ZBase are not checked. The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase, nu1RBase or nu1ZBase are not aligned on 32-bit boundaries
- {nu1XBase, u2XLength}, {nu1ZBase, 2\*u2XLength} or {nu1RBase, 2\*u2XLength} are not in Crypto RAM
- u2XLength is either: < 4, > 0xffc or not a 32-bit length
- {nu1RBase, 2\*u2XLength} overlaps {nu1XBase, u2XLength}
- {nu1RBase, 2\*u2XLength} overlaps {nu1ZBase, 2\*u2XLength} and nu1RBase > nu1ZBase

If a modular reduction is specified, the relevant parameters must be defined according to the chosen reduction and follow the description in Modular Reduction (see *Modular Reduction* from Related Links). Additional constraints to be respected and error codes are described in this section and in [Table 37-49](#).

### Multiplication with Accumulation or Subtraction

Where the options bits specify that either an Accumulation or a subtraction must be performed, this command performs the following operation:

$$R = (Z \pm (X^2 + CarryOperand)) \bmod B^{2 * XLength}$$

**Table 37-32.** Multiplication with Accumulation or Subtraction

Option AND CARRYOPTIONS	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	$R = Z \pm (X^2 + CarryIn)$
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	$R = Z \pm (X^2 - CarryIn)$
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	$R = Z \pm (X^2 + 1 + CarryIn)$
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	$R = Z \pm (X^2 + 1 - CarryIn)$
SET_CARRYOPTION(CARRY_NONE)	0	$R = Z \pm (X^2)$
SET_CARRYOPTION(ADD_1)	1	$R = Z \pm (X^2 + 1)$
SET_CARRYOPTION(SUB_1)	- 1	$R = Z \pm (X^2 - 1)$
SET_CARRYOPTION(ADD_2)	2	$R = Z \pm (X^2 + 2)$

### 37.3.4.10.9 Multiplication without Accumulation or Subtraction

Where the options bits specify that either an accumulation or a subtraction must be performed, this command performs the following operation:

$$R = (X^2 + CarryOperand) \bmod B^{2 \cdot XLength}$$

**Table 37-33.** Square Service Carry Settings

Option AND CARRYOPTIONS	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	$R = X^2 + CarryIn$
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	$R = X^2 - CarryIn$
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	$R = X^2 + 1 + CarryIn$
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	$R = X^2 + 1 - CarryIn$
SET_CARRYOPTION(CARRY_NONE)	0	$R = X^2$
SET_CARRYOPTION(ADD_1)	1	$R = X^2 + 1$
SET_CARRYOPTION(SUB_1)	- 1	$R = X^2 - 1$
SET_CARRYOPTION(ADD_2)	2	$R = X^2 + 2$

### 37.3.4.10.10 Status Returned Values

**Table 37-34.** Square Service Return Codes

Returned status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly

### 37.3.4.11 Integral (Euclidean) Division

#### 37.3.4.11.1 Purpose

The purpose of this service is to compute the Euclidean Division of two multiple precision numbers in GF(p) or polynomial in GF(2<sup>n</sup>). The Numerator is divided by the Denominator giving the Quotient “Quo” and the Remainder “R”.

The following options are available:

- Work in the GF(2<sup>n</sup>) field or in the standard integer arithmetic field GF(p)

#### 37.3.4.11.2 How to Use the Service

#### 37.3.4.11.3 Description

This service processes the calculus corresponding to:

$$Num = Mod \times Quo + R \text{ with } 0 \leq R < Mod \text{ and } Quo = \left\lfloor \frac{Num}{Mod} \right\rfloor$$

The Numerator is Num.

The Divisor (Modulus) is Mod.

The Quotient is Quo.

The Remainder is R.

The Inputs are, the Numerator Num, and the Denominator Mod. The service calculates the Quotient and the Remainder. The Remainder overwrites the Numerator and is copied to the R area.

If the parameter nu1QuoBase equals zero, the Quotient is not stored in memory.

If nu1QuoBase is different from zero, the Quotient length is (<Numerator Length> - <Denominator Length>) + 4 bytes.

In this computation, the following areas need to be provided:

- Num (pointed by {nu1NumBase,u2NumLength}) filled with the Numerator (with MSB word to zero).
- Mod (pointed by {nu1ModBase,u2ModLength}) filled with the Denominator.
- Workspace (pointed by {nu1CnsBase,64 or68}).



- Quo (pointed by {nu1QuoBase,u2NumLength - u2ModLength + 4}) to contain calculated Quotient.
  - When the quotient is not needed, the nu1QuoBase pointer can be provided as NULL. In that case, only the remainder will be provided as a result.
- R (pointed by {nu1RBase,u2ModLength}) to contain the calculated Remainder.

The service name for this operation is Div.

### 37.3.4.11.4 Parameters Definition

**Table 37-35.** Div Service Parameters

Parameter	Type	Dir.	Location	Data Length	Before Executing the Service	After Executing the Service
Specific/Gf2n	Bit	I	-	-	GF(2n) Bit	-
nu1NumBase	nu1	I	Crypto RAM	u2NumLength	Base of Num Filled with the Numerator	Base of Num Filled with the Remainder
u2NumLength	u2	I	-	-	Length of the Numerator	Length of the Numerator
nu1ModBase	nu1	I	Crypto RAM	u2ModLength	Base of the Divisor	Base of the Divisor untouched
u2ModLength	u2	I	-	-	Length of the Divisor	Length of the Divisor
nu1QuoBase (see <b>Note 1</b> )	nu1	I	Crypto RAM	u2NumLength - u2ModLength + 4	Base of the Quotient	Base of the Quotient
nu1WorkSpace	nu1	I	Crypto RAM	GF(p): 64 GF(2n): 68	Base of the WorkSpace	Base of the WorkSpace corrupted
nu1RBase ( see <b>Note 2</b> )	nu1	I	Crypto RAM	u2ModLength	Base of the Remainder	Base of the Remainder

#### Notes:

1. If the quotient is not needed, set nu1QuoBase to zero and the quotient will not be written to memory. If the quotient is needed, set the nu1QuoBase to the beginning of an area of size (u2NumLength - u2ModLength + 4) to write the whole quotient.
2. The Remainder is present in the area {nu1NumBase, u2NumLength} at the end of the calculus. The nu1RBase parameter makes it possible to copy this result in the other area {nu1RBase, u2ModLength}, if this copy is not needed, set nu1RBase to the same value as nu1NumBase and the copy will not be done.

**Note:** The parameter Num must have its most significant 32-bit word cleared to zero. The length u2NumLength is the length of Num including this zero word.

One additional word is used on the LSB side of the Num parameter, this word is restored at the end of the calculus. As a consequence the parameter nu1NumBase must never be at the beginning of the Crypto RAM, i.e., ensure that nu1NumBase ≥ <Crypto RAM Base> + 4 bytes.

One additional word is used on the MSB side of the Num parameter, this word is not corrupted. As a consequence the Area {nu1NumBase, u2NumLength} must not be at the end of the Crypto RAM, i.e., ensure that nu1NumBase+u2NumLength ≤ <Crypto RAM End> - 4.

u2ModLength must be the true length of the Modulus, i.e., the MSB word of the area {nu1ModBase, u2ModLength} must be different from zero.

The minimum value for u2ModLength is 8 bytes, so the significant length of Num must be at least 8 bytes. To divide by a 32-bit value, the divider and numerator shall be multiplied by 2<sup>32</sup>. The resulting remainder will have to be divided by 2<sup>32</sup>, the quotient will be exact.

### 37.3.4.11.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;
```

```
// Fill all the fields
// In that case, the quotient will be computed
// If it was not needed, set nu1QuoBase to NULL
PUKCL_Div(nu1NumBase) = <Base of the ram location of Num>;
PUKCL_Div(nu1ModBase) = <Base of the ram location of Mod>;
PUKCL_Div(nu1QuoBase) = <Base of the ram location of Quo>;
PUKCL_Div(nu1WorkSpace) = <Base of the workspace>;
PUKCL_Div(nu1RBase) = <Base of the ram location of R>;
PUKCL_Div(u2NumLength) = <Length of Num>;
PUKCL_Div(u2ModLength) = <Length of Mod>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Div,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    // The Division has been executed correctly
    ...
}
else // Manage the error
```

### 37.3.4.11.6 Constraints

The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1RBase, nu1QuoBase, nu1WorkSpace or nu1NumBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength}, {nu1RBase, u2ModLength}, {nu1WorkSpace, 64} or {nu1NumBase, u2NumLength} are not in Crypto RAM
- u2ModLength, u2NumLength is either: < 4, > 0xffc or not a 32-bit length
- One or more overlaps exist between two of the areas: {nu1ModBase,u2ModLength},{nu1RBase, u2ModLength} {nu1NumBase, u2NumLength}(1) or {nu1WorkSpace,64}
- If nu1QuoBase is different from zero and: {nu1QuoBase, u2NumLength - u2ModLength + 4} are not in Crypto RAM
- If nu1QuoBase is different from zero and one or more overlaps exist between two of the areas: {nu1QuoBase, u2NumLength - u2ModLength + 4}, {nu1ModBase, u2ModLength}, {nu1RBase, u2ModLength}, {nu1NumBase, u2NumLength} or {nu1WorkSpace, 64}

Overlaps between {nu1RBase, u2ModLength} and {nu1NumBase, u2NumLength} are forbidden, but the equality between nu1RBase and nu1NumBase is authorized

### 37.3.4.11.7 Status Returned Values

**Table 37-36.** Div Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly.
PUKCL_DIVISION_BY_ZERO	Severe	The operation was not performed because the Denominator value is zero.

### 37.3.4.12 GCD, Modular Inverse

#### 37.3.4.12.1 Purpose

The purpose of this command is to compute the Greatest Common Divisor (GCD) and the Modular Inverse. The algorithm used is the Extended Euclidean Algorithm for the GCD.

This command accepts as input two multiple precision numbers in GF(p) or two polynomials in GF(2<sup>n</sup>) X and Y and computes their GCD (D), if D equals one, the command also supplies the inverse of X modulo Y.

The available options are as follows:

- Work in the GF(2<sup>n</sup>) field or in the standard integer arithmetic field GF(p)

### 37.3.4.12.2 How to Use the Service

#### 37.3.4.12.3 Description

This command calculates:

$$D = GCD(X,Y).$$

and parameter A in the Bezout equation:

$$A \times X + B \times Y = D.$$

The first input, or input to inverse is X.

The second input, or modulus is Y.

The GCD is output in D.

The modular inverse if X and Y are co-primes is output A:

$$A = X^{-1} \text{mod}(Y)$$

The command calculates the GCD and the value A. The value A is the multiplicative inverse of X, only if X and Y are co-prime. As a supplemental result, Z is given back, being the quotient of Y divided by D only if D is different from zero:

$$Z = \left\lfloor \frac{Y}{D} \right\rfloor$$

At the end of the command: X is overwritten by D.

Y is cleared.

The value of A is calculated and stored.

The value of Z is calculated and stored if D is different from zero.

The service name for this operation is GCD.

In this computation, the following areas have to be provided:

- X (pointed by {nu1XBase,u2Length}) filled with X (with MSB word to zero)
- Y (pointed by {nu1YBase,u2Length}) filled with Y (with MSB word to zero)
- A (pointed by {nu1ABase,u2Length}) to contain calculated A
- Z (pointed by {nu1ZBase,u2Length}) to contain calculated Z
- The workspace (pointed by {nu1WorkSpace,32})

#### 37.3.4.12.4 Parameters Definition

**Table 37-37.** GCD Service Parameters

Parameter	Type	Dir.	Location	Data Length	Before Executing the Service	After Executing the Service
Specific/Gf2n	Bit	I	-	-	GF(2n) Bit	-
nu1XBase	nu1	I	Crypto RAM	u2Length	Base of X Number X	Base of X Filled with the GCD D
u2Length	u2	I	-	-	Length of the Areas X, Y, A, Z	Length of the Areas X, Y, A, Z
nu1YBase	nu1	I	Crypto RAM	u2Length	Base of Y Number Y	Base of Y Cleared area
nu1ABase	nu1	I	Crypto RAM	u2Length	Base of A	Base of A Filled with the result
nu1ZBase	nu1	I	Crypto RAM	u2Length + 4 (see <b>Note 1</b> )	Base of Z	Base of Z Filled with the result
nu1WorkSpace	nu1	I	Crypto RAM	32 bytes	Base of the workspace	Base of the workspace corrupted

**Note:**

1. The additional word is 4 zero bytes.

The parameters X and Y must have their most significant 32-bit word cleared to zero. The length u2Length is the length of the longer of the parameters X and Y including this zero word.

To clarify here is an example:

- X is an 8 bytes number.
- Y is a 12 bytes number.

This example is processed this way before the use of the GCD service:

- The longer number is Y so its length is taken and increased by 4 bytes for the 32-bit word cleared to zero, this gives u2Length = 16 bytes. Therefore, X, Y, A and Z areas have a length of 16 bytes.
- Y is padded with 4 bytes cleared to zero on the MSB side and the u2Length = 16 bytes are written in memory (LSB first).
- X is padded with 8 bytes cleared to zero on the MSB side and the u2Length = 16 bytes are written in memory (LSB first).
- The areas A and Z are mapped in memory with a size of u2Length = 16 bytes.
- The workspace is mapped in memory with its constant size of 32 bytes

### 37.3.4.12.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PUPUKCL_PARAM pvPUKCLParam = &PUKCLParam;
// Fill all the fields
PUKCL(u2Option) = 0;
PUKCL_GCD(nu1XBase) = <Base of the ram location of X>;
PUKCL_GCD(nu1YBase) = <Base of the ram location of Y>;
PUKCL_GCD(nu1ABase) = <Base of the ram location of A>;
PUKCL_GCD(nu1ZBase) = <Base of the ram location of Z>;
PUKCL_GCD(nu1WorkSpace) = <Base of the workspace>;
PUKCL_GCD(u2Length) = <Length of X, Y, A and Z>;
// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GCD, pvPUKCLParam);
if (PUKCL_Param.Status == PUKCL_OK)
    {
        // The GCD has been executed correctly
        ...
    }
else // Manage the error

```

### 37.3.4.12.6 Constraints

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase, nu1YBase, nu1ABase or nu1ZBase are not aligned on 32-bit boundaries
- {nu1XBase, u2Length}, {nu1YBase, u2Length}, {nu1ABase, u2Length} or {nu1ZBase, u2Length} are not in Crypto RAM
- u2Length is either: < 4, > 0xffc or not a 32-bit length
- {nu1XBase, u2Length} overlaps {nu1YBase, u2Length} or {nu1XBase, u2Length} overlaps {nu1ABase, u2Length} or {nu1XBase, u2Length} overlaps {nu1ZBase, u2Length} or {nu1YBase, u2Length} overlaps

{nu1ABase, u2Length} or {nu1YBase, u2Length} overlaps {nu1ZBase, u2Length} or {nu1ABase, u2Length} overlaps {nu1ZBase, u2Length}

### 37.3.4.12.7 Status Returned Values

**Table 37-38.** GCD Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly

### 37.3.4.13 Get Random Number

#### 37.3.4.13.1 Purpose

The purpose of this command is to provide the user with a source of entropy. The options available for this service are:

- Generation of random numbers from a Hardware Random Number Generator (TRNG).
- Generation of random numbers from a Deterministic Random Number Generator (DRNG).



**Important:**

When using this service, be sure to strictly follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG. If the directives require not to use this service, follow them and use the proposed method to get random numbers.

This service only has the option to get random numbers and does not seed, initialize or start the RNG. Other options are reserved for future use.

Neither continuous testing nor entropy testing is included in this service. If this is needed (FIPS 140, ZKA, ...), this service must not be used and the users develops their own command.

The DRNG is compatible with both ANSI X9.31 and FIPS 186-2 standards (see the important note below). The DRNG is designed according to:

- The algorithm described in the document ANSI Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA) X9.31 dated September 9, 1998.
- The Change recommendation for ANSI X9.0 - 1995 (Part 1) and ANSI X9.31 -1998:

The algorithm B.2.1 Algorithm for computing m Values of x is the one applied in the Toolbox 3 X9.31 DRNG. The DRNG is compatible with:

- The DRNG is described in the document NIST Digital Signature Standard (DSS) FIPS Pub 186-2 January 27, 2000 Appendix 3.1
- The FIPS 186-2 Change Notice 1 dated October 5, 2001 modifies this algorithm.



**Important:** To apply the FIPS 186-2 algorithm, the parameters XSeed[0] and XSeed[1] must be set to the same value.

#### 37.3.4.13.2 How to Use the Service

#### 37.3.4.13.3 Description

This service has four possible options described in [Table 37-41](#). Two of these options are reserved for future use. This service performs the following operations:

- Generation of a random number from the Hardware RNG
- Generation of a random number from the Deterministic RNG

### Generation of a Random Number from the Hardware RNG

This service, activated with the option PUKCL\_RNG\_GET, makes it possible to get a random number R from the Hardware RNG:

R = HardwareRandomGenerate()

In the Generation of random from the RNG service, the following parameters need to be provided:

- R the generated number area (pointed by {nu1RBase,u2RLength})

#### 37.3.4.13.4 Generation of a Random Number from the Deterministic RNG

This service, activated with the option PUKCL\_RNG\_X931\_GET, makes it possible to get a random number R from the Deterministic Random Number Generator with input parameters the Key XKey and the Seed XSeed:

(XKey, R) = DeterministicRandomGenerateFromSeed ( XKey, XSeed, Q)

In the generation of a random number from the Deterministic RNG service, the following parameters need to be provided:

- XKey the input and output Key (pointed by {nu1XKeyBase,u2XKeyLength})
- XSeed the input Seed (pointed by {nu1XseedBase,u2XKeyLength})
- Q the prime number (pointed by {nu1QBase, 20bytes})
- R the generated number area (pointed by {nu1RBase, 20bytes})

#### 37.3.4.13.5 Hardware RNG Parameters Definition

The parameters for the generation of random from the Hardware RNG are described in the following table. This service can easily be accessed through the use of the PUKCL\_Rng () and PUKCL () macros.

**Table 37-39.** RNG Service Hardware Generated Parameters

Parameter	Type	Dir.	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	-	-	Option (see Table 37-41)	Option (see Table 37-41)
nu1RBase	nu1	I	Crypto RAM or Device RAM	u2RLength	Base of R	Base of R filled with random values depending on the option
u2RLength	u2	I	-	-	Length of R	Length of R

#### 37.3.4.13.6 Deterministic RNG Parameters Definition

The parameters for the generation of random from the Deterministic RNG are described in the following table. This service can easily be accessed through the use of the PUKCL\_Rng () and PUKCL () macros.

**Table 37-40.** RNG Service Deterministic Generated Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	-	-	Option (see Table 37-41)	Option (see Table 37-41)
nu1XKeyBase	nu1	I/O	Crypto RAM	u2XKeyLength	Base of XKey	Base of XKey filled with the resulting XKey
nu1Workspace	nu1	NA	Crypto RAM	64 bytes	Base of the workspace	Base of the workspace corrupted
nu1Workspace2 <sup>(1)</sup>	nu1	NA	Crypto RAM	2*u1XKeyLength + 4	Base of the workspace 2	Base of the workspace corrupted
nu1XSeedBase	nu1	I/O	Crypto RAM	max ( 2*u2XKeyLength, 44 bytes)	Base of the values XSeed[0] and XSeed[1]	Base of XSeed filled with the result on 20 bytes

.....continued

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2XKeyLength	u2	I	-	-	Length of XKey, Xseed[0] and Xseed[1]	Length of XKey, Xseed[0] and Xseed[1]
nu1QBase	nu1	I	Crypto RAM	20 bytes	Base of Q	Base of Q
nu1RBase	nu1	I	Crypto RAM	u2RLength	Base of R	Base of R filled with the result on 20 bytes

**Note:**

1. The nu1 Workspace2 must be a multiple of 256.

### 37.3.4.13.7 Options

The option is set by the u2Options input parameter that must take one of the values listed in the following table.

**Note:** The values, OPTION\_RNG\_SEED and OPTION\_RNG\_GETSEED, are reserved for future use.

**Table 37-41.** RNG Service Options

Option	Purpose	Required Parameters
PUKCL_RNG_SEED	Reserved	Reserved
PUKCL_RNG_GET	Generation of a random number from the RNG	nu1RBase, u2RLength
PUKCL_RNG_X931_GET	Generation of a random number from the Deterministic RNG	nu1XKeyBase, nu1Workspace, nu1XSeedBase, u2XKeyLength, nu1QBase, nu1RBase
PUKCL_RNG_GETSEED	Reserved	Reserved

### 37.3.4.13.8 Code Example

```

PUKCL_PARAM PUKCLParam;
P_PUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL(u2Option) =...;

// Initializing parameters
PUKCL_Rng(nu1RBase) = <Base of the ram location to store the rng>;
PUKCL_Rng(u2RLength) = <Length of the rng to get>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Rng,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    // The RNG generation has been executed correctly
    ...
}
else // Manage the error

```

### 37.3.4.13.9 Constraints

#### Random Number Generation

The following conditions must be avoided to ensure that the service works correctly:

- {nu1RBase,u2RLength} not in RAM
- {nu1RBase,u2RLength} not accessible or authorized for writing

#### Deterministic Random Number Generation

The length of the parameter nu1XSeedbase is: XSeedLength = max( 2\*u2XKeyLength, 44 bytes) The max () macro takes a maximum of two values.

The following conditions must be avoided to ensure that the service works correctly:

- nu1XKeyBase, nu1Workspace, nu1Workspace2, nu1XSeedBase, nu1QBase, nu1RBase are not aligned on 32-bit boundaries
- {nu1XKeyBase, u2XKeyLength}, {nu1Workspace, 64 bytes}, {nu1Workspace2, 2\*u1XKeyLength+4}, {nu1XSeedBase, XSeedLength}, {nu1QBase, 24 bytes} or {nu1RBase, 20 bytes} are not in PUKCC RAM
- u2XKeyLength is either: < 20, > 64 or not a 32-bit length
- nu1Workspace2 not multiple of 256.
- Overlaps exist between two or more of the areas: {nu1XKeyBase, u2XKeyLength}, {nu1Workspace, 64 bytes}, {nu1XSeedBase, XSeedLength}, {nu1QBase, 24 bytes} or {nu1RBase, 20 bytes}  
The area {nu1RBase, 20} can overlap with {nu1Workspace, 64 bytes} or {nu1QBas, 24 bytes}. The pointer nu1RBase can equal the pointer nu1XSeedBase.

### 37.3.4.13.10 Status Returned Values

**Table 37-42.** RNG Service Return Codes

Returned status	Importance	Meaning
PUKCL_OK	Information	Service functioned correctly

### 37.3.5 Modular Arithmetic Services

This section provides a complete description of the modular arithmetic services, which consists of two sets:

- Modular reductions, which can be used as stand alone operations, or used as a final step of most arithmetic operations (full and small multiplications, squaring).
- Modular operations, which include modular exponentiations (with or without using the CRT) and a probabilistic prime number generation.

These operations work on general data so the modulus has no special form. The modular services are available through:

- a Fast form (may return a congruence of the result, with a high probability to have a Normalized result)
- a Normalized form (returns the exact result, strictly lower than the modulus)
- a Euclidean form (returns the exact result, strictly lower than the modulus)

The following table describes the modes of the modular reduction with the hypothesis:

- In GF(p): The modulus is N with length NLength in bytes
- In GF(2<sup>n</sup>): The modulus is P[X] with length NLength in bytes

For the exact calculus of NLength see below.

**Table 37-43.** Modular Reduction Modes

Modular Reduction Form	Input Dynamic	Result Dynamic	Comments
Fast	GF(p): $0 \leq \text{Input} < (N^2) * (2^{32})$ GF(2 <sup>n</sup> ): $\text{Input} < ((P[X])^2) * (X^{32})$	GF(p): $0 \leq \text{Res} < N * 4$ GF(2 <sup>n</sup> ): $\text{Res} < P[X] * (X^2)$	The fastest reduction available, needs a precomputed constant.
Normalized	InputLength < NLength + 4 bytes	GF(p): $0 \leq \text{Res} < N$ GF(2 <sup>n</sup> ): $\text{Res} < P[X]$	The correction step does not runs in constant time. Needs a precomputed constant.  The Normalize function cannot be applied to the product of two numbers of length u2NLength.



.....continued			
Modular Reduction Form	Input Dynamic	Result Dynamic	Comments
Using Euclidean division	InputLength < 2 * NLength + 4 bytes	GF(p): $0 \leq \text{Res} < N$ GF(2 <sup>n</sup> ): Res < P[X]	Does not need any precomputed constant.

To be able to use these modular reduction services (except the Euclidean division), first the implementer shall call the setup service, providing the modulus as well as one free memory space for the constant (this constant is used to speed up the modular reduction). In most commands (except the modular exponentiation), the quotient is stored in the high order bytes of the number to be reduced, using only eight bytes more than the maximum size of the number to be reduced.

The following rules must be respected to ensure the modular reduction services function correctly:

- The numbers to be reduced can have any significant length, given the fact it CANNOT BE GREATER than  $2 * u2\text{ModLength} + 4$  bytes.
- The modulus SHALL ALWAYS HAVE a significant length of  $<u2\text{ModLength}>$  bytes. The modulus must be provided as a  $<u2\text{ModLength} + 4>$  bytes long number, padded on the most significant side with a 32-bit word cleared to zero. Not respecting this rule leads to unexpected and wrong results from the modular reduction.
- The normalization operation ALWAYS performs a modular reduction step, and will therefore have the same memory usage as this one.
- The very first operation before any modular operation SHALL BE a modular setup.

### 37.3.5.1 Modular Reduction

#### Related Links

[37.3.3.4. Aligned Significant Length](#)

#### 37.3.5.1.1 Purpose

This service is used to perform the various steps necessary to perform a modular reduction and accepts as input numbers in GF(p) or polynomials in GF(2<sup>n</sup>).

The available options for this service are:

- Work in the GF(2<sup>n</sup>) or in the standard integer arithmetic field GF(p)
- Operation is the generation of the reduction constant.
- Operation is a Modular Reduction.
- Operation is a Normalization.

#### 37.3.5.1.2 How to Use the Service

#### 37.3.5.1.3 Description

This service performs one of the following operations:

- Setup of the Fast or Normalize functions: generation of the reduction constant
- Fast Modular Reduction
- Big Modular Reduction (using Euclidean's division)
- Normalization

The service name for this operation is RedMod.

#### 37.3.5.1.4 Modular Reduction Setup

This service calculates the constant Cns, computed from the modulus and used to speed up the modular reduction:

$$\text{Cns} = \text{SetupConstant}(N)$$

This service must be processed before the use of the Fast or Normalize functions. In the Setup computations, the following data must be provided:

- N the modulus (pointed by {nu1ModBase,u2ModLength +4}).
- Cns the Setup Constant Result (pointed by {nu1CnsBase,u2ModLength +12}).
- X used as a workspace (pointed by {nu1XBase,2 \* u2ModLength + 8}) (include the supplementary bytes; see **Note 2** in [Table 37-44](#))
- R used as a workspace (pointed by {nu1RBase,64 or 68bytes}).
- u2ModLength is the Aligned Significant Length of the modulus and is not the byte Significant Length (see *Aligned Significant Length* from Related Links).

### 37.3.5.1.5 Fast Reductions and Normalization

These commands calculate an approximated or exact Modular Reduction, that is, the result may be greater than the modulus, but is always congruent to the true result.



**Important:** Before using these functions, ensure that the constant Cns has been calculated with the setup for the Modular Reduction service.

Input and Result significant values verify:

- For the Fast Modular Reduction:

$$0 \leq X < N^2 \times 2^{32}$$

$$R = X \bmod(N) + k \times N \quad \text{with} \quad 0 \leq k \leq 4$$

- For the Normalize:

$$XLength < (NLength + 4)bytes \quad R$$

$$= X \bmod(N)$$

In these Fast Modular Reduction and Normalize computations, the following data have to be provided:

- X (pointed by {nu1XBase,2 \* u2ModLength +8})
  - The Normalize computation accept as entry a value whose length is lower or equal to u2ModLength + 4 (that is, for example, a value yet reduced but not normalized.). The u2ModLength + 4 MSB bytes are cleared at the beginning of the computation.
  - in case of Fast RedMod computations, the value X may verify:  $X < (N^2) \times (2^{32})$ .
  - include the supplementary bytes; see **Note 3** in [37.3.5.1.8. Fast Modular Reductions Service Parameters Definition](#).
- R (pointed by {nu1RBase,u2Modlength +4})
- N (pointed by {nu1ModBase,u2ModLength +4})
- Cns (pointed by {nu1CnsBase,u2ModLength +12})
- u2ModLength is the Aligned Significant Length of the modulus and is not the byte Significant Length (see *Aligned Significant Length* from Related Links).

The Fast Modular Reduction is able to reduce inputs up to  $<2 * u2ModLength + 4>$  bytes. The input can come from a multiplication of 2  $<u2ModLength + 4>$  bytes numbers. The input X is considered as a  $<2 * u2ModLength + 8>$  bytes number.



**Important:** Additionally the Fast Reduction and Normalize functions need supplemental bytes located on the MSB side of the number to be reduced but these bytes are restored at the end of the operation and are therefore unchanged. However, these bytes are to be taken into account when the mapping is created, and could lead to unexpected results if overlapping with other area used by the function.

The Fast Modular Reduction returns a  $\langle u2ModLength + 4 \rangle$  bytes number, but this number is in fact a  $\langle u2ModLength + 2 \rangle$  significant bytes number. When using the Fast Modular Reduction, the two MSB bytes of the  $\langle u2ModLength + 2 \rangle$  can have a maximum of two lsb bits set (depending on the reduced number and the modulo).

The Normalize computation accepts as entry a resulting value of Fast Modular Reduction and computes an exact result. It can not be applied to the result of the product of two numbers of size NLength: a Fast Modular Reduction must be applied before.

For the Normalize computation, the X value is limited by the preceding formula but the area in memory is bigger as described in [37.3.5.1.8. Fast Modular Reductions Service Parameters Definition](#).

As input, the Normalize sub-service only accept values, which length is lower or equal to  $u2ModLength + 4$ . The Most Significant  $u2ModLength + 4$  bytes are firstly cleared by this service.

### 37.3.5.1.6 Big Modular Reduction Using Euclide's Division

This command calculates:

$$XLength < (2 \times NLength + 4)bytes \quad R \\ = Xmod(N)$$

In this Big Modular Reduction computations, the following data must be provided:

- X (pointed by  $\{nu1XBase, 2 * u2ModLength + 8\}$ ) (include the supplementary bytes; see **Note 1** in [Table 37-46](#))
- R (pointed by  $\{nu1RBase, u2Modlength + 4\}$ )
- N (pointed by  $\{nu1ModBase, u2ModLength + 4\}$ )
- $u2ModLength$  is the Aligned Significant Length of the modulus and is not the byte Significant Length (see *Aligned Significant Length* from Related Links).
- Workspace (pointed by  $\{nu1CnsBase, 64 \text{ or } 68\}$ ).

### 37.3.5.1.7 Modular Reductions Service Parameters Definition

**Table 37-44.** RedMod Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	-	-	Options (see below)	Options (see below)
Specific/CarryIn	Bits	I	-	-	Must be set to zero.	-
Specific/Gf2n	Bit	I	-	-	GF(2 <sup>n</sup> ) Bit	-
Specific/CarryOut Zero Violation	Bits	I	-	-	-	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase <sup>(1)</sup>	nu1	I	Crypto RAM	$u2ModLength + 4$	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	$u2ModLength + 12$	Base of Cns	Base of Cns filled with the Setup Constant
u2ModLength	u2	I	-	-	Length of N	Length of N
nu1RBase	nu1	I	Crypto RAM	GF(p): 64 bytes GF(2n): 68 bytes	Base of R as a workspace	Base of R workspace corrupted

.....continued

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1XBase <sup>(2)</sup>	nu1	I	Crypto RAM	2*u2ModLength + 8	Base of X as a workspace	Base of X workspace corrupted

**Notes:**

1. The Modulus is to be given as a u2ModLength Aligned Significant Length Bytes however, it has to be provided as a u2ModLength + 4 bytes long number, having the four high-order bytes set to zero.
2. Before the X (pointed by {nu1XBase,2 \* u2ModLength + 8}) LSB bytes, four supplementary bytes will be saved/restored. Other four supplementary bytes will also be saved/restored after the X MSB bytes. All these supplementary bytes may be entirely in the Crypto RAM (therefore, do not place the X area too near the end of the Crypto RAM) and shall not overlap with other area used by the service.

**37.3.5.1.8 Fast Modular Reductions Service Parameters Definition**

**Table 37-45.** Fast RedMode and Normalize Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	-	-	Options (see below)	Options (see below)
Specific/CarryIn	Bits	I	-	-	Must be set to zero.	-
Specific/Gf2n	Bit	I	-	-	GF(2n) Bit	-
Specific/CarryOut Zero Violation	Bits	I	-	-	-	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase <sup>(1)</sup>	nu1	I	Crypto RAM	u2ModLength + 4	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 12	Base of Cns	Base of Cns untouched
u2ModLength	u2	I	-	-	Length of N	Length of N
nu1RBase <sup>(2)</sup>	nu1	I	Crypto RAM	u2ModLength + 4	Base of R	Base of R filled with the result
nu1XBase <sup>(3)</sup>	nu1	I	Crypto RAM	2*u2ModLength + 8	Base of X the number to reduce	Base of X corrupted

**Notes:**

1. The Modulus is to be given as a u2ModLength Aligned Significant Length Bytes however, it has to be provided as a u2ModLength + 4 bytes long number, having the four high-order bytes set to zero.
2. To make profitable the space memory, it is possible to set nu1RBase exactly equal to nu1XBase.
3. After the X (pointed by {nu1XBase,2 \* u2ModLength + 8}) MSB bytes, supplementary bytes will be saved/restored (8 bytes in case of Fast RedMod, otherwise; 12 bytes). These supplementary bytes may be entirely in the Crypto RAM (therefore, do not place the X area too near the end of the Crypto RAM) and shall not overlap with other area used by the service.

**37.3.5.1.9 Big Modular Reduction Parameters Definition**

**Table 37-46.** Big RedMod Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	-	-	Options (see below)	Options (see below)
Specific/CarryIn	Bits	I	-	-	Must be set to zero	-
Specific/Gf2n	Bit	I	-	-	GF(2n) Bit	-

.....continued

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
Specific/CarryOut Zero Violation	Bits	I	-	-	-	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	GF(p): 64 bytes GF(2n): 68 bytes	Base of Cns as a workspace	Base of Cns corrupted
u2ModLength	u2	I	-	-	Length of N	Length of N
nu1RBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of R	Base of R filled with the result
nu1XBase <sup>(1)</sup>	nu1	I	Crypto RAM	2*u2ModLength + 8	Base of X the number to reduce	Base of X filled with the result

**Note:**

1. Before the X (pointed by {nu1XBase, 2 \* u2ModLength + 8}) LSB bytes, four supplementary bytes will be saved/restored. Other four supplementary bytes will also be saved/restored after the X MSB bytes. All of these supplementary bytes may be entirely in the Crypto RAM (therefore, do not place the X area too near the end of the Crypto RAM) and shall not overlap with other area used by the service.

**37.3.5.1.10 Options**

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Operation Option described in [Table 37-47](#)
- if the Operation Option is PUKCL\_REDMOD\_REDUCTION, the Modular Reduction Sub-Option described in [Table 37-48](#)

The u2Options number is calculated by an Inclusive OR of the options. Some Examples in C language are:

- Operation: Setup for the ModularReductions.  
`PUKCL(u2Options) = PUKCL_REDMOD_SETUP;`
- Operation: Fast ModularReduction.  
`PUKCL(u2Options) = PUKCL_REDMOD_REDUCTION | PUKCL_REDMOD_USING_FASTRED;`

For this command three exclusive options can be specified. The following table lists the operations that can be performed.

**Table 37-47. RedMod Service Options**

Option	Purpose	Required Parameters
PUKCL_REDMOD_SETUP	Perform the Cns value computation	nu1ModBase, u2ModLength, nu1CnsBase, nu1XBase
PUKCL_REDMOD_REDUCTION	Perform $R \equiv X \text{ Mod } N$ , see sub-option for details	nu1ModBase, u2ModLength, nu1CndBase, nu1XBase, nu1RBase
PUKCL_REDMOD_NORMALIZE	Perform $R = X \text{ Mod } N$	nu1ModBase, u2ModLength, nu1CndBase, nu1XBase, nu1RBase

When selecting the PUKCL\_REDMOD\_REDUCTION option, one of the two sub-options listed in the following table must be selected.

**Table 37-48. RedMode Service Options with PUKCL\_RED\_MOD\_REDUCTION**

Option	Purpose	Required Parameters
PUKCL_REDMOD_USING_DIVISION	Perform $R = X \text{ Mod } N$	nu1ModBase, u2ModLength, nu1CndBase, nu1XBase

.....continued

Option	Purpose	Required Parameters
PUKCL_REDMOD_USING_FASTRED	Perform $R \equiv X \text{ Mod } N$ The entropy is minimized (~2 bits)	nu1ModBase, u2ModLength, nu1CndBase, nu1XBase, nu1RBase

### 37.3.5.1.11 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL(Specific).CarryIn = 0;
PUKCL(Specific).GF2n = ...;

PUKCL(u2Option) = ...;

// Depending on the option specified, not all fields must be filled
PUKCL_RedMod(nu1ModBase) = <Base of the ram location of N>;
PUKCL_RedMod(u2ModLength) = <Length of N>;
PUKCL_RedMod(nu1CnsBase) = <Base of the ram location of Cns>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(RedMod,pvPUKCLParam);
if (PUKCL_Param.Status == PUKCL_OK)
    {
        // operation has correctly been performed
        ...
    }
else // Manage the error

```

### 37.3.5.1.12 Constraints

Depending on the options chosen the lengths of the R area and Cns area differ:

- For the Setup:
  - RLength = 64bytes
  - CnsLength = u2ModLength + 12
- For the Fast Reduction and Normalize:
  - RLength = u2ModLength + 4
  - CnsLength = u2ModLength + 8
- For the BigRedMod:
  - RLength = u2ModLength + 4
  - CnsLength = 64

The following combinations of input values must be avoided in the case of a modular reduction 'alone', meaning that it has not been requested as an option of any other command:

- nu1ModBase, nu1CnsBase, nu1RBase, nu1XBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2CnsLength}, {nu1XBase, 2\*u2XLength + 8 + s} or {nu1RBase, u2RLength} are not in Crypto RAM
- u2ModLength is either: < 4, > 0xffc or not a 32-bit length
- Overlaps exist between two or more of the areas: {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2CnsLength}, {nu1XBase, 2\*u2XLength + 8 + s} or {nu1RBase, u2RLength}

**Note:** Overlaps between {nu1RBase, RLength} and {nu1XBase, 2\*u2XLength + 8} are forbidden; but if the operation is the Fast, Normalized or Big Modular Reduction, the equality between nu1RBase and nu1XBase is authorized.

### 37.3.5.1.13 Status Returned Values

**Table 37-49.** RedMod Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly
PUKCL_DIVISION_BY_ZERO	Severe	When computing an Euclidean division, the Modulus was found to be zero. This occurs ONLY when either reducing using an Euclidean division or computing the reduction constant usable for a Fast or Normalize Reduction.
PUKCL_UNEXPLOITABLE_OPTIONS	Severe	A bad combination of options has been detected.
PUKCL_MALFORMED_MODULUS	Severe	The Msw of the modulus is not zero.

### 37.3.5.2 Modular Exponentiation (Without CRT)

#### 37.3.5.2.1 Purpose

This service is used to perform the Modular Exponentiation computation. This service processes integers in GF(p) only.

The options available for this service are:

- Fast implementation
- Regular implementation
- Exponent is located in Crypto RAM or not in Crypto RAM
- Exponent window size

#### 37.3.5.2.2 How to Use the Service

#### 37.3.5.2.3 Description



**Important:** Before using these functions, ensure that the constant Cns has been calculated with the Setup of the Modular Reductions service.

This service processes the following operation:

The service name for this operation is `ExpMod`.

$$R = X^{Exp} \bmod(N)$$

In this computation, the following parameters need to be provided:

- X: input number (pointed by {nu1XBase,u2ModLength +16})
- N: modulus (pointed by {nu1ModBase,u2ModLength +4}).
- Exp: exponent (pointed by {pfu1ExpBase,u2ExpLength +4})
- Cns: Fast Modular Constant (pointed by {nu1CnsBase,u2ModLength +8})
- Precomp: precomputation workspace (pointed by {nu1PrecompBase,PrecompLen})
- Blinding: exponent blinding value (provided inu1Blinding)

The length PrecompLen depends on the lengths and options chosen; its calculus is detailed in Options below.

**Note:** The minimum value for u2ModLength is 12 bytes. Therefore, the significant length of N must be at least three 32-bit words.

### 37.3.5.2.4 Parameters Definition

**Table 37-50.** ExpMod Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	-	-	Options (see below)	Options (see below)
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns untouched
u2ModLength	u2	I	-	-	Length of N	Length of N
nu1XBase <sup>(1)</sup>	nu1	I	Crypto RAM	u2ModLength + 16	Base of X	Base of X Filled with the result
nu1PrecompBase	nu1	I	Crypto RAM	See below	Base of Precomp as a workspace	Base of Precomp workspace corrupted
pfu1ExpBase <sup>(2)</sup>	pfu1	I	Any place <sup>(3)</sup>	u2ExpLength + 4	Base of the Exponent	Base of the Exponent untouched
u2ExpLength <sup>(4)</sup>	u2	I	-	-	Significant length of Exponent	Significant length of Exponent
u1Blinding <sup>(5)</sup>	u1	I	-	-	Exponent unblinding value	Exponent unblinding value untouched

**Notes:**

1. This zone contains the number to be exponentiated (u2ModLength bytes) and is used during the computations as a workspace (four 32-bit words longer than the number to be exponentiated). At the end of the computation, it contains the correct result of the operation.
2. The exponent must be given with a supplemental word on the LSB side (low addresses). This word shall be set to zero.
3. If the PUKCL\_EXPMOD\_EXPINPUKCCRAM option is not set, the location of the exponent MUST NOT be the Crypto RAM, even partially.
4. The u2ExpLength parameter does not take into account the supplemental word needed on the LSB side of the exponent.
5. It is possible to mask the exponent in memory using an 8-bits XOR mask value. Be aware that not only the exponent, but also the supplemental word has to be masked. If masking is not desired, then this parameter must be set to 0.

### 37.3.5.2.5 Options

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Calculus Mode Option described in [Table 37-51](#)
- the mandatory Window Size Option described in [Table 37-52](#)
- the indication of the presence of the exponent in Crypto RAM

**Note:** Please check precisely if one part of the exponent is in Crypto RAM. If this is the case the PUKCL\_EXPMOD\_EXPINPUKCCRAM must be used.

The u2Options number is calculated by an “Inclusive OR” of the options. Some examples in C language are:

- Operation:Fast Modular Exponentiation with the window size equal to 1 and with no part of the Exponent in the Crypto RAM  
`PUKCL(u2Options) = PUKCL_EXPMOD_FASTRSA | PUKCL_EXPMOD_WINDOWSIZE_1;`
- Operation: Regular Modular Exponentiation with the window size equal to 2 and with one part of the Exponent in the Crypto RAM  
`PUKCL(u2Options) = PUKCL_EXPMOD_REGULARRSA | PUKCL_EXPMOD_WINDOWSIZE_2 | PUKCL_EXPMOD_EXPINPUKCCRAM;`



There is no difference on the final result when using any of the options for this service. The choice has to be made according to the available resources (RAM, Time) and also taking into account the expected security level.

For this service, two exclusive Calculus Modes are possible. The following table describes the Calculus Mode Options.

**Table 37-51.** ExpMod Service Calculus Mode Option

Option	Explanation
PUKCL_EXPMOD_FASTRSA	Performs a Fast computation
PUKCL_EXPMOD_REGULARRSA	Performs a Regular computation, slower than the Fast version, but using Regular calculus methods

For this service, four window sizes are possible. The window size in bits is those of the windowing method used for the exponent.

The choice of the window size is a balance between the size of the parameters and the computation time:

- Increasing the window size increases the precomputation workspace.
- Increasing the window size reduces the computation time (may not be relevant for very small exponents).

The following table details the size of the precomputation workspace, depending on the chosen window size option.

**Table 37-52.** ExpMod Service Window Size Options and Precomputation Space Size

Option specified	Size of the PrecompBase Workspace (bytes)	Content of the Workspace
PUKCL_EXPMOD_WINDOWSIZE_1	$3*(u2ModLength + 4) + 8$	x
PUKCL_EXPMOD_WINDOWSIZE_2	$4*(u2ModLength + 4) + 8$	$x x^3$
PUKCL_EXPMOD_WINDOWSIZE_3	$6*(u2ModLength + 4) + 8$	$x x^3 x^5 x^7$
PUKCL_EXPMOD_WINDOWSIZE_4	$10*(u2ModLength + 4) + 8$	$x x^3 x^5 x^7 x^9 x^{11} x^{13} x^{15}$

The exponent can be located in RAM or in the data space. If one part of the exponent is in Crypto RAM this must be mandatory signaled by using the option PUKCL\_EXPMOD\_EXPINPUKCCRAM.

The following table describes this option.

**Table 37-53.** ExpMod Service Exponent in Crypto RAM Option

Option	Purpose
PUKCL_EXPMOD_EXPINPUKCCRAM	The exponent can be read from any data space of memory, including Flash, RAM or even Crypto RAM. When at least one word the exponent is in Crypto RAM, this option has to be set.

### 37.3.5.2.6 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL(u2Option) = ...;

// Depending on the option specified, not all fields must be filled
PUKCL_ExpMod(nulModBase) = <Base of the ram location of N>;
PUKCL_ExpMod(u2ModLength) = <Length of N>;
PUKCL_ExpMod(nulCnsBase) = <Base of the ram location of Cns>;
PUKCL_ExpMod(nulXBase) = <Base of the ram location of X>;
PUKCL_ExpMod(nulPrecompBase) = <Base of the ram location of Precomp>;
PUKCL_ExpMod(pfufExpBase) = <Base of the location of Exp>;
PUKCL_ExpMod(u2ExpLength) = <Length of Exp>;
...

```

```
// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ExpMod, pvPUKCLParam);
if (PUKCL_Param.Status == PUKCL_OK)
    {
        // operation has been performed correctly
        ...
    }
else // Manage the error
```

### 37.3.5.2.7 Constraints

The following combinations of input values must be avoided in the case of a modular reduction 'alone', meaning that it has not been requested as an option of any other command:

- nu1ModBase, nu1CnsBase, nu1XBase, nu1PrecompBase, nu1ExpBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1XBase, u2ModLength + 16}, {nu1PrecompBase, <PrecompLength>} are not in Crypto RAM
- {nu1ExpBase, u2ExpLength + 4} has no part in Crypto RAM and PUKCL\_EXPMOD\_EXPINPUKCCRAM is specified
- u2ModLength or u2ExpLength are either: < 4, > 0xffc or not a 32-bit length
- None or both PUKCL\_EXPMOD\_REGULARRSA and PUKCL\_EXPMOD\_FASTRSA are specified.
- {nu1PrecompBase, <PrecompLength>} overlaps with either: {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8} {nu1XBase, u2ModLength + 16} or {nu1ExpBase, u2ExpLength + 4}
- {nu1XBase, u2ModLength + 16} overlaps with either: {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8} or {nu1ExpBase, u2ExpLength + 4}
- {nu1ModBase, u2ModLength + 4} overlaps {nu1CnsBase, u2ModLength + 8}

### 37.3.5.2.8 Maximum Sizes for the Modular Exponentiation

The following table provides the maximum sizes for the Modular Exponentiation, depending on the window size and the presence of the exponent in Crypto RAM.

- The figures below are calculated supposing that u2ExpLength = u2ModLength.
- In case of the PUKCL\_EXPMOD\_EXPINPUKCCRAM option is specified, for the computation of the maximum acceptable size, it is assumed the Exponent is entirely in the Crypto RAM and its length is equal to the Modulus one.
- Otherwise, the Exponent is entirely out of the Crypto RAM and so the computation do not depend on its length.

**Table 37-54.** Maximum Exponentiation Sizes

Option Specified	Maximum Modulus Size (bytes)	Maximum Modulus Size (bits)
Exponent in Crypto RAM, 1 bit window	576	4608
Exponent in Crypto RAM, 2 bits window	504	4032
Exponent in Crypto RAM, 3 bits window	400	3200
Exponent in Crypto RAM, 4 bits window	284	2272
Exponent not in Crypto RAM, 1 bit window	672	5376
Exponent not in Crypto RAM, 2 bits window	576	4608
Exponent not in Crypto RAM, 3 bits window	448	3584
Exponent not in Crypto RAM, 4 bits window	308	2464

### 37.3.5.2.9 Status Returned Values

**Table 37-55.** ExpMod Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	Service functioned correctly

### 37.3.5.3 Probable Prime Generation (Using Rabin-Miller)

#### 37.3.5.3.1 Purpose

This service is used to perform probable prime generation or test. This service processes integers in GF(p) only.

The options available for this service are:

- Choice of the number of iterations of the Rabin-Miller test
- Generation or Test of a probable prime number
- Fast Implementation
- Regular Implementation
- Exponent Window Size

#### 37.3.5.3.2 Additional Information

The Rabin-Miller test is a probable-primality testing algorithm. As a consequence, the primality of the generated number is not guaranteed at 100%, however, numerous publications have been issued explaining how to estimate the probability of getting a composite number, giving the size of the number and the number of iterations (the T parameter).

Useful information can be found in the *"Handbook of Applied Cryptography (Discrete Mathematics and Its Applications)"* by Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, in the following sections:

- 4.2.3. "Rabin-Miller Test"
- 4.4. "Prime Number Generation"

#### 37.3.5.3.3 How to Use the Service

#### 37.3.5.3.4 Description

This service processes a test for probable primality or a generation of a probable prime number.

**Note:** When using this service be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

This service processes one of the following operations: CheckProbablePrimality(N)

or

N = GenerateProbablePrimeFromSeed (NSeed)

In this computation, the following parameters need to be provided:

- N the input number (pointed by {nu1NBase,u2NLength +4})
  - If the requested operation is a test, it is untouched after the operation.
  - If the requested operation is a generation and a probable prime number was found before reaching the Maximum Increment, it contains the resulting probable prime after the operation.
  - If the requested operation was a generation and Maximum Increment was reached before a probable prime number was found, it contains no relevant information.
- Cns as a workspace (pointed by {nu1CnsBase,u2NLength +12})
- Rnd as a workspace (pointed by {nu1RndBase,u2NLength +16})

- Precomp the precomputation workspace (pointed by {nu1PrecompBase,PrecompLen})
- Exp as a workspace (pointed by {pfu1ExpBase,u2ExpLength +4})
- u1MillerRabinIterations the number of Miller Rabin Iterations requested
- u2MaxIncrement, maximum increment of the number in case of probable prime generation

The length PrecompLen depends on the lengths and options chosen; its calculus is detailed in Options below.

The service name for this operation is PrimeGen.

### 37.3.5.3.5 Parameters Definition

**Table 37-56.** PrimeGen Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1NBase <sup>(1)</sup>	nu1	I	Crypto RAM	u2NLength + 4	Base of N Number to test or Seed for the generation	Base of N unchanged if test or generation result <sup>(2)</sup>
nu1CnsBase	nu1	I	Crypto RAM	u2NLength + 12	Base of Cns as a workspace	Base of Cns workspace corrupted
u2NLength	u2	I	-	-	Length of N	Length of N
nu1RndBase	nu1	I	Crypto RAM	Max (u2NLength + 16,64)	Internal Workspace	Internal Workspace corrupted
nu1PrecompBase	nu1	I	Crypto RAM	See Options below	Base of Precomp workspace	Base of Precomp workspace corrupted
nu1RBase <sup>(2)</sup>	nu1	-	Crypto RAM	-	-	-
nu1ExpBase <sup>(3)</sup>	nu1	I	Crypto RAM	u2NLength + 4	Base of Exponent (R)	Base of Exponent (R)
u1MillerRabin-Iterations	u1	I	-	-	Miller Rabin's T parameter	Miller Rabin's T parameter
u2MaxIncrement	u2	I	-	-	Maximum Increment <sup>(4)</sup>	Maximum Increment

#### Notes:

1. This zone contains the number to be either tested or used as a seed for generation. It has to be provided with one zero word on the MSB side. This area has supplementary constraints (see the following Important note).
1. This parameter does not have to be provided and is used as an internal value for computing the reduction's constant.
2. The area {nu1ExpBase, u2NLength + 4} must be entirely in the Crypto RAM.
3. The generation starts from the number in {nu1NBase,u2NLength + 4} and increments it until a number is found as probable prime. However, the generation may stop for two reasons: The number has been incremented in a way it is bigger than <u2NLength> bytes, or the original number has been incremented by more than <u2MaxIncrement>.

In case of probable prime generation, ensure that the addition of NSeed and Maximum Increment is not a number with more bytes than u2NLength, as this would produce an overflow.



**Important:**

One additional word is used on the LSB side of the NBase parameter; this word is restored at the end of the calculus. As a consequence, the parameter nu1NBase must never be at the beginning of the Crypto RAM, but at least at one word from the beginning.

One additional word is used on the MSB side of the NBase parameter; this word is not corrupted. As a consequence the Area {nu1NBase, u2NLength} must not be at the end of the Crypto RAM but at least at one word from the end.

Prime numbers of a size lower than 96 bits (three 32-bit words) cannot be generated or tested by this service.

**37.3.5.3.6 Options**

Some of the Prime Generation options configure the Modular Exponentiation steps and so are very similar to the Modular Exponentiation options.

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Operation Option described in [Table 37-57](#)
- the mandatory Calculus Mode Option described in [Table 37-58](#)
- the mandatory Window Size Option described in [Table 37-59](#)

The u2Options number is calculated by an “Inclusive OR” of the options. Some Examples in C language are:

- Operation: Probable Prime Testing with Fast Modular Exponentiation and the window size equal to 1

```
PUKCL(u2Options) = PUKCL_PRIMEGEN_TEST | PUKCL_EXPMOD_FASTRSA |
PUKCL_EXPMOD_WINDOWSIZE_1;
```

- Operation: Probable Prime Generate with Regular Modular Exponentiation and the window size equal to 2

```
PUKCL(u2Options) = PUKCL_EXPMOD_REGULARRSA | PUKCL_EXPMOD_WINDOWSIZE_2;
```

The following table describes the PrimeGen service features available from the various options.

**Table 37-57.** PrimeGen Service Options

Option	Method Used
PUKCL_PRIMEGEN_TEST	This option is used to specify that only tests will be made on the provided number.  When this option is not specified, a prime generation algorithm is selected, starting from the given seed and incrementing it.
PUKCL_EXPMOD_WINDOWSIZE_1,2,3 or 4	Depending on this option, different bit-window sizes will be used. For long exponents, the bigger the window, the faster the computation. However, this has also an impact on the size of the precomputations table.

For this service, two exclusive Calculus Modes are possible. The following table describes the Calculus Mode Options.

**Table 37-58.** PrimeGen Service Calculus Mode Options

Option	Explanation
PUKCL_EXPMOD_FASTRSA	Perform a Fast computation.
PUKCL_EXPMOD_REGULARRSA	Performs a Regular computation, slower than the Fast version, but using regular calculus methods.

The length of the Precomp area depends on the window size W and u2NLength. The Precomp area length is:

$$\text{PrecompLen} = \max(2*(u2NLength + 4) + 2W-1 * (u2NLength + 4), u2NLength + 8 + 64) + 8$$

**Note:** Please calculate precisely the length PrecompLen with the formula and the `max()` macro, which takes a maximum of two values.

The following table shows the size of the precomputation workspace (PrecompLen), depending on the chosen window size option.

**Table 37-59.** PrimeGen Service Precomputation Space Size

Option Specified	Size of the PrecompBase Workspace (bytes)	Content of the Workspace
PUKCL_EXPMOD_WINDOWSIZE_1	$\max(3*(u2NLength + 4), u2NLength + 72) + 8$	x
PUKCL_EXPMOD_WINDOWSIZE_2	$\max(4*(u2NLength + 4), u2NLength + 72) + 8$	x x <sup>3</sup>
PUKCL_EXPMOD_WINDOWSIZE_3	$\max(6*(u2NLength + 4), u2NLength + 72) + 8$	x x <sup>3</sup> x <sup>5</sup> x <sup>7</sup>
PUKCL_EXPMOD_WINDOWSIZE_4	$\max(10*(u2NLength + 4), u2NLength + 72) + 8$	x x <sup>3</sup> x <sup>5</sup> x <sup>7</sup> x <sup>9</sup> x <sup>11</sup> x <sup>13</sup> x <sup>15</sup>

The following table provides the maximum sizes for the Prime Generation depending on the window size.

**Table 37-60.** PrimeGen Service Maximum Sizes

Characteristics of the Operation	Maximum Prime Sizes (bits)
1 bit window	4608
2 bits window	4032
3 bits window	3200
4 bits window	2272

### 37.3.5.3.7 Code Example

```

PUKCL_PARAM PUKCLParam;
P_PUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip PUKCL(u2Option) = ...;
// Depending on the option specified, not all fields must be filled
PUKCL_PrimeGen(nu1NBase) = <Base of the ram location of N>;
PUKCL_PrimeGen(u2NLength) = <Length of N>;
PUKCL_PrimeGen(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL_PrimeGen(nu1PrecompBase) = <Base of the ram location of Precomp>;
PUKCL_PrimeGen(pf1ExpBase) = <Base of the location of Exp>;
PUKCL_PrimeGen(u2ExpLength) = <Length of Exp>;
PUKCL_PrimeGen(u1MillerRabinIterations) = <Number of iterations>;
PUKCL_PrimeGen(u2MaxIncrement) = <Maximum Increment>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(PrimeGen, pvPUKCLParam);
if (PUKCL_Param.Status == PUKCL_NUMBER_IS_PRIME)
{
    // The number is probably prime
    ...
}
else if (PUKCL_Param.Status == PUKCL_NUMBER_IS_NOT_PRIME)
{
    // The number is not prime
    ...
}
else // Manage the error

```

### 37.3.5.3.8 Constraints

The following combinations of input values must be avoided in the case of a modular reduction 'alone', meaning that it has not been requested as an option of any other service:

- nu1NBase, nu1CnsBase, nu1RndBase, nu1PrecompBase, nu1ExpBase are not aligned on 32-bit boundaries

- {nu1NBase, u2NLength + 4}, {nu1CnsBase, u2NLength + 12}, {nu1RndBase, u2NLength + 12}, {nu1PrecompBase, <PrecompLength>} are not in Crypto RAM
- u2NLength is either: < 12, > 0xffc or not a 32-bit length
- Both PUKCL\_EXPMOD\_REGULARRSA and PUKCL\_EXPMOD\_FASTRSA are specified.
- {nu1PrecompBase,<PrecompLength>} overlaps with either: {nu1NBase, u2NLength + 4}, {nu1CnsBase, u2NLength + 12} {nu1RndBase, u2NLength + 12} or {nu1ExpBase, u2ExpLength + 4}
- {nu1RndBase,3\*u2NLength + 24} overlaps with either: {nu1NBase, u2NLength + 4},{nu1CnsBase, u2NLength + 12} {nu1XBase, u2NLength + 12} or {nu1ExpBase, u2ExpLength + 4}
- {nu1NBase, u2NLength + 4} overlaps {nu1CnsBase, u2NLength + 12}

### 37.3.5.3.9 Status Returned Values

Table 37-61. PrimeGen Service Return Codes

Returned Status	Importance	Meaning
PUKCL_NUMBER_IS_PRIME	Information	The generated or tested number has been detected as probably prime.
PUKCL_NUMBER_IS_NOT_PRIME	Information	The generated or tested number has been detected as composite.

### 37.3.5.4 Modular Exponentiation (With CRT)

#### 37.3.5.4.1 Purpose

The purpose of this service is to perform the Modular Exponentiation with the Chinese Remainders Theorem (CRT). This service processes integers in GF(p) only.

The options available for this service are:

- Fast implementation
- Regular implementation
- Exponent is located in Crypto RAM or not
- Exponent window size

#### 37.3.5.4.2 How to Use the Service

#### 37.3.5.4.3 Description

This service processes a Modular Exponentiation with the Chinese Remainder Theorem:

$$R = X^D \text{ mod}(N) \text{ with } N = P * Q$$



**Important:** For this service, be sure to follow the directives given for the RSA implementation on the chip you use.

This service requires that the modulus N is the product of two co-primes P and Q and that the decryption exponents D is co-prime with the product ((P-1)\*(Q-1)).

The Input data are P, Q, EP, EQ, Rvalue, and X. P and Q are the co-primes so that N = P\*Q.

X is the number to exponentiate.

EP, EQ and Rval are calculated as follows:

$$EP = D \text{ mod}(P - 1) \quad EQ = D \text{ mod}(Q - 1) \quad Rval = P^{-1} \text{ mod}(Q)$$

In some cases, the decryption exponent D may not be available and the encryption exponent E may be available instead. The possibilities to calculate the parameters are:

- Calculate D from E with the formula:  
 $D = E^{-1} \text{ mod}((P - 1) \times (Q - 1))$

- Calculate the parameters from E:  
 $EP = E^{-1} \bmod(P - 1)$   $EQ = E^{-1} \bmod(Q - 1)$   $Rval = P^{-1} \bmod(Q)$

In this computation, the following parameters need to be provided:

- X the input number (pointed by {nu1XBase, 2\*u2ModLength + 16})
- P and Q the primes (pointed by {nu1ModBase, 2\*u2ModLength + 8}).
- EP and EQ the reduced exponents (pointed by {pfu1ExpBase, 2\*u2ExpLength + 8})
- Rval and Precomp (pointed by {nu1PrecompBase, RAndPrecompLen})
- Blinding the exponent blinding value (provided in u1Blinding)

The length RAndPrecompLen depends on the lengths and options chosen; its calculus is detailed in Options below.

The service for this operation is CRT.

**Note:** The minimum value for u2ModLength is 12 bytes. Therefore, the significant length of P or Q must be at least three 32-bit words.

### 37.3.5.4.4 Parameters Definition

The following table shows the parameter block for the CRT service.

Many parameters have complex placement in memory; therefore, detailed figures are provided in CRT Service Placement below.

**Table 37-62.** CRT Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	-	-	Options (see below)	Options (see below)
nu1ModBase	nu1	I	Crypto RAM	2*u2ModLength + 8	Base of P, Q	Base of P, Q untouched
u2ModLength	u2	I	-	-	Length of P or Q greater than or equal to 12	Length of P or Q
nu1XBase <sup>(1)</sup>	nu1	I	Crypto RAM	2*u2ModLength + 16	Base of X	Base of X Filled with the result
nu1PrecompBase	nu1	I	Crypto RAM	See Options below	Base of Rvalue and Pre computations workspace	Corrupted
pfu1ExpBase <sup>(2)</sup>	pfu1	I	Any place	2*u2ExpLength + 8	Base of EP, EQ	Base of EP, EQ untouched
u2ExpLength	u2	I	-	-	Significant length of EP or EQ	Significant length of EP or EQ
u1Blinding <sup>(3)</sup>	u4	I	-	-	Exponent unblinding value	Exponent unblinding value

**Notes:**

1. This zone contains the number to be exponentiated (u2ModLength bytes) and is used during the computations as a workspace (four 32-bit words longer than the number to be exponentiated). At the end of the computation, it contains the correct result of the operation.
2. If the PUKCL\_EXPMOD\_EXPINPUKCCRAM option is not set, the location of the exponent MUST NOT be placed in the Crypto RAM, even partially.
3. It is possible to mask the exponent in memory using a 32-bit XOR mask value. Be aware that not only the exponent, but also the supplemental spill word has to be masked. If masking is not desired, the parameter must be set to 0.

### 37.3.5.4.5 Options

Most of the CRT options configure the Modular Exponentiation steps of the CRT and so are very similar to the Fast Modular Exponentiation options.



The options are set by the u2Options input parameter, which is composed of:

- the mandatory Calculus Mode Option described in [Table 37-63](#)
- the mandatory Window Size Option described in [Table 37-64](#)
- the indication of the presence of the exponent in Crypto RAM



**Important:** Please check precisely if one part of the exponent area (containing EP and EQ) is in Crypto RAM. If this is the case, the PUKCL\_EXPMOD\_EXPINPUKCCRAM option must be used.

The u2Options number is calculated by an “Inclusive OR” of the options. Some Examples in C language are:

- Operation: CRT using the Fast Modular Exponentiation with the window size equal to 1 and with no part of the Exponent area in the Crypto RAM  

$$\text{PUKCL}(u2Options) = \text{PUKCL\_EXPMOD\_FASTRSA} | \text{PUKCL\_EXPMOD\_WINDOWSIZE\_1};$$
- Operation: CRT using the Regular Modular Exponentiation with the window size equal to 2 and with one part the Exponent area in the Crypto RAM  

$$\text{PUKCL}(u2Options) = \text{PUKCL\_EXPMOD\_REGULARRSA} | \text{PUKCL\_EXPMOD\_WINDOWSIZE\_2} | \text{PUKCL\_EXPMOD\_EXPINPUKCCRAM};$$

For this service, two exclusive Calculus Modes for the Modular Exponentiation steps of the CRT are possible. The following table describes the Calculus Mode Options.

**Table 37-63.** CRT Service Calculus Mode Options

Option	Explanation
PUKCL_EXPMOD_Fastrsa	Perform a Fast computation.
PUKCL_EXPMOD_Regularrsa	Performs a Regular computation, slower than the Fast version, but using regular calculus methods.

For this service, four window sizes for the Modular Exponentiation Steps are possible. The window size in bits is those of the windowing method used for the exponent.

The choice of the window size is a balance between the size of the parameters and the computation time:

- Increasing the window size increases the precomputation workspace.
- Increasing the window size reduces the computation time (may not be relevant for very small exponents). The length of the Rval and Precomp area depends on the window size W and u2ModLength.

The Rval and Precomp area length is:

$$\text{RandPrecompLen} = 4 * (u2ModLength + 4) + \max(64, 2^{(W-1)} * (u2ModLength + 4)) + 8$$



**Important:** Please calculate precisely the length RandPrecompLen with the formula and the `max()` macro, which takes the maximum of two values.

The following table shows the size of the Rval and Precomp area, depending on the chosen window size option.

**Table 37-64.** CRT Service Window Size Options and Rval and Precomp Area Size

Option Specified	Size of the Rval and Precomp Area (bytes)	Precomputation Values
PUKCL_EXPMOD_WINDOWSIZE_1	$4 * (u2ModLength + 4) + \max(64, (u2ModLength + 4)) + 8$	x

.....continued

Option Specified	Size of the Rval and Precomp Area (bytes)	Precomputation Values
PUKCL_EXPMOD_WINDOWSIZE_2	$4*(u2ModLength + 4) + \max(64, 2*(u2ModLength + 4)) + 8$	$x x^3$
PUKCL_EXPMOD_WINDOWSIZE_3	$4*(u2ModLength + 4) + \max(64, 4*(u2ModLength + 4)) + 8$	$x x^3 x^5 x^7$
PUKCL_EXPMOD_WINDOWSIZE_4	$10*(u2ModLength + 4) + \max(64, 8*(u2ModLength + 4)) + 8$	$x x^3 x^5 x^7 x^9 x^{11} x^{13} x^{15}$

The exponent area can be located in RAM or in the data space. If one part of the exponent area is in Crypto RAM this must be mandatory signaled by using the PUKCL\_EXPMOD\_EXPINPUKCCRAM option.

The following table describes this option.

**Table 37-65.** CRT Service Crypto RAM Option Exponent Area

Option	Purpose
PUKCL_EXPMOD_EXPINPUKCCRAM	The exponent area can be read from any data space of memory, including Crypto RAM. When at least one word the exponent is in Crypto RAM, this option has to be set.

### 37.3.5.4.6 Code Example

```

PUKCL_PARAM PUKCLParam;
vPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL(u2Option) = ...;

// Depending on the option specified, not all fields must be filled PUKCL_CRT(nu1ModBase) =
// <Base of the ram location of P and Q>; PUKCL_CRT(u2ModLength) = <Length of P or Q>;
PUKCL_CRT(nu1XBase) = <Base of the ram location of X>;
PUKCL_CRT(nu1PrecompBase) = <Base of the ram location of Rval and Precomp>;
PUKCL_CRT(pf1ExpBase) = <Base of the ram location of EP and EQ>;
PUKCL_CRT(u2ExpLength) = <Length of EP or EQ>;
PUKCL_CRT(u1Blinding) = <Blinding value>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(CRT, pvPUKCLParam);
if (PUKCL_Param.Status == PUKCL_OK)
    {
        // operation has been performed correctly
        ...
    }
else // Manage the error

```

### 37.3.5.4.7 Constraints

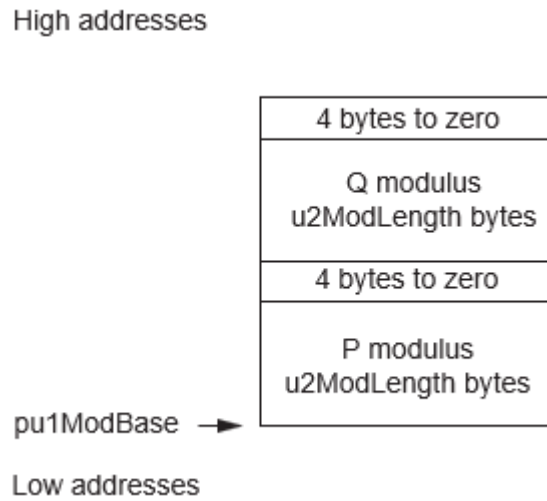
The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1XBase, nu1PrecompBase, pf1ExpBase are not aligned on 32-bit boundaries
- {nu1XBase,  $2*u2ModLength + 16$ }, {nu1ModBase,  $2*u2ModLength + 8$ }, {nu1PrecompBase, <PrecompLength>} are not in Crypto RAM
- {nu1ExpBase,  $2*u2ExpLength + 8$ } is not in Crypto RAM and PUKCL\_EXPMOD\_EXPINPUKCCRAM is specified
- u2ModLength or u2ExpLength are either:  $< 4$ ,  $> 0xffc$  or not a 32-bit length
- None or both PUKCL\_EXPMOD\_REGULARRSA and PUKCL\_EXPMOD\_FASTRSA are specified.
- {nu1XBase,  $2*u2ModLength + 16$ } overlaps with either: {nu1ModBase,  $2*u2ModLength + 8$ }, {nu1PrecompBase, <PrecompLength>} or {pf1ExpBase,  $2*u2ExpLength + 8$ }
- {nu1ModBase,  $2*u2ModLength + 8$ } overlaps with either: {nu1PrecompBase, <PrecompLength>} or {pf1ExpBase,  $2*u2ExpLength + 8$ }
- {nu1PrecompBase, <PrecompLength>} overlaps {pf1ExpBase,  $2*u2ExpLength + 8$ }

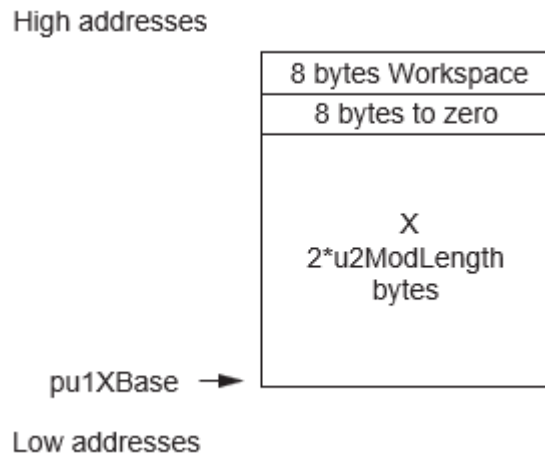
### 37.3.5.4.8 CRT Service Parameter Placement

The parameters' placements are described in detail in the following figures.

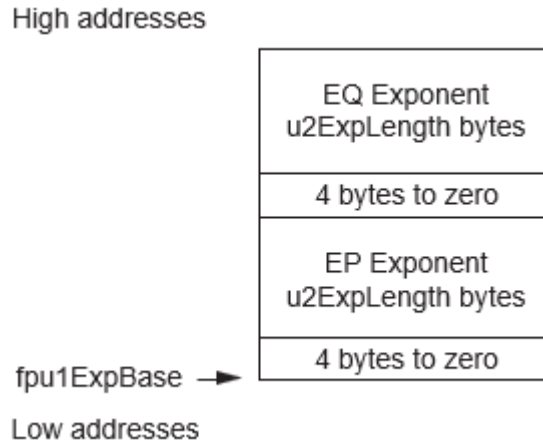
**Figure 37-2.** Modulus P and Q in {nu1ModBase, 2\*u2ModLength + 8}



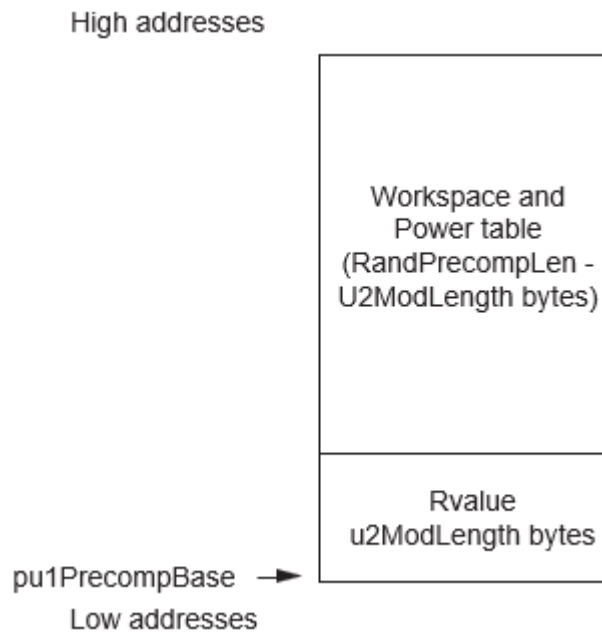
**Figure 37-3.** Value X in {nu1XBase, 2\*u2ModLength + 16}



**Figure 37-4.** Exponents EP and EQ in {fpu1ExpBase, 2\*u2ExpLength + 8}



**Figure 37-5.** Value Rval and Precomp in {nu1PrecompBase, RandPrecompLen}



#### 37.3.5.4.9 CRT Service Modular Exponentiation Maximum Size

The following table details the maximum size in bits of P or Q, of N and of EP or EQ.

- The maximum size in bits of P or Q equals:  
<Max Size Bits P> = <Max Size Bits Q> = 8 \* <Max u2ModLength bytes>
- The maximum size in bits of N=P\*Q equals:  
<Max Size Bits N> = 2 \* <Max Size Bits P>
- The maximum size in bits of EP or EQ equals:  
<Max Size Bits EP> = <Max Size Bits EQ> = 8 \* <Max u2ExpLength bytes>
- In case of the PUKCL\_EXPMOD\_EXPINPUKCCRAM option is specified, for the computation of the maximum acceptable size, it is assumed the Exponent is entirely in the Crypto RAM and its length equal the Modulus one.

- Otherwise, the Exponent is entirely out of the Crypto RAM and so the computation do not depend on its length.

**Table 37-66.** CRT Service Maximum Sizes

Characteristics of the Operation	P or Q Max Bit Sizes	N Max Bit Sizes	EP or EQ Max Bit Sizes
Exponent in Crypto RAM, 1 bit window	2912	5824	2912
Exponent in Crypto RAM, 2 bits window	2688	5376	2688
Exponent in Crypto RAM, 3 bits window	2464	4928	2464
Exponent in Crypto RAM, 4 bits window	2304	4608	2304
Exponent not in Crypto RAM, 1 bit window	3584	7168	<application dependent>
Exponent not in Crypto RAM, 2 bits window	3232	6464	<application dependent>
Exponent not in Crypto RAM, 3 bits window	2912	5824	<application dependent>
Exponent not in Crypto RAM, 4 bits window	2688	5376	<application dependent>

### 37.3.5.4.10 Status Returned Values

**Table 37-67.** CRT Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	Information	Service functioned correctly

### 37.3.6 Elliptic Curves Over GF(p) Services

This section provides a complete description of the currently available elliptic curve over Prime Fields services. These services process integers in GF(p) only.

The offered services cover the basic operations over elliptic curves such as:

- Adding two points over a curve
- Doubling a point over a curve
- Multiplying a point by an integral constant
- Converting a point's projective coordinates (resulting from a doubling or an addition) to the affine coordinates, and oppositely converting a point's affine coordinates to the projective coordinates.
- Testing the point presence on the curve.

Additionally, some higher level services covering the needs for signature generation and verification are offered:

- Generating an ECDSA signature (compliant with FIPS186-2)
- Verifying an ECDSA signature (compliant with FIPS186-2) The supported curves use the following curve equation:

$$Y^2 = X^3 + aX + b$$

#### 37.3.6.1 Coordinate Systems

##### Related Links

[37.3.5.1. Modular Reduction](#)

##### 37.3.6.1.1 General Considerations

In this implementation, several choices have been made related to the coordinate systems managed by the elliptic curve primitives.

There are two systems currently managed by the library:

- Affine Coordinates System where each curve point has two coordinates (X, Y)
- Projective Coordinates System where each point is represented with three coordinates (X,Y, Z)

Converting from the affine coordinates system to a projective coordinates system is performed by extending its representation with  $Z = 1$ :

$$(X, Y) \Rightarrow (X, Y, Z=1)$$

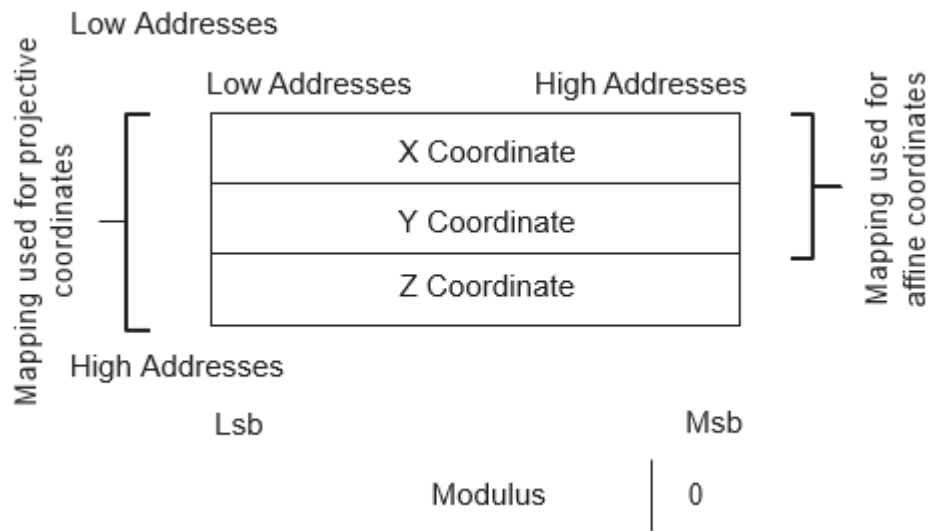
Converting from a projective coordinate to an affine one is a service offered by the PUKCL. The formula to perform this conversion is:

$$(X, Y, Z) \Rightarrow (X / Z^2, Y / Z^3)$$

### 37.3.6.1.2 Points Representations

Depending on the representation (Projective or Affine), points are represented in memory, as shown in the following figure.

Figure 37-6. Points Representation in Memory



In this figure, the modulus is represented as a reference, and to show that coordinates are always to be provided on the length of the modulus plus one 32-bit word.

The different types of representations are as follows:

Affine representation  $Pt = \begin{bmatrix} X_{Affine} < P \times 2^{15} \\ Y_{Affine} < P \times 2^{15} \end{bmatrix}$

Projective representation  $Pt = \begin{bmatrix} X_{Projective} < P \times 2^{15} \\ Y_{Projective} < P \times 2^{15} \\ Z_{Projective} < P \times 2^{15} \end{bmatrix}$

**Notes:**

1. The minimum value for u2ModLength is 12 bytes. Therefore, the significant length of the modulus must be at least three 32-bit words.
2. In some cases the point can be the infinite point. In this case, it is represented with its Z coordinates equal or congruent to zero.

### 37.3.6.1.3 Modulus and Modular Constant Parameters

In most of the services the following parameters must be provided:

- P the Modulus (often pointed by {nu1ModBase,u2ModLength + 4}): This parameter contains the Modulus Integer prime P defining the Galois Field used in points coordinates computations. The Modulus must be u2ModLength bytes long, while having a supplemental zeroed 32-bit word on the MSB side.  
**Note:** Most of the Elliptic Curve computations are reduced modulo P. In many functions the reductions are made with the Fast Reduction.
- Cns the Modular Constant (often pointed by {nu1CnsBase,u2ModLength + 12}): This parameter contains the Modular Constant associated to the Modulus



**Important:** The Modular Constant must be calculated before using the GF(p) Elliptic Curves functions by a call to the Setup for Modular Reductions with the GF(p) option (see *Modular Reduction Setup* in the *Modular Reduction* from Related Links).

### 37.3.6.2 Point Addition

#### 37.3.6.2.1 Purpose

This service is used to perform a point addition, based on a given elliptic curve over GF(p). Please note that:

- This service is not intended to add the same point twice. In this particular case, use the doubling service (see [37.3.6.4. Fast Point Doubling](#)).

#### 37.3.6.2.2 How to Use the Service

#### 37.3.6.2.3 Description

The operation performed is:

$$Pt_C = Pt_A + Pt_B$$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3\*u2ModLength + 12}). This point can be the Infinite Point.
- B the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointBBase,3\*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 5\*u2ModLength + 32})

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the very same place than the input point A. This Point can be the Infinite Point.

The service name for this operation is `ZpEccAddFast`. This service uses Fast mode and Fast Modular Reduction for computations.



**Important:** Before using this service, ensure that the constant Cns has been calculated with the Setup of the Modular Reduction functions.

### 37.3.6.2.4 Parameters Definition

**Table 37-68.** ZpEccAddFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of Modulus P	Base of Modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulo	Length of modulo
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1PointBBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point B (projective coordinates)	Input point B
nu1Workspace	nu1	I	Crypto RAM	5*u2ModLength + 32	-	Corrupted workspace

### 37.3.6.2.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PUPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;

PUKCL _ZpEccAdd(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEccAdd(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEccAdd(u2ModLength) = <Byte length of P>;
PUKCL _ZpEccAdd(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _ZpEccAdd(nu1PointBBase) = <Base of the ram location of the B point>;
PUKCL _ZpEccAdd(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccAddFast,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.6.2.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PointBBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PointBBase, 3\*u2ModLength + 12}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PointBBase, 3\*u2ModLength + 12} and {nu1Workspace, 5\*u2ModLength + 32}

### 37.3.6.2.7 Status Returned Values

**Table 37-69.** ZpEccAddFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.



### 37.3.6.3 Point Addition and Subtraction

#### 37.3.6.3.1 Purpose

This service is used to perform a point addition and point subtraction, based on a given elliptic curve over GF(p). Please note that:

- This service is not intended to add the same point twice. In this particular case, use the doubling service (see 37.3.6.4. Fast Point Doubling).

#### 37.3.6.3.2 How to Use the Service

#### 37.3.6.3.3 Description

The operation performed is:

$$Pt_C = Pt_A \pm Pt_B$$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase, 3\*u2ModLength + 12}). This point can be the Infinite Point.
- B the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointBBase, 3\*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase, u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 5\*u2ModLength + 32})
- The operator filled with the operation to perform (Addition or Subtraction)

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the very same place than the input point A. This Point can be the Infinite Point.

The service name for this operation is `ZpEccAddSubFast`. This service uses Fast mode and Fast Modular Reduction for computations.

**Note:** Before using this service, ensure that the constant Cns has been calculated with the setup of the modular reduction functions.

#### 37.3.6.3.4 Parameters Definition

Table 37-70. ZpEccAddSubFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of Modulus P	Base of Modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulo	Length of modulo
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1PointBBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point B (projective coordinates)	Input point B
u2Operator	u2	I	-	-	Addition or Subtraction	Addition or Subtraction
nu1Workspace	nu1	I	Crypto RAM	5*u2ModLength + 32	-	Corrupted workspace

#### 37.3.6.3.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;

PUKCL _ZpEccAddSub(nu1ModBase) = <Base of the ram location of P>;

```

```

PUKCL _ZpEccAddSub(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEccAddSub(u2ModLength) = <Byte length of P>;
PUKCL _ZpEccAddSub(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _ZpEccAddSub(nu1PointBBase) = <Base of the ram location of the B point>;
PUKCL _ZpEccAddSub(nu1Workspace) = <Base of the ram location of the workspace>;
PUKCL _ZpEccAddSub(u2Operator) = <Operation to perform (PUKCL_ZPECCADD or PUKCL_ZPECCSUB)>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccAddSubFast,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
    {
        ...
    }
else // Manage the error

```

### 37.3.6.3.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PointBBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PointBBase, 3\*u2ModLength + 12}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PointBBase, 3\*u2ModLength + 12} and {nu1Workspace, 5\*u2ModLength + 32}

### 37.3.6.3.7 Status Returned Values

**Table 37-71.** ZpEccAddFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.

## 37.3.6.4 Fast Point Doubling

### 37.3.6.4.1 Purpose

This service is used to perform a Point Doubling, based on a given elliptic curve over GF(p).

### 37.3.6.4.2 How to Use the Service

### 37.3.6.4.3 Description

These two services process the Point Doubling:

$$Pt_C = 2 \times Pt_A$$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3\*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength +8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})
- The workspace not initialized (pointed by {nu1WorkSpace, 4\*u2ModLength +28})
- The a parameter relative to the elliptic curve (pointed by {nu1ABase,u2ModLength +4})
- The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the same location than the input point A. This point can be the Infinite Point.

The service name for this operation is `ZpEccDblFast`. This service uses Fast mode and Fast Modular Reduction for computations.



**Important:** Before using this service, ensure that the constant `Cns` has been calculated with the setup of the Fast Modular Reduction service.

### 37.3.6.4.4 Parameters Definition

Table 37-72. `ZpEccDblFastService`

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
<code>nu1ModBase</code>	<code>nu1</code>	I	Crypto RAM	<code>u2ModLength + 4</code>	Base of modulus P	Base of modulus P
<code>nu1CnsBase</code>	<code>nu1</code>	I	Crypto RAM	<code>u2ModLength + 8</code>	Base of Cns	Base of Cns
<code>u2ModLength</code>	<code>u2</code>	I	-	-	Length of modulus P	Length of modulus P
<code>nu1ABase</code>	<code>u2</code>	I	Crypto RAM	<code>u2ModLength + 4</code>	Parameter a of the elliptic curve	Parameter a of the elliptic curve
<code>nu1PointABase</code>	<code>nu1</code>	I/O	Crypto RAM	<code>3*u2ModLength + 12</code>	Input point A (projective coordinates)	Resulting point C (projective coordinates)
<code>nu1Workspace</code>	<code>nu1</code>	I	Crypto RAM	<code>4*u2ModLength + 28</code>	-	Corrupted workspace

### 37.3.6.4.5 Code Example

```

PUKCL_PARAM PUKCLParam;
P_PUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;

PUKCL _ZpEccDbl(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEccDbl(u2ModLength) = <Byte length of P>;
PUKCL _ZpEccDbl(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEccDbl(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _ZpEccDbl(nu1ABase) = <Base of the a parameter of the elliptic curve>;
PUKCL _ZpEccDbl(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccDblFast, &PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.6.4.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- `nu1ModBase`, `nu1CnsBase`, `nu1PointABase`, `nu1ABase`, `nu1Workspace` are not aligned on 32-bit boundaries
- `{nu1ModBase, u2ModLength + 4}`, `{nu1CnsBase, u2ModLength + 8}`, `{nu1PointABase, 3*u2ModLength + 12}`, `{nu1ABase, u2ModLength + 4}`, `{nu1Workspace, <WorkspaceLength>}` are not in Crypto RAM
- `u2ModLength` is either: `< 12`, `> 0xffc` or not a 32-bit length
- All overlapping between `{nu1ModBase, u2ModLength + 4}`, `{nu1CnsBase, u2ModLength + 8}`, `{nu1PointABase, 3*u2ModLength + 12}`, `{nu1ABase, u2ModLength + 4}` and `{nu1Workspace, 4*u2ModLength + 28}`

### 37.3.6.4.7 Status Returned Values

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.

### 37.3.6.5 Fast Multiplying by a Scalar Number of a Point

#### 37.3.6.5.1 Purpose

This service is used to multiply a point by an integral constant K on a given elliptic curve over GF(p).

#### 37.3.6.5.2 How to Use the Service

#### 37.3.6.5.3 Description

These two services process the Multiplying by a scalar number:

$$Pt_C = K \times Pt_A$$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3\*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength +8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})
- The workspace not initialized (pointed by {nu1WorkSpace, 8\*u2ModLength +44})
- The a parameter relative to the elliptic curve (pointed by {nu1ABase,u2ModLength +4})
- K the scalar number (pointed by {nu1ScalarNumber,u2ScalarLength +4})

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the very same place than the input point A. This point can be the Infinite Point.

The service name for this operation is `ZpEccMulFast`. This service uses Fast mode and Fast Modular Reduction for computations.

**Note:** Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reduction service.

#### 37.3.6.5.4 Parameters Definition

Table 37-73. ZpEccMulFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1KBase	nu1	I	Crypto RAM	u2KLength	Scalar number used to multiply the point A	Unchanged
u2KLength	u2	I	-	-	Length of scalar K	Length of scalar K
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1ABas	nu1	I	Crypto RAM	u2ModLength + 4	Parameter a of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	-	Corrupted workspace

#### 37.3.6.5.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;
```

```

PUKCL_ZpEccMul(nu1ModBase) = <Base of the ram location of P>;
PUKCL_ZpEccMul(u2ModLength) = <Byte length of P>;
PUKCL_ZpEccMul(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL_ZpEccMul(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL_ZpEccMul(nu1ABase) = <Base of the ram location of the parameter A of the elliptic
curve>;
PUKCL_ZpEccMul(nu1KBase) = <Base of the ram location of the scalar number>;
PUKCL_ZpEccMul(nu1Workspace) = <Base of the ram location of the workspace>;
PUKCL_ZpEccMul(u2KLength) = <Byte length of the Scalar Number K>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccMulFast, &PUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.6.5.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1ABase, nu1ScalarNumber, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength+ 12}, {nu1ABase, u2ModLength + 4}, {nu1ScalarNumber, u2ScalarLength} or {nu1Workspace, 8\*u2ModLength + 44} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1ABase, u2ModLength + 4}, {nu1ScalarNumber, u2ScalarLength} and {nu1Workspace, 8\*u2ModLength + 44}

### 37.3.6.5.7 Status Returned Values

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.

### 37.3.6.6 Quick Dual Multiplying by Two Scalar Numbers and Two Points

#### 37.3.6.6.1 Purpose

This service is used to multiply two points by two integral constants K1 and K2, and then provide the addition of these multiplications results.



**Important:** This service has a quick implementation without additional security.

#### 37.3.6.6.2 How to Use the Service

#### 37.3.6.6.3 Description

This service processes the dual Multiplying by two scalar numbers:

$$PtC = K_1 \times Pt_A + K_2 \times Pt_B$$

In this computation, the following parameters need to be provided:

- A the first input point is filled in projective coordinates (X,Y,Z) (pointed by {pu1PointABase, (3\*(u2ModLength + 4)) \* (2(WA-2))}). This point can be the Infinite Point.

- B the 2nd input point is filled in projective coordinates (X,Y,Z) (pointed by {pu1PointBBase, (3\*(u2ModLength + 4)) \* (2(WB-2))}). This point can be the Infinite Point.
- P the modulus filled and Cns the Fast Modular Constant filled (pointed by {pu1ModCnsBase, 2\*u2ModLength + 16})
- The a parameter filled and the workspace not initialized (pointed by {pu1AWorkBase, 9\*u2ModLength + 48})
- KAB the scalar numbers (pointed by {pu1KABBase, 2\*u2KLength + 8})
- The options are set by the u2Options input parameter, which is composed of:
  - wA: Size of window for Point A between 2 and 15
  - wB: Size of window for Point B between 2 and 15
  - PUKCL\_ZPECCMUL\_SCAL\_IN\_CLASSIC\_RAM flag: to set only if the scalars are entirely in Classic RAM with no part in PUKCC RAM

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at (pu1AWorkBase + u2ModLength + 4). This point can be the Infinite Point.



**Important:** Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reduction service.

#### 37.3.6.6.4 Parameters Definition

WA is the Point A window size and WB is the Point B window size (see Options below for details).



**Important:** Please calculate precisely the length of areas with the formulas. Ensure that the pu1 type is a pointer on 4 bytes and contains the full address (see [37.3.3.4. Aligned Significant Length](#)).

**Table 37-74.** ZpEccQuickDualMulFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
pu1ModCnsBase	pu1	I	Crypto RAM	2 * u2ModLength + 16	Base of modulus P, Base of Cns	Base of modulus P, Base of Cns
u2Option	u2	I	-	-	Option related to the called service (see below)	-
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
pu1KABBase	pu1	I	Any RAM	2 * u2KLength + 8	Scalar numbers used to multiply the points A and B	Unchanged
u2KLength	u2	I	-	-	Length of scalars KA and KB	Length of scalars KA and KB
pu1PointABase	pu1	I/O	Crypto RAM	$(3*(u2ModLength + 4)) * (2^{(wA-2)}) (1)$	Input point A (projective coordinates)	Unchanged
pu1PointBBase	pu1	I	Crypto RAM	$(3*(u2ModLength + 4)) * (2^{(wB-2)}) (2)$	Input point B (projective coordinates)	Unchanged
pu1AWorkBase	pu1	I	Crypto RAM	9*u2ModLength + 48	Parameter a of the elliptic curve	Resulting point C (projective coordinates) in pu1AWorkBase Base + u2ModLength + 4

**Notes:**

1. The precalculus table size for the point A is calculated from chosen window size “WA”.
2. The precalculus table size for the point B is calculated from chosen window size “WB”.

**37.3.6.6.5 Options**

The options are set by the u2Options input parameter, which is composed of:

- the mandatory windows sizes WA and WB
- the indication of the presence of the scalars in system RAM

**Note:** Please check precisely if one part of the scalars is in Crypto RAM. If this is the case, the PUKCL\_ZPECCMUL\_SCAL\_IN\_CLASSIC\_RAM option must not be used.

The u2Options number is calculated by an “Inclusive OR” of the options. Some Examples in C language are:

- ```
// Scalars are in system RAM
// The Point A window size is 3
// The Point B window size is 4
PUKCL(u2Options) = PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM |
PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(3) |
PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(4);
```
- ```
// Scalars are in the PUKCC Cryptographic RAM
// The Point A window size is 2
// The Point B window size is 5
PUKCL(u2Options) = PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(2) |
PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(5);
```

For this service, many window sizes are possible. The window sizes in bits are those of the windowing method used for the scalar multiplying.

The choice of the window sizes is a balance between the size of the parameters and the computation time:

- Increasing the window size increases the precomputation table size.
- Increasing the window size to the optimum reduces the computation time.

The following table details the size of the point and the precomputation table, depending on the chosen window size option.

**Table 37-75.** ZpEccQuickDualMulFast Service Window Size Options and Precomputation Table Size

Option Specified	Size of the Point and the Precomputation Table
PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(WA) WA in [2, 15]	$(3*(u2ModLength + 4)) * 2^{(WA-2)}$
PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(WB) WB in [2, 15]	$(3*(u2ModLength + 4)) * 2^{(WB-2)}$

The scalars can be located in PUKCC RAM or in system RAM. If both scalars are entirely in system RAM with no part in PUKCC RAM this can be signaled by using the option PUKCL\_ZPECCMUL\_SCAL\_IN\_CLASSIC\_RAM. In all other cases this option must not be used.

The following table describes this option.

**Table 37-76.** ZpEccQuickDualMulFast Service System RAM Scalar Options

Option	Purpose
PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM	The scalars can be located in Crypto RAM or in system RAM. If both scalars are entirely in system RAM with no part in Crypto RAM this can be signaled by using this option . In all other cases this option must not be used.

### 37.3.6.6.6 Code Example

```

PUKCL_PARAM PUKCLParam;
PUPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL(u2Option) = <Configure scalar numbers location and windows sizes>;
PUKCL_ZpEccQuickDualMulFast(pu1ModCnsBase) = <Base of the ram location of P and Cns>;
PUKCL_ZpEccQuickDualMulFast(u2ModLength) = <Byte length of P>;
PUKCL_ZpEccQuickDualMulFast(u2KLength) = <Byte length of scalars>;
PUKCL_ZpEccQuickDualMulFast(pu1PointABase) = <Base of the ram location of the A point>;
PUKCL_ZpEccQuickDualMulFast(pu1PointBBase) = <Base of the ram location of the B point>;
PUKCL_ZpEccQuickDualMulFast(pu1AWorkBase) = <Base of the ram location of the parameter A of
the elliptic curve and workspace>;
PUKCL_ZpEccQuickDualMulFast(pu1KABBase) = <Base of the ram location of the scalar numbers KA
and KB>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccQuickDualMulFast, pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.6.6.7 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

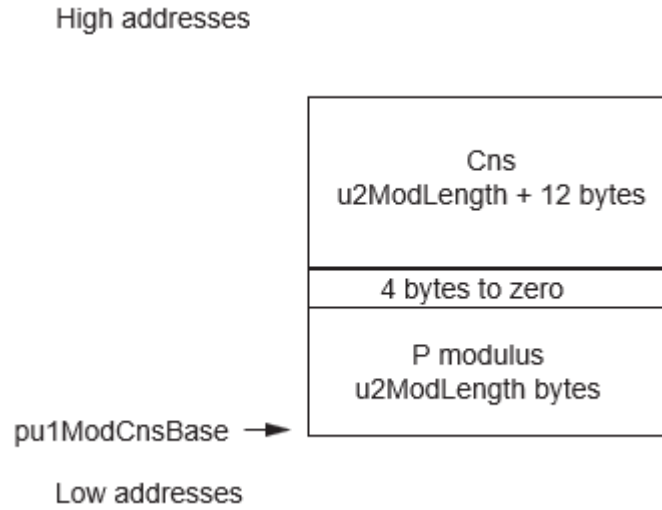
- pu1ModCnsBase, pu1PointABase, pu1PointBBase, pu1AWorkBase, pu1KABBase are not aligned on 32-bit boundaries
- {pu1ModCnsBase, 2\*u2ModLength + 16}, {pu1PointABase, (3\*(u2ModLength + 4)) \* (2<sup>(WA-2)</sup>)}, {pu1PointBBase, (3\*(u2ModLength + 4)) \* (2<sup>(WB-2)</sup>)} or { pu1AWorkBase, 9\*u2ModLength + 48} are not in PUKCC RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {pu1ModCnsBase, 2\*u2ModLength + 16}, {pu1PointABase, (3\*(u2ModLength + 4)) \* (2<sup>(WA-2)</sup>)}, {pu1PointBBase, (3\*(u2ModLength + 4)) \* (2<sup>(WB-2)</sup>)} or {pu1AWorkBase, 9\*u2ModLength + 48}.

### 37.3.6.6.8 Parameters Placement

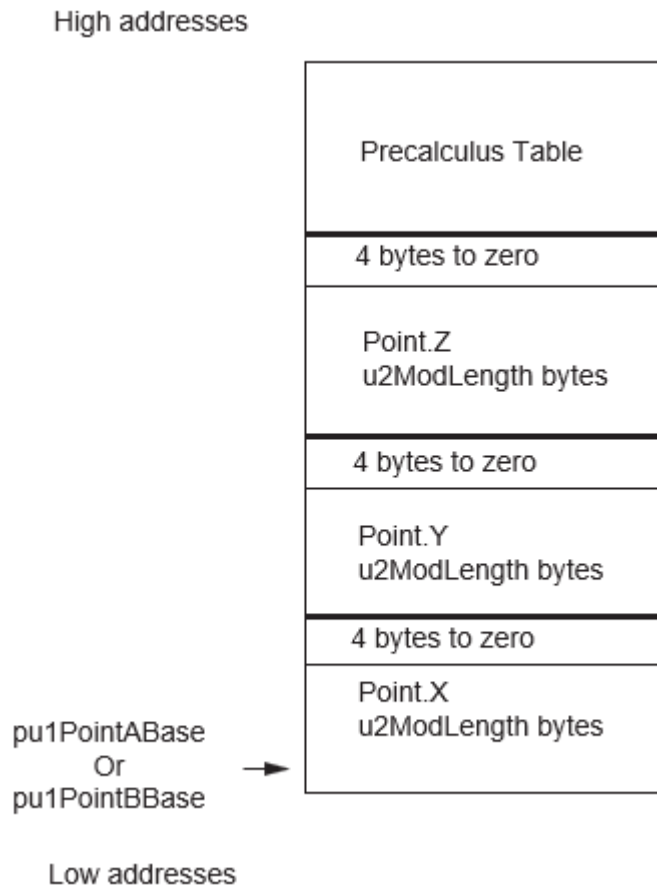
The parameters' placement is described in the following figures.



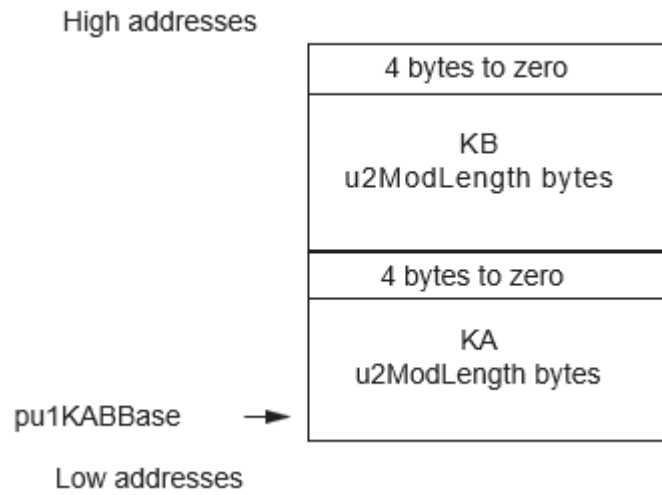
**Figure 37-7.** Modulus P and Cns{pu1ModCnsBase, 2\*u2ModLength + 16}



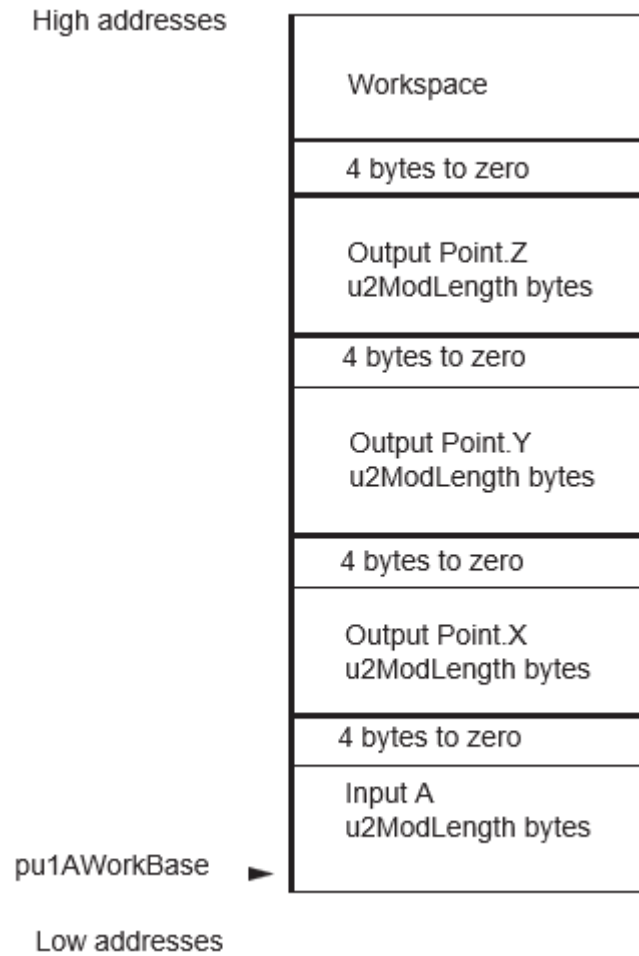
**Figure 37-8.** Points A and B {pu1PointABase, [(3\*(u2ModLength + 4)) \* (2<sup>(WA-2)</sup>)] Or [(3\*(u2ModLength + 4)) \* (2<sup>(WB-2)</sup>)]}



**Figure 37-9.** Scalars KA and KB {pu1KABBase, 2 \* u2KLength + 8}



**Figure 37-10.** The a parameter and Workspace {pu1AWorkBase, 9\*u2ModLength + 48}



### 37.3.6.6.9 Status Returned Values

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.

### 37.3.6.7 Projective to Affine Coordinates Conversion

#### 37.3.6.7.1 Purpose

This service is used to perform a point coordinates conversion from projective representation to affine.

#### 37.3.6.7.2 How to Use the Service

#### 37.3.6.7.3 Description

The operation performed is:

$$Pt_X \text{ Affine coordinate} = \left[ \frac{Pt_X \text{ Projective coordinate}}{(Pt_Z \text{ Projective coordinate})^2} \right]$$

$$Pt_Y \text{ Affine coordinate} = \left[ \frac{Pt_Y \text{ Projective coordinate}}{(Pt_Z \text{ Projective coordinate})^3} \right]$$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) or affine coordinates for X and Y, and setting Z to 1 (pointed by {nu1PointABase, 3\*u2ModLength + 12}). The Point A can be the point at infinity. In this case, the u2Status returned is PUKCL\_POINT\_AT\_INFINITY.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase, u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 4\*u2ModLength + 48})

The result is the point A with its (X,Y) coordinates converted to affine, and the Z coordinate set to 1. The service for this operation is ZpEcConvProjToAffine.



**Important:** Before using this service, ensure that the constant Cns has been calculated with the Setup of the fast Modular Reductions service.

#### 37.3.6.7.4 Parameters Definition

**Table 37-77.** ZpEcConvAffineToProjective Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1PointABase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point A	Resulting point A in affine coordinates
nu1Workspace	nu1	I	Crypto RAM	4*u2ModLength + 48	-	Workspace

#### 37.3.6.7.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;
```

```

PUKCL _ZpEcConvProjToAffine(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEcConvProjToAffine(u2ModLength) = <Byte length of P>;
PUKCL _ZpEcConvProjToAffine(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEcConvProjToAffine(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _ZpEcConvProjToAffine(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library..
vPUKCL_Process(ZpEcConvProjToAffine,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
    {
        ...
    }
else // Manage the error

```

### 37.3.6.7.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength+ 12}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12} and {nu1Workspace, 4\*u2ModLength + 48}

### 37.3.6.7.7 Status Returned Values

**Table 37-78.** ZpEccConvAffineToProjective Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.
PUKCL_POINT_AT_INFINITY	Warning	The input point has its Z equal to zero, so it's a representation of the infinite point.

### 37.3.6.8 Affine to Projective Coordinates Conversion

#### 37.3.6.8.1 Purpose

This service is used to perform a point coordinates conversion from an affine point representation to projective.

#### 37.3.6.8.2 How to Use the Service

#### 37.3.6.8.3 Description

The operation performed is:

$$\text{affine}(X_a, Y_a) \rightarrow \text{projective}(X_p, Y_p, Z_p)$$

In this computation, the following parameters need to be provided:

- A the input point is filled in affine coordinates for X and Y, and setting Z to 1 (pointed by {nu1PointABase, 3\*u2ModLength + 4}).
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase, u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 2\*u2ModLength + 16})

The result is the point A with its (X,Y,Z) projective coordinates.

The service for this operation is ZpEcConvAffineToProjective



**Important:** Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reductions service.

### 37.3.6.8.4 Parameters Definition

**Table 37-79.** ZpEccConvAffineToProjective Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1PointABase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point A	Resulting point A in affine coordinates
nu1Workspace	nu1	I	Crypto RAM	2*u2ModLength + 16	-	Workspace

### 37.3.6.8.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PUPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;

PUKCL _ZpEcConvAffineToProjective(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEcConvAffineToProjective(u2ModLength) = <Byte length of P>;
PUKCL _ZpEcConvAffineToProjective(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEcConvAffineToProjective(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _ZpEcConvAffineToProjective(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEcConvAffineToProjective, &PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.6.8.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength+ 12}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, and {nu1Workspace, 2\*u2ModLength + 16}

### 37.3.6.8.7 Status Returned Values

**Table 37-80.** ZpEccConvAffineToProjective Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.

### 37.3.6.9 Randomize a Coordinate

#### 37.3.6.9.1 Purpose

This service is used to convert the projective representation of a point to another projective representation.

#### 37.3.6.9.2 How to Use the Service

#### 37.3.6.9.3 Description

The operation performed is:

$$Projective(X_1, Y_1, Z_1) \rightarrow Projective(X_2, Y_2, Z_2)$$

In this computation, the following parameters need to be provided:

- The input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointBase, 3\*u2ModLength + 12}). This Point must not be the point at infinity.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase, u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 3\*u2ModLength + 28})
- The random number (pointed by {nu1RandomBase, u2ModLength + 4}).

The result is the point nu1PointBase with its (X,Y,Z) coordinates randomized.

The service for this operation is ZpEccRandomiseCoordinate.



**Important:** Before using this service:

- Ensure that the constant Cns has been calculated with the setup of the Modular Reduction service.
- Be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG

#### 37.3.6.9.4 Parameters Definition

**Table 37-81.** ZpEccRandomiseCoordinate Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1PointBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point	Resulting point
nu1RandomBase	nu1	I	Crypto RAM	u2ModLength + 4	Random	Corrupted
nu1Workspace	nu1	I	Crypto RAM	3*u2ModLength + 28	-	Workspace

#### 37.3.6.9.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL _ZpEccRandomiseCoordinate(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEccRandomiseCoordinate(u2ModLength) = <Byte length of P>;

```

```

PUKCL_ZpEccRandomiseCoordinate(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL_ZpEccRandomiseCoordinate(nu1RandomBase) = <Base of the ram location where the the RNG
is stored>;
PUKCL_ZpEccRandomiseCoordinate(nu1PointBase) = <Base of the ram location of the point>;
PUKCL_ZpEccRandomiseCoordinate(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccRandomiseCoordinate,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.6.9.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1RandomBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1RandomBase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1RandomBase, u2ModLength + 4} and {nu1Workspace, 3\*u2ModLength + 28}

### 37.3.6.9.7 Status Returned Values

**Table 37-82.** ZpEccRandomiseCoordinate Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.

### 37.3.6.10 Point is on Elliptic Curve

#### 37.3.6.10.1 Purpose

This service is used to test whether or not the point is on the curve.

#### 37.3.6.10.2 How to Use the Service

#### 37.3.6.10.3 Description

The operation performed is:

*Status = IsPointOnCurve(X, Y, Z)*

In this computation, the following parameters need to be provided:

- The input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointBase,3\*u2ModLength + 4}). This Point can be the point at infinity.
- AParam and BParam are the Elliptic Curve Equation parameters. (pointed by{nu1AParam, u2ModLength+4} and {nu1BParam, u2ModLength+4}).
- Cns the Fast Modular Constant filled (pointed by{nu1CnsBase,u2ModLength+8}).
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4}).
- The workspace not initialized (pointed by {nu1WorkSpace, 4\*u2ModLength +28}).

The result is the status of the point (X,Y,Z) regarding the Elliptic Curve Equation.

The service name for this operation is ZpEcPointIsOnCurve.

**Note:** Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reduction service.

### 37.3.6.10.4 Parameters Definition

**Table 37-83.** ZpEcPointIsOnCurve Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1PointBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point	unchanged
nu1AParam	nu1	I	Crypto RAM	u2ModLength + 4	The parameter a	The parameter a
nu1BParam	nu1	I	Crypto RAM	u2ModLength + 4	The parameter b	The parameter b
nu1Workspace	nu1	I	Crypto RAM	4*u2ModLength + 28	-	Workspace

### 37.3.6.10.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PUPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;

PUKCL _ZpEcPointIsOnCurve(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEcPointIsOnCurve(u2ModLength) = <Byte length of P>;
PUKCL _ZpEcPointIsOnCurve(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEcPointIsOnCurve(nu1AParam) = <Base of the ram location of the parameter a>;
PUKCL _ZpEcPointIsOnCurve(nu1BParam) = <Base of the ram location of the parameter b>;
PUKCL _ZpEcPointIsOnCurve(nu1PointBase) = <Base of the ram location of the point>;
PUKCL _ZpEcPointIsOnCurve(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEcPointIsOnCurve, &PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.6.10.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1AParam, nu1BParam, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength+4}, {nu1CnsBase, u2ModLength+8}, {nu1PointABase, 3\*u2ModLength +12}, {nu1AParam, u2ModLength + 4}, {nu1BParam, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM.
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length.
- All overlapping between {nu1ModBase, u2ModLength+4}, {nu1CnsBase, u2ModLength+8}, {nu1PointABase, 3\*u2ModLength+12}, {nu1AParam, u2ModLength+4}, {nu1AParam, u2ModLength + 4} and {nu1Workspace, 4\*u2ModLength+28}.

### 37.3.6.10.7 Status Returned Values

**Table 37-84.** ZpEcPointIsOnCurve Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The point is on the curve.
PUKCL_POINT_IS_NOT_ON_CURVE	Warning	The point is not on the curve.



.....continued

Returned Status	Importance	Meaning
PUKCL_POINT_AT_INFINITY	Warning	The input point has its Z equal to zero, so it's a representation of the infinite point.

### 37.3.6.11 Generating an ECDSA Signature (Compliant with FIPS 186-2)

#### 37.3.6.11.1 Purpose

This service is used to generate an ECDSA signature following the FIPS 186-2. It performs the second step of the Signature Generation. A hash value (*HashVal*) must be provided as input, it has to be previously computed from the message to be signed using a secure hash algorithm.

A scalar number must be provided too as described in the FIPS 186-2. The result (R,S) is computed by this service.

#### 37.3.6.11.2 How to Use the Service

#### 37.3.6.11.3 Description

The operation performed is:

$$(R, S) = EcDsaSign(Pt_A, HashVal, k, CurveParameters, PrivateKey)$$

This service processes the following checks:

- If the Scalar Number *k* is out of the range [1, PointOrder -1], the calculus is stopped and the status is set to PUKCL\_WRONG\_SELECT\_NUMBER.
- If *R* equals zero, the calculus is stopped and the status is set to PUKCL\_WRONG\_SELECT\_NUMBER.
- If *S* equals zero, the calculus is stopped and the status is set to PUKCL\_WRONG\_SELECT\_NUMBER.

In this computation, the following parameters need to be provided:

- A the input point is filled in "mixed" coordinates (X,Y) with the affine values and  $Z = 1$  (pointed by {nu1PointABase, 3\*u2ModLength + 12})
- Cns the working space for the Fast Modular Constant not initialized (pointed by {nu1CnsBase, u2ScalarLength + 8})
- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 8\*u2ModLength + 44})
- The *a* parameter relative to the elliptic curve (pointed by {nu1ABase, u2ModLength + 4})
- The order of the Point A on the elliptic curve (pointed by {nu1OrderPointBase, u2ScalarLength + 4})
- *k* the input Scalar Number beforehand generated and filled (pointed by {nu1ScalarNumber, u2ScalarLength + 4})
- HashVal the hash value beforehand generated and filled (pointed by {nu1HashBase, u2ScalarLength + 4})
- The Private Key (pointed by {nu1PrivateKey, u2ScalarLength + 4})
- Generally, u2ScalarLength is equal to (u2ModLength) or (u2ModLength + 4)



**Important:**

For the ECDSA signature generation be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

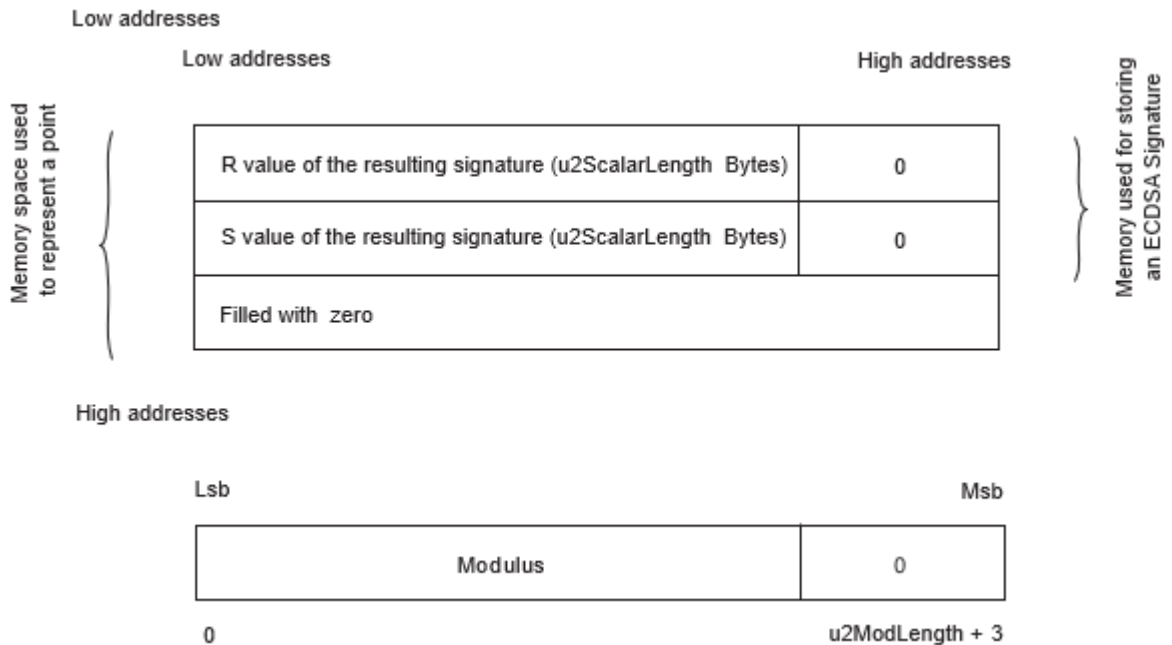
The scalar number k must be selected at random. This random must be generated before the call of the ECDSA signature. For this random generation be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

The operation performed is:

- Compute the ECDSA (R,S) as described in FIPS 186-2, but leaving the user the role of computing the input Hash Value, thus leaving the freedom of using any other algorithm than SHA-1.
- Compute a R value using the input A point and the scalar number.
- Compute a S value using R, the scalar number, the private key and the provided hash value. Note that the resulting signature (R,S) is stored at the place of the input A point.
- If all is correct and S is different from zero, the status is set to PUKCL\_OK. If all is correct and S equals zero, the status is set to PUKCL\_WRONG\_SELECT\_NUMBER. If an error occurs, the status is set to the corresponding error value (see Status Returned Values below).

The service name for this operation is `ZpEcDsaGenerateFast`. This service uses Fast mode and Fast Modular Reduction for computation.

- The signature (R,S), when resulting from a computation is given back at address of the A point:
  - R output is at offset 0 and has length (u2ScalarLength + 4) bytes.
  - S output is at offset (u2ScalarLength + 4) bytes and has length (u2ScalarLength + 4) bytes.
  - The MSB 4 zero bytes may be suppressed to get the R and S values on u2ScalarLength bytes



### 37.3.6.11.4 Parameters Definition

**Table 37-85.** ZpEcDsaGenerateFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ScalarLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1ScalarNumber	nu1	I	Crypto RAM	u2ScalarLength + 4	Scalar Number used to multiply the point A	Unchanged
nu1OrderPointBase	nu1	I	Crypto RAM	u2ScalarLength + 4	Order of the Point A in the elliptic curve	Unchanged
nu1PrivateKey	nu1	I/O	Crypto RAM	u2ScalarLength + 4	Base of the Private Key	Unchanged
nu1HashBase <sup>(1)</sup>	nu1	I	Crypto RAM	u2ScalarLength + 4	Base of the hash value resulting from the previous SHA	Unchanged
u2ScalarLength	u2	I	-	-	Length of scalar ( <u>same</u> length as the length of order)	Length of scalar
nu1PointABase <sup>(2)</sup>	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (three coordinates (X,Y) affine and Z = 1)	Resulting signature (R,S,0)
nu1ABase	nu1	I	Crypto RAM	u2ModLength + 4	Parameter a of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	-	Corrupted workspace

**Notes:**

1. The hash value calculus is defined by the ECDSA norm and depends on the elliptic curve domain parameters. To construct the input parameter, the 4 Most Significant Bytes must be set to zero.
2. The resulting signature format is different from the point A format (see Description above for information on the point A format).

### 37.3.6.11.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL _ZpEcDsaGenerate(nulModBase) = <Base of the ram location of P>; PUKCL
_ZpEcDsaGenerate(u2ModLength) = <Byte length of P>;
PUKCL _ZpEcDsaGenerate(nulCnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEcDsaGenerate(nulPointABase) = <Base of the A point>;
PUKCL _ZpEcDsaGenerate(nulPrivateKey) = <Base of the Private Key>;
PUKCL _ZpEcDsaGenerate(nulScalarNumber) = <Base of the ScalarNumber>;
PUKCL _ZpEcDsaGenerate(nulOrderPointBase) = <Base of the order of A point>;
PUKCL _ZpEcDsaGenerate(nulABase) = <Base of the a parameter of the curve>;
PUKCL _ZpEcDsaGenerate(nulWorkspace) = <Base of the workspace>;
PUKCL _ZpEcDsaGenerate(nulHashBase) = <Base of the SHA resulting hash>;
PUKCL _ZpEcDsaGenerate(u2ScalarLength) = < Length of ScalarNumber>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEcDsaGenerateFast, pvPUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
...

```

```
    }  
else // Manage the error
```

### 37.3.6.11.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PrivateKey, nu1ScalarNumber, nu1OrderPointBase, nu1ABase, nu1Workspace or nu1HashBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PrivateKey, u2ScalarLength + 4}, {nu1ScalarNumber, u2ScalarLength + 4}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} or {nu1HashBase, u2ScalarLength + 4} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PrivateKey, u2ScalarLength + 4}, {nu1ScalarNumber, u2ScalarLength + 4}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} and {nu1HashBase, u2ScalarLength + 4}

### 37.3.6.11.7 Status Returned Values

**Table 37-86.** ZpEcDsaGenerateFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem. The signature is the good one.
PUKCL_WRONG_SELECTNUMBER	Warning	The given value for nu1ScalarNumber is not good to perform this signature generation.

### 37.3.6.12 Verifying an ECDSA Signature (Compliant with FIPS186-2)

#### 37.3.6.12.1 Purpose

This service is used to verify an ECDSA signature following the FIPS 186-2. It performs the second step of the Signature Verification.

A hash value (HashVal) must be provided as input, it has to be previously computed from the message to be signed using a secure hash algorithm.

As second significant input, the Signature is provided to be checked. This service checks the signature and fills the status accordingly.

#### 37.3.6.12.2 How to Use the Service

#### 37.3.6.12.3 Description

The operation performed is:

*Verify = EcDsaVerifySignature(Pt<sub>A</sub>, HashVal, Signature, CurveParameters, PublicKey)*

The points used for this operation are represented in different coordinate systems. In this computation, the following parameters need to be provided:

- A the input point is filled with the affine values (X,Y) and Z = 1 (pointed by {nu1PointABase, 3\*u2ModLength + 12})
- Cns the working space for the Fast Modular Constant not initialized (pointed by {nu1CnsBase, u2ScalarLength + 8})
- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 8\*u2ModLength + 44})
- The a parameter relative to the elliptic curve (pointed by {nu1ABase, u2ModLength + 4})

- The order of the Point A on the elliptic curve (pointed by {nu1OrderPointBase,u2ScalarLength + 4})
- HashVal the hash value is generated prior and filled (pointed by {nu1HashBase,u2ScalarLength + 4})
- The Public Key point is filled in “mixed” coordinates (X,Y) with the affine values and Z = 1 (pointed by {nu1PointPublicKeyGen, 3\*u2ModLength + 12})
- The input signature (R,S), even if it is not a Point, is represented in memory like a point in affine coordinates (X,Y) (pointed by {nu1PointSignature, 2\*u2ScalarLength + 8})  
**Note:** For the ECDSA signature verification be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.
- The operation consists in obtaining a V value with all these input parameters and checking that V equals the provided R. If all is correct and the signature is the good one, the status is set to PUKCL\_OK. If all is correct and the signature is wrong, the status is set to PUKCL\_WRONG\_SIGNATURE. If an error occurs, the status is set to the corresponding error value (see Status Returned Values below).

### 37.3.6.12.4 Parameters Definition

**Table 37-87.** ZpEcDsaVerifyFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ScalarLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1OrderPointBase	nu1	I	Crypto RAM	u2ScalarLength + 4	Order of the Point A in the elliptic curve	Unchanged
nu1PointSignature	nu1	I	Crypto RAM	2*u2ScalarLength + 8	Signature(r, s)	Corrupted
nu1HashBase <sup>(1)</sup>	nu1	I	Crypto RAM	u2ScalarLength + 4	Base of the hash value resulting from the previous SHA	Corrupted
u2ScalarLength	u2	I	-	-	Length of scalar	Length of scalar
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Generator point	Corrupted
nu1PointPublicKeyGen	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Public point	Corrupted
nu1ABase	nu1	I	Crypto RAM	u2ModLength + 4	Parameter a of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	-	Corrupted workspace

**Note:**

1. The hash value calculus is defined by the ECDSA norm and depends on the elliptic curve domain parameters. To construct the input parameter, the 4 Most Significant Bytes must be set to zero.

### 37.3.6.12.5 Code Example

```

PUKCL_PARAM PUKCLParam;
P_PUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL(u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL_ZpEcDsaVerify(nulModBase) = <Base of the ram location of P>;
PUKCL_ZpEcDsaVerify(u2ModLength) = <Byte length of P>;
PUKCL_ZpEcDsaVerify(nulCnsBase) = <Base of the ram location of Cns>;
PUKCL_ZpEcDsaVerify(nulPointABase) = <Base of the A point>;

```

```

PUKCL_ZpEcDsaVerify(nulPrivateKey) = <Base of the Private Key>;
PUKCL_ZpEcDsaVerify(nulScalarNumber) = <Base of the ScalarNumber>;
PUKCL_ZpEcDsaVerify(nulOrderPointBase) = <Base of the order of A point>;
PUKCL_ZpEcDsaVerify(nulABase) = <Base of the a parameter of the curve>;
PUKCL_ZpEcDsaVerify(nulWorkspace) = <Base of the workspace>;
PUKCL_ZpEcDsaVerify(nulHashBase) = <Base of the SHA resulting hash>;
PUKCL_ZpEcDsaVerify(u2ScalarLength) = < Length of ScalarNumber>;

...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEcDsaVerifyFast, pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
    {
        ...
    }ou
else
    if (PUKCL(u2Status) == PUKCL_WRONG_SIGNATURE)
        {
            ...
        }
    else // Manage the error

```

### 37.3.6.12.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PointPublicKeyGen, nu1PointSignature, nu1OrderPointBase, nu1ABase, nu1Workspace or nu1HashBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PointPublicKeyGen, 3\*u2ModLength + 12}, {nu1PointSignature, 2\*u2ScalarLength + 8}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} or {nu1HashBase, u2ScalarLength + 4} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PointPublicKeyGen, 3\*u2ModLength + 12}, {nu1PointSignature, 2\*u2ScalarLength + 8}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} and {nu1HashBase, u2ScalarLength + 4}

### 37.3.6.12.7 Status Returned Values

**Table 37-88.** ZpEcDsaVerifyFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem. The signature is the good one.
PUKCL_WRONG_SIGNATURE	Warning	The signature is wrong.

### 37.3.6.13 Quick Verifying an ECDSA Signature (Compliant with FIPS 186-2)

#### 37.3.6.13.1 Purpose

This service is used to verify an ECDSA signature following the FIPS 186-2. It performs the second step of the Signature Verification using Quick Dual Multiplying to perform computation.

A hash value (HashVal) must be provided as input, it has to be previously computed from the message whose signature is verified using a secure hash algorithm.

As second significant input, the Signature is provided to be checked.

This service checks the signature and fills the status accordingly.



**Important:** This service has a quick implementation without additional security.

### 37.3.6.13.2 How to Use the Service

#### 37.3.6.13.3 Description

The operation performed is:

$Verify = EcDsaVerifySignature(Pt_A, HashVal, Signature, CurveParameters, PublicKey)$

The points used for this operation are represented in different coordinate systems.

In this computation, the following parameters need to be provided (such that  $u2MaxLength = \max(u2ModLength, u2ScalarLength)$ ):

- A the input point is filled with the affine values (X,Y) and  $Z = 1$  (pointed by  $\{pu1PointABase, (3*(u2ModLength + 4)) * (2^{(WA-2)})\}$ )
- P the modulus filled and Cns the working space for the Fast Modular Constant not initialized (pointed by  $\{pu1ModBase, u2ModLength + u2MaxLength + 16\}$ )
- The a parameter relative to the elliptic curve filled and workspace not initialized (pointed by  $\{pu1AWorkBase, 8*u2MaxLength + u2ModLength + 48\}$ )
- The order of the Point A on the elliptic curve (pointed by  $\{pu1OrderPointBase, u2ScalarLength + 4\}$ )
- HashVal the hash value beforehand generated and filled (pointed by  $\{pu1HashBase, u2MaxLength + 4\}$ )
- The Public Key point is filled in “mixed” coordinates (X,Y) with the affine values and  $Z = 1$  (pointed by  $\{nu1PointPublicKeyGen, (3*(u2ModLength + 4)) * (2^{(WB-2)})\}$ )
- The input signature (R,S), even if it is not a Point, is represented in memory like a point in affine coordinates (X,Y) (pointed by  $\{nu1PointSignature, 2*u2ScalarLength + 8\}$ )

The operation consists of obtaining a V value with all input parameters and checks that V equals the provided R. If all is correct and the signature is the good one, the status is set to PUKCL\_OK. If all is correct and the signature is wrong, the status is set to PUKCL\_WRONG\_SIGNATURE. If an error occurs, the status is set to the corresponding error value (see Status Returned Values below).

#### 37.3.6.13.4 Parameters Definition

To place the parameters correctly the maximum of  $u2ModLength$  and  $u2ScalarLength$  must be calculated:  $u2MaxLength = \max(u2ModLength, u2ScalarLength)$

WA is the Point A window size and WB is the Point Public Key window size (see Options below for details).



**Important:** Please calculate precisely the length of areas with the formulas and the  $\max()$  service which takes the maximum of two values. Ensure that the pu1 type is a pointer on 4 bytes and contains the full address (see 37.3.3.4. [Aligned Significant Length](#) for details).

**Table 37-89.** ZpEcDsaQuickVerify Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
pu1ModCnsBase	pu1	I	Crypto RAM	$u2ModLength + 4 + u2MaxLength + 12$	Base of modulus P	Base of modulus P



.....continued

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Option	u2	I	-	-	Option related to the called service (see below)	-
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
pu1OrderPointBase	pu1	I	Crypto RAM	u2ScalarLength + 4	Order of the Point A in the elliptic curve	Unchanged
pu1PointSignature	pu1	I	Any RAM	2*u2ScalarLength + 8	Signature(r, s)	Corrupted
pu1HashBase (see <b>Note 1</b> )	pu1	I	Crypto RAM	u2MaxLength + 4	Base of the hash value resulting from the previous SHA	Corrupted
u2ScalarLength	u2	I	-	-	Length of scalar	Length of scalar
pu1PointABase	pu1	I/O	Crypto RAM	$(3*u2ModLength + 12) * (2(WA-2))$	Generator point	Corrupted
pu1PointPublicKeyGen	pu1	I/O	Crypto RAM	$(3*u2ModLength + 12) * (2(WB-2))$	Public Key point	Corrupted
pu1AWorkBase	pu1	I	Crypto RAM	$(u2ModLength + 4) + (8*u2MaxLength + 44)$	Parameter a of the elliptic curve and Workspace	Corrupted

**Note:**

1. The hash value calculus is defined by the ECDSA norm and depends on the elliptic curve domain parameters. To construct the input parameter, the 4 Most Significant Bytes must be set to zero.

A suggested parameters placement in Crypto RAM is:

- ModCnsBase
- OrderPointBase
- Signature may be placed here or in Classical RAM
- HashBase
- PointABase
- PointPublicKeyGen
- AWorkBase

**37.3.6.13.5 Options**

The options are set by the u2Options input parameter, which is composed of:

- the mandatory windows sizes WA (window for Point A) and WB (window for Point Public Key)
- the indication of the presence of the Point Signature in system RAM



**Important:** Please check precisely if the Point Signature is in Crypto RAM. If this is the case the PUKCL\_ZPECCMUL\_SCAL\_IN\_CLASSIC\_RAM must not be used.

The u2Options number is calculated by an “Inclusive OR” of the options. Some Examples in C language are:

- `// Point Signature in system RAM`
- `// The Point A window size is 3`
- `// The Point Public Key window size is 4`



```

PUKCL(u2Options) = PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM |
PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(3) |
PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(4);
• // Point Signature in the Cryptographic RAM
// The Point A window size is 2
// The Point Public Key window size is 5
PUKCL(u2Options) = PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(2) |
PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(5);

```

For this service, many window sizes are possible. The window sizes in bits are those of the windowing method used for the scalar multiplying.

The choice of the window sizes is a balance between the size of the parameters and the computation time:

- Increasing the window size increases the precomputation table size.
- Increasing the window size to the optimum reduces the computation time.

The following table details the estimated windows WA and WB optimum and possible for some curves.

**Table 37-90.** ZpEcDsaQuickVerify Service Estimated WA and WB Window Size

Curve Size (bits)	Optimum Window size	Possible Window Sizes (WA, WB) or (WB, WA)
192	5	5, 5
256	5	5, 5
384	6	5, 5
521	6	4, 5

The following table details the size of the point and the precomputation table, depending on the chosen window size option.

**Table 37-91.** ZpEcDsaQuickVerify Service Window Size and Precomputation Table Size Options

Option Specified	Point and Precomputation Table Size
PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(WA) WA in [2, 15]	$(3*(u2ModLength + 4)) * (2^{(WA-2)})$
PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(WB) WB in [2, 15]	$(3*(u2ModLength + 4)) * (2^{(WB-2)})$

The Point Signature can be located in PUKCC RAM or in system RAM. If the Point Signature is entirely in system RAM with no part in PUKCC RAM this can be signaled by using the option PUKCL\_ZPECCMUL\_SCAL\_IN\_CLASSIC\_RAM. In all other cases this option must not be used.

The following table describes this option.

**Table 37-92.** ZpEcDsaQuickVerify Service Point Signature in Classical RAM Option

Option	Purpose
PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM	The Point Signature can be located in Crypto RAM or in system RAM. If the Point Signature is entirely in system RAM with no part in PUKCC RAM this can be signaled by using this option. In all other cases this option must not be used.

### 37.3.6.13.6 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;
PUKCL(u2Option) = <Point Signature location and windows sizes>;
PUKCL_ZpEcDsaQuickVerify(pulModCnsBase) = <Base of the ram location of P and Cns>;
PUKCL_ZpEcDsaQuickVerify(u2ModLength) = <Byte length of P>;
PUKCL_ZpEcDsaQuickVerify(pulPointABase) = <Base of the ram location of the A point>;
PUKCL_ZpEcDsaQuickVerify(pulPointPublicKeyGen) = <Base of the Public Key>;
PUKCL_ZpEcDsaQuickVerify(pulPointSignature) = <Base of the Signature (r, s)>;

```

```

PUKCL_ZpEcDsaQuickVerify(pu1OrderPointBase) = <Base of the order of the A point>;
PUKCL_ZpEcDsaQuickVerify(pu1AWorkBase) = <Base of the ram location of the parameter A of the
elliptic curve and workspace>;
PUKCL_ZpEcDsaQuickVerify(pu1HashBase) = <Base of the SHA resulting hash>;
PUKCL_ZpEcDsaQuickVerify(u2ScalarLength) = <Byte length of R and S in Point Signature>;
. . .
// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEcDsaQuickVerify, pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
    {
        . . .
    }
else
    if ( PUKCL(u2Status) = PUKCL_WRONG_SIGNATURE )
        {
            . . .
        }
    else // Manage the error

```

### 37.3.6.13.7 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- pu1ModCnsBase, pu1PointABase, pu1PointPublicKeyGen, pu1PointSignature, pu1OrderPointBase, pu1AWorkBase or pu1HashBase are not aligned on 32-bit boundaries
- {pu1ModCnsBase, u2ModLength + 4 + u2MaxLength + 12}, {pu1PointABase, (3 \* u2ModLength + 12) \* (2<sup>(WA-2)</sup>)}, {pu1PointPublicKeyGen, (3 \* u2ModLength + 12) \* (2<sup>(WPub-2)</sup>)}, {pu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {pu1AWorkBase, (u2ModLength + 4) + (8 \* u2MaxLength + 44)} or {nu1HashBase, u2ScalarLength + 4} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {pu1ModCnsBase, u2ModLength + 4 + u2MaxLength + 12}, {pu1PointABase, (3 \* u2ModLength + 12) \* (2<sup>(WA-2)</sup>)}, {pu1PointPublicKeyGen, (3 \* u2ModLength + 12) \* (2<sup>(WPub-2)</sup>)}, {pu1OrderPointBase, u2ScalarLength + 4}, {pu1PointSignature, 2 \* u2ScalarLength + 8}, {nu1ABase, u2ModLength + 4}, {pu1AWorkBase, (u2ModLength + 4) + (8 \* u2MaxLength + 44)} and {nu1HashBase, u2ScalarLength + 4}

### 37.3.6.13.8 Status Returned Values

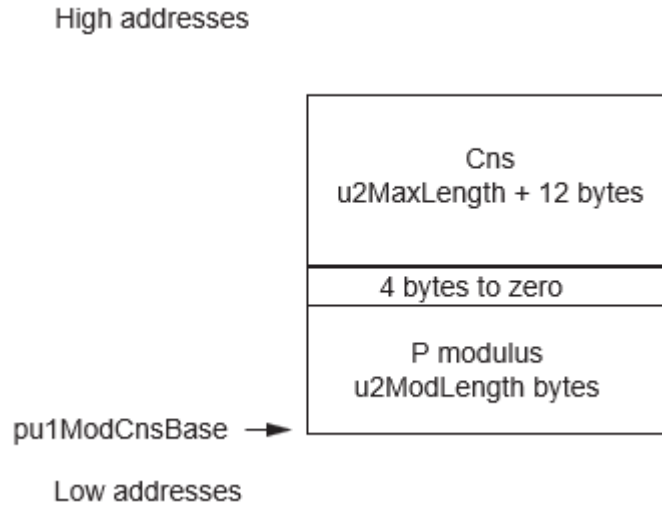
**Table 37-93.** ZpEcDsaQuickVerify Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem. The signature is the good one.
PUKCL_WRONG_SIGNATURE	Warning	The signature is wrong.

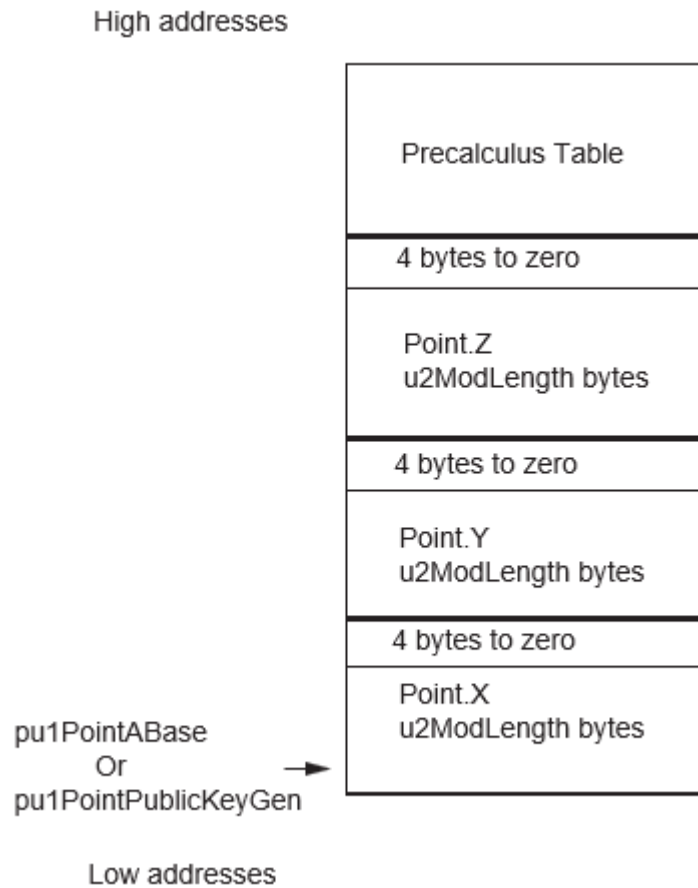
### 37.3.6.13.9 Parameter Placement

The parameters' placement is described in detail in the following figures.

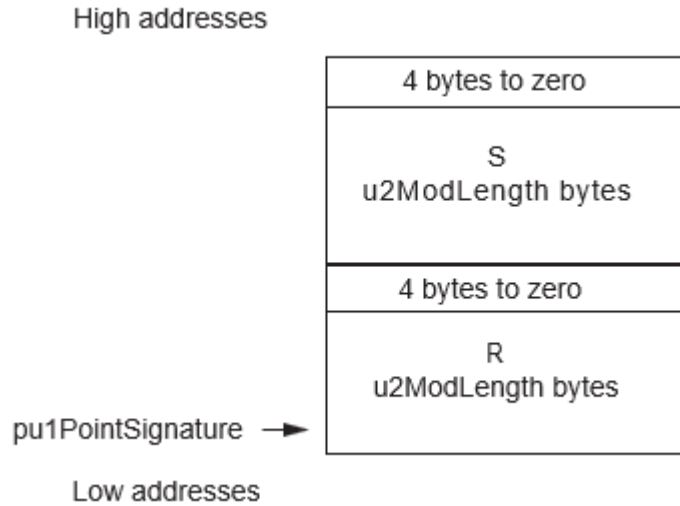
**Figure 37-11.** Modulus P and Cns{pu1ModCnsBase, u2ModLength + 4 + u2MaxLength + 12}



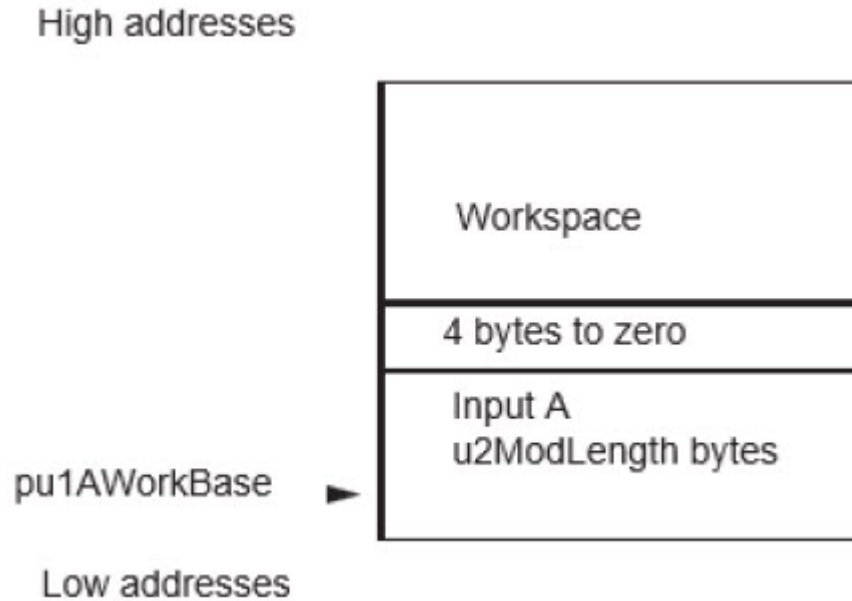
**Figure 37-12.** Points A {pu1PointABase, (3\*(u2ModLength + 4)) \* (2<sup>(WA-2)</sup>)} and Public Key Gen {pu1PointPublicKeyGen, (3\*(u2ModLength + 4)) \* (2<sup>(WB-2)</sup>)}



**Figure 37-13.** PointSignature {pu1PointSignature, 2 \* u2ScalarLength + 8}



**Figure 37-14.** The a parameter and Workspace {pu1AWorkBase, 9\*u2ModLength + 48}



### 37.3.7 Elliptic Curves Over GF(2<sup>n</sup>) Services

This section provides a complete description of the currently available elliptic curve over Polynomials in GF(2<sup>n</sup>) services.

These services process Polynomials in GF(2<sup>n</sup>) only.

The offered services cover the basic operations over elliptic curves such as:

- Adding two points over a curve
- Doubling a point over a curve
- Multiplying a point by an integral constant
- Converting a point's projective coordinates (resulting from a doubling or an addition) to the affine coordinates, and oppositely converting a point's affine coordinates to the projective coordinates.

- Testing the point presence on the curve.

Additionally, some higher level services covering the needs for signature generation and verification are offered:

- Generating an ECDSA signature (compliant with FIPS186-2)
- Verifying an ECDSA signature (compliant with FIPS 186-2) The supported curves use the following curve equation in  $GF(2^n)$ :

$$Y^2 + XY = X^3 + aX + b$$

### 37.3.7.1 Parameters Format

#### Related Links

[37.3.5.1. Modular Reduction](#)

[37.3.3.4. Aligned Significant Length](#)

#### 37.3.7.1.1 Polynomials in $GF(2^n)$

Polynomials in  $GF(2^n)$  are binary polynomials reduced modulo the polynomial  $P[X]$ . This polynomial is called the modulus and may be abbreviated to  $P$  in this document. The storage of these polynomials in memory area is described in *Aligned Significant Length* (see *Aligned Significant Length* from Related Links).

For notation simplicity the comparison signs "<" or ">" may be used for polynomials, this is to be interpreted as a comparison between the degree of the polynomials.

In  $GF(2^n)$  fully reduced polynomials are of degree strictly lower than  $\text{degree}(P[X])$ . In many cases the polynomials used in this library are only partially reduced and so have a degree higher or equal than  $\text{degree}(P[X])$ , but this degree is maintained strictly lower than  $(\text{degree}(P[X]) + 15)$ .

#### 37.3.7.1.2 Coordinates System

In this implementation, several choices have been made related to the coordinate systems managed by the elliptic curve primitives.

There are two systems currently managed by the library:

- Affine Coordinates System where each curve point has two coordinates  $(X,Y)$
- Projective Coordinates System where each point is represented with three coordinates  $(X,Y,Z)$

Converting from the affine coordinates system to a projective coordinates system and is performed by extending its representation having  $Z = 1$ :

$$(X,Y) \Rightarrow (X,Y,Z=1)$$

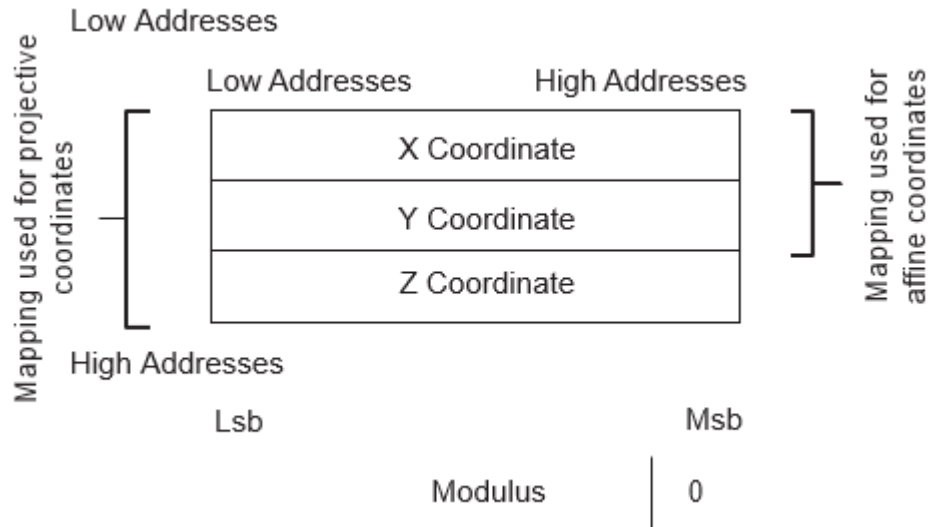
Converting from a projective coordinate to an affine one is a service offered by the library. The formula to perform this conversion is:

$$(X,Y,Z) \Rightarrow (X \Rightarrow Z, Y/Z^2)$$

#### 37.3.7.1.3 Points Representation in Memory

Depending on the representation (Projective or Affine), points are represented in memory as shown in the following figure.

Figure 37-15. Point Representation in Memory



In this figure, the modulus is represented as a reference, and to show that coordinates are always to be provided on the length of the modulus plus one 32-bit word.

Different types of representations are listed here:

$$\text{Affine representation: } Pt = \begin{bmatrix} X_{\text{Affine}} < P \times X^{15} \\ Y_{\text{Affine}} < P \times X^{15} \end{bmatrix}$$

$$\text{Projective representation: } Pt = \begin{bmatrix} X_{\text{Projective}} < P \times X^{15} \\ Y_{\text{Projective}} < P \times X^{15} \\ Z_{\text{Projective}} < P \times X^{15} \end{bmatrix}$$

**Notes:**

1. The minimum value for u2ModLength is 12 bytes. Therefore, the significant length of the modulus must be at least three 32-bit words.
2. In some cases the point can be the infinite point. In this case it is represented with its Z coordinates equal or congruent to zero.

**37.3.7.1.4 Modulus and Modular Constant Parameters**

In most of the services the following parameters must be provided:

- P the Modulus (often pointed by {nu1ModBase,u2ModLength + 4}): This parameter contains the Modulus Polynomial P[X] defining the Galois Field used in points coordinates computations. The Modulus must be u2ModLength bytes long, while having a supplemental zeroed 32-bit word on the MSB side.  
**Note:** Most of the Elliptic Curve computations are reduced modulo P. In many functions the reductions are made with the Fast Reduction.
- Cns the Modular Constant (often pointed by {nu1CnsBase,u2ModLength + 12}): This parameter contains the Modular Constant associated to the Modulus.



**Important:** The Modular Constant must be calculated before using the GF(2<sup>n</sup>) Elliptic Curves functions by a call to the Setup for Modular Reductions with the GF(2<sup>n</sup>) option (see *Modular Reduction* from Related Links).

### 37.3.7.1.5 Curve Parameters in Memory

Some services need one or both of the Elliptic Curve Equation Parameters a and b. In this case these values are organized in memory as follows:

- The a Parameter relative to the Elliptic Curve Equation (often pointed by  $\{\text{nu1ABase}, \text{u2ModLength} + 4\}$ ). The a Parameter is written in a classical way in memory. It is  $\text{u2ModLength}$  bytes long and has a supplemental zeroed 32-bit word on the MSB side.
- The a and b Parameters relative to the Elliptic Curve Equation (often pointed by  $\{\text{nu1ABase}, 2 * \text{u2ModLength} + 8\}$ ):
  - The a Parameter is written in memory on  $\text{u2ModLength}$  bytes long, with a supplemental zeroed 32-bit word on the MSB side.
  - The b Parameter is written in memory after the a Parameter at an offset of  $(\text{u2ModLength} + 4)$  bytes. It is written in memory on  $\text{u2ModLength}$  bytes long, with a supplemental zeroed 32-bit word on the MSB side.

### 37.3.7.2 Point Addition

#### 37.3.7.2.1 Purpose

This service is used to perform a point addition, based on a given elliptic curve over  $\text{GF}(2^n)$ .

Please note that this service is not intended to add the same point twice. In this particular case, use the doubling service (see [37.3.7.3. Point Doubling](#)).

#### 37.3.7.2.2 How to Use the Service

#### 37.3.7.2.3 Description

The operation performed is:

$$Pt_C = Pt_A + Pt_B$$

In this computation, the following parameters need to be provided:

- Point A the input point is filled in projective coordinates (X,Y,Z) (pointed by  $\{\text{nu1PointABase}, 3 * \text{u2ModLength} + 12\}$ ). This point can be the Infinite Point.
- Point B the input point is filled in projective coordinates (X,Y,Z) (pointed by  $\{\text{nu1PointBBase}, 3 * \text{u2ModLength} + 12\}$ ). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by  $\{\text{nu1CnsBase}, \text{u2ModLength} + 12\}$ )
- P the modulus filled (pointed by  $\{\text{nu1ModBase}, \text{u2ModLength} + 4\}$ )
- The a parameter relative to the elliptic curve equation (pointed by  $\{\text{nu1ABase}, \text{u2ModLength} + 4\}$ )
- The workspace not initialized (pointed by  $\{\text{nu1WorkSpace}, 7 * \text{u2ModLength} + 40\}$ )

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the same place than the input point A. This Point can be the Infinite Point.

The services for this operation are:

- Service GF2NEccAddFast: The fast mode is used, the fast modular reduction is used in the computations.



**Important:** Before using this service, ensure that the constant Cns has been calculated with the setup of the Modular Reductions service.

### 37.3.7.2.4 Parameters Definition

**Table 37-94.** GF2NEccAddFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of Modulus P	Base of Modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulo	Length of modulo
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1PointBBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point B (projective coordinates)	Input point B
nu1ABBase	nu1	I	Crypto RAM	u2ModLength + 4	Parameter a of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	7*u2ModLength + 40	-	Corrupted workspace

### 37.3.7.2.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;
//Depending on the function the Random Number Generator
//must be initialized and started
//following the directives given for the RNG on the chip
PUKCL(u2Option) = 0;
PUKCL_GF2NEccAdd(nu1ModBase) = <Base of the ram location of P>;
PUKCL_GF2NEccAdd(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL_GF2NEccAdd(u2ModLength) = <Byte length of P>;
PUKCL_GF2NEccAdd(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL_GF2NEccAdd(nu1PointBBase) = <Base of the ram location of the B point>;
PUKCL_GF2NEccAdd(nu1ABBase) = <Base of the ram location of the a Parameter>;
PUKCL_GF2NEccAdd(nu1Workspace) = <Base of the ram location of the workspace>;
. . .
// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEccAddFast, pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
{
    . . .
}
else // Manage the error

```

### 37.3.7.2.6 Constraints

No overlapping between either input and output are allowed The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PointBBase, nu1ABBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength+ 12}, {nu1PointBBase, 3\*u2ModLength + 12}, {nu1ABBase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PointBBase, 3\*u2ModLength + 12}, {nu1ABBase, u2ModLength + 4} and {nu1Workspace, 5\*u2ModLength + 32}

### 37.3.7.2.7 Status Returned Values

**Table 37-95.** GF2NEccAddFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without errors.



### 37.3.7.3 Point Doubling

#### 37.3.7.3.1 Purpose

This service is used to perform a Point Doubling, based on a given elliptic curve over  $GF(2^n)$ .

#### 37.3.7.3.2 How to Use the Service

#### 37.3.7.3.3 Description

The operation performed is:

$$Pt_C = 2 \times Pt_A$$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3\*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength +8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})
- The workspace not initialized (pointed by {nu1WorkSpace, 4\*u2ModLength +28})
- The a and b Parameters relative to the Elliptic Curve Equation (pointed by {nu1ABBase,2\*u2ModLength+ 8})
- The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the very same place than the input point A. This point can be the Infinite Point.

The service name for this operation is `GF2NEccDblFast`. This service uses Fast mode and Fast Modular Reduction for computation.



**Important:** Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reductions service.

#### 37.3.7.3.4 Parameters Definition

**Table 37-96.** GF2NEccDblFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1ABBase	u2	I	Crypto RAM	2*u2ModLength + 8	Parameters a and b of the elliptic curve	Parameter a and b of the elliptic curve
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1Workspace	nu1	I	Crypto RAM	4*u2ModLength + 28	-	Corrupted workspace

#### 37.3.7.3.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;

PUKCL _GF2NEccDbl(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEccDbl(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEccDbl(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEccDbl(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _GF2NEccDbl(nu1ABBase) = <Base of the a and b parameters of the elliptic curve>;
PUKCL _GF2NEccDbl(nu1Workspace) = <Base of the ram location of the workspace>;
...

```

```
// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEccDbfFast,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
    {
        ...
    }
else // Manage the error
```

### 37.3.7.3.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1ABBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength+ 12}, {nu1ABBase, 2\*u2ModLength + 8}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1ABBase, u2ModLength + 4} and {nu1Workspace, 4\*u2ModLength + 28}

### 37.3.7.3.7 Status Returned Values

**Table 37-97.** GF2NEccDbfFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.

## 37.3.7.4 Scalar Point Multiply

### 37.3.7.4.1 Purpose

This service is used to multiply a point by an integral constant K on a given elliptic curve over GF(2<sup>n</sup>).

### 37.3.7.4.2 How to Use the Service

### 37.3.7.4.3 Description

The operation performed is:

$$Pt_C = K \times Pt_A$$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3\*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 8\*u2ModLength + 44})
- The a and b parameters relative to the elliptic curve (pointed by {nu1ABBase,2\*u2ModLength + 8})
- K the scalar number (pointed by {nu1ScalarNumber,u2ScalarLength + 4})

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the very same place than the input point A. This point can be the Infinite Point.

The service name for this operation is GF2NEccMulFast. This service uses Fast mode and Fast Modular Reduction for computation.



**Important:** Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reductions service.

### 37.3.7.4.4 Parameters Definition

**Table 37-98.** GF2NEccMulFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1KBase	nu1	I	Crypto RAM	u2KLength	Scalar number used to multiply the point A	Unchanged
u2KLength	u2	I	-	-	Length of scalar K	Length of scalar K
nu1PointBase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1ABase	nu1	I	Crypto RAM	2*u2ModLength + 8	Parameters a and b of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	-	Corrupted workspace

### 37.3.7.4.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;

PUKCL _GF2NEccMul(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEccMul(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEccMul(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEccMul(nu1PointBase) = <Base of the ram location of the A point>;
PUKCL _GF2NEccMul(nu1ABase) = <Base of the ram location of the parameters a and b of the
elliptic
curve>;
PUKCL _GF2NEccMul(nu1KBase) = <Base of the ram location of the scalar number>;
PUKCL _GF2NEccMul(nu1Workspace) = <Base of the ram location of the workspace>;
PUKCL _GF2NEccMul(u2KLength) = <Length of the ram location of the scalar number>;

...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEccMulFast,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.7.4.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointBase, nu1ABase, nu1KBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointBase, 3\*u2ModLength+ 12}, {nu1ABase, 2\*u2ModLength + 8}, {nu1KBase, u2KLength} or {nu1Workspace, 8\*u2ModLength + 44} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length

- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointBase, 3\*u2ModLength + 12}, {nu1ABase, 2\*u2ModLength + 8}, {nu1KBase, u2KLength} and {nu1Workspace, 8\*u2ModLength + 44}

### 37.3.7.4.7 Status Returned Values

**Table 37-99.** GF2NEccMulFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.

### 37.3.7.5 Projective to Affine Coordinates Conversion

#### 37.3.7.5.1 Purpose

This service is used to perform a point coordinates conversion from a projective representation to an affine.

#### 37.3.7.5.2 How to Use the Service

#### 37.3.7.5.3 Description

The operation performed is:

$$Pt_X \text{ Affine coordinate} = \left[ \frac{Pt_X \text{ Projective coordinate}}{(Pt_Z \text{ Projective coordinate})} \right]$$

$$Pt_Y \text{ Affine coordinate} = \left[ \frac{Pt_Y \text{ Projective coordinate}}{(Pt_Z \text{ Projective coordinate})^2} \right]$$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) or affine coordinates for X and Y, and setting Z to 1 (pointed by {nu1PointABase, 3\*u2ModLength + 12}). The Point A can be the point at infinity. In this case, the u2Status returned is PUKCL\_POINT\_AT\_INFINITY.
- Cns the Modular Constant filled (pointed by {nu1CnsBase, u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 4\*u2ModLength + 48})

The result is the point A with its (X,Y) coordinates converted to affine, and the Z coordinate set to 1.

The service name for this operation is GF2NEcConvProjToAffine.



**Important:** Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reductions service.

#### 37.3.7.5.4 Parameters Definition

**Table 37-100.** GF2NEcConvProjToAffine Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1PointABase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point A	Resulting point A in affine coordinates
nu1Workspace	nu1	I	Crypto RAM	4*u2ModLength + 48	-	Workspace

### 37.3.7.5.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PUCCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

PUKCL _GF2NEcConvProjToAffine(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcConvProjToAffine(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcConvProjToAffine(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEcConvProjToAffine(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _GF2NEcConvProjToAffine(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEcConvProjToAffine, &PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.7.5.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12} and {nu1Workspace, 4\*u2ModLength + 48}

### 37.3.7.5.7 Status Returned Values

**Table 37-101.** GF2NEcConvProjToAffine Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.
PUKCL_POINT_AT_INFINITY	Warning	The input point has its Z equal to zero, so it is a representation of the infinite point.

### 37.3.7.6 Affine to Projective Coordinates Conversion

#### 37.3.7.6.1 Purpose

This service is used to perform a point coordinates conversion from an affine point representation to projective.

#### 37.3.7.6.2 How to Use the Service

#### 37.3.7.6.3 Description

The operation performed is:

$$\text{affine}(X_a, Y_a) \rightarrow \text{projective}(X_p, Y_p, Z_p)$$

In this computation, the following parameters need to be provided:

- A the input point is filled in affine coordinates for X and Y, and setting Z to 1 (pointed by {nu1PointABase, 3\*u2ModLength + 4}).
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase, u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 4})

- The workspace not initialized (pointed by  $\{nu1Workspace, 2*u2ModLength + 16\}$ ) The result is the point A with its (X,Y,Z) projective coordinates.

The service name for this operation is `GF2NEcConvAffineToProjective`.



**Important:** Before using this service, ensure that the constant `Cns` has been calculated with the setup of the Fast Modular Reductions service.

### 37.3.7.6.4 Parameters Definition

**Table 37-102.** GF2NEcConvAffineToProjective Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
<code>nu1ModBase</code>	<code>nu1</code>	I	Crypto RAM	$u2ModLength + 4$	Base of modulus P	Base of modulus P
<code>nu1CnsBase</code>	<code>nu1</code>	I	Crypto RAM	$u2ModLength + 8$	Base of Cns	Base of Cns
<code>u2ModLength</code>	<code>u2</code>	I	-	-	Length of modulus P	Length of modulus P
<code>nu1PointABase</code>	<code>nu1</code>	I	Crypto RAM	$3*u2ModLength + 12$	Input point A	Resulting point A in affine coordinates
<code>nu1Workspace</code>	<code>nu1</code>	I	Crypto RAM	$2*u2ModLength + 16$	-	Workspace

### 37.3.7.6.5 Code Example

```

PUKCL_PARAM PUKCLParam;
P_PUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

PUKCL _GF2NEcConvAffineToProjective(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcConvAffineToProjective(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcConvAffineToProjective(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEcConvAffineToProjective(nu1PointABase) = <Base of the ram location of the A
point>;
PUKCL _GF2NEcConvAffineToProjective(nu1Workspace) = <Base of the ram location of the
workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEcConvAffineToProjective,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.7.6.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- `nu1ModBase`, `nu1CnsBase`, `nu1PointABase`, `nu1Workspace` are not aligned on 32-bit boundaries
- $\{nu1ModBase, u2ModLength + 4\}$ ,  $\{nu1CnsBase, u2ModLength + 8\}$ ,  $\{nu1PointABase, 3*u2ModLength + 12\}$ ,  $\{nu1Workspace, <WorkspaceLength>\}$  are not in Crypto RAM
- `u2ModLength` is either:  $< 12$ ,  $> 0xffc$  or not a 32-bit length
- All overlapping between  $\{nu1ModBase, u2ModLength + 4\}$ ,  $\{nu1CnsBase, u2ModLength + 8\}$ ,  $\{nu1PointABase, 3*u2ModLength + 12\}$ , and  $\{nu1Workspace, 2*u2ModLength + 16\}$

### 37.3.7.6.7 Status Returned Values

**Table 37-103.** GF2NEcConvAffineToProjective Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.

### 37.3.7.7 Randomize Coordinate

#### 37.3.7.7.1 Purpose

This service is used to convert the Projective representation of a point to another Projective representation.

#### 37.3.7.7.2 How to Use the Service

#### 37.3.7.7.3 Description

The operation performed is:

$$Projective(X_1, Y_1, Z_1) \rightarrow Projective(X_2, Y_2, Z_2)$$

In this computation, the following parameters need to be provided:

- The input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointBase,3\*u2ModLength + 12}). This Point must not be the point at infinity.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 3\*u2ModLength + 28})
- The random number (pointed by {nu1RandomBase, u2ModLength + 4}) The result is the point nu1PointBase with its (X,Y,Z) coordinates randomized. The service for this operation is GF2NEcRandomiseCoordinate.



#### Important:

Before using this service:

- Ensure that the constant Cns has been calculated with the Setup of the fast Modular Reductions service.
- Be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

### 37.3.7.7.4 Parameters Definition

**Table 37-104.** GF2NEcRandomiseCoordinate Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1PointBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point	Resulting point
nu1RandomBase	nu1	I	Crypto RAM	u2ModLength + 4	Random	Corrupted
nu1Workspace	nu1	I	Crypto RAM	3*u2ModLength + 28	-	Workspace

### 37.3.7.7.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
```

```
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL _GF2NEcRandomiseCoordinate(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcRandomiseCoordinate(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcRandomiseCoordinate(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEcRandomiseCoordinate(nu1RandomBase) = <Base of the ram location where the the rng
is stored>;
PUKCL _GF2NEcRandomiseCoordinate(nu1PointBase) = <Base of the ram location of the point>;
PUKCL _GF2NEcRandomiseCoordinate(nu1Workspace) =
<Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEcRandomiseCoordinate,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error
```

### 37.3.7.7.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1RandomBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1RandomBase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1RandomBase, u2ModLength + 4} and {nu1Workspace, 3\*u2ModLength + 28}

### 37.3.7.7.7 Status Returned Values

**Table 37-105.** GF2NEcRandomiseCoordinate Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.

### 37.3.7.8 Point is on Elliptic Curve

#### 37.3.7.8.1 Purpose

This service is used to test whether the point is on the curve.

#### 37.3.7.8.2 How to Use the Service

#### 37.3.7.8.3 Description

The operation performed is:

Status = IsPointOnCurve(X, Y, Z);

In this computation, the following parameters need to be provided:

- The input points filled in projective coordinates (X, Y, Z) (pointed by {nu1PointBase, 3\*U2ModLength + 4}). This point can be point at infinity.
- AParam and BParam are the Elliptic Curve Equation parameters (pointed by {nu1AParam, u2ModLength+ 4} and {nu1BParam, u2ModLength + 4}).
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase, u2ModLength + 8})



- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 8})
- The workspace not initialized (pointed by {nu1WorkSpace, 4\*u2ModLength + 28})

The service name for this operation is GF2NEcPointIsOnCurve.



**Important:** Before using this service, the constant Cns must have been calculated with the Fast Modular Reduction service.

### 37.3.7.8.4 Parameters Definition

**Table 37-106.** GF2NEcPointIsOnCurve Service Parameters

Parameter	Type	Dir.	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1PointBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point	Unchanged
nu1AParam	nu1	I	Crypto RAM	u2ModLength + 4	The parameter a	Unchanged
nu1BParam	nu1	I	Crypto RAM	u2ModLength + 4	The parameter b	Unchanged
nu1Workspace	nu1	I	Crypto RAM	4*u2ModLength + 28	N/A	Workspace

### 37.3.7.8.5 Code Example

```

PUKCL_PARAM PUKCLParam;
P_PUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL _GF2NEcPointIsOnCurve(nulModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcPointIsOnCurve(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcPointIsOnCurve(nulCnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEcPointIsOnCurve(nulPointABase) = <Base of the A point>;
PUKCL _GF2NEcPointIsOnCurve(nulAParam) = <Base of the ram location of the parameter a>;
PUKCL _GF2NEcPointIsOnCurve(nulBParam) = <Base of the ram location of the parameter b>;
PUKCL _GF2NEcPointIsOnCurve(nulPointBase) = <Base of the ram location of the point>;
PUKCL _GF2NEcPointIsOnCurve(nulWorkspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEcPointIsOnCurve,
pvPUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
...
}
else // Manage the error

```

### 37.3.7.8.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1AParam, nu1BParam and nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1AParam, u2ModLength + 4}, {nu1BParam, u2ModLength + 4}, {nu1Workspace, 4\*u2ModLength + 28} are not in Crypto RAM

- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1AParam, u2ModLength + 4}, {nu1BParam, u2ModLength + 4} and {nu1Workspace, 4\*u2ModLength + 28}

### 37.3.7.8.7 Status Returned Values

**Table 37-107.** GF2NEcPointIsOnCurve Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The point is on the curve.
PUKCL_POINT_IS_NOT_ON_CURVE	Warning	The point is not on the curve.
PUKCL_POINT_AT_INFINITY	Warning	The input point has its Z equal to zero, so it's a representation of the infinite point.

### 37.3.7.9 Generating an ECDSA Signature (Compliant with FIPS 186-2)

#### 37.3.7.9.1 Purpose

This service is used to generate an ECDSA signature following the FIPS 186-2. It performs the second step of the Signature Generation. A hash value (HashVal) must be provided as input, it has to be previously computed from the message to be signed using a secure hash algorithm.

A scalar number must be provided, as described in the FIPS 186-2.

The result (R,S) is computed by this service. If S equals zero, the status is set to PUKCL\_WRONG\_SELECT\_NUMBER.

#### 37.3.7.9.2 How to Use the Service

#### 37.3.7.9.3 Description

The operation performed is:

$$(R, S) = EcDsaSign(Pt_A, HashVal, k, CurveParameters, PrivateKey)$$

This service processes the following checks:

- If the Scalar Number k is out of the range [1, PointOrder -1], the calculus is stopped and the status is set to PUKCL\_WRONG\_SELECT\_NUMBER.
- If R equals zero, the calculus is stopped and the status is set to PUKCL\_WRONG\_SELECT\_NUMBER.
- If S equals zero, the calculus is stopped and the status is set to PUKCL\_WRONG\_SELECT\_NUMBER. In this computation, the following parameters need to be provided:
- A the input point is filled in "mixed" coordinates (X,Y) with the affine values and Z = 1 (pointed by {nu1PointABase, 3\*u2ModLength + 12})
- Cns the working space for the Fast Modular Constant not initialized (pointed by {nu1CnsBase, u2ScalarLength + 8})
- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 8\*u2ModLength + 44})
- The a and b parameters relative to the elliptic curve equation (pointed by {nu1ABase, 2\*u2ModLength + 8})
- The order of the Point A on the elliptic curve (pointed by {nu1OrderPointBase, u2ScalarLength + 4})
- k the input Scalar Number beforehand generated and filled (pointed by {nu1ScalarNumber, u2ScalarLength + 4})
- HashVal the hash value beforehand generated and filled (pointed by {nu1HashBase, u2ScalarLength + 4})

- The Private Key (pointed by {nu1PrivateKey, u2ScalarLength +4})
- Generally u2ScalarLength is equal to (u2ModLength) or (u2ModLength + 4)



**Important:**

For the ECDSA signature generation be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

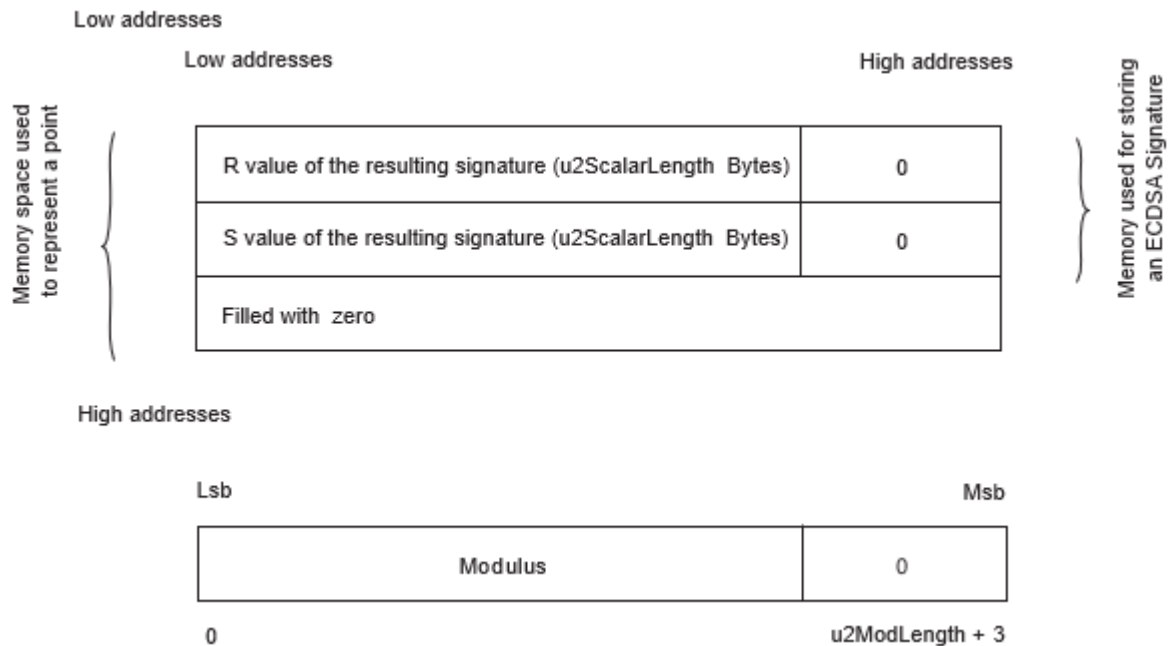
The scalar number k must be selected at random. This random must be generated before the call of the ECDSA signature. For this random generation be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

The operation performed is:

- Compute the ECDSA (R,S) as described in FIPS 186-2, but leaving the user the role of computing the input Hash Value, thus leaving the freedom of using any other algorithm than SHA-1.
- Compute a R value using the input A point and the scalar number.
- Compute a S value using R, the scalar number, the private key and the provided hash value. Note that the resulting signature (R,S) is stored at the place of the input A point.
- If all is correct and S is different from zero, the status is set to PUKCL\_OK. If all is correct and S equals zero, the status is set to PUKCL\_WRONG\_SELECT\_NUMBER. If an error occurs, the status is set to the corresponding error value (see Status Returned Values below).

The service name for this operation is GF2NEcDsaGenerateFast. The fast mode is used, the fast modular reduction is used in the computations.

- The signature (R,S), when resulting from a computation is given back at address of the A point:
  - The R value result with u2ModLength + 4 bytes (padded with zeros).
  - The S value result with u2ModLength + 4 bytes (padded with zeros)
  - The u2NLength + 4 following bytes (space for the third coordinate of A) are filled with zeros.



### 37.3.7.9.4 Parameters Definition

**Table 37-108.** GF2NEcdsaGenerateFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ScalarLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1ScalarNumber	nu1	I	Crypto RAM	u2ScalarLength + 4	Scalar Number used to multiply the point A	Unchanged
nu1OrderPointBase	nu1	I	Crypto RAM	u2ScalarLength + 4	Order of the Point A in the elliptic curve	Unchanged
nu1PrivateKey	nu1	I/O	Crypto RAM	u2ScalarLength + 4	Base of the Private Key	Unchanged
nu1HashBase <sup>(1)</sup>	nu1	I	Crypto RAM	u2ScalarLength + 4	Base of the hash value resulting from the previous SHA	Unchanged
u2ScalarLength	u2	I	-	-	Length of scalar (same length as the length of order)	Length of scalar
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (three coordinates (X,Y) affine and Z = 1)	Resulting signature (R,S,0)
nu1ABase	nu1	I	Crypto RAM	2*u2ModLength + 8	Parameter a of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	-	Corrupted workspace

**Note:**

1. Whatever the chosen SHA, the resulting hash value may have a length inferior or equal to the modulo length and be padded with zeros until its total length is u2ModLength + 4.

### 37.3.7.9.5 Code Example

```

PUKCL_PARAM PUKCLParam;
P_PUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL _GF2NEcdsaGenerate(nulModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcdsaGenerate(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcdsaGenerate(nulCnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEcdsaGenerate(nulPointABase) = <Base of the A point>;
PUKCL _GF2NEcdsaGenerate(nulPrivateKey) = <Base of the Private Key>;
PUKCL _GF2NEcdsaGenerate(nulScalarNumber) = <Base of the ScalarNumber>;
PUKCL _GF2NEcdsaGenerate(nulOrderPointBase) = <Base of the order of A point>;
PUKCL _GF2NEcdsaGenerate(nulABase) = <Base of the a parameter of the curve>; PUKCL
_GF2NEcdsaGenerate(nulWorkspace) = <Base of the workspace>;
PUKCL _GF2NEcdsaGenerate(nulHashBase) = <Base of the SHA resulting hash>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEcdsaGenerateFast, pvPUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
{
    ...
}
else // Manage the error

```

### 37.3.7.9.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PrivateKey, nu1ScalarNumber, nu1OrderPointBase, nu1ABase, nu1Workspace or nu1HashBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PrivateKey, u2ScalarLength + 4}, {nu1ScalarNumber, u2ScalarLength + 4}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} or {nu1HashBase, u2ScalarLength + 4} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PrivateKey, u2ScalarLength + 4}, {nu1ScalarNumber, u2ScalarLength + 4}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} and {nu1HashBase, u2ScalarLength + 4}

### 37.3.7.9.7 Status Returned Values

**Table 37-109.** GF2NEcdsaGenerate Fast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without problem.
PUKCL_WRONG_SELECTNUMBER	Warning	The given value for nu1ScalarNumber is not good to perform this signature generation.

### 37.3.7.10 Verifying an ECDSA Signature (Compliant with FIPS 186-2)

#### 37.3.7.10.1 Purpose

This service is used to verify an ECDSA signature following the FIPS 186-2. It performs the second step of the Signature Verification.

A hash value (HashVal) must be provided as input, it has to be previously computed from the message to be signed using a secure hash algorithm.

As second significant input, the Signature is provided to be checked. This service checks the signature and fills the status accordingly.

#### 37.3.7.10.2 How to Use the Service

#### 37.3.7.10.3 Description

The operation performed is:

*Verify = EcDsaVerifySignature(Pt<sub>A</sub>, HashVal, Signature, CurveParameters, PublicKey)*

The points used for this operation are represented in different coordinate systems. In this computation, the following parameters need to be provided:

- A the input point is filled with the affine values (X,Y) and Z = 1 (pointed by {nu1PointABase, 3\*u2ModLength + 12})
- Cns the working space for the Fast Modular Constant not initialized (pointed by {nu1CnsBase, u2ScalarLength + 8})
- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 8\*u2ModLength + 44}) The a and b parameters relative to the elliptic curve (pointed by {nu1ABase, 2\*u2ModLength + 8})
- The order of the Point A on the elliptic curve (pointed by {nu1OrderPointBase, u2ScalarLength + 4})

- HashVal the hash value beforehand generated and filled (pointed by {nu1HashBase,u2ScalarLength +4})
- The Public Key point is filled in “mixed” coordinates (X,Y) with the affine values and Z = 1 (pointed by {nu1PointPublicKeyGen, 3\*u2ModLength + 12})
- The input signature (R,S), even if it is not a Point, is represented in memory like a point in affine coordinates (X,Y) (pointed by {nu1PointSignature, 2\*u2ScalarLength + 8})



**Important:** For the ECDSA signature verification be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

- The operation consists in obtaining a V value with all these input parameter and check that V equals the provided R. If all is correct and the signature is the good one, the status is set to PUKCL\_OK. If all is correct and the signature is wrong, the status is set to PUKCL\_WRONG\_SIGNATURE. If an error occurs, the status is set to the corresponding error value (see Status Returned Values below).

The service name for this operation is `GF2NEcDsaVerifyFast`. This service uses Fast mode and Fast Modular Reduction for computation.

### 37.3.7.10.4 Parameters Definition

**Table 37-110.** GF2NEcDsaVerifyFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ScalarLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	-	-	Length of modulus P	Length of modulus P
nu1OrderPointBase	nu1	I	Crypto RAM	u2ScalarLength + 4	Order of the Point A in the elliptic curve	Unchanged
nu1PointSignature	nu1	I	Crypto RAM	2*u2ScalarLength + 8	Signature(r, s)	Corrupted
nu1HashBase <sup>(1)</sup>	nu1	I	Crypto RAM	u2ScalarLength + 4	Base of the hash value resulting from the previous SHA	Corrupted
u2ScalarLength	u2	I	-	-	Length of scalar	Length of scalar
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Generator point	Corrupted
nu1PointPublicKeyGen	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Public point	Corrupted
nu1ABase	nu1	I	Crypto RAM	2*u2ModLength + 8	Parameter a and b of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	-	Corrupted workspace

**Note:**

1. Whatever the chosen SHA, the resulting hash value may have a length inferior or equal to the modulo length and be padded with zeros until its total length is u2ModLength + 4.

### 37.3.7.10.5 Code Example

```

PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

```

```
// Depending on the option specified, not all fields must be filled PUKCL
_GF2NEcdsaVerify(nu1ModBase) = <Base of the ram location of P>;
_PUKCL _GF2NEcdsaVerify(u2ModLength) = <Byte length of P>;
_PUKCL _GF2NEcdsaVerify(nu1CnsBase) = <Base of the ram location of Cns>;
_PUKCL _GF2NEcdsaVerify(nu1PointABase) = <Base of the A point>;
_PUKCL _GF2NEcdsaVerify(nu1PrivateKey) = <Base of the Private Key>;
_PUKCL _GF2NEcdsaVerify(nu1ScalarNumber) = <Base of the ScalarNumber>;
_PUKCL _GF2NEcdsaVerify(nu1OrderPointBase) = <Base of the order of A point>;
_PUKCL _GF2NEcdsaVerify(nu1ABase) = <Base of the a parameter of the curve>; _PUKCL
_GF2NEcdsaVerify(nu1Workspace) = <Base of the workspace>;
_PUKCL _GF2NEcdsaVerify(nu1HashBase) = <Base of the SHA resulting hash>;
...

// v_PUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
v_PUKCL_Process(GF2NEcdsaVerifyFast, &_PUKCLParam);
if (_PUKCL (u2Status) == _PUKCL_OK)
{
    ...
}
else
    if (_PUKCL (u2Status) == _PUKCL_WRONG_SIGNATURE)
    {
        ...
    }
else // Manage the error
```

### 37.3.7.10.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PointPublicKeyGen, nu1PointSignature, nu1OrderPointBase, nu1ABase, nu1Workspace or nu1HashBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PointPublicKeyGen, 3\*u2ModLength + 12}, {nu1PointSignature, 2\*u2ScalarLength + 8}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, 2\*u2ModLength + 8}, {nu1Workspace, <WorkspaceLength>} or {nu1HashBase, u2ScalarLength + 4} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3\*u2ModLength + 12}, {nu1PointPublicKeyGen, 3\*u2ModLength + 12}, {nu1PointSignature, 2\*u2ScalarLength + 8}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, 2\*u2ModLength + 8}, {nu1Workspace, <WorkspaceLength>} and {nu1HashBase, u2ScalarLength + 4}

### 37.3.7.10.7 Status Returned Values

**Table 37-111.** GF2NEcdsaVerifyFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	-	The computation passed without errors. The signature is correct.
PUKCL_WRONG_SIGNATURE	Warning	The signature is incorrect.

## 37.3.8 PUKCL Requirements and Performance

### 37.3.8.1 Services Stack Usage

This library is using the main core to execute its computations, and therefore is also sharing some resources with the application.

It may be important for the application to know RAM usage by the library functions and to be aware that the library does not use any global variables.

The following table provides the minimum number of bytes used by the library that have to be available on the stacks to ensure that the functionality can be executed correctly. In some cases, the library may use less bytes than the specified number for some options. This table contains estimated values.

**Table 37-112.** Services Stack Usage

PUKCL Service	STACK Usage (Bytes)
SelfTest	112
ClearFlags	0
Swap	8
Fill	8
CondCopy	24
FastCopy	16
Smult	16
Smult (with reduction)	88
Comp	8
Fmult	24
Fmult (with reduction)	96
Square	16
Square (with reduction)	88
Div	144
GCD	136
RedMod (Setup)	160
RedMod (using fast reduction)	80
RedMod (randomize)	80
RedMod (Normalize)	80
RedMod (Using Division)	184
ExpMod	200
PrimeGen	416
CRT	304
ZpEccAddFast	104
ZpEccAddSubFast	92
ZpEcConvProjToAffine	280
ZpEcConvAffineToProjective	64
ZpEccDblFast	96
ZpEccMulFast	168
ZpEccQuickDualMulFast	216
ZpEcDsaGenerateFast	392
ZpEcDsaVerifyFast	456
ZpEcDsaQuickVerify	368
ZpEcRandomiseCoordinate	56
GF2NEccAddFast	128
GF2NEcConvProjToAffine	264
GF2NEcConvAffineToProjective	56
GF2NEccDblFast	136
GF2NEccMulFast	208
GF2NEcDsaGenerateFast	376
GF2NEcDsaVerifyFast	440



.....continued

PUKCL Service	STACK Usage (Bytes)
GF2NEcRandomiseCoordinate	56

### 37.3.8.2 Parameter Size Limits for Different Services

The following table lists parameter size limits for different services.

For the services ModExp, PrimeGen, and CRT, additional details are available in the service description.

**Table 37-113.** Parameter Size Limits

API	Min/Max Sizes	Comments
SelfTest	—	—
ClearFlags	—	—
Swap	4 bytes to 2044 bytes	Per block to be swapped
Fill	4 bytes to 4088 bytes	—
Fast Copy/Clear	4 bytes to 2044 bytes	Supposing Length(R) = Length(X)
Conditional Copy/Clear	4 bytes to 2044 bytes	Supposing Length(R) = Length(X)
Smult	4 bytes to 2040 bytes	Supposing Length(R) = Length(X) + 4 Bytes, No Z Parameter, No Reduction
Compare	4 bytes to 2044 bytes	Supposing Length(X) = Length(Y)
FMult	Input: 4 bytes to 1020 bytes Output: 4bytes to 2040 bytes	Supposing Length(Y) = Length(X), No Z Parameter, No Reduction
Square	Input: 4 bytes to 1020 bytes Output: 4 bytes to 2040 bytes	Supposing No Z Parameter, No Reduction
Euclidean Division	Divider: 8 to 1016 bytes Num.: 8 to 2032 bytes	Supposing Length(Num) = 2*Length(Divider)
Mod. inv. / GCD	8 to 1012 bytes	—
ModRed	Modulus: 12 to 1016 bytes Input: 24 to 2032 bytes	Supposing RBase = XBase
Fast ModExp Exp in Crypto RAM	12 to 576 bytes (96 to 4608 bits)	Supposing Length(Exponent) = Length(Modulus), Window Size = 1 With the Exponent in Crypto RAM
Fast ModExp Exp not in Crypto RAM	12 to 672 bytes (96 to 5376 bits)	Supposing Length(Exponent) = Length(Modulus), Window Size = 1 With the Exponent not in Crypto RAM
Prime Gen.	Prime Number: 12 to 448 bytes (96 to 3584 bits)	Supposing Window Size = 1
CRT	Modulus = Two Primes: Size of one prime from 24 to 448 bytes Modulus = from 48 to 896 bytes (384 to 7168 bits)	Supposing Length(Exponent) = Length(Modulus), Window Size = 1
ECC Addition and Subtraction GF(p)	Modulus: 12 to 308 bytes	—
ECC Doubling GF(p)	Modulus: 12 to 400 bytes	—
ECC Multiplication GF(p)	Modulus: 12 to 264 bytes	Supposing Length(Scalar) = Length(Modulus)
ECC Quick Dual Multiplication GF(p)	Modulus: 12 to 152 bytes	—

.....continued

API	Min/Max Sizes	Comments
ECDSA Generate GF(p)	Modulus: 12 to 220 bytes (up to 521 bits for common curves)	Supposing Length(Scalar) = Length(Modulus)
ECDSA Verify GF(p)	Modulus: 12 to 188 bytes (up to 521 bits for common curves)	Supposing Length(Scalar) = Length(Modulus)
ECC Addition GF(2n)	Modulus: 12 to 248 bytes	—
ECC Doubling GF(2n)	Modulus: 12 to 364 bytes	—
ECC Multiplication GF(2n)	Modulus: 12 to 250 bytes	Supposing Length(Scalar) = Length(Modulus)
ECDSA Generate GF(2n)	Modulus: 12 to 208 bytes (up to 571 bits for common curves)	Supposing Length(Scalar) = Length(Modulus)
ECDSA Verify GF(2n)	Modulus: 12 to 180 bytes (up to 571 bits for common curves)	Supposing Length(Scalar) = Length(Modulus)
ECDSA Quick Verify GF(2n)	Modulus: 12 to 140 bytes (up to 571 bits for common curves)	Supposing Length(Scalar) = Length(Modulus)

### 37.3.8.3 Service Timing

The values in the following tables are estimated performances for CPU clock of 64 MHz. The CPU and PUKCC are operated at the same frequency. Due to possible change in the parameters values, the measurements show approximated values.

Other test conditions:

- PUKCL library data in Crypto RAM
- Test code and test data in SRAM
- ICache and DCache are disabled

#### 37.3.8.3.1 Service Timing for RSA

RSA uses the ExpMod service for encryption and decryption. Following tables show service timing, where 'W' indicates window size.

**Table 37-114. RSA1024**

Operation	Clock Cycles	Timing one block
RSA 1024 decryption / signature generation. No CRT, Regular implementation, W=4	3.05 MCycles	47.799 ms
RSA 1024 decryption / signature generation. With CRT, Regular implementation, W=4	1.09 MCycles	17.109 ms
RSA 1024 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=3	0.07 MCycles	1.141 ms
RSA 1024 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=0x10001	0.07 MCycles	1.129 ms

**Table 37-115. RSA2048**

Operation	Clock Cycles	Timing One block
RSA 2048 decryption / signature generation. No CRT, Regular implementation, W=4	21.6 MCycles	338.249 ms
RSA 2048 decryption / signature generation. With CRT, Regular implementation, W=4	6.36 MCycles	99.408 ms
RSA 2048 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=3	0.24 MCycles	3.843 ms

.....continued

Operation	Clock Cycles	Timing One block
RSA 2048 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=0x10001	0.24 MCycles	3.827 ms

**Table 37-116. RSA4096**

Operation	Clock Cycles	Timing One block
RSA 4096 Decryption / signature generation. No CRT, Regular implementation, W=1	209 MCycles	3.2742s
RSA 4096 Decryption / signature generation. With CRT, Regular implementation, W=3	46.1 MCycles	720.95 ms
RSA 4096 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=3	0.91 MCycles	14.346 ms
RSA 4096 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=0x10001	0.91 MCycles	14.337 ms

### 37.3.8.3.2 Service Timing for Prime Generation

Prime generation uses the PrimeGen service.

**Table 37-117. Prime Generation**

Operation	Clock Cycles	Timing One Block
Regular Generation of two primes, Prime_Length=512 bits, W=4, Rabin Miller Iterations Number = 3, (average of 200 samples)	Mean = 47.4 MCycles	Mean = 0.4s
Regular Generation of two primes, Prime_Length=512 bits, W=4, Rabin Miller Iterations Number = 3, (Standard Deviation for 200 samples)	Std Dev = 30.3 MCycles	Std Dev = 0.47s
Regular Generation of two primes, Prime_Length=1024 bits, W=4, Rabin Miller Iterations Number = 3, (average of 200 samples)	Mean = 419.71 MCycles	Mean = 6.558s
Regular Generation of two primes, Prime_Length=1024 bits, W=4, Rabin Miller Iterations Number = 3, (Standard Deviation for 200 samples)	Std Dev = 294 MCycles	Std Dev = 4.59s
Regular Generation of two primes, Prime_Length=2048 bits, W=4, Rabin Miller Iterations Number = 3, (average of 200 samples)	Mean = 4.78 GCycles	Mean = 74.68s
Regular Generation of two primes, Prime_Length=2048 bits, W=4, Rabin Miller Iterations Number = 3, (Standard Deviation for 200 samples)	Std Dev = 3.05 GCycles	Std Dev = 47.65s

### 37.3.8.3.3 Service Timing for ECDSA on Prime Field

In the following table, ECDSA signature generation uses the ZpEcDsaGenerateFast service and signature verification uses ZpEcDsaQuickVerify

**Table 37-118. ECDSA GF(p)**

Operation	Clock Cycles	Timing One block
ECDSA GF(p) 256 Generate Fast	2.67 MCycles	41.864 ms
ECDSA GF(p) 256 Verify Quick W=(4,4) Scalar in PUKCC RAM	1.84 MCycles	28.888 ms
ECDSA GF(p) 384 Generate Fast	6.18 MCycles	96.712 ms
ECDSA GF(p) 384 Verify Quick W=(4,4) Scalar in PUKCC RAM	4.15 MCycles	64.868 ms
ECDSA GF(p) 521 Generate Fast	13.36 MCycles	208.869 ms
ECDSA GF(p) 521 Verify Quick W=(4,4) Scalar in PUKCC RAM	8.81 MCycles	137.711 ms

### 37.3.8.3.4 Service Timing for ECDSA on Binary Field

In the following table, ECDSA signature generation uses the GF2NEcDsaGenerateFast service and signature verification uses GF2NEcDsaVerifyFast

**Table 37-119. ECDSA GF(2<sup>n</sup>)**

Operation	CPU Cycles	Timing One block
ECDSA GF(2 <sup>n</sup> ) B283 Generate Fast	3.21 MCycles	50.301 ms
ECDSA GF(2 <sup>n</sup> ) B283 Verify	6.40 MCycles	100.150 ms
ECDSA GF(2 <sup>n</sup> ) B409 Generate Fast	6.94 MCycles	108.554 ms
ECDSA GF(2 <sup>n</sup> ) B409 Verify	13.73 MCycles	214.571 ms
ECDSA GF(2 <sup>n</sup> ) B571 Generate Fast	15.08 MCycles	235.704 ms
ECDSA GF(2 <sup>n</sup> ) B571 Verify	30.07 MCycles	469.972 ms

## 38. Analog-to-Digital Converter (ADC)

### 38.1 Overview

The PIC32CX-BZ2 12-bit High Speed Successive Approximation Register (SAR) Analog-to-Digital Converter (ADC) includes the following features:

- 12-bit resolution
- One ADC module, up to 2 Msps conversion rate
- Single-ended and/or differential input
- Supported in Sleep mode
- Two digital comparators
- Two digital filters supporting two modes:
  - Oversampling mode
  - Averaging mode
- Designed for motor control, power conversion and general purpose applications

The PIC32CX-BZ2 has one shared ADC module. This ADC module incorporates a multiplexer on the input to facilitate a group of inputs and provides a flexible automated scanning option through the input scan logic.

For the ADC module, the analog inputs are connected to the Sample and Hold (S&H) capacitor. The ADC module performs the conversion of the input analog signal based on the configurations set in the registers. When the conversion is complete, the final result is stored in the result buffer for the specific analog input and is passed to the digital filter and digital comparator if configured to use data from this particular sample.

**Equation 38-1.** ADC Throughput Rate

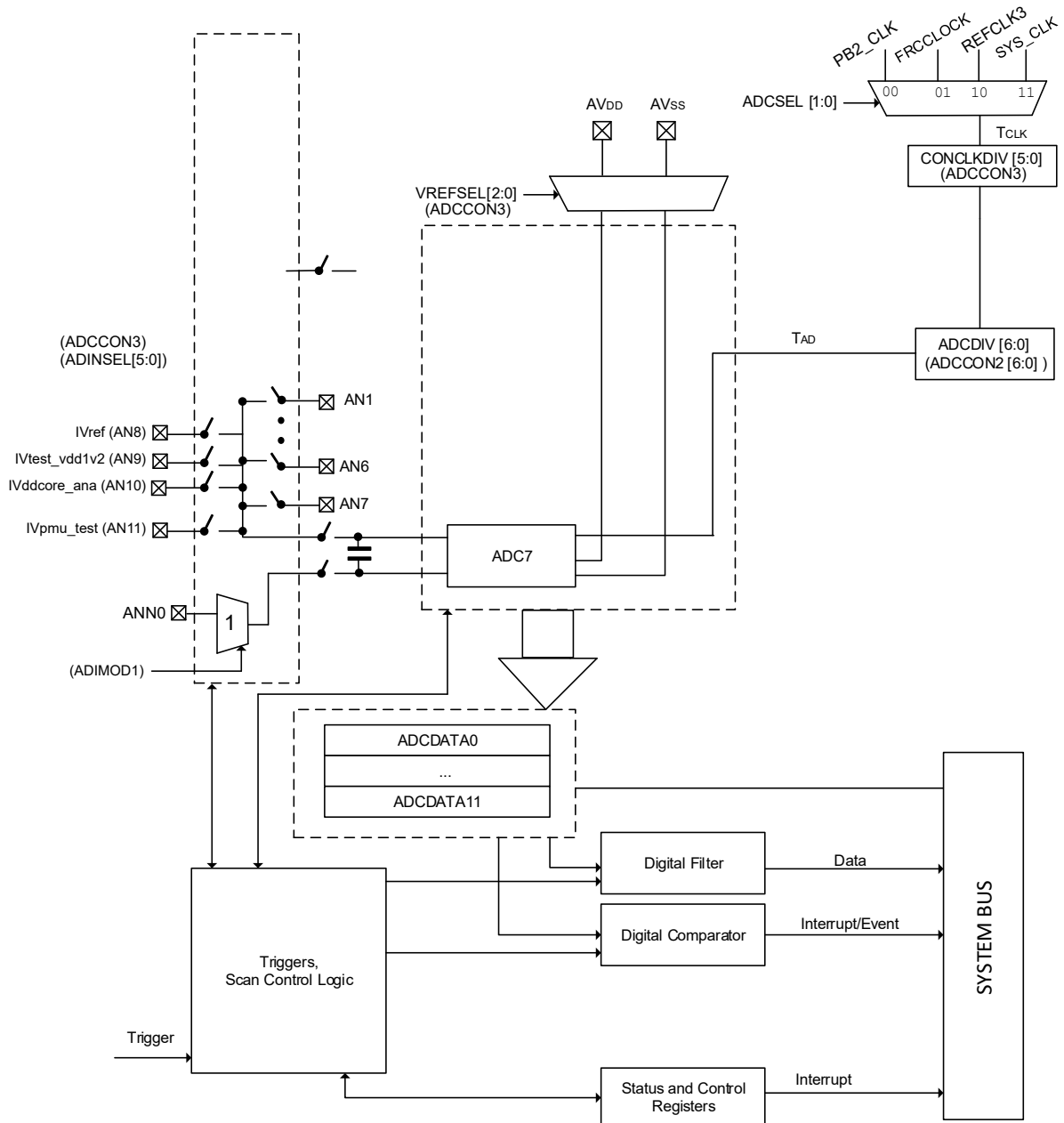
$$FTP = \frac{T_{AD}}{T_{SAMP} + T_{CONV}}$$

Where,

- $T_{AD}$  = The frequency of the individual ADC module.

A block diagram of the ADC module is illustrated in the following figure.

Figure 38-1. ADC Block Diagram



## 38.2 ADC Operation

The High Speed Successive Approximation Register (SAR) ADC is designed to support power conversion and motor control applications and consists of one shared ADC module. The shared ADC module has multiple analog inputs connected to its S&H circuit through a multiplexer. Multiple analog inputs share this ADC; therefore, it is termed the shared ADC module. The shared ADC module is used to measure analog signals of lower frequencies and signals that are static in nature (in other words, do not change significantly with time). However, this ADC module is capable of up to 2 Msps sample rate.

The analog inputs connected to the shared ADC module are Class 2 and Class 3 inputs. The number of inputs designated for each class depends on the specific device. For the PIC32CX-BZ2, the following arrangement is provided.

- Class 2 = AN0 to AN5
- Class 3 = AN6 to AN7

The property of each class of analog input is described in the following table.

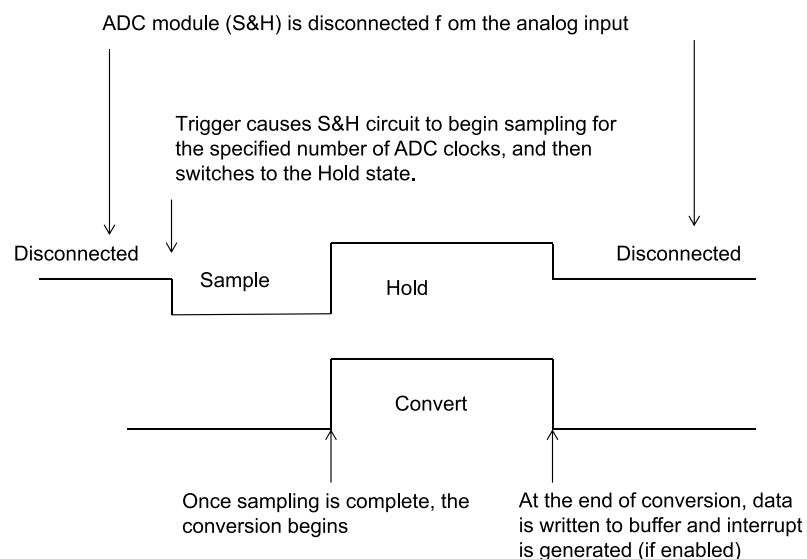
**Table 38-1.** Analog Input Class

ADC Module	Analog Input Class	Trigger	Trigger Action
Shared ADC module	Class 2	Individual trigger source or scan trigger	Starts sampling sequence or begins scan sequence
Shared ADC module with input scan	Class 3	Scan trigger	Starts scan sequence

Class 2 and Class 3 analog input properties:

- Class 2 inputs are used on the shared ADC module, either individually triggered or as part of a scan list. When used individually, they are triggered by their unique trigger selected by the ADCTRGx register.
- The analog inputs on the shared ADC have a natural order of priority (for example, AN6 has a higher priority than AN7).
- Class 3 inputs are used exclusively for scanning and share a common trigger source (scan trigger).
- Class 3 analog inputs share both the ADC module and the trigger source; therefore, the only method possible to convert them is to scan them sequentially for each incoming scan trigger event, where scanning occurs in the natural order of priority.
- The arrival of a trigger in the shared ADC module only starts the sampling. When the trigger arrives, the ADC module goes into sampling mode for the sampling time decided by the SAMC[9:0] bits (ADCCON2[25:16]). At the end of sampling, the ADC starts conversion. Upon completion of conversion, the ADC module is used to convert the next in line Class 2 or Class 3 inputs according to the natural order of priority. When a shared analog input (Class 2 or Class 3) has completed all conversion and no trigger is pending, the ADC module is disconnected from all analog inputs

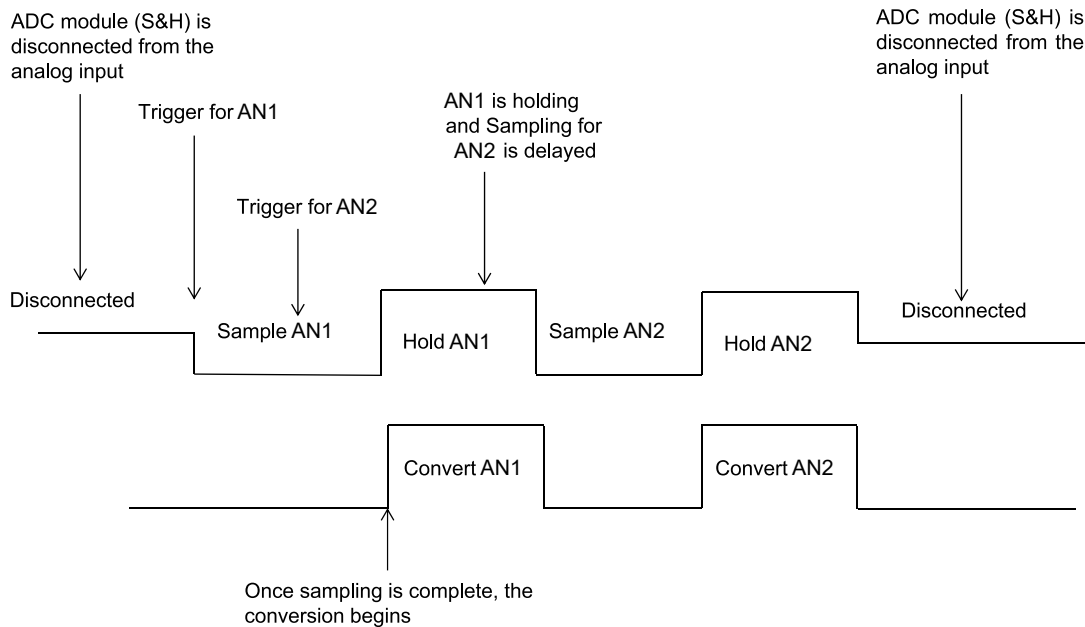
**Figure 38-2.** Sample and Conversion Sequence for Shared ADC Modules



### 38.2.1 Class 2 Triggering

When a single Class 2 input is triggered, it is sampled and converted by the shared S&H using the sequence illustrated in Sample and Conversion Sequence for the Shared ADC Modules figure; see *Sample and Conversion Sequence for Shared ADC Modules* figure in the *ADC Operation* from Related Links. When multiple Class 2 inputs are triggered, it is important to understand the consequences of trigger timing. If a conversion is underway and another Class 2 trigger occurs, then the sample-hold-conversion for the new trigger is stalled until the in-process, sample-hold cycle is complete, as shown in the following figure.

**Figure 38-3.** Multiple Independent Class 2 Trigger Conversion Sequence



When multiple inputs to the shared S&H are triggered simultaneously, the processing order is determined by their natural priority (the lowest numbered input has the highest priority). As an example, if AN1, AN2 and AN3 are triggered simultaneously, AN1 is sampled and converted first, followed by AN2 and finally, AN3. When using the independent Class 2 triggering on the shared S&H, the SAMC[9:0] bits (ADCCON2[25:16]) determine the sample time for all inputs while the appropriate TRGSRC[4:0] bits in the ADCTRGx Register (see *ADCTRG1* register from Related Links) determine the trigger source for each input.

#### Related Links

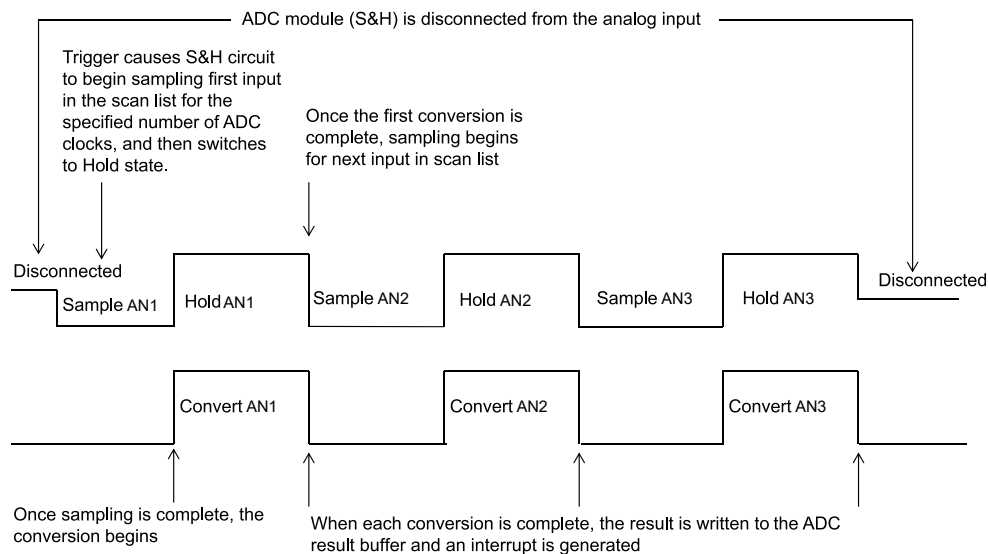
- [38.11.15. ADCTRG1](#)
- [38.2. ADC Operation](#)

### 38.2.2 Input Scan

Input scanning is a feature that allows an automated scanning sequence of multiple Class 2 or Class 3 inputs. All Class 2 and Class 3 inputs are scanned using the single shared S&H. The selection of analog inputs for scanning is done with the CSSx bits of the ADCCSS1 registers. Class 2 inputs are triggered using STRIG selection in the ADCTRGx register, and Class 3 inputs are triggered using the TRGSRC[4:0] of the ADCCON1[20:16] register. When a trigger occurs for Class 2 or Class 3 inputs, the sampling and conversion occur in the natural input order is used; lower number inputs are sampled before higher number inputs.



**Figure 38-4.** Input Scan Conversion Sequence for Three Class 2 Inputs

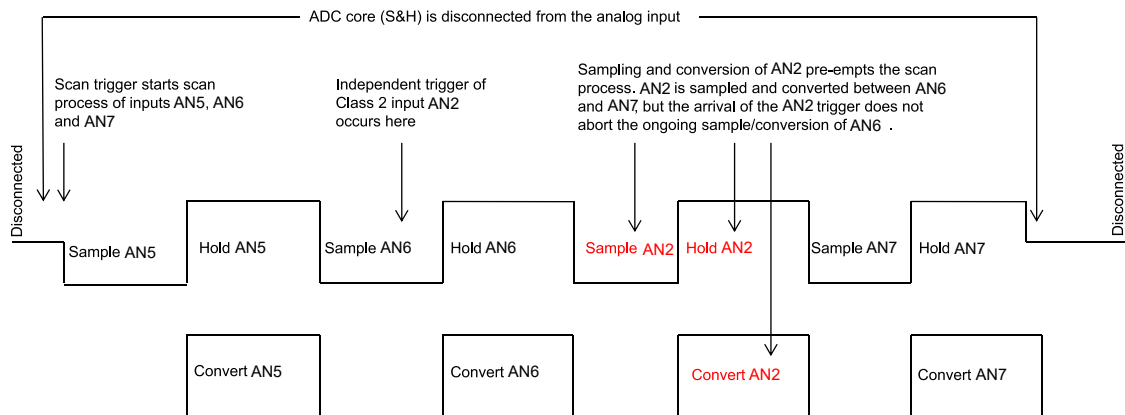


When using the shared analog inputs in scan mode, the SAMC[9:0] bits in the ADC Control Register 2 (ADCCON2[25:16]) determine the sample time for all inputs, while the Scan Trigger Source Selection bits (STRGSRC[4:0]) in the ADC Control Register 1 (ADCCON1[20:16]) determine the trigger source.

To ensure predictable results, a scan must not be retriggered until a sampling of all inputs is complete. Ensure system design to preclude retriggering a scan while a scan is in progress.

Individual Class 2 triggers that occur during a scan preempts the scan sequence if they are a higher priority than the sample currently being processed. In the following figure, a scan of AN5, AN6 and AN7 is underway when an independent trigger of Class 2 input AN2 takes place. The scan is interrupted for the sampling and conversion of AN2.

**Figure 38-5.** Scan Conversion Pre-empted by Class 2 Input Trigger



### 38.3 ADC Module Configuration

Operation of the ADC module is directed through bit settings in the specific registers. The following instructions summarize the actions and the settings. The options and details for each configuration step are provided in the subsequent sections.

To configure the ADC module, perform the following steps:

1. Configure the analog port pins as described in [38.3.1. Configuring the Analog Port Pins](#).
2. Select the analog inputs to the ADC multiplexers as described in [38.3.2. Selecting the ADC Multiplexer Analog Inputs](#).
3. Select the format of the ADC result as described in [38.3.3. Selecting the Format of the ADC Result](#).
4. Select the conversion trigger source as described in [38.3.4. Selecting the Conversion Trigger Source](#).
5. Select the voltage reference source as described in [38.3.5. Selecting the Voltage Reference Source](#).
6. Select the scanned inputs as described in [38.3.6. Selecting the Scanned Inputs](#).
7. Select the analog-to-digital conversion clock source and prescaler as described in [38.3.7. Selecting the Analog-to-Digital Conversion Clock Source and Prescaler](#).
8. Specify any additional acquisition time (if required) as described in [38.9. ADC Sampling Requirements](#).
9. Turn on the ADC module as described in [38.3.8. Turning ON the ADC](#).
10. Poll (or wait for the interrupt) for the voltage reference to be ready as described in [38.3.5. Selecting the Voltage Reference Source](#).
11. Enable the analog and bias circuit for the required ADC modules, and, after the ADC module wakes up, enable the digital circuit as described in [38.6.3. Low-Power Mode](#).
12. Configure the ADC interrupts (if required) as described in [38.5. Interrupts](#).

### 38.3.1 Configuring the Analog Port Pins

The ANSELx registers for the I/O ports associated with the analog inputs are used to configure the corresponding pin as an analog or a digital pin. A pin is configured as an analog input when the corresponding ANSELx bit = '1'. When the ANSELx bit = '0', the pin is set to digital control. The ANSELx registers are set when the device comes out of Reset, causing the ADC input pins to be configured as analog inputs by default.

The TRISx registers control the digital function of the port pins. The port pins that are required as analog inputs must have their corresponding bit set in the specific TRISx register, configuring the pin as an input. If the I/O pin associated with an ADC input is configured as an output by clearing the TRISx bit, the port's digital output level ( $V_{OH}$  or  $V_{OL}$ ) is converted. After a device Reset, all of the TRISx bits are set. For more information on port pin configuration, see *I/O Ports and Peripheral Pin Select (PPS)* from Related Links.

**Note:** When reading a PORT register that shares pins with the ADC, any pin configured as an analog input reads as '0' when the PORT latch is read. Analog levels on any pin that is defined as a digital input but not configured as an analog input, may cause the input buffer to consume the current that exceeds the device specification.

#### Related Links

[6. I/O Ports and Peripheral Pin Select \(PPS\)](#)

### 38.3.2 Selecting the ADC Multiplexer Analog Inputs

The ADC module has two inputs, referred to as the positive and negative inputs. Input selection options vary as described in the following sections.

#### 38.3.2.1 Selection of Positive Inputs

For the shared ADC module, the positive input is shared among all Class 2 and Class 3 inputs. Input connection of the analog input ANx to the shared ADC is automatic for either the Class 2 input trigger or during a scan of Class 2 and or Class 3 inputs. Selecting inputs for scanning is described in *Selecting the Scanned Inputs* from Related Links.

## Related Links

### 38.3.6. Selecting the Scanned Inputs

#### 38.3.2.2 Selection of Negative Inputs

Negative input selection is determined by the setting of the DIFFx bit of the ADCIMCON1 register. The DIFFx bit allows the inputs to be rail-to-rail and either single-ended or differential. The SIGNx and DIFFx bits in the ADCIMCON1 register scale the internal ADC analog inputs and reference voltages and configure the digital result to align with the expected full-scale output range.

For the shared ADC module, the analog inputs have individual settings for the DIFFx bit. Therefore, the user has the ability to select certain inputs as single-ended and others as differential while being connected to the same shared ADC module. While sampling, the signal changes on-the-fly as single-ended or differential according to its corresponding DIFFx bit setting.

**Table 38-2.** Negative Input Selection

ADCIMCON1		Input Configuration	Input Voltage		Output
DIFFx	SIGNx				
1	1	Differential 2's complement	Minimum input	$V_{INP} - V_{INN} = -V_{REF}$	-2048
			Maximum input	$V_{INP} - V_{INN} = V_{REF}$	+2047
1	0	Differential unipolar	Minimum input	$V_{INP} - V_{INN} = -V_{REF}$	0
			Maximum input	$V_{INP} - V_{INN} = V_{REF}$	+4095
0	1	Single-ended 2's complement	Minimum input	$V_{INP} = V_{REF}$	-2048
			Maximum input	$V_{INP} - V_{INN} = V_{REF}$	+2047
0	0	Single-ended unipolar	Minimum input	$V_{INP} = V_{REF}$	0
			Maximum input	$V_{INP} - V_{INN} = V_{REF}$	+4095

#### Legend:

- $V_{INP}$  = Positive S&H input
- $V_{INN}$  = Negative S&H input
- $V_{REF} = V_{REFH} - V_{REFL}$

**Note:** For proper operation and to prevent device damage, input voltage levels must not exceed the limits listed in the Electrical Specifications.

#### 38.3.3 Selecting the Format of the ADC Result

The data in the ADC Result register can be read in any of the four supported data formats. The user can select from unsigned integer, signed integer, unsigned fractional or signed fractional. Integer data is right-justified and fractional data is left-justified.

- The integer or fractional data format selection is specified globally for all analog inputs using the Fractional Data Output Format bit, FRACT (ADCCON1[23]).
- The signed or unsigned data format selection can be independently specified for each individual analog input using the SIGNx bits in the ADCIMCONx registers

The following table provides how a result is formatted.

**Table 38-3.** ADC Result Format

FRACT	SIGNx	Description	32-bit Output Data Format			
0	0	Unsigned integer	0000	0000	0000	0000
			0000	dddd	dddd	dddd

.....continued

FRACT	SIGNx	Description	32-bit Output Data Format			
0	1	Signed integer	ssss	ssss	ssss	ssss
			ssss	sddd	dddd	dddd
1	0	Fractional	dddd	dddd	dddd	0000
			0000	0000	0000	0000
1	1	Signed fractional	sddd	dddd	dddd	dddd
			0000	0000	0000	0000

The following code is an example for ADC Class 2 configuration and fractional format.

```
int main(int argc, char** argv) {
    int result[3];

    /* Configure ADCCON1 */
    ADCCON1bits.FRACT = 1; // use Fractional output format ADCCON1bits.SELRES = 3; // ADC
    resolution is 12 bits ADCCON1bits.STRGSRC = 0; // No scan trigger.

    /* Configure ADCCON2 */
    ADCCON2bits.SAMC = 5; // ADC sampling time = 5 * TAD7
    ADCCON2bits.ADCDIV = 1; // ADC clock freq is half of control clock = TAD7

    /* Initialize warm up time register */ ADCCON3 = 0;
    ADCCON3bits.WKUPCLCNT = 5; // Wakeup exponent = 32 * TADx

    /* Clock setting */ ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0; // Select input clock source
    ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0; // Select AVDD and AVSS as reference source

    /* No selection for dedicated ADC modules, no presync trigger, not sync sampling */
    ADCTRGMODEbits = 0;

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN7 = 0; // unsigned data format ADCIMCON1bits.DIFF7 = 0; // Single ended
    mode ADCIMCON1bits.SIGN8 = 0; // unsigned data format ADCIMCON1bits.DIFF8 = 0; // Single
    ended mode ADCIMCON1bits.SIGN9 = 0; // unsigned data format ADCIMCON1bits.DIFF9 = 0; //
    Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0; // No interrupts are used
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0; // No scanning is used
    ADCCSS2 = 0;

    /* Configure ADCCMPCONx */
    ADCCMPCON1 = 0; // No digital comparators are used. Setting the ADCCMPCONx
    ADCCMPCON2 = 0; // register to '0' ensures that the comparator is disabled.
    ADCCMPCON3 = 0; // Other registers are "don't care".
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0; ADCCMPCON6 = 0;

    /* Configure ADCFLTRx */
    ADCFLTR1 = 0; // No oversampling filters are used. ADCFLTR2 = 0;
    ADCFLTR3 = 0; ADCFLTR4 = 0; ADCFLTR5 = 0; ADCFLTR6 = 0;
    /* Set up the trigger sources */
    ADCTRGSNSbits.LVL7 = 0; // Edge trigger ADCTRGSNSbits.LVL8 = 0; // Edge trigger
    ADCTRGSNSbits.LVL9 = 0; // Edge trigger
    ADC1TRG2bits.TRGSRC7 = 1; // Set AN7 to trigger from software
    ADC2TRG3bits.TRGSRC8 = 1; // Set AN8 to trigger from software
    ADC2TRG3bits.TRGSRC9 = 1; // Set AN9 to trigger from software
    /* Early interrupt */
    ADCEIEN1 = 0; // No early interrupt
    ADCEIEN2 = 0;

    /* Turn the ADC on */ ADCCON1bits.ON = 1;

    /* Wait for voltage reference to be stable */
```

```

while(!ADCCON2bits.BGVRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT); // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCANCONbits.ANEN7 = 1; // Enable the clock to analog bias

/* Wait for ADC to be ready */
while(!ADCANCONbits.WKRDY7); // Wait until ADC7 is ready

/* Enable the ADC module */ ADCCON3bits.DIGEN7 = 1; // Enable ADC7

while (1) {
/* Trigger a conversion */ ADCCON3bits.GSWTRG = 1;

/* Wait the conversions to complete */
while (ADCDSTAT1bits.ARDY7 == 0);
/* fetch the result */
result[0] = ADCDATA7;

while (ADCDSTAT1bits.ARDY8 == 0);
/* fetch the result */
result[1] = ADCDATA8;

while (ADCDSTAT1bits.ARDY9 == 0);
/* fetch the result */
result[2] = ADCDATA9;

/*
* Process results here
*
* Note 1: Loop time determines the sampling time since all inputs are Class 2.
* If the loop time happens is small and the next trigger happens before the
* completion of set sample time, the conversion will happen only after the
* sample time has elapsed.
*
* Note 2: Results are in fractional format
*/
}
return (1);
}

```

### 38.3.4 Selecting the Conversion Trigger Source

Class 2 inputs to the ADC module can be triggered for conversion either individually or as part of a scan sequence. Class 3 inputs can only be triggered as part of a scan sequence. Individual or scan triggers can originate from an on-board timer or output compare peripheral event, from external digital circuits connected to INT0, from external analog circuits connected to an analog comparator or through software by setting a trigger bit in an SFR.

**Note:** When conversion triggers for multiple Class 2 analog inputs occur simultaneously, they are prioritized according to a natural order priority scheme based on the analog input used. AN6 has the highest priority, AN7 has the next highest priority and so on.

#### 38.3.4.1 Trigger Selection Class 2 Inputs

For each one of the Class 2 inputs, the user application can independently specify a conversion trigger source. The individual trigger source for an analog input 'x' is specified by the TRGSRC[4:0] bits located in registers ADCTRG1 through ADCTRG3. For example, these trigger sources may include:

- **General Purpose (GP) Timers:** When a period match occurs for the 32-bit timer, Timer3/2 or Timer5/4, or the 16-bit Timer1, Timer3 or Timer5, a special ADC trigger event signal is generated by the timer. This feature does not exist for other timers. For more information, see *Timer/Counter (TC)* from Related Links.
- **Output Compare:** The Output Compare peripherals, OC1, OC3 and OC5, can be used to generate an ADC trigger, then the output transitions from a low to high state. For more information, see *Timer/Counter (TC)* from Related Links.

- **Comparators:** The analog Comparators can be used to generate an ADC trigger when the output transitions from a low state to a high state. For more information, see *Digital Comparator* from Related Links.
- **External INTO Pin Trigger:** In this mode, the ADC module starts a conversion on an active transition on the INTO pin. The INTO pin may be programmed for either a rising edge input or a falling edge input to trigger the conversion process.
- **Global Software Trigger:** The ADC module can be configured for manually triggering a conversion for all inputs that have selected this trigger option. The user can manually trigger a conversion by setting the Global Software Trigger bit, GSWTRG (ADCCON3[6]).

#### Related Links

[38.4.1. Digital Comparator](#)

[40. Timer/Counter \(TC\)](#)

### 38.3.4.2 Conversion Trigger Sources and Control

The following are the possible sources for each trigger signal:

- External trigger selection through the TRGSRCx[4:0] bits in the ADCTRGx registers. This capability is supported only for Class 2 analog inputs. Typically, the user specifies a particular trigger source to initiate a conversion for specific input. All of the analog inputs may select the same trigger source if desired. In such an event, the result resembles a “scanned conversion”, which has its order of completion enforced by the priority of the inputs associated with the same trigger source. The first trigger selection is 00000 (no trigger), which amounts to temporarily disabling that particular trigger and, consequently, temporarily disabling that analog input from being converted. The next two selections for trigger source (GSWTRG and GLSWTRG) are software-generated trigger sources. The second software-generated trigger selection is the Global Software Trigger (GSWTRG). This trigger links to the GSWTRG bit in the ADCCON3 register, which may be used to enable the user application to initiate a single conversion. GSWTRG is a self-clearing bit; therefore, it clears itself on the next ADC clock cycle after being set by the user application. The third software-generated trigger selection is the Global Level Software Trigger (GLSWTRG), which is linked to the GLSWTRG bit in the ADCCON3 register. This trigger may be used by the user application to initiate a burst of consecutive samples as the GLSWTRG bit is not self-clearing. The fourth trigger selection is a special selection, the Scan Trigger selection, which allows the Class 2 analog inputs to be included as members of a global scan of all inputs.
- Scanned trigger selection via the STRGSRC[4:0] bits in the ADCCON1 register and select bits in the ADCCSS1 registers. This mode is typically used to initiate the conversion of a group of analog inputs. This capability works for 2 and 3 analog inputs but is typically used for Class 3 inputs because they do not have individual associated TRGSRC bits. One of the trigger selections is the GSWTRG bit in the ADCCON3 register, which may be used to enable the user software to initiate a conversion.
- User initiated trigger via the ADINSEL[5:0] bits and the RQCNVRT bit in the ADCCON3 register. This mode enables the user application to create an individual conversion trigger request for a specified analog input. Using this mode enables the user application to trigger the conversion of an input without changing the trigger source configuration of the ADC. This is useful in handling error situations where another software module wants ADC information without disrupting the normal operation of the ADC. This is also the preferred method to generate the initial trigger to start a digital filter sequence.
- User-controlled sampling of Class 2 and Class 3 inputs via the ADINSEL[5:0] bits and the SAMP bit in the ADCCON3 register. Setting the SAMP bit causes the Class 2 and Class 3 inputs to be in Sampling mode while ignoring the selection of the SAMC[9:0] bits. This mode is also useful in software conversion of ADC with software-selectable sample time.
- External module (such as PTG) may specify an analog input for conversion via the setting of the ECIEN bit in the ADCCON2 register. This method operates independently of the normal TRGSRC

and STRGSRC methods. External modules may still use individual trigger signals and initiate conversions via the normal TRGSRC and STRGSRC methods.

### 38.3.4.3 User-Requested Individual Conversion Trigger (Software ADC Conversion) (Only for Class 2 and Class 3 Inputs)

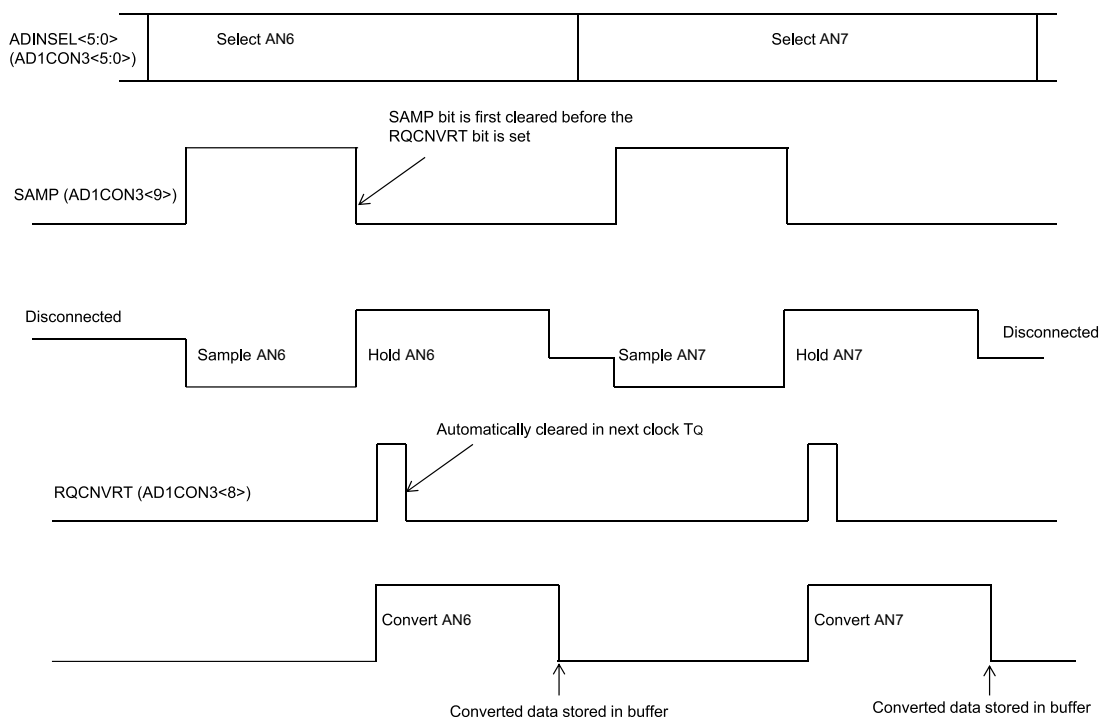
The user can explicitly request a single conversion (by software) of any selected analog input at any time during program execution without changing the trigger source configuration of the ADC.

The steps to be followed for conversion are as follows:

1. The analog input ID to be converted is specified by the ADC Input Select bits, ADINSEL[5:0] (ADCCON3[5:0]).
2. The sampling of analog input is started by setting the SAMP bit (ADCCON3[9]).
3. After the required sampling time (time delay), the SAMP bit is cleared.
4. The conversion of sampled signal is started by setting the RQCNVRT bit (ADCCON3[8]).
5. Once the conversion is complete, the ARDYx bit of the ADCSTATx register is set. The data can be read from the ADCDATAx register.

The following figure illustrates the conversion process in graphical form.

**Figure 38-6.** Individual Conversion Trigger Process



### 38.3.5 Selecting the Voltage Reference Source

The user application can select the voltage reference for the ADC module, which can be internal or external. The Voltage Reference Input Selection bits, VREFSEL[2:0] (ADCCON3[15:13]), select the voltage reference for analog-to-digital conversions. The upper voltage reference ( $V_{REFH}$ ) and the lower voltage reference ( $V_{REFL}$ ) may be the internal  $AV_{DD}$  and  $AV_{SS}$  voltage rails or the band gap reference generator or the external  $V_{REFH+}$  and  $V_{REF-}$  input pins. When the voltage reference and band gap reference are ready, the BGVRDY (ADCCON2[31]) bit is set. If a Fault occurs in the voltage



reference (such as a brown-out), the REFFLT bit (ADCCON2[30]) is set. The BGVRDY and REFFLT bits can also generate interrupts if the BGVRIEN bit (ADCCON2[15]) and REFFLTIE bit (ADCCON2[14]) are set, respectively.

The voltages applied to the external reference pins must comply with certain specifications. See *Electrical Characteristics* from Related Links.

The Analog Input Charge Pump Enable bit, AICMPEN (ADCCON1[12]), must be set when the difference between the selected reference voltages ( $V_{REFH} - V_{REFL}$ ) is less than  $0.65 * (AV_{DD} - AV_{SS})$ . Setting this bit does not increase the magnitude of the reference voltage; however, setting this bit reduces the series source resistance to the sampling capacitors. This maximizes the SNR for analog-to-digital conversions using small reference voltage rails.

### Related Links

[43. Electrical Characteristics](#)

## 38.3.6 Selecting the Scanned Inputs

All available analog inputs can be configured for scanning. Class 2 and Class 3 inputs are sampled using the shared ADC module. A single conversion trigger source is selected for all of the inputs selected for scanning using the TRGSRC[4:0] bits (ADCCON1[20:16]). On each conversion trigger, the ADC module starts converting (in the natural priority) all inputs specified in the user-specified scan list (ADCCSS1 or ADCCSS2). For Class 2 and Class 3 inputs, the trigger initiates a sequential sample/conversion process in the natural priority order.

An analog input belongs to the scan if it is:

- A Class 3 input. For Class 3 inputs, scan is the only mechanism for conversion.
- A Class 2 input that has the scan trigger selected as the trigger source by selecting the STRIG option in the TRGSRCx[4:0] bits located in the ADCTRG1 through ADCTRG8 registers.

The trigger options available for scan are identical to those available for independent triggering of Class 2 inputs. Any Class 2 inputs that are part of the scan must have the STRIG option selected as their trigger source in the TRGSRCx[4:0] bits.

**Note:** The end-of-scan (EOS) is generated only if the last shared input conversion has completed. Until this condition is met, the scan sequence is still in effect. Therefore, the EOS Interrupt can be used for any scan sequence with any combination of input types.

The following code is an example for ADC scanning multiple inputs.

```
int main(int argc, char** argv) {
    int result[3];

    /* Configure ADCCON1 */
    ADCCON1 = 0; // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
    // CVD mode, Fractional mode and scan trigger source. ADCCON1bits.SELRES = 3; // ADC7
    resolution is 12 bits
    ADCCON1bits.TRGSRC = 1; // Select scan trigger.

    /* Configure ADCCON2 */
    ADCCON2bits.SAMC = 5; // ADC7 sampling time = 5 * TAD7
    ADCCON2bits.ADCDIV = 1; // ADC7 clock freq is half of control clock = TAD7

    /* Initialize warm up time register */ ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3bits.ADCSEL = 0; // Select input clock source
    ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0; // Select AVDD and AVSS as reference source

    ADC0TIMEbits.ADCDIV = 1; // ADC0 clock frequency is half of control clock = TAD0
    ADC0TIMEbits.SAMC = 5; // ADC0 sampling time = 5 * TAD0
    ADC0TIMEbits.SELRES = 3; // ADC0 resolution is 12 bits

    /* Select analog input for ADC modules, no presync trigger, not sync sampling */
```



```

ADCTRGMODEbits.SH0ALT = 0;    // ADC0 = AN0

/* Select ADC input mode */
ADCIMCON1bits.SIGN0 = 0;      // unsigned data format ADCIMCON1bits.DIFF0 = 0;    //
Single ended mode ADCIMCON1bits.SIGN8 = 0;    // unsigned data format ADCIMCON1bits.DIFF8
= 0;    // Single ended mode ADCIMCON1bits.SIGN40 = 0;    // unsigned data format
ADCIMCON1bits.DIFF40 = 0;    // Single ended mode

/* Configure ADCGIRQENx */
ADCGIRQEN1 = 0;              // No interrupts are used. ADCGIRQEN2 = 0;

/* Configure ADCCSSx */
ADCCSS1 = 0;                 // Clear all bits
ADCCSS2 = 0;
ADCCSS1bits.CSS0 = 1;        // AN0 (Class 1) set for scan ADCCSS1bits.CSS8 = 1;    //
AN8 (Class 2) set for scan ADCCSS2bits.CSS40 = 1;    // AN40 (Class 3) set for scan

/* Configure ADCCMPCONx */
ADCCMPCON1 = 0;              // No digital comparators are used. Setting the ADCCMPCONx
ADCCMPCON2 = 0;              // register to '0' ensures that the comparator is disabled.
ADCCMPCON3 = 0;              // Other registers are 'don't care'.
ADCCMPCON4 = 0;
ADCCMPCON5 = 0; ADCCMPCON6 = 0;

/* Configure ADCFLTRx */
ADCFLTR1 = 0;                // No oversampling filters are used. ADCFLTR2 = 0;
ADCFLTR3 = 0; ADCFLTR4 = 0; ADCFLTR5 = 0; ADCFLTR6 = 0;
/* Set up the trigger sources */
ADCTRGL1bits.TRGSRC0 = 3;    // Set AN0 (Class 1) to trigger from scan source
ADCTR3bits.TRGSRC8 = 3;     // Set AN8 (Class 2) to trigger from scan source
// AN40 (Class 3) always uses scan trigger source

/* Early interrupt */
ADCEIEN1 = 0;                // No early interrupt
ADCEIEN2 = 0;

/* Turn the ADC on */ ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT);  // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCANCONbits.ANEN0 = 1;     // Enable the clock to analog bias ADC0
ADCANCONbits.ANEN7 = 1;     // Enable, ADC7

/* Wait for ADC to be ready */
while(!ADCANCONbits.WKRDY0); // Wait until ADC0 is ready while(!
ADCANCONbits.WKRDY7);      // Wait until ADC7 is ready

/* Enable the ADC module */
ADCCON3bits.DIGEN0 = 1;     // Enable ADC0
ADCCON3bits.DIGEN7 = 1;     // Enable ADC7

while (1) {
/* Trigger a conversion */ ADCCON3bits.GSWTRG = 1;

/* Wait the conversions to complete */
while (ADCDSTAT1bits.ARDY0 == 0);
/* fetch the result */
result[0] = ADCDATA0;

while (ADCDSTAT1bits.ARDY8 == 0);
/* fetch the result */
result[1] = ADCDATA8;

while (ADCDSTAT2bits.ARDY40 == 0);
/* fetch the result */
result[2] = ADCDATA40;

/*
* Process results here
*
*
*/
}

```

```
return (1);
}
```

### 38.3.7 Selecting the Analog-to-Digital Conversion Clock Source and Prescaler

The ADC module can use the internal Fast RC (FRC) oscillator output, system clock (SYSCLK), reference clock (REFCLK3) or peripheral bus clock (PBCLK) as the conversion clock source ( $T_Q$ ). See *ADCCON3* register from Related Links.

When the ADCSEL[1:0] bits (ADCCON2[31:30]) = '01', the internal FRC oscillator is used as the ADC clock source. When using the internal FRC oscillator, the ADC module can continue to function in Sleep and Idle modes.

**Note:** It is recommended that applications that require precise timing of ADC acquisitions use SYSCLK as the clock source for the ADC.

For correct analog-to-digital conversions, the conversion clock limits must not be exceeded. Clock frequencies from 1 MHz to 28 MHz are supported by the ADC module.

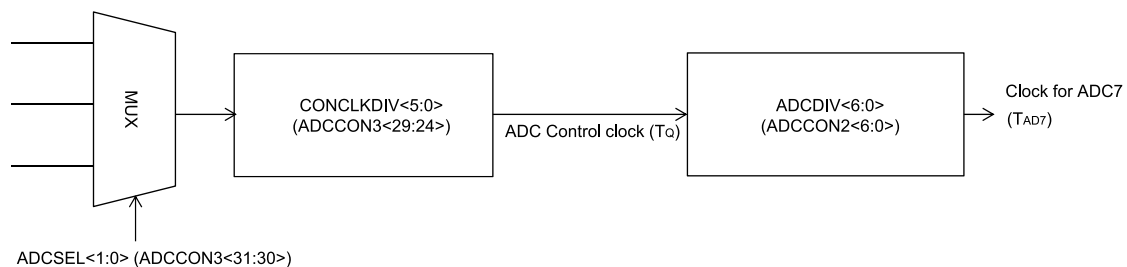
The maximum rate that analog-to-digital conversions may be completed by the ADC module (effective conversion throughput) is 2 Msps. However, the maximum rate that a single input can be converted is dependent on the sampling time requirements. In addition, the sampling time depends on the output impedance of the analog signal source. For more information on sampling time, see *ADC Sampling Requirements* from Related Links.

The input clock source for the ADC is selected using the ADCSEL[1:0] bits (ADCCON3[31:30]). The input clock is further divided by the control clock divider CONCLKDIV[5:0] bits (ADCCON3[29:24]). The output clock is called the "ADC control clock" with a time period of  $T_Q$ .

The ADC control clock is divided by the ADCDIV[6:0] bits (ADCxTIME[22:16]). This acts as the clock source for the respective dedicated ADC modules with a time period of  $T_{ADx}$ .

The ADC control clock is divided before it is used for the shared ADC by the ADCDIV[6:0] bits (ADCCON2[6:0]). The time period for this clock is denoted as  $T_{AD7}$ .

**Figure 38-7.** Clock Derivation for Shared ADC Modules



**Equation 38-2.** Sample Time for the Shared ADC Module

$$t_{SAMC} = ADCCON2 \langle 25:16 \rangle T_{AD}$$

$$t_{conversion} = 2 + ADCCON2 \langle 22:21 \rangle T_{AD}$$

#### Related Links

[38.11.4. ADCCON3](#)

[38.9. ADC Sampling Requirements](#)

### 38.3.8 Turning ON the ADC

Turning ON the ADC module involves the following procedure.

When the ADC module enable bit, ON (ADCCON1[15]), is set to '1', the module is in Active mode and is fully powered and functional. When the ON bit is '0', the ADC module is disabled. Once disabled, the digital and analog portions of the ADC are turned off for maximum current savings. In addition to setting the ON bit, the analog and digital circuits of ADC must be turned ON. See *Low-power Mode* from Related Links.

**Note:** Writing to the ADC control bits that control the ADC clock, input assignments, scanning, voltage reference selection, S&H circuit operating modes and interrupt configuration is not recommended while the ADC module is enabled.

#### Related Links

[38.6.3. Low-Power Mode](#)

### 38.3.9 ADC Status Bits

The ADC module includes the WKRDYx/WKRDY7 status bit in the ADCANCON register, which indicates the current state of ADC Analog and bias circuit. The user application must not perform any ADC operations until this bit is set.

## 38.4 Additional ADC Functions

This section describes some additional features of the ADC module, which includes:

- Digital comparator
- Oversampling filter

### 38.4.1 Digital Comparator

The ADC module features digital comparators that can be used to monitor selected analog input conversion results and generate interrupts when a conversion result is within the user-specified limits. Conversion triggers are still required to initiate conversions. The comparison occurs automatically once the conversion is complete. This feature is enabled by setting the Digital Comparator Module Enable bit, ENDCMP (ADCCMPCONx[7]).

The user application makes use of an interrupt that is generated when the analog-to-digital conversion result is higher or lower than the specified high and low limit values in the ADCCMPx register. The high and low limit values are specified in the DCMPhi[15:0] bits (ADCCMPx[31:16]) and the DCMPlO[15:0] bits (ADCCMPx[15:0]).

The CMPEx bits ('x' = 0 through 31) in the ADCCMPENx registers are used to specify which analog inputs are monitored by the digital comparator (for the first 8 analog inputs, ANx, where 'x' = 0 through 31). The ADCCMPCONx register specifies the comparison conditions that generates an interrupt, as follows:

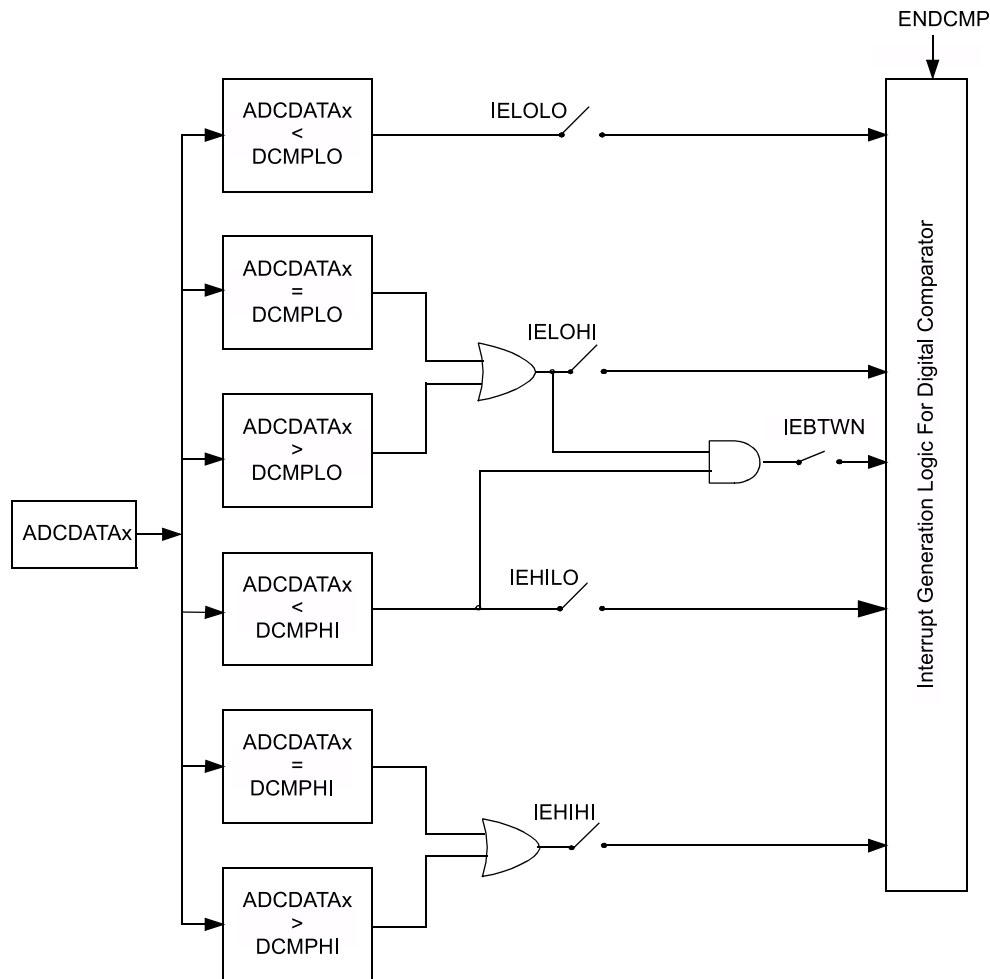
- When IEBTWN = 1, an interrupt is generated when  $DCMPLO \leq ADCDATA < DCMPhi$
- When IEHIHI = 1, an interrupt is generated when  $DCMPhi \leq ADCDATA$
- When IEHILO = 1, an interrupt is generated when  $ADCDATA < DCMPhi$
- When IELOHI = 1, an interrupt is generated when  $DCMPLO \leq ADCDATA$
- When IELOLO = 1, an interrupt is generated when  $ADCDATA < DCMPlO$

The comparator event generation is illustrated in the following figure. When the ADC module generates a conversion result, the conversion result is provided to the comparator. The comparator uses the DIFFx and SIGNx bits of the ADCIMCONx register (depending on the analog input used) to determine the data format used and to appropriately select whether the comparison must be signed or unsigned. The global ADC setting, which is specified by the FRACT bit (ADCCON1[23]), is also used to set the fractional or integer format. The digital comparator compares the ADC result with the high and low limit values (depending on the selected comparison criteria) in the ADCCMPx register.

Depending on the comparator results, a digital comparator interrupt event may be generated. If a comparator event occurs, the Digital Comparator Interrupt Event Detected status bit, DCMPEL (ADCCMPCONx[5]), is set, and the Analog Input Identification (ID) bits, AINID[4:0] (ADCCMPCONx[12:8]), are automatically updated so that the user application knows which analog input generated the interrupt event.

**Note:** The user software must format the values contained in the ADCCMPx registers to match converted data format as either signed or unsigned, and fractional or integer.

**Figure 38-8.** Digital Comparator



The following code is an example for ADC digital comparator.

```
int main(int argc, char** argv) {
    int result = 0, eventFlag = 0;

    /* Configure ADCCON1 */
    ADCCON1 = 0; // No ADCCON1 features are enabled including: Stop-in-Idle,
    // turbo, CVD mode, Fractional mode and scan trigger source. ADCCON1bits.SELRES = 3; //
    ADC resolution is 12 bits
    ADCCON1bits.STRGSRC = 0; // No scan trigger.

    /* Configure ADCCON2 */
    ADCCON2bits.SAMC = 5; // ADC7 sampling time = 5 * TAD7
    ADCCON2bits.ADCDIV = 1; // ADC7 clock freq = TAD7

    /* Initialize warm up time register */ ADCANCON = 0;
}
```

```

/* No selection for dedicated ADC modules, no presync trigger, not sync sampling */
ADCTRGMODEbits = 0;

/* Select ADC input mode */
ADCIMCON1bits.SIGN8 = 0; // unsigned data format
ADCIMCON1bits.DIFF8 = 0; // Single ended mode

/* Configure ADCCIRQENx */
ADCCIRQEN1 = 0; // No interrupts are used
ADCCIRQEN2 = 0;

/* Configure ADCCSSx */
ADCCSS1 = 0; // No scanning is used
ADCCSS2 = 0;

/* Configure ADCCMPCONx */
ADCCMP1 = 0; // Clear the register ADCCMP1bits.DCMPHI = 0xC00; // High
limit is a 3072 result. ADCCMP1bits.DCMPLO = 0x500; // Low limit is a 1280 result.
ADCCMPCON1bits.IEBTWN = 1; // Create an event when the measured result is
// >= low limits and < high limit. ADCCMPEN1 = 0; // Clear all enable bits
ADCCMPEN1bits.CMPE8 = 1; // set the bit corresponding to AN8
ADCCMPCON1bits.ENDCMP = 1; // enable comparator
ADCCMPCON2 = 0; ADCCMPCON3 = 0; ADCCMPCON4 = 0; ADCCMPCON5 = 0; ADCCMPCON6 = 0;

/* Configure ADCFLTRx */
ADCFLTR1 = 0; // No oversampling filters are used. ADCFLTR2 = 0;
ADCFLTR3 = 0; ADCFLTR4 = 0; ADCFLTR5 = 0; ADCFLTR6 = 0;
/* Set up the trigger sources */
ADCTR3bits.TRGSRC8 = 3; // Set AN8 (Class 2) to trigger from scan source

/* Early interrupt */
ADCEIEN1 = 0; // No early interrupt
ADCEIEN2 = 0;

/* Turn the ADC on */ ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT); // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCANCONbits.ANEN7 = 1; // Enable the clock to analog bias

/* Wait for ADC to be ready */
while(!ADCANCONbits.WKRDY7); // Wait until ADC7 is ready

/* Enable the ADC module */
ADCCON3bits.DIGEN7 = 1; // Enable ADC7

while (1) {
/* Trigger a conversion */ ADCCON3bits.GSWTRG = 1;

while (ADCDSTAT1bits.ARDY8 == 0);
/* fetch the result */
result = ADCDATA8;

/* Note: It is not necessary to fetch the result for the digital
* comparator to work. In this example we are triggering from
* software so we are using the ARDY8 to gate our loop. Reading the
* data clears the ARDY bit.
*/
/* See if we have a comparator event*/
if (ADCCMPCON1bits.DCMPED == 1) {
eventFlag = 1;
/*
* Process results here
*/
}
}
}

```

```
return (1);
}
```

### 38.4.2 Oversampling Digital Filter

The ADC module supports two oversampling digital filters. The oversampling digital filter consists of an accumulator and a decimator (down-sampler), which function together as a low-pass filter. By sampling an analog input at a higher-than-required sample rate, then processing the data through the oversampling digital filter, the effective resolution of the ADC module can be increased at the expense of decreased conversion throughput.

To obtain 'x' bits of extra resolution, the number of samples required (over and above the Nyquist rate) =  $(2^x)^2$ :

- 4x oversampling yields one extra bit of resolution (total 13 bits resolution)
- 16x oversampling yields two extra bits of resolution (total 14 bits resolution)
- 64x oversampling provides three extra bits of resolution (total 15 bits resolution)
- 256x oversampling provides four extra bits of resolution (total 16 bits resolution)

The digital filter also has an averaging mode, where it accumulates the samples and divides it by the number of samples.

**Note:**

1. Only Class 2 analog inputs can engage the digital filter. Therefore, the CHNLID[2:0] bits are 3 bits wide (0 to 7).
2. During the burst conversion process (repeated trigger until all required data for oversampling is obtained), in the case of filtering Class 2 input using the shared ADC module, higher priority ADC inputs may still process conversions; lower priority ADC conversion requests are held waiting until the filter burst sequence is completed.
3. If higher priority requests occur during the digital filter sequence, they delay the completion of the filtering process. This delay may affect the accuracy of the result because the multiple samples cannot be contiguous. The user must arrange the initiation trigger for the oversampling filters to occur while there are no expected interruptions from higher priority ADC conversion requests.

The user application must configure the following bits to perform an oversampling conversion:

- Select the amount of oversampling through the Oversampling Filter Oversampling Ratio (OVSAM[2:0]) bits in the ADC Filter register (ADCFLTRx[28:26]).
- Set the filter mode to either Oversampling mode or Averaging mode using the DFMODE bit(ADCFLTRx[29]).
- If the filter is set to Averaging mode and the data format is set to fractional (FRACT bit), set or clear the DATA16EN bit (ADCFLTRx[30]) to set the output resolution.
- Set the sample time for subsequent samples:
  - If using Class 2 inputs, select the sample time using the SAMC[9:0] bits (ADC- CON2[25:16]).
- Select the specific analog input to be oversampled by configuring the Analog Input ID Selection bits, CHNLID[4:0] (ADCFLTRx[20:16]).
- If needed, include the oversampling filter interrupt event in the global ADC interrupt by setting the Accumulator Filter Global Interrupt Enable bit, AFGIEN (ADCFLTRx[25]).
- Enable the oversampling filter by setting the Oversampling Filter Accumulator Enable bit, AFEN (ADCFLTRx[31]).

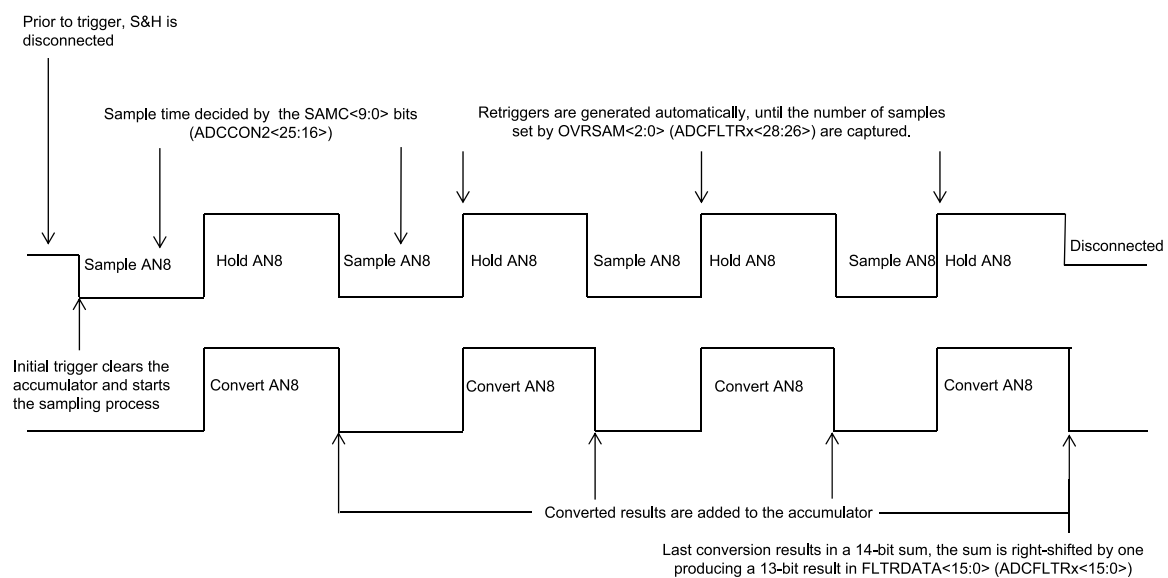
When the digital filter module is configured, the filter's control logic waits for an external trigger to initiate the process. The trigger signal for the analog input to be oversampled causes the accumulator to be cleared and initiates the first conversion. The trigger also forces the trigger sensitivity into level mode and forces the trigger itself to 1 as long as the filter needs to acquire

the user-specified number of samples via the OVRSAM[2:0] bits (ADCFLTRx[28:26]). The time delay between each acquired sample is decided by the set sample time in the SAMC[9:0] bits in the ADCCON2 register for Class 2 and the time for conversion. When the required number set by OVRSAM[2:0] are received and processed, the data stored in the FLTRDATA[15:0] bit (ADCFLTRx[15:0]) and the AFRDY bit (ADCFLTRx[24]) is set and the interrupt is generated (if enabled).

The following figure illustrates 4x oversampling using a Class 2 input. Triggering a Class 2 input initiates sampling for the length of time defined by the SAMC[9:0] bits. Retriggers generated by the oversampling logic use the SAMC[9:0] bits to set the sample time.

Class 2 inputs use the shared S&H; therefore, oversampling blocks lower priority Class 2 and Class 3 triggers. Higher priority Class 2 triggers completely disrupt the oversampling process; therefore, they must be avoided completely. The same priority rule applies to two Class 2 inputs that use two digital filters. In such a case, the higher priority input also uses the shared ADC module in Burst mode and prevents the lower priority input from using the shared ADC. Only after all required samples are obtained by the higher priority input can the lower priority input use the shared ADC to acquire samples for its own digital filtering.

**Figure 38-9.** 4x Oversampling of a Class 2 Input



The following code is an example for ADC digital oversampling filter.

```
int main(int argc, char** argv) {
    int result;

    /* Configure ADCCON1 */
    ADCCON1 = 0; // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
    // CVD mode, Fractional mode and scan trigger source.

    /* Configure ADCCON2 */
    ADCCON2 = 0; // Since, we are using only the Class 1 inputs, no setting is
    // required for ADCDIV

    /* Initialize warm up time register */ ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5; // Wake-up exponent = 32 * TADx

    /* Clock setting */ ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0; // Select input clock source
    ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0; // Select AVDD and AVSS as reference source
}
```

```

ADCOTIMEbits.ADCDIV = 1; // ADC0 clock frequency is half of control clock = TAD0
ADCOTIMEbits.SAMC = 5; // ADC0 sampling time = 5 * TAD0
ADCOTIMEbits.SELRES = 3; // ADC0 resolution is 12 bits

/* Select analog input for ADC modules, no presync trigger, not sync sampling */
ADCTRGMODEbits.SHOALT = 0; // ADC0 = AN0

/* Select ADC input mode */
ADCIMCONbits.SIGN0 = 0; // unsigned data format
ADCIMCONbits.DIFF0 = 0; // Single ended mode

/* Configure ADCGIRQENx */
ADCGIRQEN1 = 0; // No interrupts are used
ADCGIRQEN2 = 0;

/* Configure ADCCSSx */
ADCCSS1 = 0; // No scanning is used
ADCCSS2 = 0;
/* Configure ADCCMPCONx */
ADCCMPCON1 = 0; // No digital comparators are used. Setting the ADCCMPCONx
ADCCMPCON2 = 0; // register to '0' ensures that the comparator is disabled.
ADCCMPCON3 = 0; // Other registers are 'don't care'.
ADCCMPCON4 = 0; ADCCMPCON5 = 0; ADCCMPCON6 = 0;

/* Configure ADCFLTRx */
ADCFLTR1 = 0; // Clear all bits ADCFLTR1bits.CHNLID = 0; // Use AN0 as
the source ADCFLTR1bits.OVRSAM = 3; // 16x oversampling ADCFLTR1bits.DFMODE = 0; //
Oversampling mode ADCFLTR1bits.AFEN = 1; // Enable filter 1
ADCFLTR2 = 0; // Clear all bits
ADCFLTR3 = 0; ADCFLTR4 = 0; ADCFLTR5 = 0; ADCFLTR6 = 0;

/* Set up the trigger sources */ ADCTGNSNSbits.LVL0 = 0; // Edge trigger
ADCTRGLbits.TRGSR0 = 1; // Set AN0 to trigger from software.

/* Turn the ADC on */ ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT); // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCANCONbits.ANEN0 = 1; // Enable the clock to analog bias and digital control

/* Wait for ADC to be ready */
while(!ADCANCONbits.WKRDY0); // Wait until ADC0 is ready

/* Enable the ADC module */ ADCCON3bits.DIGEN0 = 1; // Enable ADC0

while (1) {
/* Trigger a conversion */ ADCCON3bits.GSWTRG = 1;

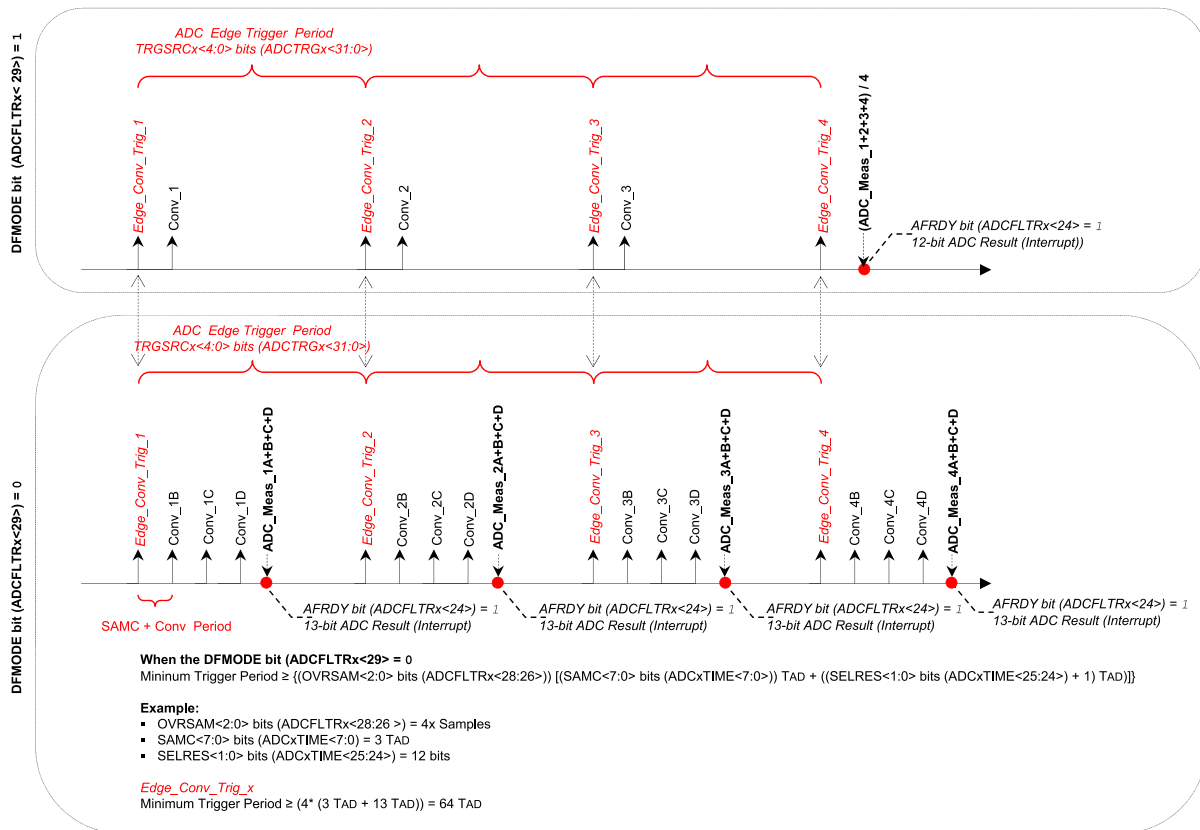
/* Wait for the oversampling process to complete */
while (ADCFLTR1bits.AFRDY == 0);
/* fetch the result */
result = ADCFLTR1bits.FLTRDATA;

/*
* Process result Here
*
* Note 1: Loop time determines the sampling time for the first sample.
* remaining samples sample time is determined by set sampling + conversion time.
*
* Note 2: The first 5 samples may have reduced accuracy.
*/
}
return (1);
}

```



Figure 38-10. ADC Filter Comparisons Example



## 38.5 Interrupts

The ADC module supports interrupts triggered from a variety of sources that can be processed individually or globally. An early interrupt feature is also available to compensate for interrupt servicing latency.

After an enabled interrupt is generated, the CPU jumps to the vector assigned to that interrupt. The CPU begins executing code at the vector address. The user software at this vector address must perform the required operations, such as processing the data results, clearing the interrupt flag, then exiting. See *Nested Vector Interrupt Controller (NVIC)* from Related Links for more information on interrupts and the vector address table details.

### Related Links

[10.2. Nested Vector Interrupt Controller \(NVIC\)](#)

### 38.5.1 Interrupt Sources

The ADC is capable of generating interrupts from the events listed in the following table.

Table 38-4. ADC Interrupt Sources

Interrupt Event	Description	Interrupt Enable Bit	Interrupt Status Bit
ANx Data Ready Event	Interrupt is generated upon a completion of a conversion from an analog input source (ANx). Each of the ARDYx bits is capable of generating a unique interrupt when set using the ADCBASE register.	AGIENx of ADCGIRQEN1	ARDYx of ADCDSTAT1 register

.....continued

Interrupt Event	Description	Interrupt Enable Bit	Interrupt Status Bit
Digital Comparator Event	When an conversion's comparison criteria are met by a configured and enabled digital comparator. Each of the digital comparators is capable of generating a unique interrupt when its DCMPE bit is set.	DCMPGIEN of ADCCMPCONx register	DCMPED of ADCCMPCONx register
Oversampling Filter Data Ready Event	When an oversampling filter has completed the accumulation/decimation process and has stored the result.	AFGIEN of ADCFLTRx register	AFRDY of ADCFLTRx register
Both Band Gap Voltage and ADC Reference Voltage Ready Event	Interrupt is generated when both band gap voltage and ADC reference voltage are ready.	BGVRIEN of ADCCON2 register	BGVRDY of ADCCON2 register
Band Gap Fault/ Reference Voltage Fault/ AV <sub>DD</sub> Brown-out Fault Event	Interrupt is generated when Band Gap Fault/ Reference Voltage Fault/AV <sub>DD</sub> Brown-out occurs.	REFFLTEN of ADCCON2 register	REFFLT of ADCCON2 register
ADC Module Wake-up Event	Interrupt is generated when ADC wakes up after being enabled.	WKIEN0 of ADCANCON register	WKRDY0 of ADCANCON register
Update Ready Event	Interrupt is generated when ADC SFRs are ready to be (and can be safely) updated with new values.	UPDIEN of ADCCON3 register	UPDRDY of ADCCON3 register

### 38.5.2 ADC Base Register (ADCBASE) Usage

After conversion of ADC is complete, if the interrupt is vectored to a function that is common to all analog inputs, it takes some significant time to find the ADC input by evaluating the ARDYx bits in the ADCDSTATx. To avoid this time spent, the ADCBASE register is provided, which contains the base address of the user's ADC ISR jump table. When read, the ADCBASE register provides a sum of the contents of the ADCBASE register plus an encoding of the ARDYx bits set in the ADCDSTATx registers. This use of the ADCBASE register supports the creation of an interrupt vector address that can be used to improve the performance of an ISR.

The ARDYx bits are binary priority encoded with ARDY1 being the highest priority and ARDY8 being the lowest priority. The encoded priority result is, then, shifted left the amount specified by the IRQVS[2:0] bits in the ADCCON1 register, then added to the contents of the ADCBASE register. If there are no ARDYx bits set, then reading the ADCBASE register equals the value written into the ADCBASE register.

The ADCBASE register is typically loaded with the base address of a jump table that contains the address of the appropriate ISR. The k<sup>th</sup> interrupt request is enabled via the AGIENx bit (1-8) in one of ADCGIRQENx SFRs ('x' = 1 or 2).

The following codes are examples for the ADCBASE register usage.

#### Case 1:

```
ADCBASE = 0x1234; // Set the address
ADCCON1bits.IRQVS = 2; // left shift by 2
ADCGIRQEN1bits.AGIEN0 = 1; // enable interrupt when AN0 completion is done.
```

When the ADC conversion for AN0 is complete, bit 0 of ADCDSTAT1 = ARDY0 is set.

Read value of ADCBASE = 0x1234 + (0 << 2) = 0x1234.

Therefore, the ISR must be placed at address 0x1234 for AN0.

**Case 2:**

```
ADCBASE = 0x1234; // Set the address
ADCCON1bits.IRQVS = 2; // left shift by 2
ADCGIRQEN1bits.AGIEN0 = 2; // enable interrupt when AN2 completion is done.
```

When the ADC conversion for AN2 is complete, bit 2 of ADCDSTAT1 = ARDY2 is set.

Read value of ADCBASE =  $0x1234 + (2 \ll 2) = 0x123C$ .

Therefore, the ISR must be placed at address  $0x123C$  for AN2.

**Note:** The contents of the ADCBASE register are not altered. Summation is performed when the ADCBASE register is read and the summation result is the returned read value from the ADCBASE SFR.

**38.5.3 Interrupt Enabling, Priority and Vectoring**

Each of the ADC events previously mentioned generates an interrupt when its associate Interrupt Enable bit, IE, is set. Each of the ADC events previously listed also has an associated interrupt vector. See *Nested Vector Interrupt Controller (NVIC)* from Related Links for more information on the vector location and control/status bits associated with each individual interrupt.

**Related Links**

[10.2. Nested Vector Interrupt Controller \(NVIC\)](#)

**38.5.4 Individual and Global Interrupts**

The use of the individual interrupts previously listed can significantly optimize the servicing of multiple ADC events by keeping each ISR focused on efficiently handling a specific event. In addition, different ISRs can be easily segregated according to the tasks performed, thereby making user software easier to implement and maintain. There may be cases where it is desirable to have a single ISR service multiple interrupt events. To facilitate this, each ADC event can be logically "ORed" to create a single global ADC interrupt. When an ADC event is enabled for a global interrupt, it vectors to a single interrupt routine. It is the responsibility of this single global ISR to determine the source of the interrupt through polling and process it accordingly.

Use of the Global Interrupt requires configuration of its own unique IE, IF, IP and IS bits as well as configuration of its interrupt vector as described in *Interrupt Enabling, Priority and Vectoring*. See *Interrupt Enabling, Priority and Vectoring* from Related Links.

Interrupts for the ADC can be configured as individual or global, or utilized as both where some are processed individually and others in the global ISR.

**Related Links**

[38.5.3. Interrupt Enabling, Priority and Vectoring](#)

**38.6 Power-Saving Modes of Operation**

The Power-Saving, Sleep and Idle modes are useful for reducing the conversion noise by minimizing the digital activity of the CPU, buses and other peripherals.

**38.6.1 Sleep Mode**

When the device enters Sleep mode, the system clock (SYCCLK) is halted. If an ADC module selects SYSCLK as its clock source or selects REFCLK3 as its clock source (REFCLK3 is generated from SYSCLK), the ADC enters the Sleep mode.

When the SYSCLK is the source (directly or indirectly) and Sleep mode occurs during a conversion, the conversion is aborted. The converter cannot resume a partially completed conversion on exiting from Sleep mode. The ADC register contents are not affected by the device entering or leaving Sleep mode. The ADC module can operate during Sleep mode if the ADC clock source is derived from a source other than SYSCLK that is active during Sleep mode. The FRC clock source is a logical choice

for operation during Sleep; however, the REFCLK3 clock source can also be used, provided it has an input clock that is operational during Sleep mode.

ADC operation during Sleep mode reduces the digital switching noise from the conversion. When the conversion is completed, the ARDYx status bit for that analog input is set and the result is loaded into the corresponding ADC Result register (ADCDATAx).

If any of the ADC interrupts are enabled, the device is woken up from Sleep mode when the ADC interrupt occurs. The program execution resumes at the ADC ISR if the ADC interrupt is greater than the current CPU priority. Otherwise, execution continues from the instruction after the WAIT instruction that placed the device in Sleep mode.

To minimize the effects of digital noise on the ADC module operation, the user must select a conversion trigger source that ensures that the analog-to-digital conversion take places in Sleep mode. For example, the external interrupt pin (INT0) conversion trigger option (TRGSRC[4:0] = 00100) can be used for performing sampling and conversion while the device is in Sleep mode.

**Note:** For the ADC module to operate in Sleep mode, the ADC clock source must be set to Internal FRC (ADCSEL[1:0] bits (ADCCON2[31:30]) = 01). Alternately, the REFCLK3 source can be used; however, the clock source used for REFCLK3 must operate during Sleep mode. Any changes to the ADC clock configuration require that the ADC be disabled.

### 38.6.2 Operation During Idle Mode

For the ADC, the stop in Idle Mode bit, SIDL (ADCCON1[13]), specifies whether the ADC module stops on Idle or continues on Idle. If SIDL = 0, the ADC module continues normal operation when the device enters Idle mode. If any of the ADC interrupts are enabled, the device wakes up from Idle mode when the ADC interrupt occurs. The program execution resumes at the ADC ISR if the ADC interrupt is greater than the current CPU priority. Otherwise, execution continues from the instruction after the WAIT instruction that placed the device in Idle mode.

If SIDL = 1, the ADC module stops in Idle mode. If the device enters Idle mode during a conversion, the conversion is aborted. The converter cannot resume a partially completed conversion on exiting from Idle mode.

### 38.6.3 Low-Power Mode

The ADC module can be placed in a low-power state by disabling the digital circuit for individual ADC modules that are not running. This is possible by clearing the DIGENx bits and the DIGEN7 bit in the ADCCON3 register. (See *ADCCON3* register from Related Links.)

An even lower power state is possible by disabling the analog and bias circuit for individual ADC modules that are not running. This is possible by clearing the ANENx bits and the ANEN7 bit in the ADCANCON register. (See *ADCANCON* register from Related Links.) Disabling the digital circuit to achieve Low-Power mode provides a significantly faster module restart compared to disabling and re-enabling the analog and bias circuit of the ADC module. This is because disabling and re-enabling the analog and bias circuit using the ANENx bits and the ANEN7 bit requires a wake-up time (typical minimum wake-up time of 20  $\mu$ s) for the ADC module before it can be used. Refer to the Electrical Specifications in the specific device data sheet for more information on the stabilization time.

When the analog and bias circuit for an ADC module is enabled, the wake-up must be polled (or through an interrupt) using the wake-up ready bits, WKRDY6:WKRDY0 and WKRDY7, which must be equal to '1'.

#### Related Links

[38.11.4. ADCCON3](#)

[38.11.24. ADCANCON](#)

[43. Electrical Characteristics](#)

### 38.7 Effects of Reset

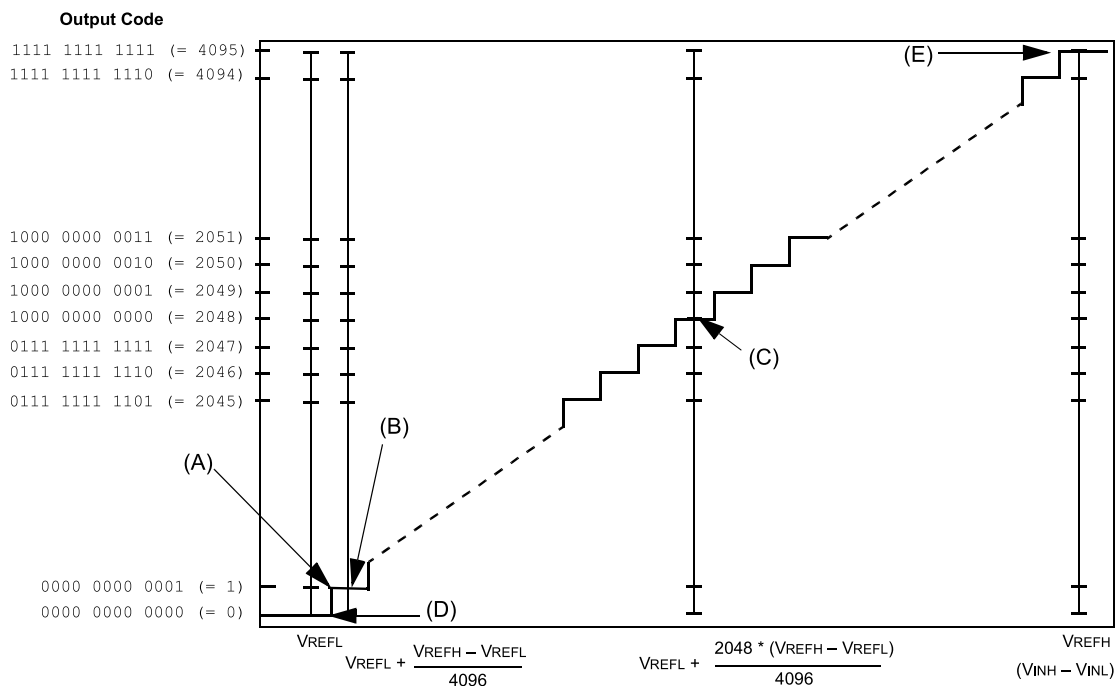
Following any Reset event, all the ADC control and status registers are reset to their default values with control bits in a non-active state. This disables the ADC module and sets the analog input pins to Analog Input mode. Any conversion that was in progress terminates, and the result cannot be written to the result buffer. The values in the ADCDATAx registers are initialized to 0x00000000 during a device Reset. The bias circuits are also turned off, so the ADC resuming operations wait for the bias circuits to stabilize by polling (or requesting to be interrupted by) the BGVRDY bit (ADCCON2 register).

### 38.8 Transfer Function

A typical transfer function of the 12-bit ADC is illustrated in the following figure. The difference of the input voltages ( $V_{INH} - V_{INL}$ ) is compared with the reference ( $V_{REFH} - V_{REFL}$ ).

- The first code transition (A) occurs when the input voltage is  $(V_{REFH} - V_{REFL}/8192)$  or 0.5 LSb.
- The 0000 0000 0001 code is centered at  $(V_{REFH} - V_{REFL}/4096)$  or 1.0 LSb (B).
- The 1000 0000 0000 code is centered at  $(2048 * (V_{REFH} - V_{REFL})/4096)$  (C).
- An input voltage less than  $(1 * (V_{REFH} - V_{REFL})/8192)$  converts as 0000 0000 0000 (D).
- An input greater than  $(8192 * (V_{REFH} - V_{REFL})/8192)$  converts as 1111 1111 1111 (E).

Figure 38-11. Analog-to-Digital Transfer Function



### 38.9 ADC Sampling Requirements

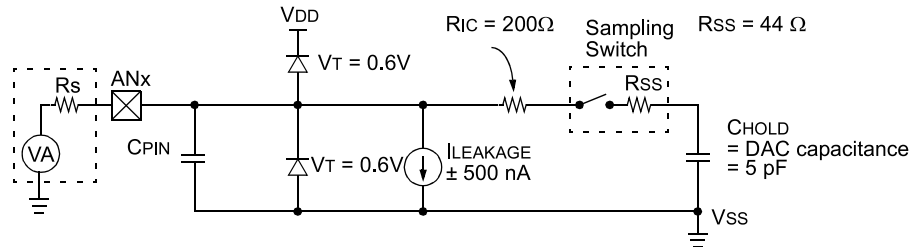
The analog input model of the 12-bit ADC is illustrated in the following figure. The total acquisition time for the analog-to-digital conversion is a function of the internal circuit settling time and the holding capacitor charge time.

For the ADC module to meet its specified accuracy, the charge holding capacitor ( $C_{HOLD}$ ) must be allowed to fully charge to the voltage level on the analog input pin. The analog output source impedance ( $R_S$ ), the interconnect impedance ( $R_{IC}$ ) and the internal sampling switch ( $R_{SS}$ ) impedance combine to directly affect the time required to charge the  $C_{HOLD}$ . The combined impedance of the analog sources must, therefore, be small enough to fully charge (to within one-fourth LSB of

the desired voltage) the holding capacitor within the selected sample time. The internal holding capacitor is in the discharged state prior to each sample operation.

At least 1  $T_{AD}$  time period must be allowed between conversions for the acquisition time. Refer to the *Electrical Characteristics* from the Related Links.

**Figure 38-12.** 12-bit ADC Analog Input Model



**Note:** The  $C_{PIN}$  value depends on the device package and is not tested. The effect of the  $C_{PIN}$  is negligible if  $R_S$  5 k.

**Legend:**

- $C_{PIN}$  = Input capacitance
- $R_{SS}$  = Sampling switch resistance
- $R_S$  = Source resistance
- $I_{LEAKAGE}$  = Leakage current at the pin due to various junctions
- $V_T$  = Threshold voltage
- $R_{IC}$  = Interconnect resistance
- $C_{HOLD}$  = Sample/hold capacitance

**Related Links**

[43. Electrical Characteristics](#)

**38.10 Connection Considerations**

Because the analog inputs employ Electrostatic Discharge (ESD) protection, they have diodes to  $V_{DD}$  and  $V_{SS}$ ; therefore, the analog input must be between  $V_{DD}$  and  $V_{SS}$ . If the input voltage exceeds this range by greater than 0.3V (either direction), one of the diodes becomes forward biased, and it may damage the device if the input current specification is exceeded.

An external RC filter is sometimes added for antialiasing of the input signal. The R (resistive) component must be selected to ensure that the acquisition time is met. Any external components connected (through high-impedance) to an analog input pin (capacitor, Zener diode and so on) must have very little leakage current at the pin.

### 38.11 Register Description

**Notes:** 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:** CLR/SET/INV registers for each register are located at offset <register offset> + 0x04, 0x08, 0x0C, respectively.

### 38.11.1 Register Summary

The PIC32CX-BZ2 12-bit High Speed SAR ADC module has the following Special Function Registers (SFRs):

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
0x00 ... 0x13FF	Reserved										
0x1400	ADCCON1	7:0	IRQVS[2:0]			STRGLVL		DMABL[2:0]			
		15:8	ON	FRZ	SIDL	FSYDMA		FSYUPB	SCANEN		
		23:16	FRACT	SELRES[1:0]			STRGSRC[4:0]				
		31:24									
0x1404 ... 0x140F	Reserved										
0x1410	ADCCON2	7:0	ADCDIV[6:0]								
		15:8	BGVRIEN	REFFLTEN	EOSIEN	ENXCNVRT					
		23:16	SAMC[7:0]								
		31:24	BGVRRDY	REFFLT	EOSRDY				SAMC[9:8]		
0x1414 ... 0x141F	Reserved										
0x1420	ADCCON3	7:0	GLSWTRG	GSWTRG	ADINSEL[5:0]						
		15:8	VREFSEL[2:0]			TRGSUSP	UPDIEN	UPDRDY	SAMP	RQCNVRT	
		23:16	CHN_EN_SHR								
		31:24	ADCSEL[1:0]			CONCLKDIV[5:0]					
0x1424 ... 0x143F	Reserved										
0x1440	ADCIMCON1	7:0	DIFF3	SIGN3	DIFF2	SIGN2	DIFF1	SIGN1	DIFF0	SIGN0	
		15:8	DIFF7	SIGN7	DIFF6	SIGN6	DIFF5	SIGN5	DIFF4	SIGN4	
		23:16	DIFF11	SIGN11	DIFF10	SIGN10	DIFF9	SIGN9	DIFF8	SIGN8	
		31:24									
0x1444 ... 0x147F	Reserved										
0x1480	ADCGIRQEN1	7:0	AGIEN7	AGIEN6	AGIEN5	AGIEN4	AGIEN3	AGIEN2	AGIEN1	AGIEN0	
		15:8				AGIEN11	AGIEN10	AGIEN9	AGIEN8		
		23:16									
		31:24									
0x1484 ... 0x149F	Reserved										
0x14A0	ADCCSS1	7:0	CSS7	CSS6	CSS5	CSS4	CSS3	CSS2	CSS1	CSS0	
		15:8				CSS11	CSS10	CSS9	CSS8		
		23:16									
		31:24									
0x14A4 ... 0x14BF	Reserved										
0x14C0	ADCDSTAT1	7:0	ARDY7	ARDY6	ARDY5	ARDY4	ARDY3	ARDY2	ARDY1	ARDY0	
		15:8				ARDY11	ARDY10	ARDY9	ARDY8		
		23:16									
		31:24									
0x14C4 ... 0x14DF	Reserved										
0x14E0	ADCCMPEN1	7:0	CMPEX[7:0]								
		15:8									
		23:16									
		31:24									



.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x14E4 ... 0x14EF	Reserved									
0x14F0	ADCCMP1	7:0	DCMPLO[7:0]							
		15:8	DCMPLO[15:8]							
		23:16	DCMPHI[7:0]							
		31:24	DCMPHI[15:8]							
0x14F4 ... 0x14FF	Reserved									
0x1500	ADCCMPEN2	7:0	CMPEX[7:0]							
		15:8								
		23:16								
		31:24								
0x1504 ... 0x150F	Reserved									
0x1510	ADCCMP2	7:0	DCMPLO[7:0]							
		15:8	DCMPLO[15:8]							
		23:16	DCMPHI[7:0]							
		31:24	DCMPHI[15:8]							
0x1514 ... 0x159F	Reserved									
0x15A0	ADCFLTR1	7:0	FLTRDATA[7:0]							
		15:8	FLTRDATA[15:8]							
		23:16	CHNLID[4:0]							
		31:24	AFEN	DATA16EN	DFMODE	OVRSAM[2:0]			AFGIEN	AFRDY
0x15A4 ... 0x15AF	Reserved									
0x15B0	ADCFLTR2	7:0	FLTRDATA[7:0]							
		15:8	FLTRDATA[15:8]							
		23:16	CHNLID[4:0]							
		31:24	AFEN	DATA16EN	DFMODE	OVRSAM[2:0]			AFGIEN	AFRDY
0x15B4 ... 0x15FF	Reserved									
0x1600	ADCTRG1	7:0	TRGSRC0[4:0]							
		15:8	TRGSRC1[4:0]							
		23:16	TRGSRC2[4:0]							
		31:24	TRGSRC3[4:0]							
0x1604 ... 0x160F	Reserved									
0x1610	ADCTRG2	7:0	TRGSRC4[4:0]							
		15:8	TRGSRC5[4:0]							
		23:16	TRGSRC6[4:0]							
		31:24	TRGSRC7[4:0]							
0x1614 ... 0x167F	Reserved									
0x1680	ADCCMPCON1	7:0	ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO
		15:8								
		23:16	AINID[5:0]							
		31:24								
0x1684 ... 0x168F	Reserved									

.....continued										
Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1690	ADCCMPCON2	7:0	ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO
		15:8				AINID[4:0]				
		23:16								
		31:24								
0x1694 ... 0x16FF	Reserved									
0x1700	ADCBASE	7:0	ADCBASE[7:0]							
		15:8	ADCBASE[15:8]							
		23:16								
		31:24								
0x1704 ... 0x170F	Reserved									
0x1710	ADCDMASTAT	7:0								RAFO
		15:8	DMACNTEN							RAFOIEN
		23:16	WROVRERR							RBF0
		31:24	DMAEN							RBF0IEN
0x1714 ... 0x171F	Reserved									
0x1720	ADCCNTB	7:0	ADCCNTB[7:0]							
		15:8	ADCCNTB[15:8]							
		23:16	ADCCNTB[23:16]							
		31:24	ADCCNTB[31:24]							
0x1724 ... 0x172F	Reserved									
0x1730	ADCDMAB	7:0	ADDMAB[7:0]							
		15:8	ADDMAB[15:8]							
		23:16	ADDMAB[23:16]							
		31:24	ADDMAB[31:24]							
0x1734 ... 0x173F	Reserved									
0x1740	ADCTRGSNS	7:0	LVL7	LVL6	LVL5	LVL4	LVL3	LVL2	LVL1	LVL0
		15:8								
		23:16								
		31:24								
0x1744 ... 0x17FF	Reserved									
0x1800	ADCANCON	7:0	ANEN7							ANENO
		15:8	WKRDY7							WKRDY0
		23:16	WKIEN7							WKIENO
		31:24						WKUPCLKCNT[3:0]		
0x1804 ... 0x1AFF	Reserved									
0x1B00	ADCSYSCFG0	7:0	AN[7:0]							
		15:8	AN[15:8]							
		23:16	AN[19:16]							
		31:24								
0x1B04 ... 0x1DFF	Reserved									
0x1E00	ADCDATAx	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							

### 38.11.2 ADCCON1 – ADC Control Register 1

**Name:** ADCCON1  
**Offset:** 0x1400  
**Reset:** 0x00601000  
**Property:** -

This register controls the basic operation of the ADC module, including behavior in Sleep and Idle modes, and data formatting. This register also specifies the vector shift amounts for the Interrupt Controller. Additional ADCCON1 functions include the RAM buffer length in DMA mode.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access	FRACT	SELRES[1:0]		STRGSRC[4:0]				
Reset	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	ON	FRZ	SIDL			FSYDMA	FSYUPB	SCANEN
Reset	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
Access		IRQVS[2:0]			STRGLVL	DMABL[2:0]		
Reset		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

#### Bit 23 – FRACT Fractional Data Output Format bit

Value	Description
0	Integer
1	Fractional

#### Bits 22:21 – SELRES[1:0] Shared ADC (ADC2) Resolution bits

**Note:** Changing the resolution of the ADC does not shift the result in the corresponding ADCDATAx register. The result occupies 12 bits, with the corresponding lower unused bits set to '0'. For example, a resolution of 6 bits results in ADCDATAx[5:0] being set to '0' and ADCDATAx[11:6] holding the result.

Value	Description
11	12 bits (default)
10	10 bits
01	8 bits
00	6 bits

#### Bits 20:16 – STRGSRC[4:0] ScanTrigger Source Select bits

Value	Description
10001 - 11111	Reserved
10000	EVSYS_47
01111	EVSYS_46
01110	EVSYS_45
01101	EVSYS_44
01100	EVSYS_43

Value	Description
01011	EVSYS_42
01010	EVSYS_41
01001	EVSYS_40
01000	EVSYS_39
00111	EVSYS_38
00110	EVSYS_37
00101	EVSYS_36
00100	INT0 External interrupt
00011	Reserved
00010	Global level software trigger (GLSWTRG)
00001	Global software edge trigger (GSWTRG)
00000	No Trigger

**Bit 15 – ON** ADC Module Enable bit

**Note:** The ON bit must be set only after the ADC module is configured.

Value	Description
0	ADC module is disabled
1	ADC module is enabled

**Bit 14 – FRZ** Freeze in Debug Mode

Value	Description
0	Do not freeze in Debug mode
1	Freeze in Debug mode

**Bit 13 – SIDL** Stop in Idle Mode bit

Value	Description
0	Continue module operation in Idle mode
1	Discontinue module operation when device enters Idle mode

**Bit 10 – FSYDMA** Fast Synchronous DMA System Clock bit

Value	Description
0	Fast synchronous DMA system clock is disabled
1	Fast synchronous DMA system clock is enabled

**Bit 9 – FSYUPB** Fast Synchronous UPB Clock bit

Value	Description
0	Fast synchronous UPB clock is disabled
1	Fast synchronous UPB clock is enabled

**Bit 8 – SCANEN** SCAN Enable bit

**Bits 6:4 – IRQVS[2:0]** Interrupt Vector Shift bits

To determine the interrupt vector address, this bit specifies the amount of left-shift done to the ARDY<sub>x</sub> status bits in the ADCDSTAT1 and ADCDSTAT2 registers prior to adding with the ADCBASE register.

Interrupt Vector Address = Read Value of ADCBASE, and Read Value of ADCBASE = Value written to ADCBASE + x << IRQVS[2:0], where 'x' is the smallest active input ID from the ADCDSTAT1 or ADCDSTAT2 registers (which has highest priority).

Value	Description
111	Shift x left 7 bit position
110	Shift x left 6 bit position
101	Shift x left 5 bit position
100	Shift x left 4 bit position
011	Shift x left 3 bit position
010	Shift x left 2 bit position

Value	Description
001	Shift x left 1 bit position
000	Shift x left 0 bit position

**Bit 3 – STRGLVL** ScanTrigger High Level/Positive Edge Sensitivity bit

Value	Description
0	Scan trigger is positive edge sensitive. Once STRIG mode is selected (TRGSRCx[4:0] in the ADCTRGr register), only a single scan trigger is generated, which completes the scan of all selected analog inputs.
1	Scan trigger is high level sensitive. Once STRIG mode is selected (TRGSRCx[4:0] in the ADCTRGr register), the scan trigger continues for all selected analog inputs, until the STRIG option is removed.

**Bits 2:0 – DMABL[2:0]** DMA to System RAM Buffer Length Size

Defines the number of locations in system memory allocated per analog input for DMA interface use. As each output data is 16-bit wide, one location consists of 2 bytes. Therefore, the actual size reserved in the system RAM follows the formula: RAM Buffer Length in bytes =  $2_{(DMABL+1)}$ .

### 38.11.3 ADCCON2 – ADC Control Register 2

**Name:** ADCCON2  
**Offset:** 0x1410  
**Reset:** 0x00000000  
**Property:** -

This register controls the reference selection for the ADC module, the sample time for the shared ADC module, interrupt enable for reference, early interrupt selection and clock division selection for the shared ADC.

Bit	31	30	29	28	27	26	25	24
	BGVRDY	REFFLT	EOSRDY				SAMC[9:8]	
Access	R/HS/HC	R/HS/HC	R/HS/HC				R/W	R/W
Reset	0	0	0				0	0
Bit	23	22	21	20	19	18	17	16
	SAMC[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
	BGVRIEN	REFFLTEN	EOSIEN		ENXCNVRT			
Access	R/W	R/W	R/W		R/W			
Reset	0	0	0		0			
Bit	7	6	5	4	3	2	1	0
		ADCDIV[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 31 – BGVRDY Band Gap Voltage/ADC Reference Voltage Status bit

Data processing is valid only after BGVRDY is set by hardware, so the application code must check that the BGVRDY bit is set to ensure data validity. This bit set to '0' when ON (ADCCON1[15]) = 0.

Value	Description
0	Either or both band gap voltage and ADC reference voltages ( $V_{REF}$ ) are not ready
1	Both band gap voltage and ADC reference voltages ( $V_{REF}$ ) are ready

#### Bit 30 – REFFLT Band Gap/ $V_{REF}$ / $A_{VDD}$ BOR Fault Status bit

This bit is cleared when the ON bit (ADCCON1[15]) = 0 and the BGVRDY bit = 1.

Value	Description
0	Band gap and $V_{REF}$ voltage are working properly
1	Fault in band gap or the $V_{REF}$ voltage while the ON bit (ADCCON1[15]) was set. Most likely a band gap or $V_{REF}$ fault is caused by a BOR of the analog $V_{DD}$ supply.

#### Bit 29 – EOSRDY End of Scan Interrupt Status bit

This bit is cleared when ADCCON2[31:24] are read in software.

Value	Description
0	Scanning has not completed
1	All analog inputs are considered for scanning through the scan trigger (all analog inputs specified in the ADCCSS1 and ADCCSS2 registers) have completed scanning

#### Bits 25:16 – SAMC[9:0] SampleTime for the Shared ADC (ADC2) bits

Where  $T_{AD7}$  = Period of the ADC conversion clock for the Shared ADC (ADC2) controlled by the ADCDIV[6:0] bits.

Value	Description
11111111 1	1025 $T_{AD7}$
...	
00000000 1	3 $T_{AD7}$
00000000 0	2 $T_{AD7}$

**Bit 15 – BGVRIEN** Band Gap/ $V_{REF}$  Voltage Ready Interrupt Enable bit

Value	Description
0	No interrupt is generated when the BGVRDY bit is set
1	Interrupt is generated when the BGVRDY bit is set

**Bit 14 – REFFLTEN** Band Gap/ $V_{REF}$  Voltage Fault Interrupt Enable bit

Value	Description
0	No interrupt is generated when the REFFLT bit is set
1	Interrupt is generated when the REFFLT bit is set

**Bit 13 – EOSIEN** End of Scan Interrupt Enable bit

Value	Description
0	No interrupt is generated when the EOSRDY bit is set
1	Interrupt is generated when the EOSRDY bit is set

**Bit 11 – ENXCNVRT** Enable External Conversion Request Interface

Setting this bit enables another module (such as the PTG) to specify and request conversion of an ADC input.

**Note:** The external module (such as the PTG) is responsible for asserting only the proper trigger signals. This ADC module has no method to block specific trigger requests from the external module.

**Bits 6:0 – ADCDIV[6:0]** Division Ratio for the Shared SAR ADC Core Clock bits

The ADCDIV[6:0] bits divide the ADC control clock ( $T_Q$ ) to generate the clock for the shared SAR ADC.

Value	Description
1111111	$254 * T_Q = T_{AD2}$
...	
0000011	$6 * T_Q = T_{AD2}$
0000010	$4 * T_Q = T_{AD2}$
0000001	$2 * T_Q = T_{AD2}$
0000000	Reserved

### 38.11.4 ADCCON3 – ADC Control Register 3

**Name:** ADCCON3  
**Offset:** 0x1420  
**Reset:** 0x00000000  
**Property:** -

This register enables ADC clock selection, enables/disables the digital feature for the shared ADC module and controls the manual (software) sampling and conversion.

Bit	31	30	29	28	27	26	25	24
	ADCSEL[1:0]		CONCLKDIV[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	23	22	21	20	19	18	17	16
	CHN_EN_SHR							
Access	R/W							
Reset	0							
Bit	15	14	13	12	11	10	9	8
	VREFSEL[2:0]			TRGSUSP	UPDIEN	UPDRDY	SAMP	RQCNVRT
Access	R/W	R/W	R/W	R/W	R/W	R/HS/HC	R/W	R/HS/HC
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	GLSWTRG	GSWTRG	ADINSEL[5:0]					
Access	R/W	R/W, HC	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

#### Bits 31:30 – ADCSEL[1:0] Analog-to-Digital Clock Source ( $T_{CLK}$ ) bits

Value	Description
00	Peripheral Bus Clock
01	FRC Clock
10	REFO3 Clock Output
11	System Clock (SYS_CLK)

#### Bits 29:24 – CONCLKDIV[5:0] Analog-to-Digital Control Clock ( $T_Q$ ) Divider bits

Value	Description
111111	$64 * T_{CLK} = T_Q$
...	
000011	$4 * T_{CLK} = T_Q$
000010	$3 * T_{CLK} = T_Q$
000001	$2 * T_{CLK} = T_Q$
000000	$T_{CLK} = T_Q$

#### Bit 23 – CHN\_EN\_SHR Shared ADC Digital Enable bit

Value	Description
1	ADC is digital enabled
0	ADC is digital disabled



### Bits 15:13 – VREFSEL[2:0] Voltage Reference ( $V_{REF}$ ) Input Selection bits

Table 38-5.

VREFSEL[2:0]	AD <sub>REF+</sub>	AD <sub>REF-</sub>
000	AV <sub>DD</sub>	AV <sub>SS</sub>
001–111	RESERVED FOR FUTURE USE	

### Bit 12 – TRGSUSP Trigger Suspend bit

Value	Description
1	Triggers are blocked from starting a new analog-to-digital conversion, but the ADC module is not disabled
0	Triggers are not blocked

### Bit 11 – UPDIEN Update Ready Interrupt Enable bit

Value	Description
1	Interrupt is generated when the UPDRDY bit is set by hardware
0	No interrupt is generated

### Bit 10 – UPDRDY ADC Update Ready Status bit

**Note:** This bit is only active while the TRGSUSP bit is set and there are no more running conversions of any ADC modules.

Value	Description
1	ADC SFRs can be updated
0	ADC SFRs cannot be updated

### Bit 9 – SAMP Class 2 and Class 3 Analog Input Sampling Enable bit<sup>(1,2,3,4)</sup>

Value	Description
1	The ADC S&H amplifier is sampling
0	The ADC S&H amplifier is holding

### Bit 8 – RQCNVRT Individual ADC Input Conversion Request bit

This bit and its associated ADINSEL[5:0] bits enable the user to individually request an analog-to-digital conversion of an analog input through software.

**Note:** This bit is automatically cleared in the next ADC clock cycle.

Value	Description
1	Trigger the conversion of the selected ADC input as specified by the ADINSEL[5:0] bits
0	Do not trigger the conversion

### Bit 7 – GLSWTRG Global Level Software Trigger bit

Value	Description
1	Trigger conversion for ADC inputs that have selected the GLSWTRG bit as the trigger signal, either through the associated TRGSRC[4:0] bits in the ADTRGx registers or through the STRGSRC[4:0]bits in the ADCCON1 register
0	Do not trigger an analog-to-digital conversion

### Bit 6 – GSWTRG Global Software Trigger bit

This bit is automatically cleared in the next ADC clock cycle.

Value	Description
0	Trigger conversion for ADC inputs that have selected the GSWTRG bit as the trigger signal, either through the associated TRGSRC[4:0] bits in the ADTRGx registers or through the STRGSRC[4:0]bits in the ADCCON1 register
1	Do not trigger an analog-to-digital conversion

### Bits 5:0 – ADINSEL[5:0] Analog Input Select bits

These bits select the analog input to be converted when the RQCNVRT bit is set.

**Note:**

1. The SAMP bit has the highest priority and setting this bit keeps the S&H circuit in Sample mode until the bit is cleared. Also, usage of the SAMP bit causes settings of SAMC[9:0] bits (ADCCON2[25:16]) to be ignored.
2. The SAMP bit only connects Class 2 and Class 3 analog inputs to the shared ADC.
3. The SAMP bit is not a self-clearing bit and it is the responsibility of application software to first clear this bit and, only after setting the RQCNVRT bit, to start the analog-to-digital conversion.
4. Normally, when the SAMP and RQCNVRT bits are used by software routines, all TRGSRCx[4:0] bits and STRGSRC[4:0] bits must be set to '00000' to disable all external hardware triggers and prevent them from interfering with the software-controlled sampling command signal SAMP and with the software-controlled trigger RQCNVRT.

Value	Description
111111	Reserved
...	
001011	PMU Test Output
001010	VddCore (Internal)
001001	CP_Test_1.2V (Internal)
001000	BandGap Reference (Internal)
000111	AN7 is being monitored
...	
000001	AN1 is being monitored
000000	AN0 is being monitored

### 38.11.5 ADCIMCON1 – ADC Input Mode Control Register 1

**Name:** ADCIMCON1  
**Offset:** 0x1440  
**Reset:** 0x00000000  
**Property:** -

This register enables the user to select between single-ended and differential operation as well as select between signed and unsigned data format.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access	DIFF11	SIGN11	DIFF10	SIGN10	DIFF9	SIGN9	DIFF8	SIGN8
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	DIFF7	SIGN7	DIFF6	SIGN6	DIFF5	SIGN5	DIFF4	SIGN4
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	DIFF3	SIGN3	DIFF2	SIGN2	DIFF1	SIGN1	DIFF0	SIGN0
Reset	0	0	0	0	0	0	0	0

#### Bit 23 – DIFF11 AN11 Mode bit

Value	Description
1	AN11 is using Differential mode
0	AN11 is using Single-ended mode

#### Bit 22 – SIGN11 AN11 Signed Data Mode bit

Value	Description
1	AN11 is using Signed Data mode
0	AN11 is using Unsigned Data mode

#### Bit 21 – DIFF10 AN10 Mode bit

Value	Description
1	AN10 is using Differential mode
0	AN10 is using Single-ended mode

#### Bit 20 – SIGN10 AN10 Signed Data Mode bit

Value	Description
1	AN10 is using Signed Data mode
0	AN10 is using Unsigned Data mode

#### Bit 19 – DIFF9 AN9 Mode bit

Value	Description
1	AN9 is using Differential mode
0	AN9 is using Single-ended mode

#### Bit 18 – SIGN9 AN9 Signed Data Mode bit

Value	Description
1	AN9 is using Signed Data mode
0	AN9 is using Unsigned Data mode

**Bit 17 – DIFF8 AN8 Mode bit**

Value	Description
1	AN8 is using Differential mode
0	AN8 is using Single-ended mode

**Bit 16 – SIGN8 AN8 Signed Data Mode bit**

Value	Description
1	AN8 is using Signed Data mode
0	AN8 is using Unsigned Data mode

**Bit 15 – DIFF7 AN7 Mode bit**

Value	Description
1	AN7 is using Differential mode
0	AN7 is using Single-ended mode

**Bit 14 – SIGN7 AN7 Signed Data Mode bit**

Value	Description
1	AN7 is using Signed Data mode
0	AN7 is using Unsigned Data mode

**Bit 13 – DIFF6 AN6 Mode bit**

Value	Description
1	AN6 is using Differential mode
0	AN6 is using Single-ended mode

**Bit 12 – SIGN6 AN6 Signed Data Mode bit**

Value	Description
1	AN6 is using Signed Data mode
0	AN6 is using Unsigned Data mode

**Bit 11 – DIFF5 AN5 Mode bit**

Value	Description
1	AN5 is using Differential mode
0	AN5 is using Single-ended mode

**Bit 10 – SIGN5 AN5 Signed Data Mode bit**

Value	Description
1	AN5 is using Signed Data mode
0	AN5 is using Unsigned Data mode

**Bit 9 – DIFF4 AN4 Mode bit**

Value	Description
1	AN4 is using Differential mode
0	AN4 is using Single-ended mode

**Bit 8 – SIGN4 AN4 Signed Data Mode bit**

Value	Description
1	AN4 is using Signed Data mode
0	AN4 is using Unsigned Data mode

**Bit 7 – DIFF3 AN3 Mode bit**

Value	Description
1	AN3 is using Differential mode

Value	Description
0	AN3 is using Single-ended mode

**Bit 6 – SIGN3** AN3 Signed Data Mode bit

Value	Description
1	AN3 is using Signed Data mode
0	AN3 is using Unsigned Data mode

**Bit 5 – DIFF2** AN2 Mode bit

Value	Description
1	AN2 is using Differential mode
0	AN2 is using Single-ended mode

**Bit 4 – SIGN2** AN2 Signed Data Mode bit

Value	Description
1	AN2 is using Signed Data mode
0	AN2 is using Unsigned Data mode

**Bit 3 – DIFF1** AN1 Mode bit

Value	Description
1	AN1 is using Differential mode
0	AN1 is using Single-ended mode

**Bit 2 – SIGN1** AN1 Signed Data Mode bit

Value	Description
1	AN1 is using Signed Data mode
0	AN1 is using Unsigned Data mode

**Bit 1 – DIFF0** AN0 Mode bit

Value	Description
1	AN0 is using Differential mode
0	AN0 is using Single-ended mode

**Bit 0 – SIGN0** AN0 Signed Data Mode bit

Value	Description
1	AN0 is using Signed Data mode
0	AN0 is using Unsigned Data mode

### 38.11.6 ADCGIRQEN1 – ADC Global Interrupt Enable Register 1

**Name:** ADCGIRQEN1  
**Offset:** 0x1480  
**Reset:** 0x00000000  
**Property:** -

This register specifies which of the individual input conversion interrupts can generate the global ADC interrupt.

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					AGIEN11	AGIEN10	AGIEN9	AGIEN8
Reset					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	AGIEN7	AGIEN6	AGIEN5	AGIEN4	AGIEN3	AGIEN2	AGIEN1	AGIEN0
Reset	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 – AGIEN ADC Global Interrupt Enable bits

Value	Description
1	Interrupts are enabled for the selected analog input. The interrupt is generated after the converted data is ready (indicated by the ARDYx bit ('x' = 8-1) of the ADCDSTAT1 register)
0	Interrupts are disabled

### 38.11.7 ADCCSS1 – ADC Common Scan Select Register 1

**Name:** ADCCSS1  
**Offset:** 0x14A0  
**Reset:** 0x00000000  
**Property:** -

This register specifies the analog inputs to be scanned by the common scan trigger.

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					CSS11	CSS10	CSS9	CSS8
Reset					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	CSS7	CSS6	CSS5	CSS4	CSS3	CSS2	CSS1	CSS0
Reset	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 – CSS Analog Common Scan Select bits

##### Notes:

1. In addition to setting the appropriate bits in this register, Class 2 analog inputs must select the STRIG input as the trigger source if they are to be scanned through the CSSx bits. Refer to the bit descriptions in the ADCTRGx registers for selecting the STRIG option.
2. If a Class 2 input is included in the scan by setting the CSSx bit to '1' and by setting the TRGSRCx[4:0] bits to STRIG mode (0b11), the user application must ensure that no other triggers are generated for that input using the RQCNVRT bit in the ADCCON3 register or the hardware input or any digital filter. Otherwise, the scan behavior is unpredictable.

Value	Description
1	Select ANx for input scan
0	Skip ANx for input scan

### 38.11.8 ADCDSTAT1 – ADC Data Ready Status Register 1

**Name:** ADCDSTAT1  
**Offset:** 0x14C0  
**Reset:** 0x00000000  
**Property:** -

This register contains the interrupt status of the individual analog input conversions. Each bit represents the data-ready status for its associated conversion result.

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					ARDY11	ARDY10	ARDY9	ARDY8
Reset					R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
					0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	ARDY7	ARDY6	ARDY5	ARDY4	ARDY3	ARDY2	ARDY1	ARDY0
Reset	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
	0	0	0	0	0	0	0	0

**Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 – ARDY** Conversion Data Ready for Corresponding Analog Input Ready bits

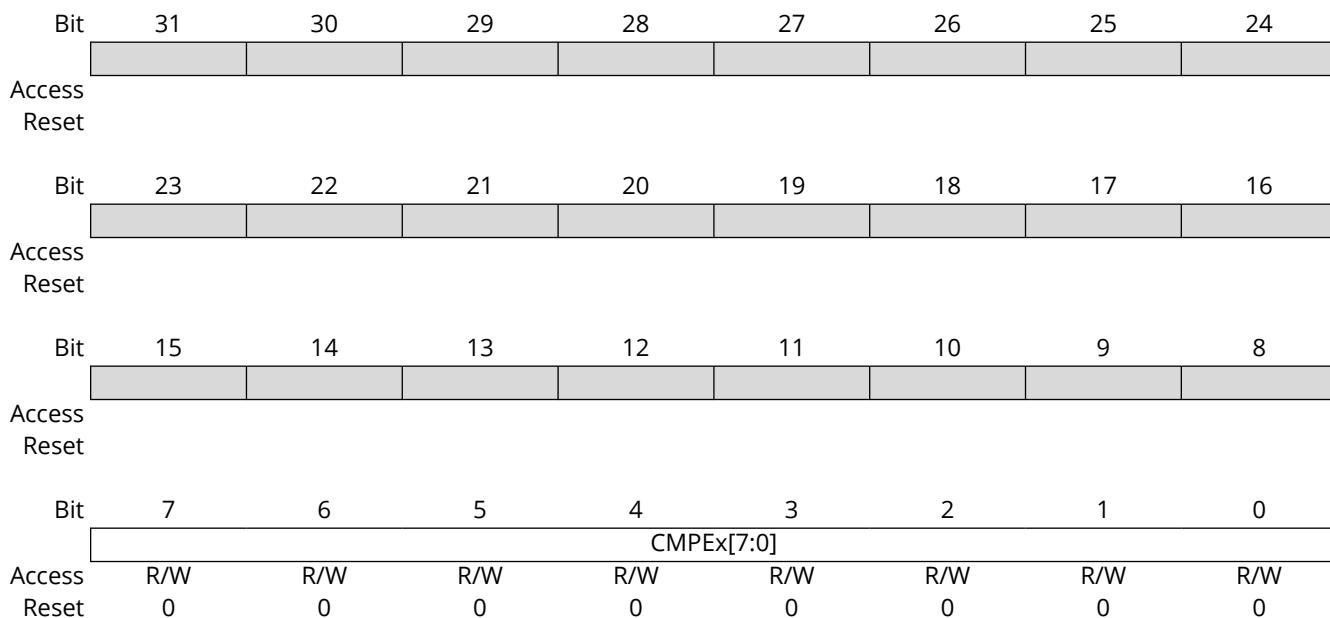
Value	Description
1	This bit is set when converted data is ready in the data register
0	This bit is cleared when the associated data register is read



### 38.11.9 ADCCMPEN1 – ADC Digital Comparator 1 Enable Register

**Name:** ADCCMPEN1  
**Offset:** 0x14E0  
**Reset:** 0x00000000  
**Property:** -

These registers select which analog input conversion results is processed by the digital comparator.



#### Bits 7:0 – CMPE<sub>x</sub>[7:0] ADC Digital Comparator 'x' Enable bits

**Note:** CMPE<sub>x</sub> = where "x" stands for bit value from 0 to 7.

These bits enable conversion results corresponding to the analog input to be processed by the digital comparator. CMPE0 enables AN0, CMPE1 enables AN1 and so on.

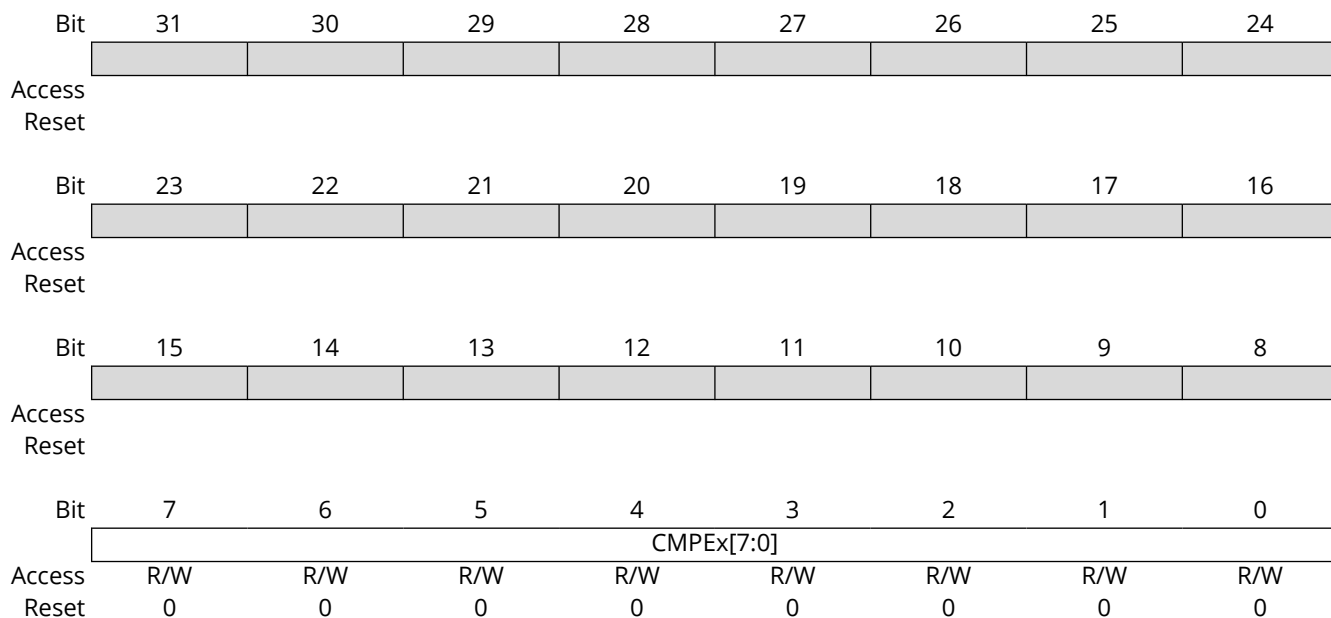
**Notes:**

1. CMPE<sub>x</sub> = AN<sub>x</sub>, where 'x' = 0-31 (Digital Comparator inputs are limited to AN0 through AN31).
2. Changing the bits in this register while the Digital Comparator is enabled (ENDCMP = 1) can result in unpredictable behavior.

### 38.11.10 ADCCMPEN2 – ADC Digital Comparator 2 Enable Register

**Name:** ADCCMPEN2  
**Offset:** 0x1500  
**Reset:** 0x00000000  
**Property:** -

These registers select which analog input conversion results is processed by the digital comparator.



#### Bits 7:0 – CMPE<sub>x</sub>[7:0] ADC Digital Comparator 'x' Enable bits

**Note:** CMPE<sub>x</sub> = where 'x' stands for bit value from 0 to 7.

These bits enable conversion results corresponding to the analog input to be processed by the digital comparator. CMPE<sub>0</sub> enables AN<sub>0</sub>, CMPE<sub>1</sub> enables AN<sub>1</sub> and so on.

**Notes:**

1. CMPE<sub>x</sub> = AN<sub>x</sub>, where 'x' = 0-31 (Digital Comparator inputs are limited to AN<sub>0</sub> through AN<sub>31</sub>).
2. Changing the bits in this register while the Digital Comparator is enabled (ENDCMP = 1) can result in unpredictable behavior.

### 38.11.11 ADCCMP1 – ADC Digital Comparator 1 Limit Value Register

**Name:** ADCCMP1  
**Offset:** 0x14F0  
**Reset:** 0x00000000  
**Property:** -

These registers contain the high and low digital comparison values for use by the digital comparator.

**Notes:**

1. Changing these bits while the Digital Comparator is enabled (ENDCMP = 1) can result in unpredictable behavior.
2. The format of the limit values must match the format of the ADC converted value in terms of sign and fractional settings.
3. For Digital Comparator 0 used in CVD mode, the DCM PHI[15:0] and DCM PLO[15:0] bits must always be specified in signed format as the CVD output data is differential and is always signed.

Bit	31	30	29	28	27	26	25	24
	DCMPHI[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	23	22	21	20	19	18	17	16
	DCMPHI[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
	DCMPLO[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
	DCMPLO[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 31:16 – DCM PHI[15:0]** Digital Comparator ‘x’ High Limit Value bits<sup>(1,2,3)</sup>

These bits store the high limit value, which is used by digital comparator for comparisons with ADC converted data.

**Bits 15:0 – DCM PLO[15:0]** Digital Comparator ‘x’ Low Limit Value bits<sup>(1,2,3)</sup>

These bits store the low limit value, which is used by digital comparator for comparisons with ADC converted data.

### 38.11.12 ADCCMP2 – ADC Digital Comparator 2 Limit Value Register

**Name:** ADCCMP2  
**Offset:** 0x1510  
**Reset:** 0x00000000  
**Property:** -

These registers contain the high and low digital comparison values for use by the digital comparator.

**Notes:**

1. Changing these bits while the Digital Comparator is enabled (ENDCMP = 1) can result in unpredictable behavior.
2. The format of the limit values must match the format of the ADC converted value in terms of sign and fractional settings.
3. For Digital Comparator 0 used in CVD mode, the DCMPhi[15:0] and DCMplo[15:0] bits must always be specified in signed format as the CVD output data is differential and is always signed.

Bit	31	30	29	28	27	26	25	24
	DCMPHI[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	23	22	21	20	19	18	17	16
	DCMPHI[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
	DCMPLO[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
	DCMPLO[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 31:16 – DCMPhi[15:0]** Digital Comparator ‘x’ High Limit Value bits<sup>(1,2,3)</sup>

These bits store the high limit value, which is used by digital comparator for comparisons with ADC converted data.

**Bits 15:0 – DCMplo[15:0]** Digital Comparator ‘x’ Low Limit Value bits<sup>(1,2,3)</sup>

These bits store the low limit value, which is used by digital comparator for comparisons with ADC converted data.

### 38.11.13 ADCFLTR1 – ADC Digital Filter 1 Register

**Name:** ADCFLTR1  
**Offset:** 0x15A0  
**Reset:** 0x00000000  
**Property:** -

These registers provide control and status bits for the oversampling filter accumulator, and also includes the 16-bit filter output data.

Bit	31	30	29	28	27	26	25	24
	AFEN	DATA16EN	DFMODE	OVRSAM[2:0]			AFGIEN	AFRDY
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
	CHNLID[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
	FLTRDATA[15:8]							
Access	R/HS/HC	R/HS/HC	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	0	0
Bit	7	6	5	4	3	2	1	0
	FLTRDATA[7:0]							
Access	R/HS/HC	R/HS/HC	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	0	0

#### Bit 31 – AFEN Digital Filter ‘x’ Enable bit

Value	Description
1	Digital filter is enabled
0	Digital filter is disabled and the AFRDY status bit is cleared

#### Bit 30 – DATA16EN Filter Significant Data Length bit

**Note:** This bit is significant only if DFMODE = 1 (Averaging Mode) and FRACT (ADCCON1[23]) = 1 (Fractional Output Mode).

Value	Description
1	All 16 bits of the filter output data are significant
0	Only the first 12 bits are significant, followed by four zeros

#### Bit 29 – DFMODE ADC Filter Mode bit

Value	Description
1	Filter ‘x’ works in Averaging mode
0	Filter ‘x’ works in Oversampling Filter mode (default)

#### Bits 28:26 – OVRSAM[2:0] Oversampling Filter Ratio bits

Value	Description
	<b>If DFMODE is ‘o’</b>
111	128 samples (shift sum 3 bits to right, output data is in 15.1 format)
110	32 samples (shift sum 2 bits to right, output data is in 14.1 format)
101	8 samples (shift sum 1 bit to right, output data is in 13.1 format)
100	2 samples (shift sum 0 bits to right, output data is in 12.1 format)
011	256 samples (shift sum 4 bits to right, output data is 16 bits)

Value	Description
010	64 samples (shift sum 3 bits to right, output data is 15 bits)
001	16 samples (shift sum 2 bits to right, output data is 14 bits)
000	4 samples (shift sum 1 bit to right, output data is 13 bits)
<b>If DFMODE is '1'</b>	
111	256 samples (256 samples to be averaged)
110	128 samples (128 samples to be averaged)
101	64 samples (64 samples to be averaged)
100	32 samples (32 samples to be averaged)
011	16 samples (16 samples to be averaged)
010	8 samples (8 samples to be averaged)
001	4 samples (4 samples to be averaged)
000	2 samples (2 samples to be averaged)

**Bit 25 – AFGIEN** Digital Filter 'x' Interrupt Enable bit

Value	Description
1	Digital filter interrupt is enabled and is generated by the AFRDY status bit
0	Digital filter is disabled

**Bit 24 – AFRDY** Digital Filter 'x' Data Ready Status bit

**Note:** This bit is cleared by reading the FLTRDATA[15:0] bits or by disabling the Digital Filter module (by setting AFEN to '0').

Value	Description
1	Data is ready in the FLTRDATA[15:0] bits
0	Data is not ready

**Bits 20:16 – CHNLID[4:0]** Digital Filter Analog Input Selection bits

**Note:** Only the first 12 analog inputs, Class 2 (AN0 -AN11), can use a digital filter.

These bits specify the analog input to be used as the oversampling filter data source.

Value	Description
11111	Reserved
...	
...	
...	
01100	Reserved
01011	AN11
...	
...	
...	
00010	AN2
00001	AN1
00000	AN0

**Bits 15:0 – FLTRDATA[15:0]** Digital Filter 'x' Data Output Value bits

The filter output data is as per the fractional format set in the FRACT bit (ADCCON1[23]). The FRACT bit must not be changed while the filter is enabled. Changing the state of the FRACT bit after the operation of the filter ended must not update the value of the FLTRDATA[15:0] bits to reflect the new format.

### 38.11.14 ADCFLTR2 – ADC Digital Filter 2 Register

**Name:** ADCFLTR2  
**Offset:** 0x15B0  
**Reset:** 0x00000000  
**Property:** -

These registers provide control and status bits for the oversampling filter accumulator, and also include the 16-bit filter output data.

Bit	31	30	29	28	27	26	25	24
	AFEN	DATA16EN	DFMODE	OVRSAM[2:0]			AFGIEN	AFRDY
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
				CHNLID[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
	FLTRDATA[15:8]							
Access	R/HS/HC	R/HS/HC	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	0	0
Bit	7	6	5	4	3	2	1	0
	FLTRDATA[7:0]							
Access	R/HS/HC	R/HS/HC	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	0	0

#### Bit 31 – AFEN Digital Filter ‘x’ Enable bit

Value	Description
1	Digital filter is enabled
0	Digital filter is disabled and the AFRDY status bit is cleared

#### Bit 30 – DATA16EN Filter Significant Data Length bit

**Note:** This bit is significant only if DFMODE = 1 (Averaging Mode) and FRACT (ADCCON1[23]) = 1 (Fractional Output Mode).

Value	Description
1	All 16 bits of the filter output data are significant
0	Only the first 12 bits are significant, followed by four zeros

#### Bit 29 – DFMODE ADC Filter Mode bit

Value	Description
1	Filter ‘x’ works in Averaging mode
0	Filter ‘x’ works in Oversampling Filter mode (default)

#### Bits 28:26 – OVRSAM[2:0] Oversampling Filter Ratio bits

Value	Description
	<b>If DFMODE is ‘o’</b>
111	128 samples (shift sum 3 bits to right, output data is in 15.1 format)
110	32 samples (shift sum 2 bits to right, output data is in 14.1 format)
101	8 samples (shift sum 1 bit to right, output data is in 13.1 format)
100	2 samples (shift sum 0 bits to right, output data is in 12.1 format)
011	256 samples (shift sum 4 bits to right, output data is 16 bits)

Value	Description
010	64 samples (shift sum 3 bits to right, output data is 15 bits)
001	16 samples (shift sum 2 bits to right, output data is 14 bits)
000	4 samples (shift sum 1 bit to right, output data is 13 bits)
<b>If DFMODE is '1'</b>	
111	256 samples (256 samples to be averaged)
110	128 samples (128 samples to be averaged)
101	64 samples (64 samples to be averaged)
100	32 samples (32 samples to be averaged)
011	16 samples (16 samples to be averaged)
010	8 samples (8 samples to be averaged)
001	4 samples (4 samples to be averaged)
000	2 samples (2 samples to be averaged)

**Bit 25 – AFGIEN** Digital Filter 'x' Interrupt Enable bit

Value	Description
1	Digital filter interrupt is enabled and is generated by the AFRDY status bit
0	Digital filter is disabled

**Bit 24 – AFRDY** Digital Filter 'x' Data Ready Status bit

**Note:** This bit is cleared by reading the FLTRDATA[15:0] bits or by disabling the Digital Filter module (by setting AFEN to '0').

Value	Description
1	Data is ready in the FLTRDATA[15:0] bits
0	Data is not ready

**Bits 20:16 – CHNLID[4:0]** Digital Filter Analog Input Selection bits

**Note:** Only the first 12 analog inputs, Class 2 (AN0-AN11), can use a digital filter.

These bits specify the analog input to be used as the oversampling filter data source.

Value	Description
11111	Reserved
...	
...	
...	
01100	Reserved
01011	AN11
...	
...	
...	
00010	AN2
00001	AN1
00000	AN0

**Bits 15:0 – FLTRDATA[15:0]** Digital Filter 'x' Data Output Value bits

The filter output data is as per the fractional format set in the FRACT bit (ADCCON1[23]). The FRACT bit must not be changed while the filter is enabled. Changing the state of the FRACT bit after the operation of the filter ended must not update the value of the FLTRDATA[15:0] bits to reflect the new format.



### 38.11.15 ADCTRG1 – ADC Trigger Source 1 Register

**Name:** ADCTRG1  
**Offset:** 0x1600  
**Reset:** 0x00000000  
**Property:** -

This register controls the trigger source selection for AN0 through AN3 analog inputs.

Bit	31	30	29	28	27	26	25	24
					TRGSRC3[4:0]			
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
					TRGSRC2[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
					TRGSRC1[4:0]			
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
					TRGSRC0[4:0]			
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

#### Bits 28:24 – TRGSRC3[4:0] Trigger Source for Conversion of Analog Input AN3 Select bits

**Note:** For STRIG, in addition to setting the trigger, it also requires programming of the STRGSRC[4:0] bits (ADCCON1[20:16]) to select the trigger source, and requires the appropriate CSS bits to be set in the ADCCSSx registers.

Value	Description
10001 – 11111	Reserved
10000	EVSYS_47
01111	EVSYS_46
01110	EVSYS_45
01101	EVSYS_44
01100	EVSYS_43
01011	EVSYS_42
01010	EVSYS_41
01001	EVSYS_40
01000	EVSYS_39
00111	EVSYS_38
00110	EVSYS_37
00101	EVSYS_36
00100	INT0 External interrupt
00011	STRIG
00010	Global level software trigger (GLSWTRG)
00001	Global software edge trigger (GSWTRG)
00000	No Trigger

**Bits 20:16 – TRGSRC2[4:0]** Trigger Source for Conversion of Analog Input AN2 Select bits

**Note:** See bits 28-24 for bit value definitions.

**Bits 12:8 – TRGSRC1[4:0]** Trigger Source for Conversion of Analog Input AN1 Select bits

**Note:** See bits 28-24 for bit value definitions.

**Bits 4:0 – TRGSRC0[4:0]** Trigger Source for Conversion of Analog Input AN0 Select bits

**Note:** See bits 28-24 for bit value definitions.

### 38.11.16 ADCTRG2 – ADC Trigger Source 2 Register

**Name:** ADCTRG2  
**Offset:** 0x1610  
**Reset:** 0x00000000  
**Property:** -

This register controls the trigger source selection for AN4 through AN7 analog inputs.

Bit	31	30	29	28	27	26	25	24
				TRGSRC7[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
				TRGSRC6[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
				TRGSRC5[4:0]				
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
				TRGSRC4[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

#### Bits 28:24 – TRGSRC7[4:0] Trigger Source for Conversion of Analog Input AN7 Select bits

**Note:** For STRIG, in addition to setting the trigger, it also requires programming of the STRGSRC[4:0] bits (ADCCON1[20:16]) to select the trigger source, and requires the appropriate CSS bits to be set in the ADCCSSx registers.

Value	Description
10001 – 11111	Reserved
10000	EVSYS_47
01111	EVSYS_46
01110	EVSYS_45
01101	EVSYS_44
01100	EVSYS_43
01011	EVSYS_42
01010	EVSYS_41
01001	EVSYS_40
01000	EVSYS_39
00111	EVSYS_38
00110	EVSYS_37
00101	EVSYS_36
00100	INT0 External interrupt
00011	STRIG
00010	Global level software trigger (GLSWTRG)
00001	Global software edge trigger (GSWTRG)
00000	No Trigger

**Bits 20:16 – TRGSRC6[4:0]** Trigger Source for Conversion of Analog Input AN6 Select bits

**Note:** See bits 28-24 for bit value definitions.

**Bits 12:8 – TRGSRC5[4:0]** Trigger Source for Conversion of Analog Input AN5 Select bits

**Note:** See bits 28-24 for bit value definitions.

**Bits 4:0 – TRGSRC4[4:0]** Trigger Source for Conversion of Analog Input AN4 Select bits

**Note:** See bits 28-24 for bit value definitions.

### 38.11.17 ADCCMPCON1 – ADC Digital Comparator 1 Control Register

**Name:** ADCCMPCON1  
**Offset:** 0x1680  
**Reset:** 0x00000000  
**Property:** -

This register controls the operation of Digital Comparator 1, including the generation of interrupts, comparison criteria to be used and provides status when a comparator event occurs.

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			AINID[5:0]						
Reset			R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	
			0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
Access	ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO	
Reset	R/W	R/W	R/HS/HC	R/W	R/W	R/W	R/W	R/W	
	0	0	0	0	0	0	0	0	

#### Bits 13:8 – AINID[5:0] Digital Comparator 1 Analog Input Identification (ID) bits

When a digital comparator event occurs (DCMPED = 1), these bits identify the analog input being monitored by digital comparator 1.

**Note:** In normal ADC mode, only analog inputs [8:1] can be processed by the digital comparator 1.

Value	Description
111111	Reserved
...	
...	
...	
101101	Reserved
101100	Reserved
101011	Reserved
000111	AN7 is being monitored
...	
000001	AN1 is being monitored
000000	AN0 is being monitored

#### Bit 7 – ENDCMP Digital Comparator 1 Enable bit

Value	Description
1	Digital comparator 1 is enabled
0	Digital comparator 1 is not enabled, and the DCMPED status bit (ADCCMP0CON[5]) is cleared

#### Bit 6 – DCMPGIEN Digital Comparator 1 Global Interrupt Enable bit

Value	Description
1	A Digital comparator 1 interrupt is generated when the DCMPED status bit (ADCCMP0CON[5]) is set

Value	Description
0	A Digital comparator 1 interrupt is disabled

**Bit 5 - DCMPEL** Digital Comparator 1 “Output True” Event Status bit

The logical conditions under which the digital comparator becomes “True” are defined by the IEBTWN, IEHIHI, IEHILO, IELOHI and IELOLO bits.

**Note:** This bit is cleared by reading the AINID[5:0] bits or by disabling the Digital Comparator module (by setting ENDCMP to ‘0’).

Value	Description
1	Digital comparator 1 output true event has occurred (output of comparator is ‘1’)
0	Digital comparator 1 output is false (output of comparator is ‘0’)

**Bit 4 - IEBTWN** Between Low/High Digital Comparator 1 Event bit

Value	Description
1	Generate a digital comparator event when DATA[31:0] is less than DCMPHI[15:0] AND greater than DCMPL0[15:0]
0	Do not generate a digital comparator event

**Bit 3 - IEHIHI** High/High Digital Comparator 1 Event bit

Value	Description
1	Generate a digital comparator 1 event when DCMPHI[15:0] bits are less than or equal to DATA[31:0] bits
0	Do not generate an event

**Bit 2 - IEHILO** High/Low Digital Comparator 1 Event bit

Value	Description
1	Generate a digital comparator 1 event when DATA[31:0] bits are less than DCMPHI[15:0] bits
0	Do not generate an event

**Bit 1 - IELOHI** Low/High Digital Comparator 1 Event bit

Value	Description
1	Generate a digital comparator 1 event when DCMPL0[15:0] bits are less than or equal to DATA[31:0] bits
0	Do not generate an event

**Bit 0 - IELOLO** Low/Low Digital Comparator 1 Event bit

Value	Description
1	Generate a digital comparator 1 event when DATA[31:0] bits are less than DCMPL0[15:0] bits
0	Do not generate an event

### 38.11.18 ADCCMPCON2 – ADC Digital Comparator 2 Control Register

**Name:** ADCCMPCON2  
**Offset:** 0x1690  
**Reset:** 0x00000000  
**Property:** -

These registers control the operation of Digital Comparator 2, including the generation of interrupts and the comparison criteria to be used. This register also provides the status when a comparator event occurs.

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				AINID[4:0]					
Reset				R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	
				0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
Access	ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO	
Reset	R/W	R/W	R/HS/HC	R/W	R/W	R/W	R/W	R/W	
	0	0	0	0	0	0	0	0	

#### Bits 12:8 – AINID[4:0] Digital Comparator 2 Analog Input Identification (ID) bits

When a digital comparator event occurs (DCMPED = 1), these bits identify the analog input being monitored by the digital comparator.

**Note:** Only analog inputs [8:1] can be processed by the Digital Comparator module 'x' ('x' = 1-2).

Value	Description
11111	Reserved
11110	Reserved
...	
...	
...	
00011	Reserved
000111	AN7 is being monitored
...	
00001	AN1 is being monitored
00000	AN0 is being monitored

#### Bit 7 – ENDCMP Digital Comparator 2 Enable bit

Value	Description
1	Digital comparator 2 is enabled
0	Digital comparator 2 is not enabled, and the DCMPED status bit (ADCCMP0CON[5]) is cleared

#### Bit 6 – DCMPGIEN Digital Comparator 2 Global Interrupt Enable bit

Value	Description
1	Digital comparator 2 interrupt is generated when the DCMPED status bit (ADCCMP0CON[5]) is set

Value	Description
0	Digital comparator 2 interrupt is disabled

**Bit 5 – DCMPEd** Digital Comparator 2 “Output True” Event Status bit

The logical conditions where the digital comparator gets “True” are defined by the IEBTWN, IEHIHI, IEHILO, IELOHI and IELOLO bits.

**Note:** This bit is cleared by reading the AINID[5:0] bits (ADCCMP0CON[13:8]) or by disabling the Digital Comparator module (by setting ENDCMP to ‘0’).

Value	Description
1	Digital comparator 2 output true event has occurred (output of comparator is ‘1’)
0	Digital comparator 2 output is false (output of comparator is ‘0’)

**Bit 4 – IEBTWN** Between Low/High Digital Comparator 2 Event bit

Value	Description
1	Generate a digital comparator event when DCMPL0[15:0] bits DATA[31:0] bits [DCMPHI[15:0] bits
0	Do not generate a digital comparator event

**Bit 3 – IEHIHI** High/High Digital Comparator 2 Event bit

Value	Description
1	Generate a digital comparator 2 event when DCMPHI[15:0] bits are less than or equal to DATA[31:0] bits
0	Do not generate an event

**Bit 2 – IEHILO** High/Low Digital Comparator 2 Event bit

Value	Description
1	Generate a digital comparator 2 event when DATA[31:0] bits are less than DCMPHI[15:0] bits
0	Do not generate an event

**Bit 1 – IELOHI** Low/High Digital Comparator 2 Event bit

Value	Description
1	Generate a digital comparator 2 event when DCMPL0[15:0] bits are less than or equal to DATA[31:0] bits
0	Do not generate an event

**Bit 0 – IELOLO** Low/Low Digital Comparator 2 Event bit

Value	Description
1	Generate a digital comparator 2 event when DATA[31:0] bits are less than DCMPL0[15:0] bits
0	Do not generate an event



### 38.11.19 ADCBASE – ADC Base Register

**Name:** ADCBASE  
**Offset:** 0x1700  
**Reset:** 0x00000000  
**Property:** -

This register specifies the base address of the user ADC Interrupt Service Routine (ISR) jump table.

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	ADCBASE[15:8]							
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	ADCBASE[7:0]							
Reset	0	0	0	0	0	0	0	0

#### Bits 15:0 – ADCBASE[15:0] ADCISR Base Address bits

This register, when read, contains the base address of the user's ADC ISR jump table. The interrupt vector address is determined by the IRQVS[2:0] bits of the ADCCON1 register specifying the amount of left shift done to the ARDYx status bits in the ADCDSTAT1 register, prior to adding with ADCBASE register.

Interrupt vector address = Read value of ADCBASE

Read value of ADCBASE = Value written to ADCBASE +  $x \ll \text{IRQVS}[2:0]$ , where 'x' is the smallest active analog input ID from the ADCDSTAT1 register (which has the highest priority).

### 38.11.20 ADCDMASSTAT – ADC DMA Status Register

**Name:** ADCDMASSTAT  
**Offset:** 0x1710  
**Reset:** 0x00000000  
**Property:** -

This register contains the DMA status bits.

Bit	31	30	29	28	27	26	25	24
	DMAEN							RBF0IEN
Access	R/W							R/W
Reset	0							0
Bit	23	22	21	20	19	18	17	16
	WROVRERR							RBF0
Access	R/HS/HC							R/HS/HC
Reset	0							0
Bit	15	14	13	12	11	10	9	8
	DMACNTEN							RAFOIEN
Access	R/W							R/W
Reset	0							0
Bit	7	6	5	4	3	2	1	0
								RAFO
Access								R/HS/HC
Reset								0

#### Bit 31 – DMAEN DMA Interface Enable bit

When DMAEN = 0, no data is being saved into the DMA FIFO, no SRAM writes occur, and the DMA interface logic is being kept in Reset state.

Value	Description
1	DMA interface is enabled
0	DMA interface is disabled

#### Bit 24 – RBF0IEN RAM Buffer B Full Interrupt Enable for channel 0

Value	Description
1	Interrupts are enabled and generated when the RBFx Status bit is set
0	Interrupts are disabled

#### Bit 23 – WROVRERR Write Overflow Error in the DMA FIFO

Set by hardware, cleared by hardware after a software read of the ADDMAST register.

**Note:** The write always occurs and the old data is replaced with new data because the software missed reading the old data on time.

#### Bit 16 – RBF0 RAM Buffer B FULL status bit for channel 0

This bit is self-clearing upon being read by software.

#### Bit 15 – DMACNTEN DMA Buffer Sample Count Enable bit

The DMA interface saves the current sample count for each buffer in the table starting at the ADCNTB address after each sample write into the corresponding buffer in the SRAM.

#### Bit 8 – RAFOIEN RAM Buffer A FULL Interrupt Enable for channel 0

Value	Description
1	Interrupts are enabled and generated when the RAFx Status bit is set
0	Interrupts are disabled

**Bit 0 – RAF0** RAM Buffer A FULL status bit for channel 0  
This bit is self-clearing upon being read by software.

### 38.11.21 ADCCNTB – ADC Channel Sample Count Base Address Register

**Name:** ADCCNTB  
**Offset:** 0x1720  
**Reset:** 0x00000000  
**Property:** -

This register contains the base address of the sample count in RAM. In addition to storing the converted data of the ADC module in RAM, DMA also stores the converted sample count.

Bit	31	30	29	28	27	26	25	24
	ADCCNTB[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
	ADCCNTB[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
	ADCCNTB[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
	ADCCNTB[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bits 31:0 – ADCCNTB[31:0] ADC Channel Count Base Address

SRAM address for the DMA interface at which to save the first class channel buffer A sample count values into the System RAM. If first class channel  $x$ ,  $x=0\dots6$ , is ready with a new available sample data and the DMA interface is currently saving data for channel  $x$  to RAM Buffer  $z$  (where  $z=0$  means Buffer A and  $z=1$  means Buffer B,  $z$  depending on  $x$ ), then the DMA interface will increment (+1) the 1 byte count value stored at the System RAM address ( $ADCCNTB + 2*x + z$ ).

ADCCNTB works in conjunction with ADDMAB. The DMA interface uses ADCCNTB to save the buffer sample counts only if ADDMAST.DMA\_CNT\_EN is set to 1.

### 38.11.22 ADCDMAB – ADC DMA Base Address Register

**Name:** ADCDMAB  
**Offset:** 0x1730  
**Reset:** 0x00000000  
**Property:** -

This register contains the base address of RAM for the DMA engine.

Bit	31	30	29	28	27	26	25	24
	ADDMAB[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
	ADDMAB[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
	ADDMAB[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
	ADDMAB[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bits 31:0 – ADDMAB[31:0] Base Address for the DMA Interface

BASE ADDRESS for the DMA interface at which to save first class channel data into the System RAM. If first class channel  $x$ ,  $x == 0...6$ , is ready with new available sample data and the DMA interface is currently saving data for channel  $x$  to RAM Buffer  $z$  (where  $z == 0$  means Buffer A and  $z == 1$  means Buffer B,  $z$  depending on  $x$ ) and the current DMA  $x$ -counter value is  $y$  ( $y$  depending on  $x$ ), then the DMA interface stores the 2-byte output data value at System RAM address  $(ADDMAB + (2*x + z)*2(DMABL+1) + 2*y)$ . Also, if  $ADDMAST.DMA\_CNT\_EN$  is set to 1, the DMA interface stores (without delay) the value  $y$  itself at the System RAM address  $(ADCCNTB + 2*x + z)$ .

### 38.11.23 ADCTRGSNS – ADC Trigger Level/Edge Sensitivity Register

**Name:** ADCTRGSNS  
**Offset:** 0x1740  
**Reset:** 0x00000000  
**Property:** -

This register contains the setting for trigger level for each ADC analog input.

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	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 – LVL Trigger Level and Edge Sensitivity bits

**Notes:**

1. This register specifies the trigger level for analog inputs 0 to 7.
2. The higher analog input ID belongs to Class 3, and, therefore, is only scan triggered. All Class 3 analog inputs use the scan trigger, for which the level/edge is defined by the STRGLVL bit (ADCCON1[3]).

Value	Description
1	Analog input is sensitive to the high level of its trigger (level sensitivity implies retriggering as long as the trigger signal remains high)
0	Analog input is sensitive to the positive edge of its trigger (this is the value after a reset)

### 38.11.24 ADCANCON – ADC Analog Warm-up Control Register

**Name:** ADCANCON  
**Offset:** 0x1800  
**Reset:** 0x00000000  
**Property:** -

This register contains the warm-up control settings for the analog and bias circuit of the ADC module.

Bit	31	30	29	28	27	26	25	24
	WKUPCLKCNT[3:0]							
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
	WKIEN7							WKIEN0
Access	R/W							R/W
Reset	0							0
Bit	15	14	13	12	11	10	9	8
	WKRDY7							WKRDY0
Access	R/HS/HC							R/HS/HC
Reset	0							0
Bit	7	6	5	4	3	2	1	0
	ANEN7							ANEN0
Access	R/W							R/W
Reset	0							0

#### Bits 27:24 – WKUPCLKCNT[3:0] Wake-up Clock Count bits

These bits represent the number of ADC clocks required to warm-up the ADC module before it can perform conversion. Although the clocks are specific to each ADC, the WKUPCLKCNT bit is common to all ADC modules.

Value	Description
1111	$2^{15}$ = 32,768 clocks
...	
...	
...	
0110	$2^6$ = 64 clocks
0101	$2^5$ = 32 clocks
0100	$2^4$ = 16 clocks
0011	$2^4$ = 16 clocks
0010	$2^4$ = 16 clocks
0001	$2^4$ = 16 clocks
0000	$2^4$ = 16 clocks

#### Bit 23 – WKIEN7 Shared ADC (ADC2) Wake-up Interrupt Enable bit

Value	Description
1	Enable interrupt and generate interrupt when the WKRDY2 status bit is set
0	Disable interrupt

#### Bit 16 – WKIEN0 ADC1 Wake-up Interrupt Enable bit

Value	Description
1	Enable interrupt and generate interrupt when the WKRDYx status bit is set

Value	Description
0	Disable interrupt

**Bit 15 - WKRDY7** Shared ADC (ADC2) Wake-up Status bit

**Note:** This bit is cleared by hardware when the ANEN2 bit is cleared.

Value	Description
1	ADC2 Analog and bias circuitry ready after the wake-up count number $2^{WKUPEXP}$ clocks after setting ANEN2 to '1'
0	ADC2 Analog and bias circuitry is not ready

**Bit 8 - WKRDY0** ADC1 Wake-up Status bit

**Note:** This bit is cleared by hardware when the ANENx bit is cleared.

Value	Description
1	ADC1 Analog and bias circuitry ready after the wake-up count number $2^{WKUPEXP}$ clocks after setting ANEN1 to '1'
0	ADC1 Analog and bias circuitry is not ready

**Bit 7 - ANEN7** Shared ADC (ADC2) Analog and Bias Circuitry Enable bit

Value	Description
1	Analog and bias circuitry enabled. Once the analog and bias circuit is enabled, the ADC module needs a warm-up time, as defined by the WKUPCLKCNT[3:0] bits.
0	Analog and bias circuitry disabled

**Bit 0 - ANEN0** ADC1 Analog and Bias Circuitry Enable bits

Value	Description
1	Analog and bias circuitry enabled. Once the analog and bias circuit is enabled, the ADC module needs a warm-up time, as defined by the WKUPCLKCNT[3:0] bits.
0	Analog and bias circuitry disabled



### 38.11.25 ADCSYSCFG0 – ADC System Configuration Register 0

**Name:** ADCSYSCFG0  
**Offset:** 0x1B00  
**Reset:** 0x00000000  
**Property:** -

This register contains read-only bits corresponding to the analog input.

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

#### Bits 19:0 – AN[19:0] ADC Analog Input bits

These bits reflect the system configuration and are updated during boot-up time. By reading these read-only bits, the user application can determine whether or not an analog input in the device is available.

### 38.11.26 ADCDATAx – ADC Output Data Register ('x' = 0 to 11)

**Name:** ADCDATAx  
**Offset:** 0x1Exx  
**Reset:** 0x00000000  
**Property:** -

These registers are the analog-to-digital conversion output data registers. The ADCDATAx register is associated with each external analog input, 0-7, plus internal analog inputs 8-11.

Bit	31	30	29	28	27	26	25	24
	DATA[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
	DATA[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
	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 31:0 – DATA[31:0] ADC Converted Data Output bits

##### Notes:

1. When an alternate input is used as the input source for a dedicated ADC module, the data output is still read from the Primary input Data Output register.
2. Reading the ADCDATAx register value after changing the FRACT bit converts the data into the format specified by the FRACT bit.

## 39. Analog Comparators (AC)

### 39.1 Overview

The Analog Comparator (AC) supports two individual comparators: one shared (with built-in supply voltage monitor (MVREF)) AC\_CMP1 and one dedicated AC\_CMP0.

A continuous mode of operation on a shared comparator is supported only if MVREF is disabled: PMU.CLKCTRL.MVREFFSMEN bit = 0 (Disabled). Each comparator (COMP) compares the voltage levels on two inputs and provides a digital output based on the comparison. Each comparator may be configured to generate interrupt requests and/or peripheral events upon several different combinations of input change.

The input selection includes up to four shared analog port pins and several internal signals. Each Comparator Output state can also be output on a pin for use by external devices.

The comparators are grouped in pairs on each port. The AC peripheral implements a pair of comparators. These are called Comparator 0 (COMP0) and Comparator 1 (COMP1). They have identical behaviors but separate Control registers. The pair can be set in Window mode to compare a signal to a voltage range instead of a single voltage level.

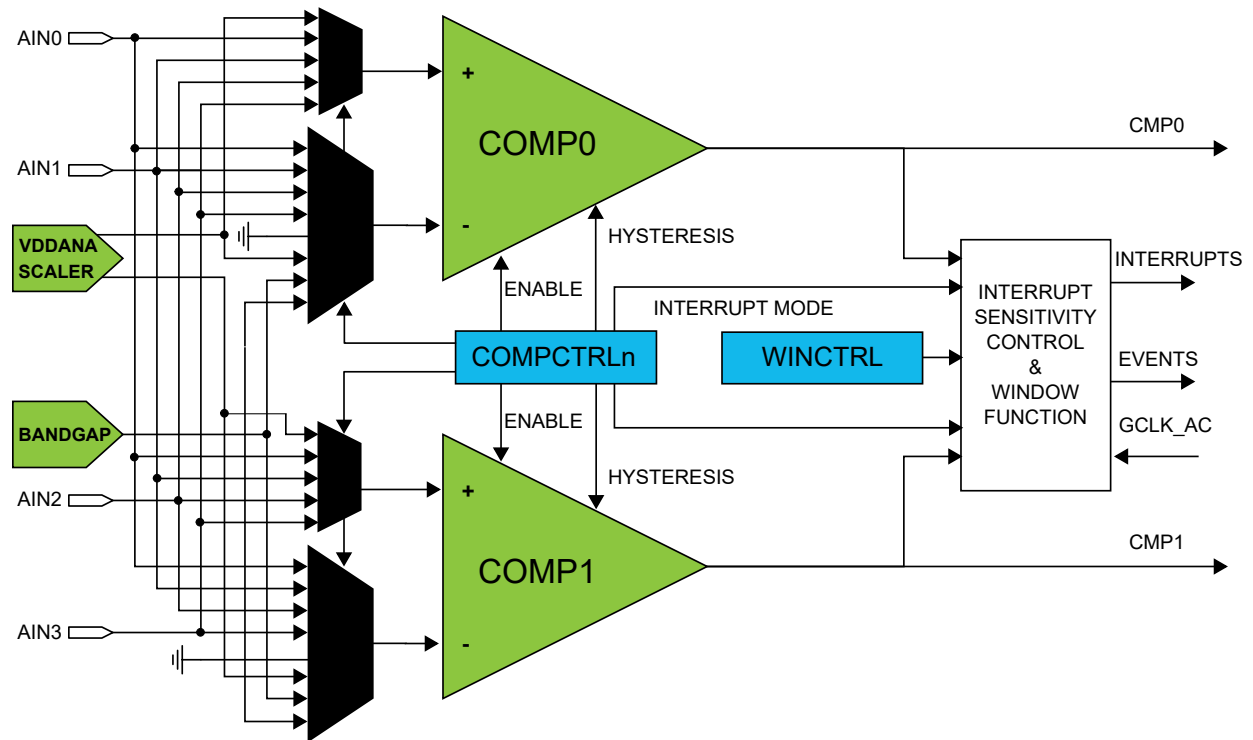
**Note:** MVREF will not be able to monitor the supply voltage for 275  $\mu$ s from the time  $\overline{\text{MCLR}}$  is released.

### 39.2 Features

- Two individual comparators (Single pair configuration)
- 48 pin variants have COMP0 and COMP1
- No COMP0 in 32 pins
- Analog comparator outputs available on pins
  - Asynchronous or synchronous
- Flexible input selection:
  - Up to four pins selectable for positive or negative inputs
  - Ground (for zero crossing)
  - Bandgap reference voltage
  - Programmable VDD scaler for AC\_CMP1 (shared with Programmable Low Voltage Detector (PLVD)) and fixed VDD/2 for AC\_CMP0
- Interrupt generation on:
  - Rising or falling edge
  - Toggle
  - End of comparison
- Window function interrupt generation on:
  - Signal above window
  - Signal inside window
  - Signal below window
  - Signal outside window
- Event generation on:
  - Comparator output
  - Window function inside/outside window
- Optional digital filter on comparator output

### 39.3 Block Diagram

Figure 39-1. Analog Comparator Block Diagram



### 39.4 Product Dependencies

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

#### 39.4.1 I/O Lines

Using the AC's I/O lines requires the I/O pins to be configured as analog pins for AC\_AINx inputs. See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links.

Table 39-1. I/O LINES

Signal	Peripheral Function
AC_AIN0	—
AC_AIN1	—
AC_AIN2	—
AC_AIN3	—
AC_CMP0	CFGCON1.CMP0_OE
AC_CMP1	CFGCON1.CMP1_OE

#### Related Links

[6. I/O Ports and Peripheral Pin Select \(PPS\)](#)

#### 39.4.2 Power Management

The AC will continue to operate in any Sleep mode where the selected source clock is running. The AC'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.

### 39.4.3 Clocks

The AC bus clock (PB2\_CLK) can be enabled and disabled in the Main Clock module, CRU (see *Clock and Reset Unit (CRU)* from Related Links), and the Analog Comparator module can be enabled or disabled via the PMD1 register. See *Peripheral Module Disable Register (PMD)* from Related Links.

A generic clock (GCLK\_AC) is required to clock the AC. This clock must be configured and enabled in the generic clock controller before using the AC.

This generic clock is asynchronous to the bus clock (PB2\_CLK). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. See *Synchronization* from Related Links.

#### Related Links

- [13. Clock and Reset Unit \(CRU\)](#)
- [20. Peripheral Module Disable Register \(PMD\)](#)
- [39.5.13. Synchronization](#)

### 39.4.4 DMA

Not applicable.

### 39.4.5 Interrupts

The interrupt request lines are connected to the interrupt controller. Using the AC 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 \(NVIC\)](#)

### 39.4.6 Events

The events are connected to the Event System. See *Event System (EVSYS)* from Related Links for details on how to configure the Event System.

#### Related Links

- [28. Event System \(EVSYS\)](#)

### 39.4.7 Debug Operation

When the CPU is halted in debug mode, the AC will halt normal operation after any ongoing comparison is completed. The AC can be forced to continue normal operation during debugging. See *DBGCTRL* register from Related Links. If the AC 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.

#### Related Links

- [39.7.9. DBGCTRL](#)

### 39.4.8 Register Access Protection

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

- Control B register (CTRLB)
- Interrupt Flag register (INTFLAG)

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.

### 39.4.9 Analog Connections

Each comparator has up to four I/O pins that can be used as analog inputs. Each pair of comparators shares the same four pins. These pins must be configured for analog operation before using them as comparator inputs.

Any internal reference source, such as a bandgap voltage reference, must be configured and enabled prior to its use as a comparator input.

## 39.5 Functional Description

### 39.5.1 Principle of Operation

Each comparator has one positive input and one negative input. Each positive input may be chosen from a selection of analog input pins. Each negative input may be chosen from a selection of both analog input pins and internal inputs, such as INTREF voltage reference.

The digital output from the comparator is '1' when the difference between the positive and the negative input voltage is positive, and '0' otherwise.

The individual comparators can be used independently (Normal mode) or paired to form a window comparison (Window mode).

### 39.5.2 Basic Operation

#### 39.5.2.1 Initialization

Some registers are enable-protected, meaning they can only be written when the module is disabled.

The following register is enable-protected:

- Event Control register (EVCTRL)

Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

#### 39.5.2.2 Enabling, Disabling and Resetting

The AC is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The AC is disabled by writing a '0' to CTRLA.ENABLE.

The AC is reset by writing a '1' to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the AC will be reset to their initial state, and the AC will be disabled. See CTRLA register from Related Links.

#### Related Links

[39.7.1. CTRLA](#)

#### 39.5.2.3 Comparator Configuration

Each individual comparator must be configured by its respective Comparator Control register (COMPCTRLx) before that comparator is enabled. These settings cannot be changed while the comparator is enabled.

- Select the desired measurement mode with COMPCTRLx.SINGLE. See *Starting a Comparison* from Related Links.
- Fixed hysteresis does not support programmable Hysteresis.
- Fixed speed of operation
- Select the interrupt source with COMPCTRLx.INTSEL.
- Select the positive and negative input sources with the COMPCTRLx.MUXPOS and COMPCTRLx.MUXNEG bits. See *Selecting Comparator Inputs* from Related Links.

- Select the filtering option with COMPCTRLx.FLEN.
- Select the standby operation with the Run in Standby bit (COMPCTRLx.RUNSTDBY).

The individual comparators are enabled by writing a '1' to the Enable bit in the Comparator x Control registers (COMPCTRLx.ENABLE). The individual comparators are disabled by writing a '0' to COMPCTRLx.ENABLE. Writing a '0' to CTRLA.ENABLE will also disable all the comparators but will not clear their COMPCTRLx.ENABLE bits.

### Related Links

[39.5.2.4. Starting a Comparison](#)

[39.5.3. Selecting Comparator Inputs](#)

### 39.5.2.4 Starting a Comparison

Each comparator channel can be in one of two different measurement modes, which is determined by the COMPCTRLx.SINGLE bit:

- Continuous measurement
- Single-shot

After being enabled, a start-up delay is required before the result of the comparison is ready. This start-up time is measured automatically to account for environmental changes, such as temperature or voltage supply level, and is specified in the Electrical Specifications. During the start-up time, the COMP output is not available.

The comparator can be configured to generate interrupts when the output toggles, when the output changes from '0' to '1' (rising edge), when the output changes from '1' to '0' (falling edge) or at the end of the comparison. An end-of-comparison interrupt can be used with the Single-Shot mode to chain further events in the system, regardless of the state of the comparator outputs. The Interrupt mode is set by the Interrupt Selection bit group in the Comparator Control register (COMPCTRLx.INTSEL). Events are generated using the comparator output state regardless of whether the interrupt is enabled or not.

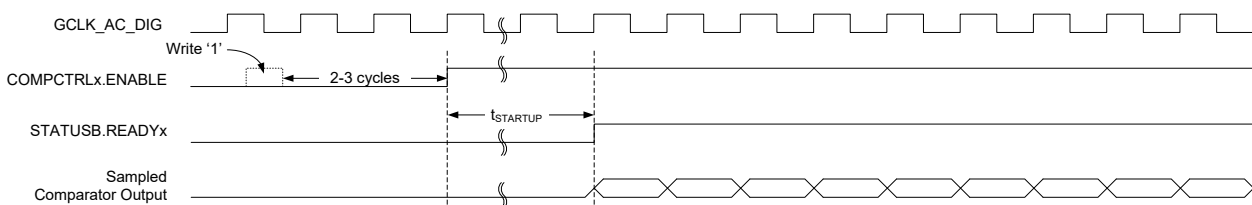
#### 39.5.2.4.1 Continuous Measurement

Continuous measurement is selected by writing COMPCTRLx.SINGLE to zero. In continuous mode, the comparator is continuously enabled and performing comparisons. This ensures that the result of the latest comparison is always available in the Current State bit in the Status A register (STATUSA.STATEx).

After the start-up time has passed, a comparison is done and STATUSA is updated. The Comparator x Ready bit in the Status B register (STATUSB.READYx) is set, and the appropriate peripheral events and interrupts are also generated. New comparisons are performed continuously until the COMPCTRLx.ENABLE bit is written to zero. The start-up time applies only to the first comparison.

In the continuous operation, edge detection of the comparator output for interrupts is done by comparing the current and previous sample. The sampling rate is the GCLK\_AC frequency. An example of continuous measurement is shown in the following figure.

**Figure 39-2.** Continuous Measurement Example



For low-power operation, comparisons can be performed during sleep mode without a clock. The comparator is enabled continuously, and changes of the comparator state are detected

asynchronously. When a toggle occurs, the CRU will start GCLK\_AC to register the appropriate peripheral events and interrupts. The GCLK\_AC clock is, then, disabled again automatically unless configured to wake up the system from sleep.

### 39.5.2.4.2 Single-Shot

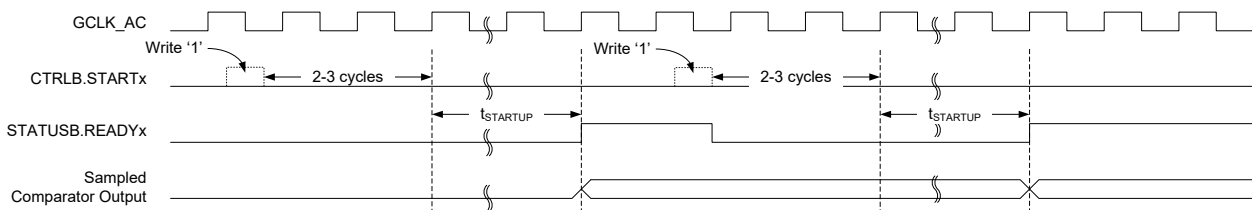
Single-shot operation is selected by writing COMPCTRLx.SINGLE to '1'. During single-shot operation, the comparator is normally idle. The user starts a single comparison by writing '1' to the respective Start Comparison bit in the write-only Control B register (CTRLB.STARTx). The comparator is enabled and, after the start-up time has passed, a single comparison is done and STATUSA is updated. Appropriate peripheral events and interrupts are also generated. No new comparisons will be performed.

Writing '1' to CTRLB.STARTx also clears the Comparator x Ready bit in the Status B register (STATUSB.READYx). STATUSB.READYx is set automatically by hardware when the single comparison has completed.

A single-shot measurement can also be triggered by the Event System. Setting the Comparator x Event Input bit in the Event Control Register (EVCTRL.COMPEIx) enables triggering on incoming peripheral events. Each comparator can be triggered independently by separate events. Event-triggered operation is similar to user-triggered operation; the difference is that a peripheral event from another hardware module causes the hardware to automatically start the comparison and will not clear STATUSB.READYx.

To detect an edge of the comparator output in single-shot operation for the purpose of interrupts, the result of the current measurement is compared with the result of the previous measurement (one sampling period earlier). An example of single-shot operation is shown in the following figure.

**Figure 39-3.** Single-Shot Example



For low-power operation, event-triggered measurements can be performed during sleep modes. When the event occurs, the CRU will start GCLK\_AC. The comparator is enabled and, after the start-up time has passed, a comparison is done and appropriate peripheral events and interrupts are also generated. The comparator and GCLK\_AC are, then, disabled again automatically unless configured to wake up the system from sleep.

### 39.5.3 Selecting Comparator Inputs

Each comparator has one positive and one negative input. The positive input is one of the external input pins (AINx). The negative input can be fed either from an external input pin (AINx) or from one of the internal reference voltage sources common to all comparators. The user selects the input source as follows:

- The positive input is selected by the Positive Input MUX Select bit group in the Comparator Control register (COMPCTRLx.MUXPOS).
- The negative input is selected by the Negative Input MUX Select bit group in the Comparator Control register (COMPCTRLx.MUXNEG).

In the case of using an external I/O pin, the selected pin must be configured for analog use in the GPIO by disabling the digital input and output. The switching of the analog input multiplexers is controlled to minimize crosstalk between the channels. The input selection must be changed only while the individual comparator is disabled.



**Note:** For internal use of the comparison results by the CCL module (see *Configurable Custom Logic (CCL)* from Related Links), COMPCTRLx.OUT must be 0x1 or 0x2.

### Related Links

[34. Configurable Custom Logic \(CCL\)](#)

### 39.5.4 Window Operation

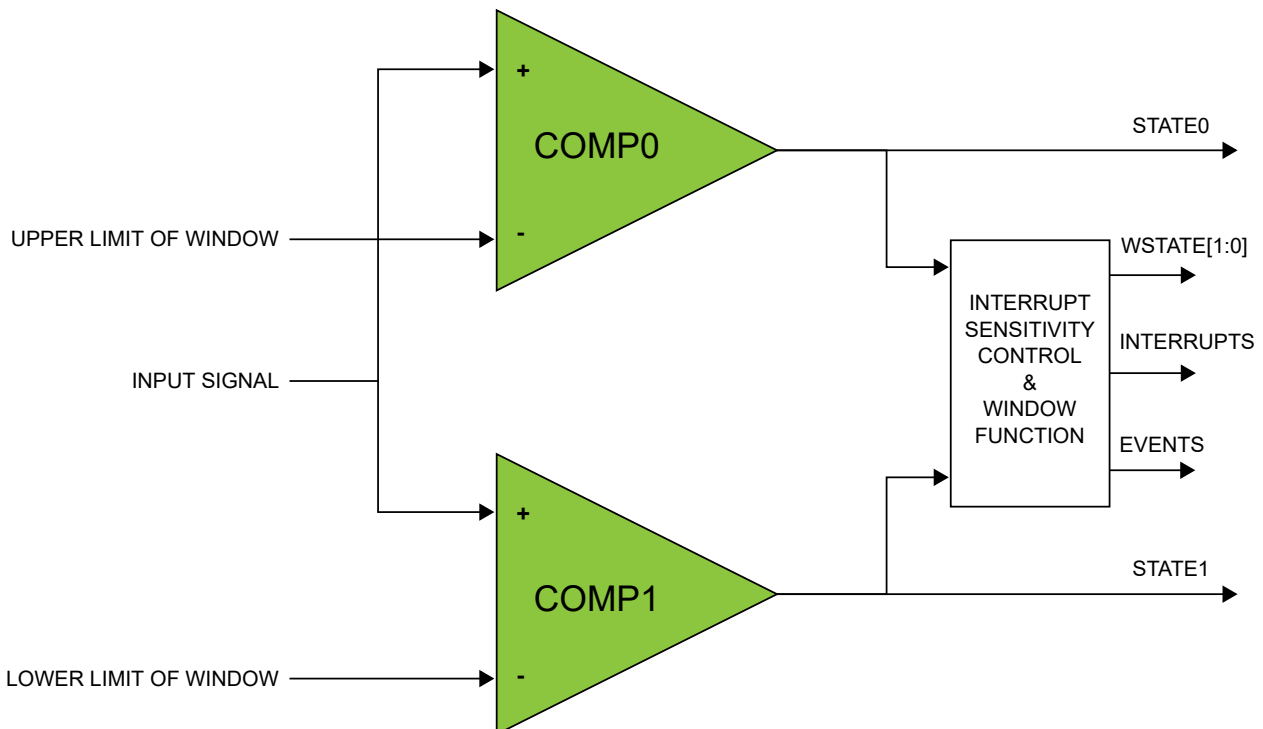
Each comparator pair can be configured to work together in Window mode. In this mode, a voltage range is defined, and the comparators give information about whether an input signal is within this range or not. Window mode is enabled by the Window Enable x bit in the Window Control register (WINCTRL.WENx). Both comparators in a pair must have the same measurement mode setting in their respective Comparator Control Registers (COMPCTRLx.SINGLE).

To physically configure the pair of comparators for Window mode, the same I/O pin must be chosen as positive input for each comparator, providing a shared input signal. The negative inputs define the range for the window. In [Figure 39-4](#), COMP0 defines the upper limit and COMP1 defines the lower limit of the window, as shown but the window will also work in the opposite configuration with COMP0 lower and COMP1 higher. The current state of the window function is available in the Window x State bit group of the Status register (STATUSA.WSTATEx).

Window mode can be configured to generate interrupts when the input voltage reaches values below the window, above the window, inside the window or outside of the window. The interrupt selections are set by the Window Interrupt Selection bit field in the Window Control register (WINCTRL.WINTSEL). Events are generated using the inside/outside state of the window, regardless of whether the interrupt is enabled or not. Note that the individual comparator outputs, interrupts and events continue to function normally during Window mode.

When the comparators are configured for Window mode and Single-shot mode, measurements are performed simultaneously on both comparators. Writing '1' to either Start Comparison bit in the Control B register (CTRLB.STARTx) will start a measurement. Likewise either peripheral event can start a measurement.

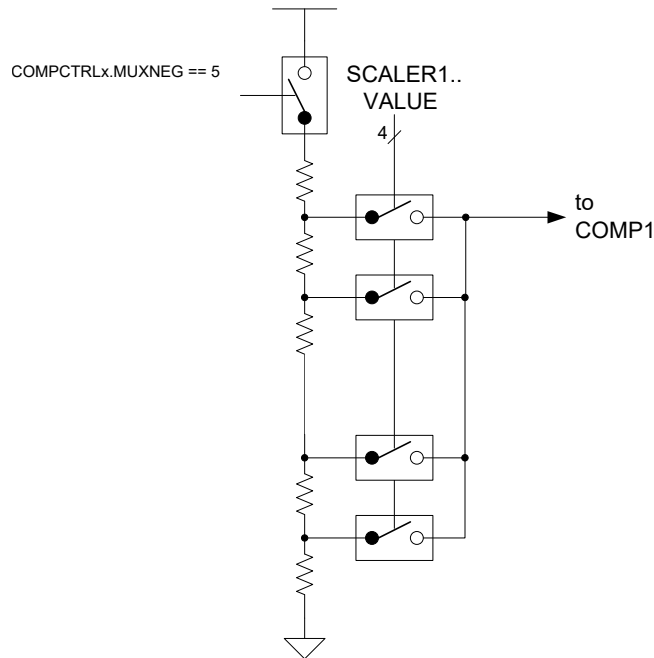
**Figure 39-4.** Comparators in Window Mode



### 39.5.5 VDD Scaler

The VDD scaler generates a reference voltage that is a fraction of the device's supply voltage with 64 levels. The programmable VDD scaler (PLVD Scaler) is available for AC\_CMP1 only. AC\_CMP0 uses a fixed VDD/2 reference. The scaler of a comparator is enabled when the Negative Input Mux bit field or the Positive Input Mux in the respective Comparator Control register (COMPCTRLx.MUXNEG) is set to 0x5 and the comparator is enabled. The voltage of each channel is selected by the Value bit field in the SCALER1 registers (SCALER1.VALUE) using the VDD resistor ladder.

Figure 39-5. PLVD Scaler

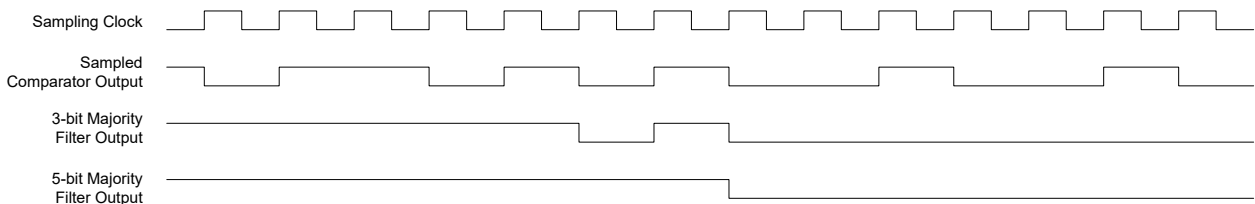


### 39.5.6 Filtering

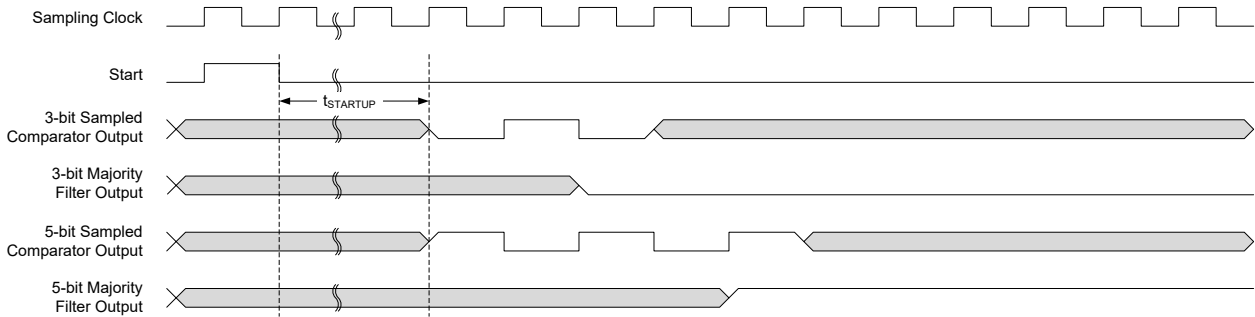
The output of the comparators can be filtered digitally to reduce noise. The filtering is determined by the Filter Length bits in the Comparator Control x register (COMPCTRLx.FLEN), and is independent for each comparator. Filtering is selectable from none, 3-bit majority (N=3) or 5-bit majority (N=5) functions. Any change in the comparator output is considered valid only if N/2+1 out of the last N samples agree. The filter sampling rate is the GCLK\_AC frequency.

Note that filtering creates an additional delay of N-1 sampling cycles from when a comparison is started until the comparator output is validated. For Continuous mode, the first valid output will occur when the required number of filter samples is taken. Subsequent outputs will be generated every cycle based on the current sample plus the previous N-1 samples, as shown in Figure 39-6. For Single-shot mode, the comparison completes after the Nth filter sample, as shown in Figure 39-7.

Figure 39-6. Continuous Mode Filtering



**Figure 39-7. Single-Shot Filtering**



During Sleep modes, filtering is supported only for single-shot measurements. Filtering must be disabled if continuous measurements will be done during Sleep modes, or the resulting interrupt/event may be generated incorrectly.

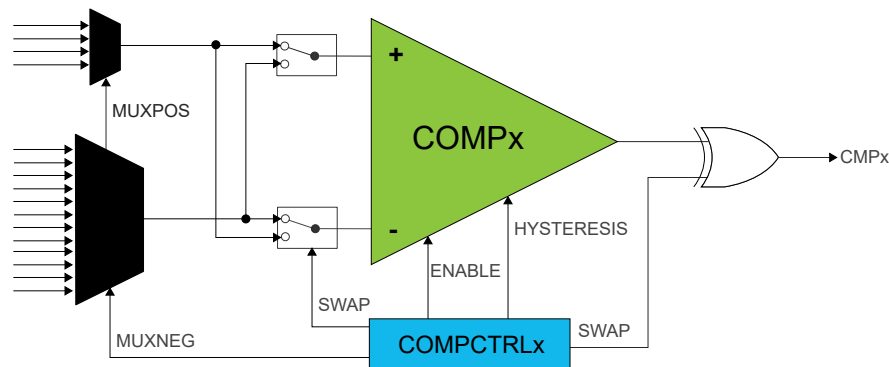
### 39.5.7 Comparator Output

The output of each comparator can be routed to an I/O pin by setting the Output bit group in the Comparator Control x register (COMPCTRLx.OUT). To get the analog comparator output on the I/O line, CFGCON1.CMP0\_OE/CFGCON1.CMP1\_OE also needs to be set or enabled. This allows the comparator to be used by external circuitry. Either the raw, non-synchronized output of the comparator or the GCLK\_AC-synchronized version, including filtering, can be used as the I/O signal source. The output appears on the corresponding AC\_CMPx pin. The AC\_CMP1 can be output on an alternate pin by configuring the CFGCON0.ACCMP1\_ALTEN configuration.

### 39.5.8 Offset Compensation

The Swap bit in the Comparator Control registers (COMPCTRLx.SWAP) controls switching of the input signals to a comparator's positive and negative terminals. When the comparator terminals are swapped, the output signal from the comparator is also inverted, as shown in Figure 39-8. This allows the user to measure or compensate for the comparator input offset voltage. As part of the input selection, COMPCTRLx.SWAP can be changed only while the comparator is disabled.

**Figure 39-8. Input Swapping for Offset Compensation**



### 39.5.9 DMA Operation

Not applicable.

### 39.5.10 Interrupts

The AC has the following interrupt sources:

- Comparator (COMP0, COMP1): Indicates a change in comparator status
- Window (WIN0): Indicates a change in the window status

Comparator interrupts are generated based on the conditions selected by the Interrupt Selection bit group in the Comparator Control registers (COMPCTRLx.INTSEL). Window interrupts are generated based on the conditions selected by the Window Interrupt Selection bit group in the Window Control register (WINCTRL.WINTSEL[1:0]).

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 one to the corresponding bit in the Interrupt Enable Set (INTENSET) register and disabled by writing a one 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 AC is Reset. See *INTFLAG* register from Related Links for details on how to clear interrupt flags. All interrupt requests from the peripheral are OR'ed together on the 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. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

**Note:** Interrupts must be globally enabled for interrupt requests to be generated.

#### Related Links

[10.2. Nested Vector Interrupt Controller \(NVIC\)](#)

[39.7.6. INTFLAG](#)

### 39.5.11 Events

The AC can generate the following output events:

- Comparator (COMP0, COMP1): Generated as a copy of the comparator status
- Window (WIN0): Generated as a copy of the window inside/outside status

Writing a one to an Event Output bit in the Event Control Register (EVCTRL.xxEO) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event. See *Event System (EVSYS)* from Related Links for details on how to configure the Event System.

The AC can take the following action on an input event:

- Start comparison (START0, START1): Start a comparison

Writing a one to an Event Input bit into the Event Control register (EVCTRL.COMPEIx) enables the corresponding action on an input event. Writing a zero to this bit disables the corresponding action on an input event. Note that if several events are connected to the AC, the enabled action will be taken on any of the incoming events. See *Event System (EVSYS)* from Related Links for details on how to configure the Event System.

When EVCTRL.COMPEIx is one, the event will start a comparison on COMPx after the start-up time delay. In normal mode, each comparator responds to its corresponding input event independently. For a pair of comparators in window mode, either comparator event will trigger a comparison on both comparators simultaneously.

#### Related Links

[28. Event System \(EVSYS\)](#)

### 39.5.12 Sleep Mode Operation

The Run in Standby bits in the Comparator x Control registers (COMPCTRLx.RUNSTDBY) control the behavior of the AC during standby sleep mode. Each RUNSTDBY bit controls one comparator. When the bit is zero, the comparator is disabled during sleep, but maintains its current configuration. When the bit is one, the comparator continues to operate during sleep. Note that when RUNSTDBY

is zero, the analog blocks are powered off for the lowest power consumption. This necessitates a start-up time delay when the system returns from sleep.

For Window Mode operation, both comparators in a pair must have the same RUNSTDBY configuration.

When RUNSTDBY is one, any enabled AC interrupt source can wake up the CPU. The AC can also be used during sleep modes where the clock used by the AC is disabled, provided that the AC is still powered (not in shutdown). In this case, the behavior is slightly different and depends on the measurement mode, as listed in [Table 39-2](#).

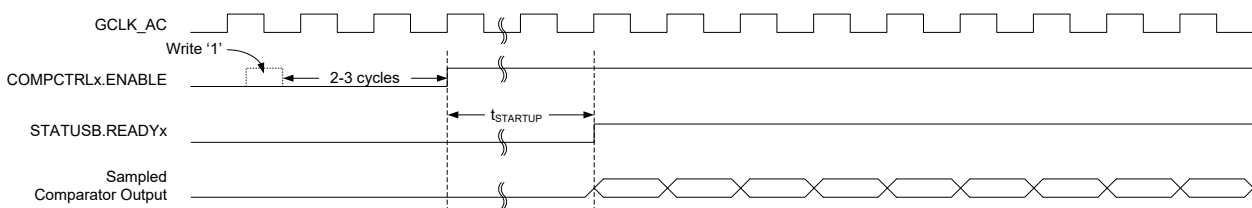
**Table 39-2.** Sleep Mode Operation

COMPCTRLx.MODE	RUNSTDBY=0	RUNSTDBY=1
0 (Continuous)	COMPx disabled	GCLK_AC stopped, COMPx enabled
1 (Single-shot)	COMPx disabled	GCLK_AC stopped, COMPx enabled only when triggered by an input event

### 39.5.12.1 Continuous Measurement during Sleep

When a comparator is enabled in continuous measurement mode and GCLK\_AC is disabled during sleep, the comparator will remain continuously enabled and will function asynchronously. The current state of the comparator is asynchronously monitored for changes. If an edge matching the interrupt condition is found, GCLK\_AC is started to register the interrupt condition and generate events. If the interrupt is enabled in the Interrupt Enable registers (INTENCLR/SET), the AC can wake up the device; otherwise GCLK\_AC is disabled until the next edge detection. Filtering is not possible with this configuration.

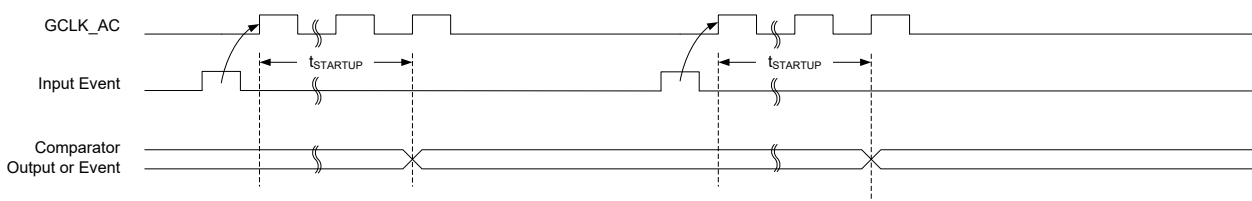
**Figure 39-9.** Continuous Mode SleepWalking



### 39.5.12.2 Single-Shot Measurement during Sleep

For low-power operation, event-triggered measurements can be performed during sleep modes. When the event occurs, the CRU will start GCLK\_AC. The comparator is enabled and, after the start-up time has passed, a comparison is done, with filtering if desired, and the appropriate peripheral events and interrupts are also generated as shown in the following figure. The comparator and GCLK\_AC are, then, disabled again automatically unless configured to wake the system from sleep. Filtering is allowed with this configuration.

**Figure 39-10.** Single-Shot SleepWalking



### 39.5.13 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)
- Enable bit in Comparator Control register (COMPCTRLn.ENABLE)

The following registers are synchronized when written:

- Window Control register (WINCTRL)

Required write synchronization is denoted by the "Write-Synchronized" property in the register description.

## 39.6 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0							ENABLE	SWRST
0x01	CTRLB	7:0							START1	START0
0x02	EVCTRL	7:0				WINEO0			COMPEO1	COMPEO0
		15:8			INVEI1	INVEI0			COMPEI1	COMPEI0
0x04	INTENCLR	7:0				WIN0			COMP1	COMP0
0x05	INTENSET	7:0				WIN0			COMP1	COMP0
0x06	INTFLAG	7:0				WIN0			COMP1	COMP0
0x07	STATUSA	7:0			WSTATE0[1:0]				STATE1	STATE0
0x08	STATUSB	7:0							READY1	READY0
0x09	DBGCTRL	7:0								DBGRUN
0x0A	WINCTRL	7:0						WINTSEL0[1:0]		WEN0
0x0B	...									
0x0C	Reserved									
0x0D	SCALER1	7:0					VALUE[3:0]			
0x0E	...									
0x0F	Reserved									
0x10	COMPCTRL0	7:0		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE	
		15:8	SWAP		MUXPOS[2:0]			MUXNEG[2:0]		
		23:16								
		31:24			OUT[1:0]			FLEN[2:0]		
0x14	COMPCTRL1	7:0		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE	
		15:8	SWAP		MUXPOS[2:0]			MUXNEG[2:0]		
		23:16								
		31:24			OUT[1:0]			FLEN[2:0]		
0x18	...									
0x1F	Reserved									
0x20	SYNCBUSY	7:0				COMPCTRL1	COMPCTRL0	WINCTRL	ENABLE	SWRST
		15:8								
		23:16								
		31:24								

## 39.7 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. See *Register Access Protection* from Related Links.

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. See *Synchronization* 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

[39.4.8. Register Access Protection](#)

[39.5.13. Synchronization](#)

### 39.7.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

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

#### Bit 1 – ENABLE Enable

Due to synchronization, there is delay from updating the register until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately and the corresponding bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE is cleared when the peripheral is enabled/disabled.

Value	Description
0	The AC is disabled.
1	The AC is enabled. Each comparator must also be enabled individually by the Enable bit in the Comparator Control register (COMPCTRLn.ENABLE).

#### 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 AC to their initial state, and the AC 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.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.



### 39.7.2 Control B

**Name:** CTRLB  
**Offset:** 0x01  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
							START1	START0
Access							R/W	R/W
Reset							0	0

#### Bits 0, 1 – STARTx Comparator x Start Comparison

Writing a '0' to this field has no effect.

Writing a '1' to STARTx starts a single-shot comparison on COMPx if both the Single-Shot and Enable bits in the Comparator x Control Register are '1' (COMPCTRLx.SINGLE and COMPCTRLx.ENABLE). If comparator x is not implemented, or if it is not enabled in single-shot mode, Writing a '1' has no effect.

This bit always reads as zero.

### 39.7.3 Event Control

**Name:** EVCTRL  
**Offset:** 0x02  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
			INVEI1	INVEI0			COMPEI1	COMPEI0
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0
Bit	7	6	5	4	3	2	1	0
				WINEO0			COMPEO1	COMPEO0
Access				R/W			R/W	R/W
Reset				0			0	0

#### Bits 12, 13 – INVEIx Inverted Event Input Enable x

Value	Description
0	Incoming event is not inverted for comparator x.
1	Incoming event is inverted for comparator x.

#### Bits 8, 9 – COMPEIx Comparator x Event Input

Note that several actions can be enabled for incoming events. If several events are connected to the peripheral, the enabled action will be taken for any of the incoming events. There is no way to tell which of the incoming events caused the action.

These bits indicate whether a comparison will start or not on any incoming event.

Value	Description
0	Comparison will not start on any incoming event.
1	Comparison will start on any incoming event.

#### Bit 4 – WINEO0 Window 0 Event Output Enable

These bits indicate whether the window 0 function can generate a peripheral event or not.

Value	Description
0	Window 0 Event is disabled.
1	Window 0 Event is enabled.

#### Bits 0, 1 – COMPEOx Comparator x Event Output Enable

These bits indicate whether the comparator x output can generate a peripheral event or not.

Value	Description
0	COMPx event generation is disabled.
1	COMPx event generation is enabled.

### 39.7.4 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x04  
**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
				WIN0			COMP1	COMP0
Access				R/W			R/W	R/W
Reset				0			0	0

#### Bit 4 – WIN0 Window 0 Interrupt Enable

Reading this bit returns the state of the Window 0 interrupt enable.  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit disables the Window 0 interrupt.

Value	Description
0	The Window 0 interrupt is disabled.
1	The Window 0 interrupt is enabled.

#### Bits 0, 1 – COMPx Comparator x Interrupt Enable

Reading this bit returns the state of the Comparator x interrupt enable.  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit disables the Comparator x interrupt.

Value	Description
0	The Comparator x interrupt is disabled.
1	The Comparator x interrupt is enabled.

### 39.7.5 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x05  
**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
Access				WIN0			COMP1	COMP0
Reset				R/W			R/W	R/W
				0			0	0

#### Bit 4 – WIN0 Window 0 Interrupt Enable

Reading this bit returns the state of the Window 0 interrupt enable.  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit enables the Window 0 interrupt.

Value	Description
0	The Window 0 interrupt is disabled.
1	The Window 0 interrupt is enabled.

#### Bits 0, 1 – COMPx Comparator x Interrupt Enable

Reading this bit returns the state of the Comparator x interrupt enable.  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit will set the Ready interrupt bit and enable the Ready interrupt.

Value	Description
0	The Comparator x interrupt is disabled.
1	The Comparator x interrupt is enabled.

### 39.7.6 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x06  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
Access				WIN0			COMP1	COMP0
Reset				R/W			R/W	R/W
				0			0	0

#### Bit 4 - WIN0 Window 0

This flag is set according to the Window 0 Interrupt Selection bit group in the [WINCTRL](#) register (WINCTRL.WINTSELx) and will generate an interrupt if INTENCLR/SET.WINx is also one. Writing a '0' to this bit has no effect. Writing a '1' to this bit clears the Window 0 interrupt flag.

#### Bits 0, 1 - COMPx Comparator x

Reading this bit returns the status of the Comparator x interrupt flag. If comparator x is not implemented, COMPx always reads as zero. This flag is set according to the Interrupt Selection bit group in the Comparator x Control register (COMPCTRLx.INTSEL) and will generate an interrupt if INTENCLR/SET.COMPx is also one. Writing a '0' to this bit has no effect. Writing a '1' to this bit clears the Comparator x interrupt flag.

### 39.7.7 Status A

**Name:** STATUSA  
**Offset:** 0x07  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
			WSTATE0[1:0]				STATE1	STATE0
Access			R	R			R	R
Reset			0	0			0	0

#### Bits 5:4 – WSTATE0[1:0] Window 0 Current State

These bits show the current state of the signal if the window 0 mode is enabled.

Value	Name	Description
0x0	ABOVE	Signal is above window
0x1	INSIDE	Signal is inside window
0x2	BELOW	Signal is below window
0x3		Reserved

#### Bits 0, 1 – STATEx Comparator x Current State

This bit shows the current state of the output signal from COMPx. STATEx is valid only when STATUSB.READYx is one.

### 39.7.8 Status B

**Name:** STATUSB  
**Offset:** 0x08  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
							READY1	READY0
Access							R	R
Reset							0	0

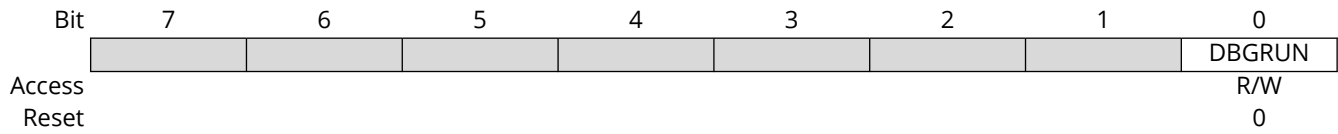
#### Bits 0, 1 – READYx Comparator x Ready

This bit is cleared when the comparator x output is not ready.

This bit is set when the comparator x output is ready.

### 39.7.9 Debug Control

**Name:** DBGCTRL  
**Offset:** 0x09  
**Reset:** 0x00  
**Property:** PAC Write-Protection



#### 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 AC is halted when the CPU is halted by an external debugger. Any on-going comparison will complete.
1	The AC continues normal operation when the CPU is halted by an external debugger.



### 39.7.10 Window Control

**Name:** WINCTRL  
**Offset:** 0x0A  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized

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

#### Bits 2:1 – WINTSELO[1:0] Window 0 Interrupt Selection

These bits configure the interrupt mode for the comparator window 0 mode.

Value	Name	Description
0x0	ABOVE	Interrupt on signal above window
0x1	INSIDE	Interrupt on signal inside window
0x2	BELOW	Interrupt on signal below window
0x3	OUTSIDE	Interrupt on signal outside window

#### Bit 0 – WENO Window 0 Mode Enable

Value	Description
0	Window mode is disabled for comparators 0 and 1.
1	Window mode is enabled for comparators 0 and 1.

### 39.7.11 Scaler 1

**Name:** SCALER1  
**Offset:** 0x0D  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

#### Bits 3:0 – VALUE[3:0] Scaler Value

These bits define the scaling factor for channel 1 of the VDD voltage scaler. The output voltage,  $V_{SCALE}$ , is:

$$V_{SCALE} = V_{DD} \times \left( \frac{R_{Bottom}}{R_{Total}} \right)$$

Where,  $R_{Total} = 900$ .

Refer to the following table for  $R_{Bottom}$  for different VALUE[3:0]

For example,  $V_{SCALE}$  for VALUE[3:0] = 0x02 at VDD = 3.3V

$$V_{SCALE} = 3.3 \times \left( \frac{598.5}{900} \right) = 2.1945V$$

**Table 39-3.** Scaler Value

Value[3:0]	R_Bottom
0x0	Reserved
0x1	Reserved
0x2	598.5
0x3	634.5
0x4	306
0x5	324
0x6	328.5
0x7	360
0x8	387
0x9	400.5
0xA	432
0xB	450
0xC	468
0xD	490.5
0xE	481.5
0xF	External Reference on LVDIN pin

### 39.7.12 Comparator Control n

**Name:** COMPCTRL  
**Offset:** 0x10 + n\*0x04 [n=0..1]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection

**Note:** 32 pins variants only have COMP1.

Bit	31	30	29	28	27	26	25	24
			OUT[1:0]				FLEN[2:0]	
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0
Bit	23	22	21	20	19	18	17	16
Access								
Reset								
Bit	15	14	13	12	11	10	9	8
	SWAP	MUXPOS[2:0]					MUXNEG[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	6	5	4	3	2	1	0
		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE	
Access		R/W		R/W	R/W	R/W	R/W	
Reset		0		0	0	0	0	

#### Bits 29:28 – OUT[1:0] Output

These bits configure the output selection for comparator n. COMPCTRLn.OUT can be written only while COMPCTRLn.ENABLE is zero.

**Note:** For internal use of the comparison results by the CCL, this must be 0x1 or 0x2.

These bits are not synchronized.

Value	Name	Description
0x0	OFF	The output of COMPn is not routed to the COMPn I/O port
0x1	ASYN	The asynchronous output of COMPn is routed to the COMPn I/O port
0x2	SYNC	The synchronous output (including filtering) of COMPn is routed to the COMPn I/O port
0x3	N/A	Reserved

#### Bits 26:24 – FLEN[2:0] Filter Length

These bits configure the filtering for comparator n. COMPCTRLn.FLEN can only be written while COMPCTRLn.ENABLE is zero.

These bits are not synchronized.

Value	Name	Description
0x0	OFF	No filtering
0x1	MAJ3	3-bit majority function (2 of 3)
0x2	MAJ5	5-bit majority function (3 of 5)
0x3–0x7	N/A	Reserved

#### Bit 15 – SWAP Swap Inputs and Invert

This bit swaps the positive and negative inputs to COMPn and inverts the output. This function can be used for offset cancellation. COMPCTRLn.SWAP can be written only while COMPCTRLn.ENABLE is zero.

These bits are not synchronized.

Value	Description
0	The output of MUXPOS connects to the positive input, and the output of MUXNEG connects to the negative input.
1	The output of MUXNEG connects to the positive input, and the output of MUXPOS connects to the negative input.

#### Bits 14:12 – MUXPOS[2:0] Positive Input Mux Selection

These bits select which input will be connected to the positive input of comparator n. COMPCTRLn.MUXPOS can be written only while COMPCTRLn.ENABLE is zero. These bits are not synchronized.

Value	Name	Description
0x0	PIN0	I/O pin 0
0x1	PIN1	I/O pin 1
0x2	PIN2	I/O pin 2
0x3	PIN3	I/O pin 3
0x4	VSCALE	VDD scaler
0x5–0x7	-	Reserved

#### Bits 10:8 – MUXNEG[2:0] Negative Input Mux Selection

These bits select which input will be connected to the negative input of comparator n. COMPCTRLn.MUXNEG can only be written while COMPCTRLn.ENABLE is zero. These bits are not synchronized.

Value	Name	Description
0x0	PIN0	I/O pin 0
0x1	PIN1	I/O pin 1
0x2	PIN2	I/O pin 2
0x3	PIN3	I/O pin 3
0x4	GND	Ground
0x5	VSCALE	VDD scaler
0x6	BANDGAP	Internal bandgap voltage

#### Bit 6 – RUNSTDBY Run in Standby

This bit controls the behavior of the comparator during standby sleep mode. This bit is not synchronized

Value	Description
0	The comparator is disabled during sleep.
1	The comparator continues to operate during sleep.

#### Bits 4:3 – INTSEL[1:0] Interrupt Selection

These bits select the condition for comparator n to generate an interrupt or event. COMPCTRLn.INTSEL can be written only while COMPCTRLn.ENABLE is zero. These bits are not synchronized.

Value	Name	Description
0x0	TOGGLE	Interrupt on comparator output toggle
0x1	RISING	Interrupt on comparator output rising
0x2	FALLING	Interrupt on comparator output falling
0x3	EOC	Interrupt on end of comparison (single-shot mode only)

#### Bit 2 – SINGLE Single-Shot Mode

This bit determines the operation of comparator n. COMPCTRLn.SINGLE can be written only while COMPCTRLn.ENABLE is zero. These bits are not synchronized.

Value	Description
0	Comparator n operates in continuous measurement mode.
1	Comparator n operates in single-shot mode.

**Bit 1 - ENABLE** Enable

Writing a zero to this bit disables comparator n.

Writing a one to this bit enables comparator n.

Due to synchronization, there is a delay from updating the register until the comparator is enabled/disabled. The value written to COMPCTRLn.ENABLE will read back immediately after being written. SYNCBUSY.COMPCTRLn is set. SYNCBUSY.COMPCTRLn is cleared when the peripheral is enabled/disabled.

Writing a one to COMPCTRLn.ENABLE will prevent further changes to the other bits in COMPCTRLn. These bits remain protected until COMPCTRLn.ENABLE is written to zero and the write is synchronized.

### 39.7.13 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x20  
**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				COMPCTRL1	COMPCTRL0	WINCTRL	ENABLE	SWRST
Reset				R	R	R	R	R
				0	0	0	0	0

#### Bits 3, 4 – COMPCTRLx COMPCTRLx Synchronization Busy

This bit is cleared when the synchronization of the COMPCTRLx register between the clock domains is complete.

This bit is set when the synchronization of the COMPCTRLx register between clock domains is started.

#### Bit 2 – WINCTRL WINCTRL Synchronization Busy

This bit is cleared when the synchronization of the WINCTRL register between the clock domains is complete.

This bit is set when the synchronization of the WINCTRL register between clock domains is started.

#### Bit 1 – ENABLE Enable Synchronization Busy

This bit is cleared when the synchronization of the CTRLA.ENABLE bit between the clock domains is complete.

This bit is set when the synchronization of the CTRLA.ENABLE bit between clock domains is started.

#### Bit 0 – SWRST Software Reset Synchronization Busy

This bit is cleared when the synchronization of the CTRLA.SWRST bit between the clock domains is complete.

This bit is set when the synchronization of the CTRLA.SWRST bit between clock domains is started.

## 40. Timer/Counter (TC)

### 40.1 Overview

There are up to four TC peripheral instances. Up to four TCs (TC[3:0]) are in PD1, whereas TC4, present in all device configurations, is always located in power domain PD0.

Each TC consists of a counter, a prescaler, compare/capture channels and control logic. The counter can be set to count events, or clock pulses. The counter, together with the compare/capture channels, can be configured to time stamp input events or IO pin edges, allowing for capturing of frequency and/or pulse width.

A TC can also perform waveform generation, such as frequency generation and pulse-width modulation.

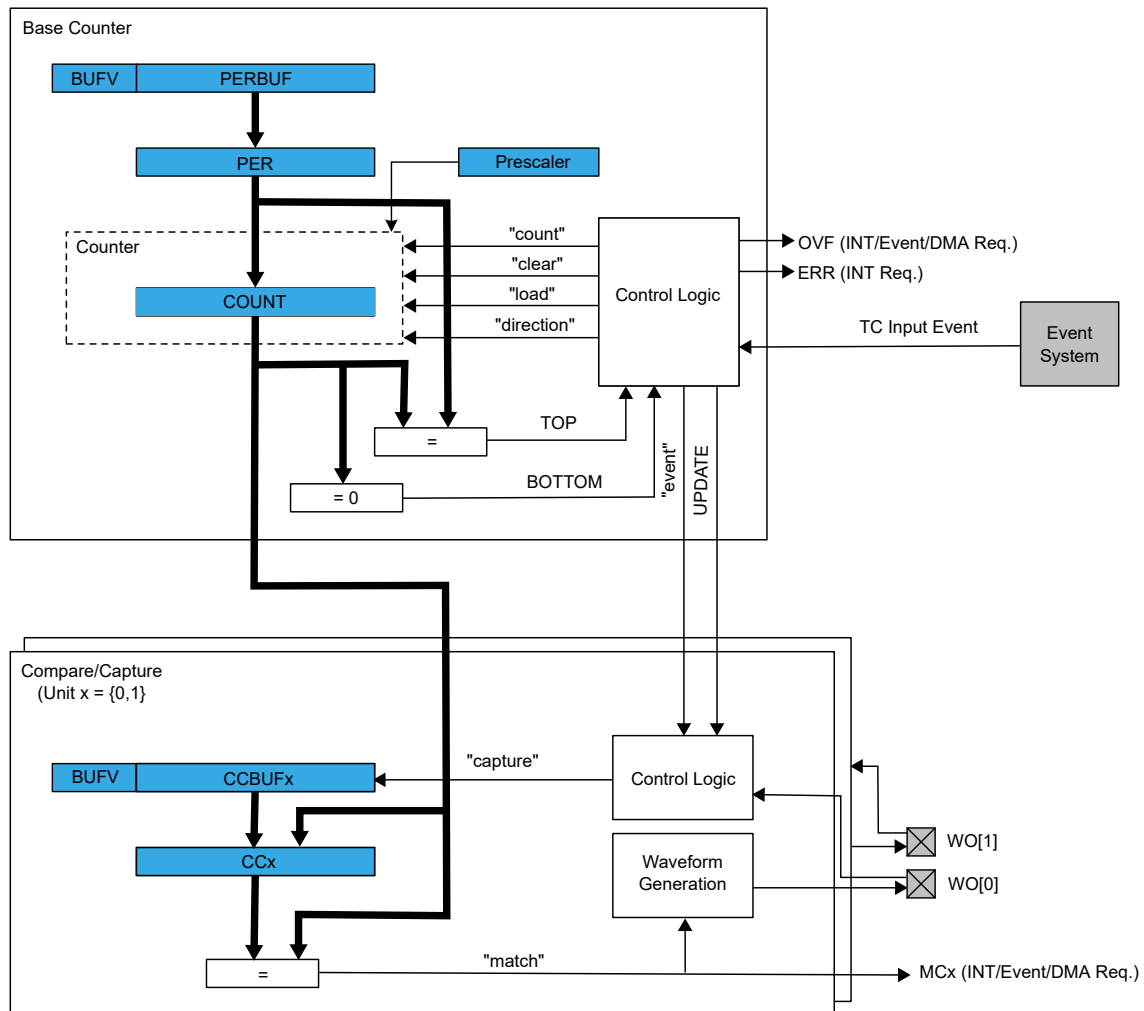
**Note:** Traditional Timer/Counter (TC) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

### 40.2 Features

- Selectable Configuration
  - 8-, 16- or 32-bit TC operation, with compare/capture channels
- 2 Compare/Capture Channels (CC) with:
  - Double buffered timer period setting (in 8-bit mode only)
  - Double buffered compare channel
- Waveform Generation
  - Frequency generation
  - Single-slope pulse-width modulation
- Input Capture
  - Event / IO pin edge capture
  - Frequency capture
  - Pulse-width capture
  - Time-stamp capture
  - Minimum and maximum capture
- One Input Event
- Interrupts/Output Events ON:
  - Counter overflow/underflow
  - Compare match or capture
- Internal Prescaler
- DMA Support

## 40.3 Block Diagram

Figure 40-1. Timer/Counter Block Diagram



## 40.4 Signal Description

Table 40-1. Signal Description for TC

Signal Name	Type	Description
WO[1:0]	Digital output	Waveform output
	Digital input	Capture input

See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

### Related Links

[6. I/O Ports and Peripheral Pin Select \(PPS\)](#)



## 40.5 Product Dependencies

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

### 40.5.1 I/O Lines

In order to use the I/O lines of this peripheral, the I/O pins must be configured using the I/O Peripheral Pin Select (PPS).

### 40.5.2 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. Events connected to the event system can trigger other operations in the system without exiting Sleep modes.

### 40.5.3 Clocks

The TC bus clocks (PB1\_CLK) can be enabled and disabled in the PMD Registers. For more details and default status of this clock, see *Peripheral Module Disable Register (PMD)* from Related Links.

The generic clocks (GCLK\_TCx) are asynchronous to the user interface clock (PB1\_CLK). Due to this asynchronicity, accessing certain registers will require synchronization between the clock domains.

**Note:** Two instances of the TC may share a peripheral clock channel. In this case, they cannot be set to different clock frequencies. See *System and Peripheral Clock Generation (CLKGEN)* from Related Links to identify shared peripheral clocks.

#### Related Links

[13.4. System and Peripheral Clock Generation \(CLKGEN\)](#)

[20. Peripheral Module Disable Register \(PMD\)](#)

### 40.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). In order 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 \(DMAC\)](#)

### 40.5.5 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 \(NVIC\)](#)

### 40.5.6 Events

The events of this peripheral are connected to the Event System.

#### Related Links

[28. Event System \(EVSYS\)](#)

### 40.5.7 Debug Operation

When the CPU is halted in Debug mode, this peripheral will halt normal operation. This peripheral can be forced to continue operation during debugging. For more details, see *DBGCTRL* from Related Links.

## Related Links

[40.7.2.11. DBGCTRL](#)

### 40.5.8 Register Access Protection

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

- Interrupt Flag Status and Clear register (INTFLAG)
- Status register (STATUS)
- Period and Period Buffer registers (PER, PERBUF)
- Compare/Capture Value registers and Compare/Capture Value Buffer registers (CCx, CCBUFx)

**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.

### 40.5.9 Analog Connections

Not applicable.

## 40.6 Functional Description

### 40.6.1 Principle of Operation

The following definitions are used throughout the documentation:

**Table 40-2.** Timer/Counter Definitions

Name	Description
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence. The TOP value can be the same as Period (PER) or the Compare Channel 0 (CC0) register value depending on the waveform generator mode in Waveform Output Operations. See <i>Waveform Output Operations</i> from Related Links.
ZERO	The counter is ZERO when it contains all zeros.
MAX	The counter reaches MAX when it contains all ones.
UPDATE	The timer/counter signals an update when it reaches ZERO or TOP, depending on the direction settings.
Timer	The timer/counter clock control is handled by an internal source.
Counter	The clock control is handled externally (e.g., counting external events).
CC	For compare operations, the CC are referred to as "compare channels." For capture operations, the CC are referred to as "capture channels."

Each TC instance has up to two compare/capture channels (CC0 and CC1).

The counter in the TC can either count events from the Event System or clock ticks of the GCLK\_TCx clock, which may be divided by the prescaler.

The counter value is passed to the CCx where it can be either compared to user-defined values or captured.

For optimized timing, the CCx and CCBUFx registers share a common resource. When writing into CCBUFx, lock the access to the corresponding CCx register (SYNCBUSY.CCX = 1) until the CCBUFx register value is not loaded into the CCx register (BUFVx == 1). Each buffer register has a buffer valid (BUFV) flag that indicates when the buffer contains a new value.

The Counter register (COUNT) and the Compare and Capture registers with buffers (CCx and CCBUFx) can be configured as 8-, 16- or 32-bit registers, with corresponding MAX values. Mode settings (CTRLA.MODE) determine the maximum range of the Counter register.

In 8-bit mode, a Period Value (PER) register and its Period Buffer Value (PERBUF) register are also available. The counter range and the operating frequency determine the maximum time resolution achievable with the TC peripheral.

The TC can be set to count up or down. Under normal operation, the counter value is continuously compared to the TOP or ZERO value to determine whether the counter has reached that value. On a comparison match, the TC can request DMA transactions, or generate interrupts or events for the Event System.

In a compare operation, the counter value is continuously compared to the values in the CCx registers. In the case of a match, the TC can request DMA transactions, or generate interrupts or events for the Event System. In waveform generator mode, these comparisons are used to set the waveform period or pulse width.

Capture operation can be enabled to perform input signal period and pulse width measurements, or to capture selectable edges from an IO pin or internal event from Event System.

### Related Links

[40.6.2.6.1. Waveform Output Operations](#)

## 40.6.2 Basic Operation

### 40.6.2.1 Initialization

The following registers are enable-protected, meaning that they can only be written when the TC is disabled (CTRLA.ENABLE=0):

- Control A register (CTRLA), except the Enable (ENABLE) and Software Reset (SWRST) bits
- Drive Control register (DRVCTRL)
- Wave register (WAVE)
- Event Control register (EVCTRL)

Writing to enable-protected bits and setting the CTRLA.ENABLE bit can be performed in a single 32-bit access of the CTRLA register. Writing to enable-protected bits and clearing the CTRLA.ENABLE bit cannot be performed in a single 32-bit access.

Before enabling the TC, the peripheral must be configured by the following steps:

1. Enable the TC bus clock if not already enabled by default (PB1\_CLK).
2. Select 8-, 16- or 32-bit counter mode via the TC Mode bit group in the Control A register (CTRLA.MODE). The default mode is 16-bit.
3. Select one wave generation operation in the Waveform Generation Operation bit group in the WAVE register (WAVE.WAVEGEN).
4. If desired, the GCLK\_TCx clock can be prescaled via the Prescaler bit group in the Control A register (CTRLA.PRESCALER).
  - If the prescaler is used, select a prescaler synchronization operation via the Prescaler and Counter Synchronization bit group in the Control A register (CTRLA.PRESYNC).
5. If desired, select one-shot operation by writing a '1' to the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT).
6. If desired, configure the counting direction 'down' (starting from the TOP value) by writing a '1' to the Counter Direction bit in the Control B register (CTRLBSET.DIR).
7. For capture operation, enable the individual channels to capture in the Capture Channel x Enable bit group in the Control A register (CTRLA.CAPTEN).
8. If desired, enable inversion of the waveform output or IO pin input signal for individual channels via the Invert Enable bit group in the Drive Control register (DRVCTRL.INVEN).

### 40.6.2.2 Enabling, Disabling, and Resetting

The TC is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The TC is disabled by writing a zero to CTRLA.ENABLE.

The TC is reset by writing a '1' to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the TC, except DBGCTRL, will be reset to their initial state. See CTRLA from Related Links.

The TC must be disabled before the TC is reset in order to avoid undefined behavior.

#### Related Links

[40.7.2.1. CTRLA](#)

### 40.6.2.3 Prescaler Selection

The GCLK\_TCx is fed into the internal prescaler.

The prescaler consists of a counter that counts up to the selected prescaler value, whereupon the output of the prescaler toggles.

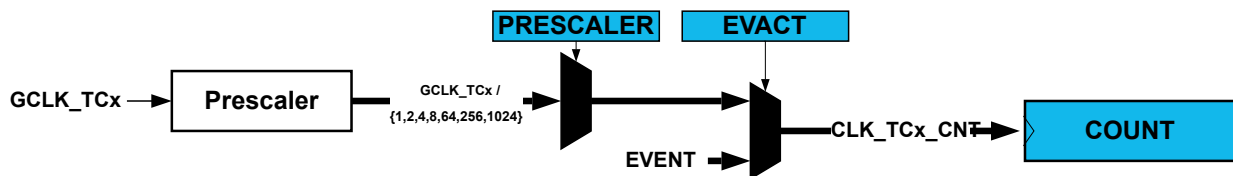
If the prescaler value is higher than one, the Counter Update condition can be optionally executed on the next GCLK\_TCx clock pulse or the next prescaled clock pulse. For further details, refer to Prescaler (CTRLA.PRESCALER) and Counter Synchronization (CTRLA.PRESYNC) description.

Prescaler outputs from 1 to 1/1024 are available. For a complete list of available prescaler outputs, see the register description for the Prescaler bit group in the Control A register (CTRLA.PRESCALER).

**Note:** When counting events, the prescaler is bypassed.

The joint stream of prescaler ticks and event action ticks is called CLK\_TCx\_CNT.

Figure 40-2. Prescaler



### 40.6.2.4 Counter Mode

The counter mode is selected by the Mode bit group in the Control A register (CTRLA.MODE). By default, the counter is enabled in the 16-bit counter resolution. Three counter resolutions are available:

- COUNT8: The 8-bit TC has its own Period Value and Period Buffer Value registers (PER and PERBUF).
- COUNT16: 16-bit is the default counter mode. There is no dedicated period register in this mode.
- COUNT32: 32-bit mode is achieved by pairing two 16-bit TC peripherals. TC(2n) is paired with TC(n+1).

When paired, the TC peripherals are configured using the registers of the even-numbered TC (TC0 or TC2, respectively).

The TC bus clocks (PB1\_CLK) for both host and client TCs need to be enabled.

The odd-numbered partner (TC1 or TC3, respectively) will act as a client, and the Client bit in the Status register (STATUS.SLAVE) will be set. The register values of a client will not reflect the registers of the 32-bit counter. Writing to any of the client registers will not affect the 32-bit counter. Normal access to the client COUNT and CCx registers is not allowed.

### 40.6.2.5 Counter Operations

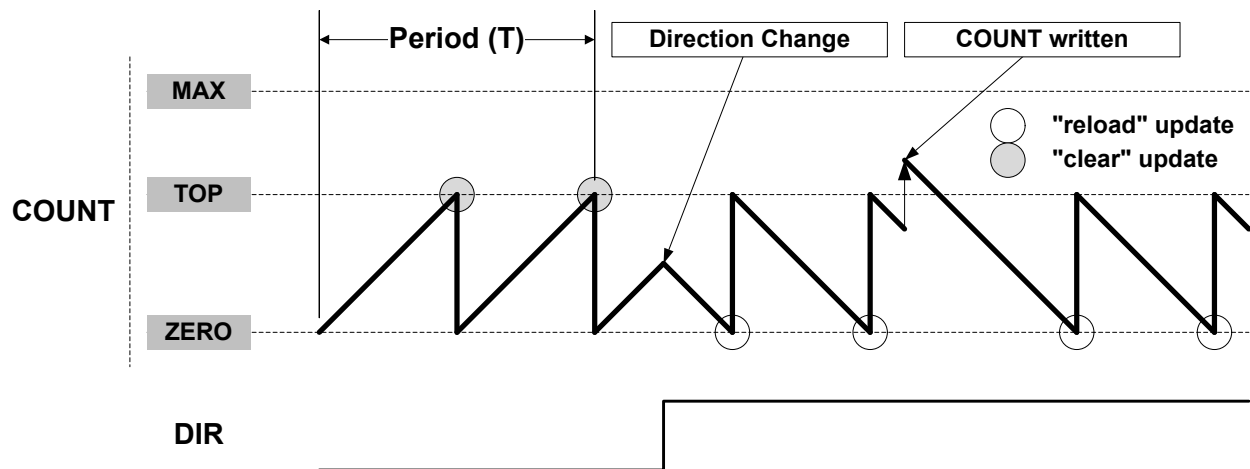
Depending on the mode of operation, the counter is cleared, reloaded, incremented, or decremented at each TC clock input (CLK\_TCx\_CNT). A counter clear or reload marks the end of the current counter cycle and the start of a new one.

The counting direction is set by the Direction bit in the Control B register (CTRLB.DIR). If this bit is zero the counter is counting up, and counting down if CTRLB.DIR=1. The counter will count up or down for each tick (clock or event) until it reaches TOP or ZERO. When it is counting up and TOP is reached, the counter will be set to zero at the next tick (overflow) and the Overflow Interrupt Flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF) will be set. When it is counting down, the counter is reloaded with the TOP value when ZERO is reached (underflow), and INTFLAG.OVF is set.

INTFLAG.OVF can be used to trigger an interrupt, a DMA request, or an event. An overflow/underflow occurrence (i.e., a compare match with TOP/ZERO) will stop counting if the One-Shot bit in the Control B register is set (CTRLBSET.ONESHOT).

It is possible to change the counter value (by writing directly in the COUNT register) even when the counter is running. When starting the TC, the COUNT value will be either ZERO or TOP (depending on the counting direction set by CTRLBSET.DIR or CTRLBCLR.DIR), unless a different value has been written to it, or the TC has been stopped at a value other than ZERO. The write access has higher priority than count, clear, or reload. The direction of the counter can also be changed when the counter is running. See also the following figure.

Figure 40-3. Counter Operation



Due to asynchronous clock domains, the internal counter settings are written when the synchronization is complete. Normal operation must be used when using the counter as timer base for the capture channels.

#### 40.6.2.5.1 Stop Command and Event Action

A Stop command can be issued from software by using Command bits in the Control B Set register (CTRLBSET.CMD = 0x2, STOP). When a Stop is detected while the counter is running, the counter will not retain its current value. All waveforms are cleared and the Stop bit in the Status register is set (STATUS.STOP).

#### 40.6.2.5.2 Re-Trigger Command and Event Action

A re-trigger command can be issued from software by writing the Command bits in the Control B Set register (CTRLBSET.CMD = 0x1, RETRIGGER), or from event when a re-trigger event action is configured in the Event Control register (EVCTRL.EVACT = 0x1, RETRIGGER).

When the command is detected during counting operation, the counter will be reloaded or cleared, depending on the counting direction (CTRLBSET.DIR or CTRLBCLR.DIR). When the re-trigger

command is detected while the counter is stopped, the counter will resume counting from the current value in the COUNT register.

**Note:** When a re-trigger event action is configured in the Event Action bits in the Event Control register (EVCTRL.EVACT=0x1, RETRIGGER), enabling the counter will not start the counter. The counter will start on the next incoming event and restart on corresponding following event.

#### 40.6.2.5.3 Count Event Action

The TC can count events. When an event is received, the counter increases or decreases the value, depending on direction settings (CTRLBSET.DIR or CTRLBCLR.DIR). The count event action can be selected by the Event Action bit group in the Event Control register (EVCTRL.EVACT=0x2, COUNT).

**Note:** If this operation mode is selected, PWM generation is not supported.

#### 40.6.2.5.4 Start Event Action

The TC can start counting operation on an event when previously stopped. In this configuration, the event has no effect if the counter is already counting. When the peripheral is enabled, the counter operation starts when the event is received or when a re-trigger software command is applied.

The Start TC on Event action can be selected by the Event Action bit group in the Event Control register (EVCTRL.EVACT=0x3, START).

#### 40.6.2.6 Compare Operations

By default, the Compare/Capture channel is configured for compare operations.

When using the TC and the Compare/Capture Value registers (CCx) for compare operations, the counter value is continuously compared to the values in the CCx registers. This can be used for timer or for waveform operation.

The Channel x Compare Buffer (CCBUFx) registers provide double buffer capability. The double buffering synchronizes the update of the CCx register with the buffer value at the UPDATE condition or a forced update command (CTRLBSET.CMD=UPDATE). See *Double Buffering* from Related Links. The synchronization prevents the occurrence of odd-length, non-symmetrical pulses and ensures glitch-free output.

##### Related Links

[40.6.2.7. Double Buffering](#)

#### 40.6.2.6.1 Waveform Output Operations

The compare channels can be used for waveform generation on output port pins. To make the waveform available on the connected pin, the following requirements must be fulfilled:

1. Choose a Waveform Generation mode in the Waveform Generation Operation bit in Waveform register (WAVE.WAVEGEN).
2. Optionally invert the waveform output WO[x] by writing the corresponding Output Waveform x Invert Enable bit in the Driver Control register (DRVCTRL.INVENx).
3. Configure the pins with the I/O Peripheral Pin Select (PPS). See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links.

**Note:** Event must not be used when the compare channel is set in waveform output operating mode.

The counter value is continuously compared with each CCx value. On a comparison match, the Match or Capture Channel x bit in the Interrupt Flag Status and Clear register (INTFLAG.MCx) will be set on the next zero-to-one transition of CLK\_TC\_CNT (see Normal Frequency Operation). An interrupt/and or event can be generated on comparison match if enabled. The same condition generates a DMA request.

There are four waveform configurations for the Waveform Generation Operation bit group in the Waveform register (WAVE.WAVEGEN). This will influence how the waveform is generated and impose restrictions on the top value. The configurations are:

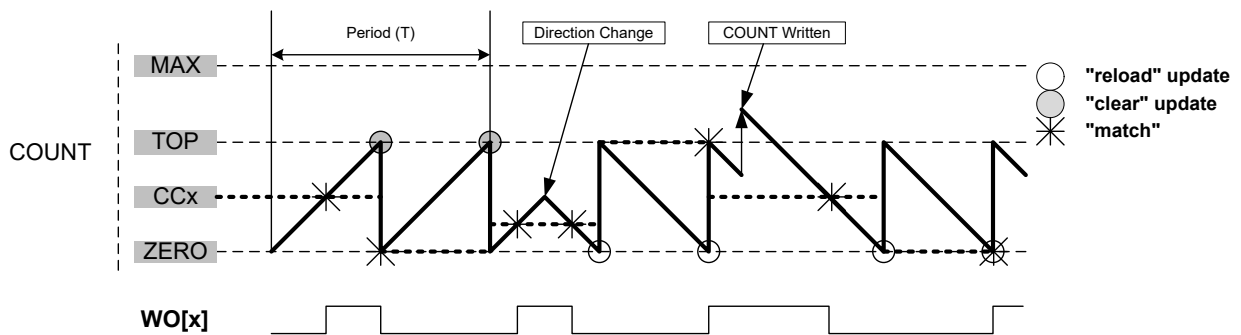
- Normal frequency (NFRQ)
- Match frequency (MFRQ)
- Normal pulse-width modulation (NPWM)
- Match pulse-width modulation (MPWM)

When using NPWM or NFRQ configuration, the TOP will be determined by the counter resolution. In 8-bit Counter mode, the Period register (PER) is used as TOP, and the TOP can be changed by writing to the PER register. In 16- and 32-bit Counter mode, TOP is fixed to the maximum (MAX) value of the counter.

### Normal Frequency Generation (NFRQ)

For Normal Frequency Generation, the period time (T) is controlled by the period register (PER) for 8-bit Counter mode and MAX for 16- and 32-bit mode. The waveform generation output (WO[x]) is toggled on each compare match between COUNT and CCx, and the corresponding Match or Capture Channel x Interrupt Flag (INTFLAG.MCx) will be set.

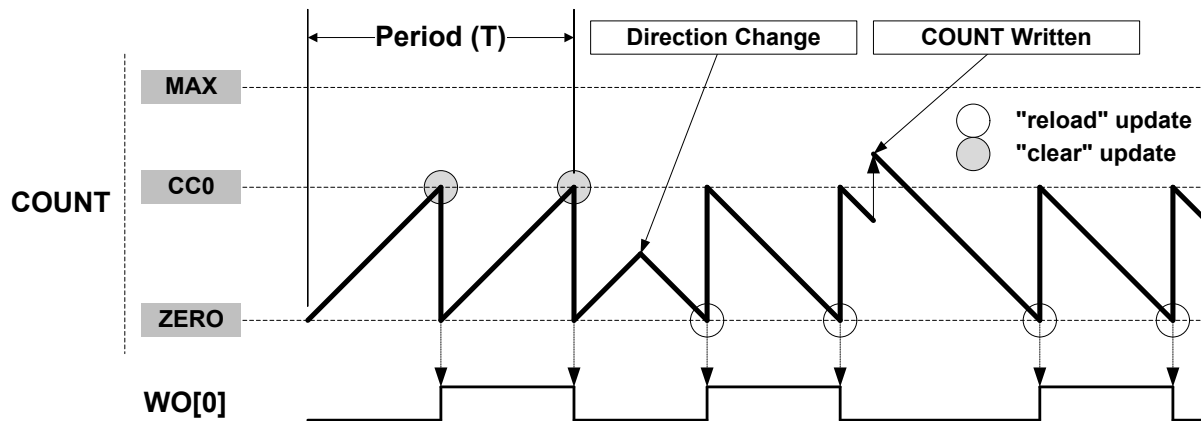
Figure 40-4. Normal Frequency Operation



### Match Frequency Generation (MFRQ)

For Match Frequency Generation, the period time (T) is controlled by the CC0 register instead of PER or MAX. WO[0] toggles on each Update condition.

Figure 40-5. Match Frequency Operation



### Normal Pulse-Width Modulation Operation (NPWM)

NPWM uses single-slope PWM generation.

For single-slope PWM generation, the period time (T) is controlled by the TOP value, and CCx controls the duty cycle of the generated waveform output. When up-counting, the WO[x] is set at start or compare match between the COUNT and TOP values, and cleared on compare match



between COUNT and CCx register values. When down-counting, the WO[x] is cleared at start or compare match between the COUNT and ZERO values, and set on compare match between COUNT and CCx register values.

The following equation calculates the exact resolution for a single-slope PWM ( $R_{PWM\_SS}$ ) waveform:

$$R_{PWM\_SS} = \frac{\log(TOP+1)}{\log(2)}$$

The PWM frequency ( $f_{PWM\_SS}$ ) depends on TOP value and the peripheral clock frequency ( $f_{GCLK\_TC}$ ), and can be calculated by the following equation:

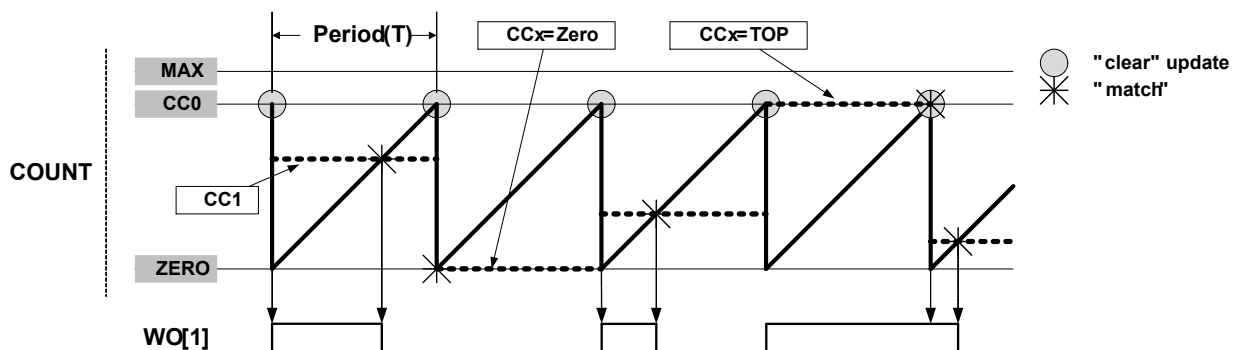
$$f_{PWM\_SS} = \frac{f_{GCLK\_TC}}{N(TOP+1)}$$

Where N represents the prescaler divider used (1, 2, 4, 8, 16, 64, 256, 1024).

### Match Pulse-Width Modulation Operation (MPWM)

In MPWM, the output of WO[1] is depending on CC1 as shown in the figure below. On every overflow/underflow, a one-TC-clock-cycle negative pulse is put out on WO[0] (not shown in the figure).

Figure 40-6. Match PWM Operation



The following table shows the Update Counter and Overflow Event/Interrupt Generation conditions in different operation modes.

Table 40-3. Counter Update and Overflow Event/interrupt Conditions in TC

Name	Operation	TOP	Update	Output Waveform		OVFIF/Event	
				On Match	On Update	Up	Down
NFRQ	Normal Frequency	PER	TOP/ ZERO	Toggle	Stable	TOP	ZERO
MFRQ	Match Frequency	CC0	TOP/ ZERO	Toggle	Stable	TOP	ZERO
NPWM	Single-slope PWM	PER	TOP/ ZERO	See description above.		TOP	ZERO
MPWM	Single-slope PWM	CC0	TOP/ ZERO	Toggle	Toggle	TOP	ZERO

### Related Links

#### [6. I/O Ports and Peripheral Pin Select \(PPS\)](#)

#### 40.6.2.7 Double Buffering

The Compare Channels (CCx) registers, and the Period (PER) register in 8-bit mode are double buffered. Each buffer register has a buffer valid bit (CCBUFVx or PERBUFV) in the STATUS register, which indicates that the buffer register contains a new valid value that can be copied into the corresponding register. As long as the respective buffer valid status flag (PERBUFV or CCBUFVx) are set to '1', related syncbusy bits are set (SYNCBUSY.PER or SYNCBUSY.CCx), a write to the respective PER/PERBUF or CCx/CCBUFx registers will generate a PAC error, and access to the respective PER or CCx register is invalid.

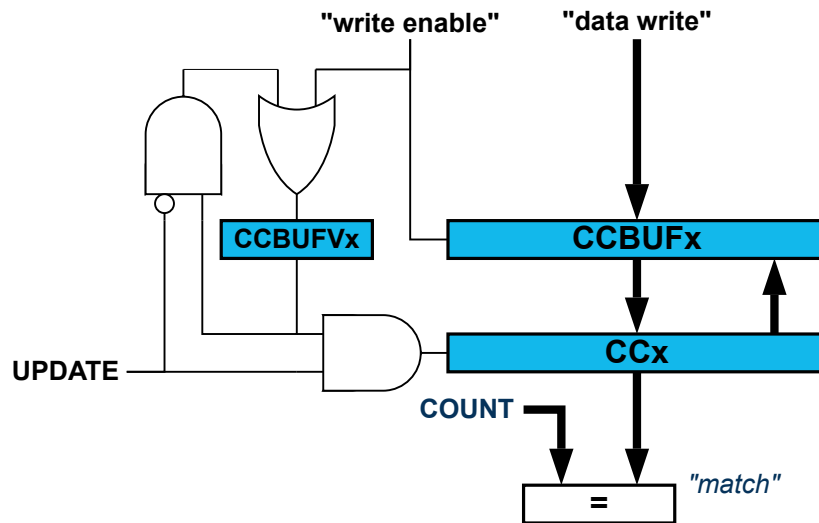


When the buffer valid flag bit in the STATUS register is '1' and the Lock Update bit in the CTRLB register is set to '0', (writing CTRLBCLR.LUPD to '1'), double buffering is enabled: the data from buffer registers will be copied into the corresponding register under hardware UPDATE conditions, then the buffer valid flags bit in the STATUS register are automatically cleared by hardware.

**Note:** The software update command (CTRLBSET.CMD=0x3) is acting independently of the LUPD value.

A compare register is double buffered as in the following figure.

**Figure 40-7.** Compare Channel Double Buffering



Both the registers (PER/CCx) and corresponding buffer registers (PERBUF/CCBUFx) are available in the I/O register map, and the double buffering feature is not mandatory. The double buffering is disabled by writing a '1' to CTRLBSET.LUPD.

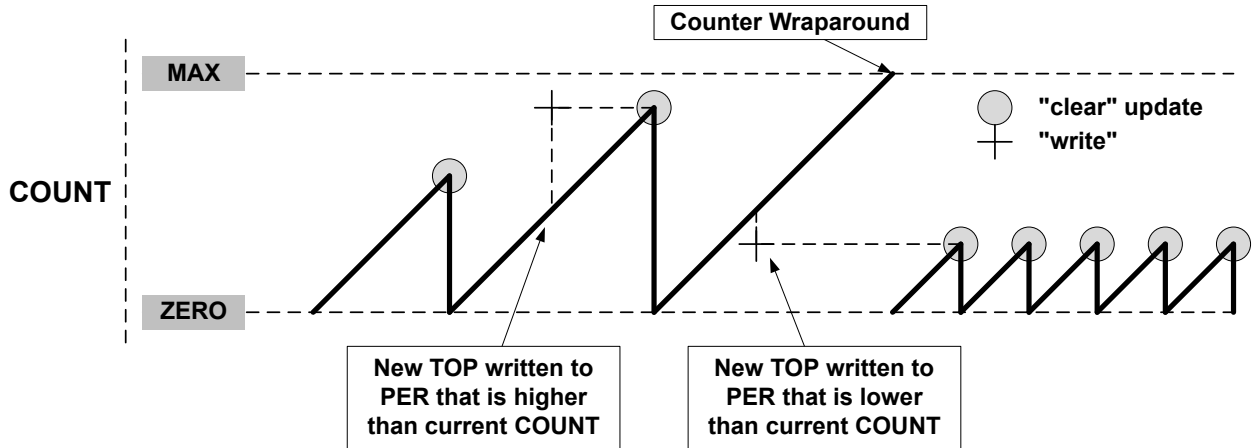
**Note:** In NFRQ, MFRQ or PWM, down-counting counter mode (CTRLBSET.DIR=1), when double buffering is enabled (CTRLBCLR.LUPD=1), PERBUF register is continuously copied into the PER independently of update conditions.

### Changing the Period

The counter period can be changed by writing a new TOP value to the Period register (PER or CC0, depending on the waveform generation mode), which is available in 8-bit mode. Any period update on registers (PER or CCx) is effective after the synchronization delay.

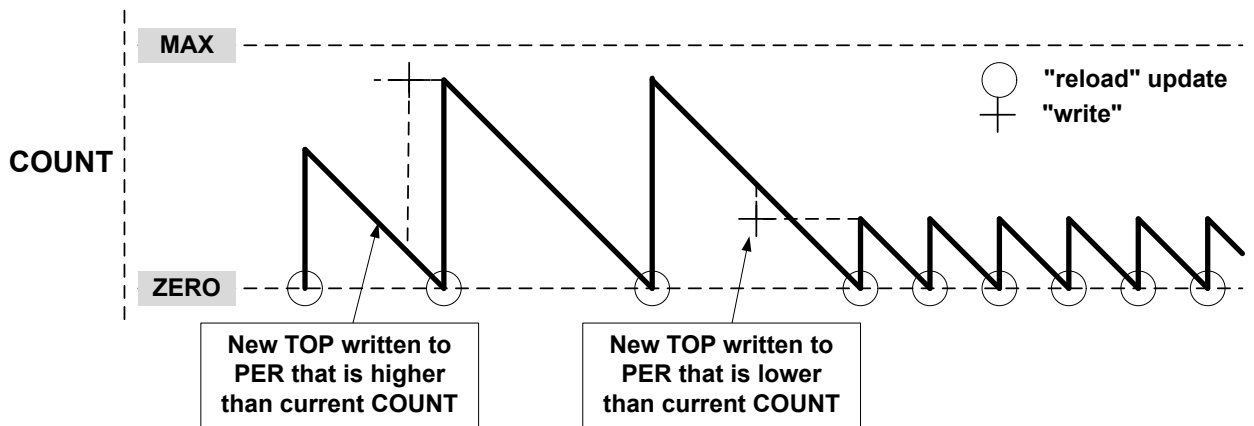
A counter wraparound can occur in any operation mode when up-counting without buffering (see the following figure).

Figure 40-8. Unbuffered Single-Slope Up-Counting Operation



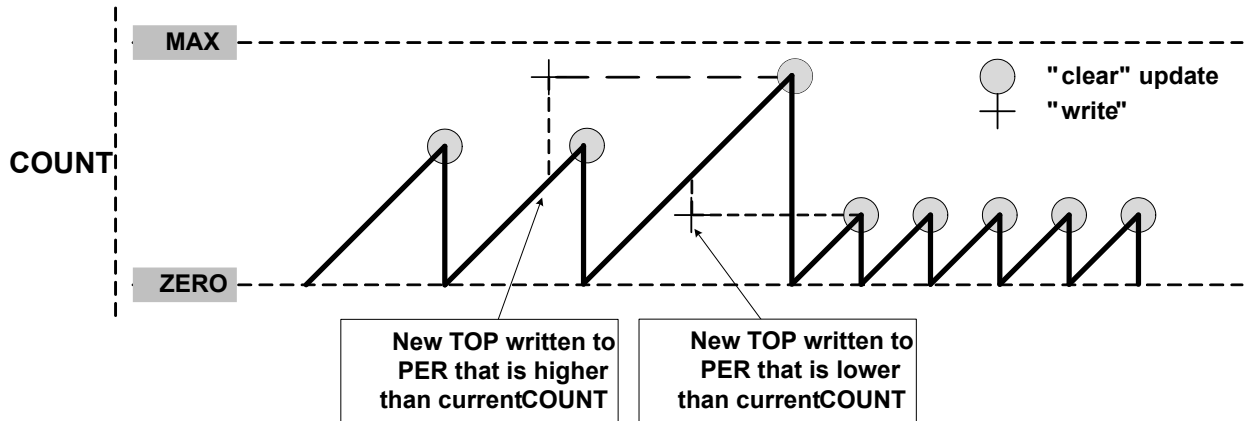
COUNT and TOP are continuously compared, so when a new TOP value that is lower than current COUNT is written to TOP, COUNT will wrap, before a compare match.

Figure 40-9. Unbuffered Single-Slope Down-Counting Operation



When double buffering is used, the buffer can be written at any time and the counter will still maintain correct operation. The period register is always updated on the update condition, as shown in the following figure. This prevents wraparound and the generation of odd waveforms.

Figure 40-10. Changing the Period Using Buffering



### 40.6.2.8 Capture Operations

To enable and use capture operations, the corresponding Capture Channel x Enable bit in the Control A register (CTRLA.CAPTENx) must be written to '1'.

A capture trigger can be provided by input event line TC\_EV or by asynchronous IO pin WO[x] for each capture channel or by a TC event. To enable the capture from input event line, Event Input Enable bit in the Event Control register (EVCTRL.TCEI) must be written to '1'. To enable the capture from the IO pin, the Capture On Pin x Enable bit in CTRLA register (CTRLA.COPENx) must be written to '1'.

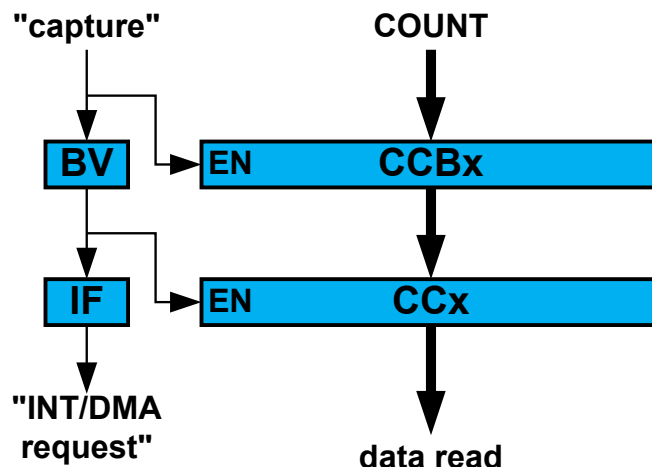
#### Notes:

1. The RETRIGGER, COUNT and START event actions are available only on an event from the Event System.
2. Event system channels must be configured to operate in asynchronous mode of operation when used for capture operations.

By default, a capture operation is done when a rising edge is detected on the input signal. Capture on falling edge is available, its activation is depending on the input source:

- When the channel is used with a IO pin, write a '1' to the corresponding Invert Enable bit in the Drive Control register (DRVCTRL.INVENx).
- When the channel is counting events from the Event System, write a '1' to the TC Event Input Invert Enable bit in Event Control register (EVCTRL.TCINV).

Figure 40-11. Capture Double Buffering



For input capture, the buffer register and the corresponding CCx act like a FIFO. When CCx is empty or read, any content in CCBUFx is transferred to CCx. The buffer valid flag is passed to set the CCx interrupt flag (IF) and generate the optional interrupt, event or DMA request. The CCBUFx register value can't be read, all captured data must be read from CCx register.

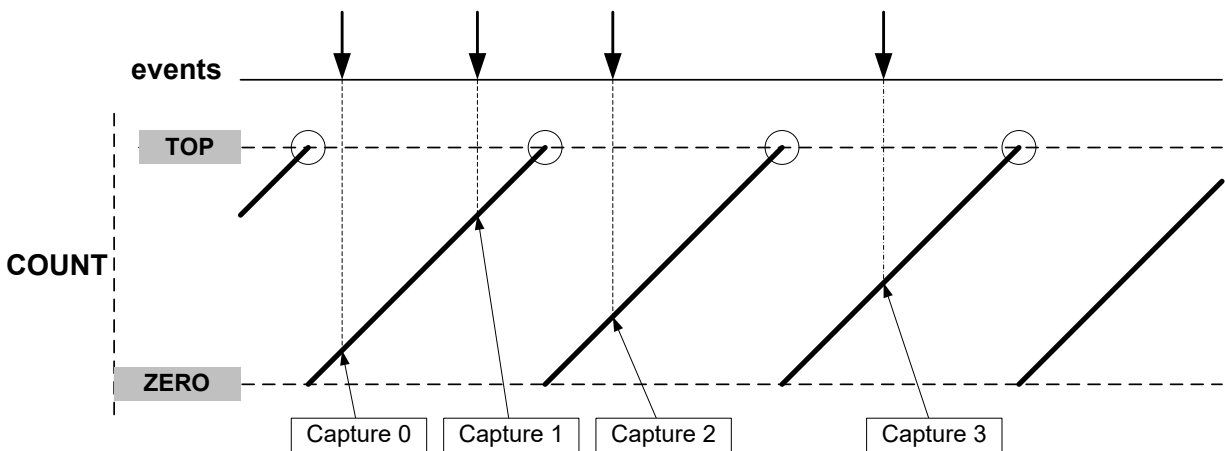
**Note:**

When up-counting (CTRLBSET.DIR=0), counter values lower than 1 cannot be captured. To capture the full range including value 0, the TC must be in down-counting mode (CTRLBSET.DIR=0).

**40.6.2.8.1 Event Capture Action**

The compare/capture channels can be used as input capture channels to capture events from the Event System and give them a timestamp. The following figure shows four capture events for one capture channel.

**Figure 40-12.** Input Capture Timing



The TC can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Interrupt flag (INTFLAG.MCx) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

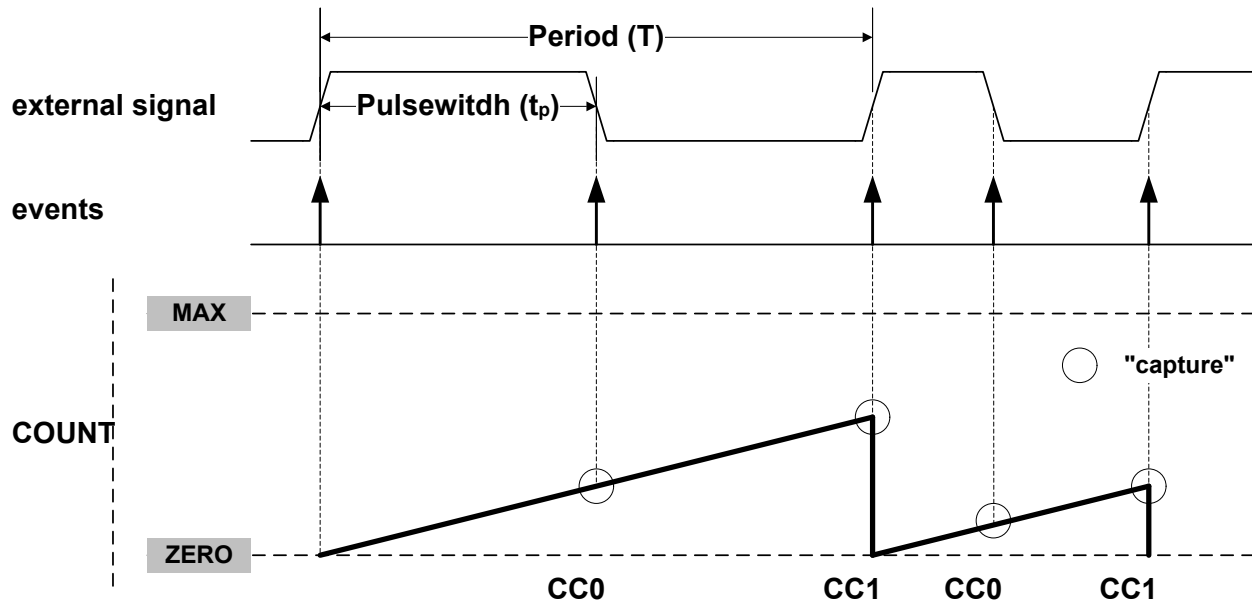
**40.6.2.8.2 Period and Pulse-Width (PPW) Capture Action**

The TC can perform two input captures and restart the counter on one of the edges. This enables the TC to measure the pulse width and period and to characterize the frequency  $f$  and duty cycle of an input signal:

$$f = \frac{1}{T}$$

$$\text{dutyCycle} = \frac{t_p}{T}$$

Figure 40-13. PWP Capture



Selecting PWP in the Event Action bit group in the Event Control register (EVCTRL.EVACT) enables the TC to perform one capture action on the rising edge and the other one on the falling edge. The period  $T$  will be captured into CC1 and the pulse width  $t_p$  in CC0. EVCTRL.EVACT=PPW (period and pulse-width) offers identical functionality, but will capture  $T$  into CC0 and  $t_p$  into CC1.

The TC Event Input Invert Enable bit in the Event Control register (EVCTRL.TCINV) is used to select whether the wraparound must occur on the rising edge or the falling edge. If EVCTRL.TCINV=1, the wraparound will happen on the falling edge. In case pin capture is enabled, this can also be achieved by modifying the value of the DRVCTRL.INVENx bit.

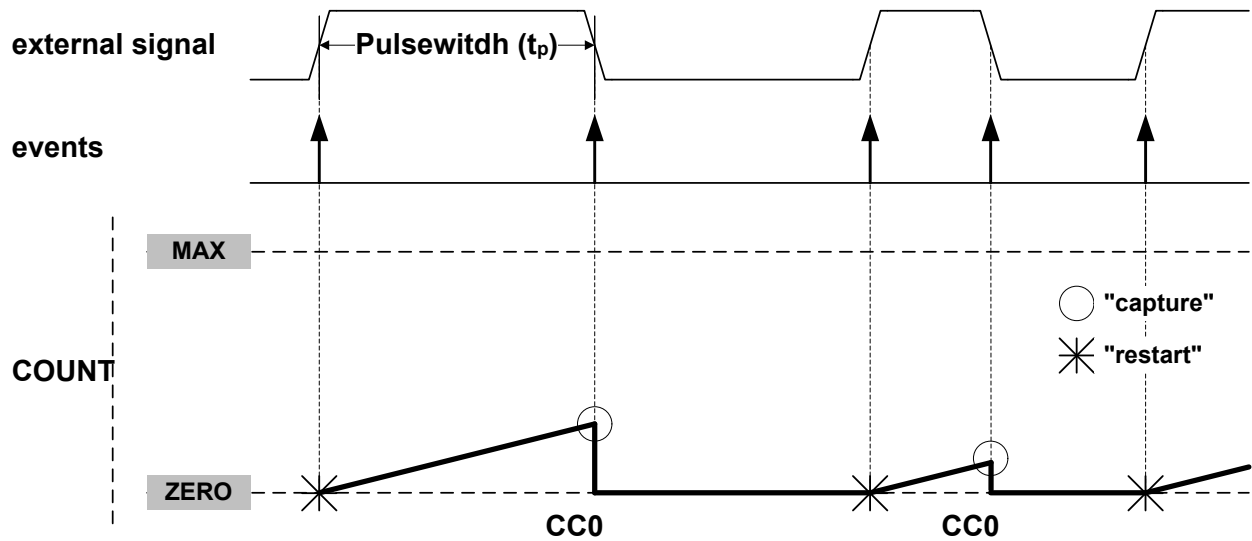
The TC can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Interrupt flag (INTFLAG.MCx) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

**Note:** The corresponding capture is working only if the channel is enabled in capture mode (CTRLA.CAPTENx=1). If not, the capture action is ignored and the channel is enabled in compare mode of operation. Consequently, both channels must be enabled in order to fully characterize the input.

#### 40.6.2.8.3 Pulse-Width Capture Action

The TC performs the input capture on the falling edge of the input signal. When the edge is detected, the counter value is cleared and the TC stops counting. When a rising edge is detected on the input signal, the counter restarts the counting operation. To enable the operation on opposite edges, the input signal to capture must be inverted (refer to DRVCTRL.INVEN or EVCTRL.TCINV).

Figure 40-14. Pulse-Width Capture on Channel 0



The TC can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Interrupt flag (INTFLAG.MC<sub>x</sub>) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

### 40.6.3 Additional Features

#### 40.6.3.1 One-Shot Operation

When one-shot is enabled, the counter automatically stops on the next Counter Overflow or Underflow condition. When the counter is stopped, the Stop bit in the Status register (STATUS.STOP) is automatically set and the waveform outputs are set to zero.

One-shot operation is enabled by writing a '1' to the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT), and disabled by writing a '1' to CTRLBCLR.ONESHOT. When enabled, the TC will count until an overflow or underflow occurs and stops counting operation. The one-shot operation can be restarted by a re-trigger software command, a re-trigger event, or a start event. When the counter restarts its operation, STATUS.STOP is automatically cleared.

#### 40.6.3.2 Time-Stamp Capture

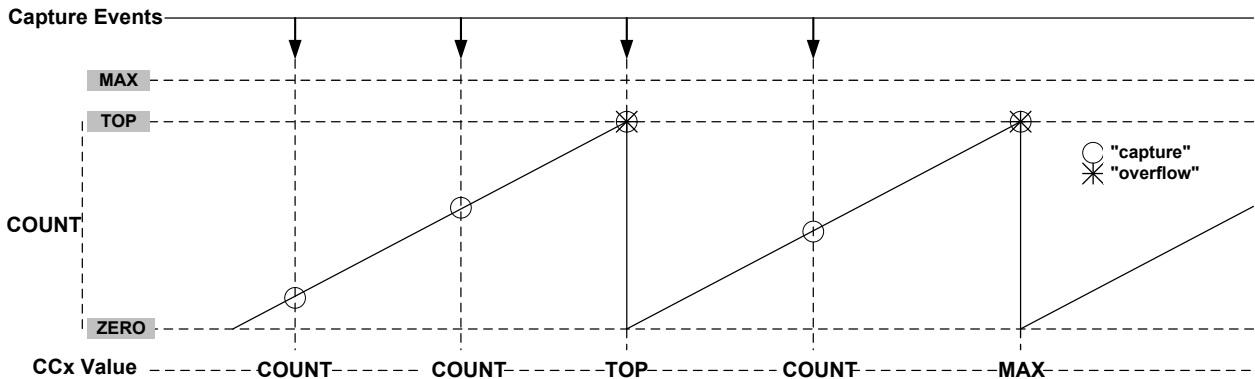
This feature is enabled when the Capture Time Stamp (STAMP) Event Action in Event Control register (EVCTRL.EVACT) is selected. The counter TOP value must be smaller than MAX.

When a capture event is detected, the COUNT value is copied into the corresponding Channel x Compare/Capture Value (CC<sub>x</sub>) register. In case of an overflow, the MAX value is copied into the corresponding CC<sub>x</sub> register.

When a valid captured value is present in the capture channel register, the corresponding Capture Channel x Interrupt Flag (INTFLAG.MC<sub>x</sub>) is set.

The timer/counter can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Channel interrupt flag (INTFLAG.MC<sub>x</sub>) is still set, the new time-stamp will not be stored and INTFLAG.ERR will be set.

Figure 40-15. Time-Stamp



#### 40.6.3.3 Minimum Capture

The minimum capture is enabled by writing the CAPTMIN mode in the Channel n Capture Mode bits in the Control A register (CTRLA.CAPTMODEn = CAPTMIN).

##### CCx Content:

In CAPTMIN operations, CCx keeps the Minimum captured values. Before enabling this mode of capture, the user must initialize the corresponding CCx register value to a value different from zero. If the CCx register initial value is zero, no captures will be performed using the corresponding channel.

##### MCx Behaviour:

In CAPTMIN operation, capture is performed only when on capture event time. The counter value is lower than the last captured value. The MCx interrupt flag is set only when on capture event time. The counter value is higher or equal to the value captured on the previous event. Therefore, the interrupt flag is set when a new absolute local Minimum value is detected.

#### 40.6.3.4 Maximum Capture

The maximum capture is enabled by writing the CAPTMAX mode in the Channel n Capture Mode bits in the Control A register (CTRLA.CAPTMODEn = CAPTMAX).

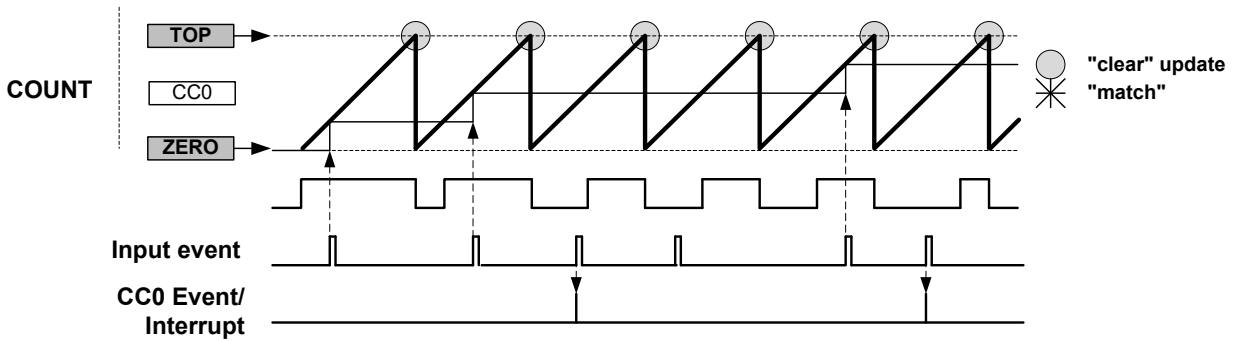
##### CCx Content:

In CAPTMAX operations, CCx keeps the Maximum captured values. Before enabling this mode of capture, the user must initialize the corresponding CCx register value to a value different from TOP. If the CCx register initial value is TOP, no captures will be performed using the corresponding channel.

##### MCx Behaviour:

In CAPTMAX operation, capture is performed only when on capture event time. The counter value is higher than the last captured value. The MCx interrupt flag is set only when on capture event time. The counter value is lower or equal to the value captured on the previous event. Therefore, the interrupt flag is set when a new absolute local Maximum value is detected.

**Figure 40-16.** Maximum Capture Operation with CC0 Initialized with ZERO Value



#### 40.6.4 DMA Operation

The TC can generate the following DMA requests:

- Overflow (OVF): the request is set when an update condition (overflow, underflow or re-trigger) is detected, the request is cleared by hardware on DMA acknowledge.
- Match or Capture Channel x (MCx): for a compare channel, the request is set on each compare match detection, the request is cleared by hardware on DMA acknowledge. For a capture channel, the request is set when valid data is present in the CCx register, and cleared when CCx register is read.

#### 40.6.5 Interrupts

The TC has the following interrupt sources:

- Overflow/Underflow (OVF)
- Match or Capture Channel x (MCx)
- Capture Overflow Error (ERR)

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) 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 register (INTENSET), and disabled by writing a '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 either the interrupt flag is cleared, the interrupt is disabled, or the TC is reset. See *INTFLAG* from Related Links for more details on how to clear the interrupt flags.

The TC has one common interrupt request line for all the interrupt sources. 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. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

#### Related Links

[10.2. Nested Vector Interrupt Controller \(NVIC\)](#)

[40.7.4.7. INTFLAG](#)

#### 40.6.6 Events

The TC can generate the following output events:

- Overflow/Underflow (OVF)



- Match or Capture Channel x (MCX0-1)

Writing a '1' to an Event Output bit in the Event Control register (EVCTRL.MCEOx) enables the corresponding output event. The output event is disabled by writing EVCTRL.MCEOx=0.

One of the following event actions can be selected by the Event Action bit group in the Event Control register (EVCTRL.EVACT):

- Disable event action (OFF)
- Start TC (START)
- Re-trigger TC (RETRIGGER)
- Count on event (COUNT)
- Capture time stamp (STAMP)
- Capture Period (PPW and PWP)
- Capture Pulse Width (PW)

Writing a '1' to the TC Event Input bit in the Event Control register (EVCTRL.TCEI) enables input events (EVU0-2) to the TC. Writing a '0' to this bit disables input events to the TC. The TC requires only asynchronous event inputs. See *Event System (EVSYS)* from Related Links for additional information on configuring the asynchronous events.

#### Related Links

[28. Event System \(EVSYS\)](#)

### 40.6.7 Sleep Mode Operation

The TC can be configured to operate in any sleep mode. To be able to run in standby, the RUNSTDBY bit in the Control A register (CTRLA.RUNSTDBY) must be '1'. This peripheral can wake up the device from any sleep mode using interrupts or perform actions through the Event System.

If the On Demand bit in the Control A register (CTRLA.ONDEMAND) is written to '1', the module stops requesting its peripheral clock when the STOP bit in STATUS register (STATUS.STOP) is set to '1'. When a re-trigger or start condition is detected, the TC requests the clock before the operation starts.

### 40.6.8 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset and Enable bits in Control A register (CTRLA.SWRST and CTRLA.ENABLE)
- Capture Channel Buffer Valid bit in STATUS register (STATUS.CCBUFVx)

The following registers are synchronized when written:

- Control B Clear and Control B Set registers (CTRLBCLR and CTRLBSET)
- Count Value register (COUNT)
- Period Value and Period Buffer Value registers (PER and PERBUF)
- Channel x Compare/Capture Value and Channel x Compare/Capture Buffer Value registers (CCx and CCBUFx)

The following registers are synchronized when read:

- Count Value register (COUNT): synchronization is done on demand through READSYNC command (CTRLBSET.CMD)
- Control B Clear and Control B Set registers (CTRLBCLR and CTRLBSET)

- Channel x Compare/Capture Value (CCx)

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.

## 40.7 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.

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](#)

## 40.7.1 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
0x00	CTRLA	7:0	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST	
		15:8	DMAOS				ALOCK	PRESCALER[2:0]			
		23:16			COPEN1	COPEN0			CAPTEN1	CAPTEN0	
		31:24				CAPTMODE1[1:0]			CAPTMODE0[1:0]		
0x04	CTRLBCLR	7:0	CMD[2:0]					ONESHOT	LUPD	DIR	
0x05	CTRLBSET	7:0	CMD[2:0]					ONESHOT	LUPD	DIR	
0x06	EVCTRL	7:0			TCEI	TCINV		EVACT[2:0]			
		15:8			MCEO1	MCEO0				OVFEO	
0x08	INTENCLR	7:0			MC1	MC0			ERR	OVF	
0x09	INTENSET	7:0			MC1	MC0			ERR	OVF	
0x0A	INTFLAG	7:0			MC1	MC0			ERR	OVF	
0x0B	STATUS	7:0			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP	
0x0C	WAVE	7:0							WAVEGEN[1:0]		
0x0D	DRVCTRL	7:0							INVEN1	INVEN0	
0x0E	Reserved										
0x0F	DBGCTRL	7:0								DBGRUN	
0x10	SYNCBUSY	7:0	CC1	CC0	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST	
0x11	Reserved										
...											
0x13											
0x14	COUNT	7:0	COUNT[7:0]								
0x15	Reserved										
...											
0x1A											
0x1B	PER	7:0	PER[7:0]								
0x1C	CC0	7:0	CC[7:0]								
0x1D	CC1	7:0	CC[7:0]								
0x1E	Reserved										
...											
0x2E											
0x2F	PERBUF	7:0	PERBUF[7:0]								
0x30	CCBUF0	7:0	CCBUF[7:0]								
0x31	CCBUF1	7:0	CCBUF[7:0]								

## 40.7.2 Register Description - 8-bit Mode

## 40.7.2.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized, Enable-Protected

Bit	31	30	29	28	27	26	25	24
			CAPTMODE1[1:0]				CAPTMODE0[1:0]	
Access			R/W		R/W		R/W	
Reset			0		0		0	
Bit	23	22	21	20	19	18	17	16
			COPEN1	COPEN0			CAPTEN1	CAPTEN0
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0
Bit	15	14	13	12	11	10	9	8
	DMAOS					ALOCK	PRESCALER[2:0]	
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0
Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	W
Reset	0	0	0	0	0	0	0	0

### Bits 28:27 – CAPTMODE1[1:0] Capture mode Channel 1

These bits select the channel 1 capture mode.

Value	Name	Description
0x0	DEFAULT	Default capture
0x1	CAPTMIN	Minimum capture
0x2	CAPTMAX	Maximum capture
0x3		Reserved

### Bits 25:24 – CAPTMODE0[1:0] Capture mode Channel 0

These bits select the channel 0 capture mode.

Value	Name	Description
0x0	DEFAULT	Default capture
0x1	CAPTMIN	Minimum capture
0x2	CAPTMAX	Maximum capture
0x3		Reserved

### Bits 20, 21 – COPENx Capture On Pin x Enable [x=1..0]

Bit x of COPEN[1:0] selects the trigger source for capture operation, either events or I/O pin input. This bit is not synchronized.

Value	Description
0	Event from Event System is selected as trigger source for capture operation on channel x.
1	I/O pin is selected as trigger source for capture operation on channel x.

### Bits 16, 17 – CAPTENx Capture Channel x Enable [x=1..0]

Bit x of CAPTEN[1:0] selects whether channel x is a capture or a compare channel. These bits are not synchronized.

Value	Description
0	CAPTEN disables capture on channel x.
1	CAPTEN enables capture on channel x.

**Bit 15 – DMAOS** DMA One-Shot Trigger Mode

This bit enables the DMA One-shot Trigger Mode.  
Writing a '1' to this bit will generate a DMA trigger on TC cycle following a TC\_CTRLBSET\_CMD\_DMAOS command.  
Writing a '0' to this bit will generate DMA triggers on each TC cycle.  
This bit is not synchronized.

**Bit 11 – ALOCK** Auto Lock

When this bit is set, Lock bit update (LUPD) is set to '1' on each overflow/underflow or re-trigger event.  
This bit is not synchronized.

Value	Description
0	The LUPD bit is not affected on overflow/underflow, and re-trigger event.
1	The LUPD bit is set on each overflow/underflow or re-trigger event.

**Bits 10:8 – PRESCALER[2:0]** Prescaler

These bits select the counter prescaler factor.  
These bits are not synchronized.

Value	Name	Description
0x0	DIV1	Prescaler: GCLK_TC
0x1	DIV2	Prescaler: GCLK_TC/2
0x2	DIV4	Prescaler: GCLK_TC/4
0x3	DIV8	Prescaler: GCLK_TC/8
0x4	DIV16	Prescaler: GCLK_TC/16
0x5	DIV64	Prescaler: GCLK_TC/64
0x6	DIV256	Prescaler: GCLK_TC/256
0x7	DIV1024	Prescaler: GCLK_TC/1024

**Bit 7 – ONDEMAND** Clock On Demand

This bit selects the clock requirements when the TC is stopped.  
In standby mode, if the Run in Standby bit (CTRLA.RUNSTDBY) is '0', ONDEMAND is forced to '0'.  
This bit is not synchronized.

Value	Description
0	The On Demand is disabled. If On Demand is disabled, the TC will continue to request the clock when its operation is stopped (STATUS.STOP=1).
1	The On Demand is enabled. When On Demand is enabled, the stopped TC will not request the clock. The clock is requested when a software re-trigger command is applied or when an event with start/re-trigger action is detected.

**Bit 6 – RUNSTDBY** Run in Standby

This bit is used to keep the TC running in standby mode.  
This bit is not synchronized.

Value	Description
0	The TC is halted in standby.
1	The TC continues to run in standby.

**Bits 5:4 – PRESCSYNC[1:0]** Prescaler and Counter Synchronization

These bits select whether the counter must wrap around on the next GCLK\_TCx clock or the next prescaled GCLK\_TCx clock. It also makes it possible to reset the prescaler.  
These bits are not synchronized.

Value	Name	Description
0x0	GCLK	Reload or reset the counter on next generic clock

Value	Name	Description
0x1	PRESC	Reload or reset the counter on next prescaler clock
0x2	RESYNC	Reload or reset the counter on next generic clock. Reset the prescaler counter
0x3	-	Reserved

**Bits 3:2 – MODE[1:0]** Timer Counter Mode

These bits select the counter mode.  
 These bits are not synchronized.

Value	Name	Description
0x0	COUNT16	Counter in 16-bit mode
0x1	COUNT8	Counter in 8-bit mode
0x2	COUNT32	Counter in 32-bit mode
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 Synchronization Busy bit in the SYNCBUSY 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.
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 TC, except DBGCTRL, to their initial state, and the TC will be disabled.  
 Writing a '1' to CTRLA.SWRST will always take precedence; all other writes in the same write-operation will be discarded.  
 This bit is not enable-protected.

## 40.7.2.2 Control B Clear

**Name:** CTRLBCLR  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Read-Synchronized, Write-Synchronized

This register allows the user to clear bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set register (CTRLBSET).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

### Bits 7:5 – CMD[2:0] Command

These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK\_TC clock cycle. When a command has been executed, the CMD bit group will be read back as zero.

Writing 0x0 to these bits has no effect.

Writing a '1' to any of these bits will clear the pending command.

### Bit 2 – ONESHOT One-Shot on Counter

This bit controls one-shot operation of the TC.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will disable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

### Bit 1 – LUPD Lock Update

This bit controls the update operation of the TC buffered registers.

When CTRLB.LUPD is set, no update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.

This bit has no effect when input capture operation is enabled.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the LUPD bit.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

### Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the bit and make the counter count down.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

### 40.7.2.3 Control B Set

**Name:** CTRLBSET  
**Offset:** 0x05  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Read-synchronized, Write-Synchronized

This register allows the user to set bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Clear register (CTRLBCLR).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

#### Bits 7:5 – CMD[2:0] Command

These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK\_TC clock cycle. When a command has been executed, the CMD bit group will be read back as zero.

Writing 0x0 to these bits has no effect.

Writing a value different from 0x0 to these bits will issue a command for execution.

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force a read synchronization of COUNT

#### Bit 2 – ONESHOT One-Shot on Counter

This bit controls one-shot operation of the TC.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will enable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

#### Bit 1 – LUPD Lock Update

This bit controls the update operation of the TC buffered registers.

When CTRLB.LUPD is set, no update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the LUPD bit.

This bit has no effect when input capture operation is enabled.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

#### Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will set the bit and make the counter count down.



Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

## 40.7.2.4 Event Control

**Name:** EVCTRL  
**Offset:** 0x06  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
			MCEO1	MCEO0				OVFEO
Access			R/W	R/W				R/W
Reset			0	0				0
Bit	7	6	5	4	3	2	1	0
			TCEI	TCINV		EVACT[2:0]		
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

### Bits 12, 13 – MCEOx Match or Capture Channel x Event Output Enable [x = 1..0]

These bits enable the generation of an event for every match or capture on channel x.

Value	Description
0	Match/Capture event on channel x is disabled and will not be generated.
1	Match/Capture event on channel x is enabled and will be generated for every compare/capture.

### Bit 8 – OVFEO Overflow/Underflow Event Output Enable

This bit enables the Overflow/Underflow event. When enabled, an event will be generated when the counter overflows/underflows.

Value	Description
0	Overflow/Underflow event is disabled and will not be generated.
1	Overflow/Underflow event is enabled and will be generated for every counter overflow/underflow.

### Bit 5 – TCEI TC Event Enable

This bit is used to enable asynchronous input events to the TC.

Value	Description
0	Incoming events are disabled.
1	Incoming events are enabled.

### Bit 4 – TCINV TC Inverted Event Input Polarity

This bit inverts the asynchronous input event source.

Value	Description
0	Input event source is not inverted.
1	Input event source is inverted.

### Bits 2:0 – EVACT[2:0] Event Action

These bits define the event action the TC will perform on an event.

Value	Name	Description
0x0	OFF	Event action disabled
0x1	RETRIGGER	Start, restart or retrigger TC on event
0x2	COUNT	Count on event
0x3	START	Start TC on event
0x4	STAMP	Time stamp capture
0x5	PPW	Period captured in CC0, pulse width in CC1
0x6	PWP	Period captured in CC1, pulse width in CC0
0x7	PW	Pulse width capture

### 40.7.2.5 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
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

#### Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable [x = 1..0]

Writing a '0' to these bits has no effect.

Writing a '1' to MCx will clear the corresponding Match or Capture Channel x Interrupt Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

#### Bit 1 – ERR Error Interrupt Disable

Writing a '0' to these bits has no effect.

Writing a '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

#### Bit 0 – OVF Overflow Interrupt Disable

Writing a '0' to these bits has no effect.

Writing a '1' to this bit will clear the Overflow Interrupt Enable bit, which disables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

### 40.7.2.6 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 register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

#### Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable [x = 1..0]

Writing a '0' to these bits has no effect.

Writing a '1' to MCx will set the corresponding Match or Capture Channel x Interrupt Enable bit, which enables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

#### Bit 1 – ERR Error Interrupt Enable

Writing a '0' to these bits has no effect.

Writing a '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

#### Bit 0 – OVF Overflow Interrupt Enable

Writing a '0' to these bits has no effect.

Writing a '1' to this bit will set the Overflow Interrupt Enable bit, which enables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

### 40.7.2.7 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x0A  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

#### Bits 4, 5 – MCx Match or Capture Channel x [x = 1..0]

This flag is set on a comparison match, or when the corresponding CCx register contains a valid capture value. This flag is set on the next CLK\_TC\_CNT cycle, and will generate an interrupt request if the corresponding Match or Capture Channel x Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.MCx) is '1'.

Writing a '0' to these bits has no effect.

Writing a '1' to one of these bits will clear the corresponding Match or Capture Channel x interrupt flag

In capture operation, this flag is automatically cleared when CCx register is read.

#### Bit 1 – ERR Error Interrupt Flag

This flag is set when a new capture occurs on a channel while the corresponding Match or Capture Channel x interrupt flag is set, in which case there is no place to store the new capture.

Writing a '0' to these bits has no effect.

Writing a '1' to this bit clears the Error interrupt flag.

#### Bit 0 – OVF Overflow Interrupt Flag

This flag is set on the next CLK\_TC\_CNT cycle after an overflow condition occurs, and will generate an interrupt request if INTENCLR.OVF or INTENSET.OVF is '1'.

Writing a '0' to these bits has no effect.

Writing a '1' to this bit clears the Overflow interrupt flag.

### 40.7.2.8 Status

**Name:** STATUS  
**Offset:** 0x0B  
**Reset:** 0x01  
**Property:** Read-Synchronized

Bit	7	6	5	4	3	2	1	0
			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
Access			R/W	R/W	R/W		R	R
Reset			0	0	0		0	1

#### Bits 4, 5 – CCBUFVx Channel x Compare or Capture Buffer Valid [x = 1..0]

For a compare channel x, the bit x is set when a new value is written to the corresponding CCBUFx register.

The bit x is cleared by writing a '1' to it when CTRLB.LUPD is set, or it is cleared automatically by hardware on UPDATE condition.

For a capture channel x, the bit x is set when a valid capture value is stored in the CCBUFx register. The bit x is cleared automatically when the CCx register is read.

#### Bit 3 – PERBUFV Period Buffer Valid

This bit is set when a new value is written to the PERBUF register. The bit is cleared by writing '1' to the corresponding location when CTRLB.LUPD is set, or automatically cleared by hardware on UPDATE condition. This bit is available only in 8-bit mode and will always read zero in 16- and 32-bit modes.

#### Bit 1 – SLAVE Client Status Flag

This bit is only available in 32-bit mode on the Client TC (i.e., TC1, TC3, TC5 and/or TC7). The bit is set when the associated Host TC (TC0, TC2, TC4 and/or TC6, respectively) is set to run in 32-bit mode.

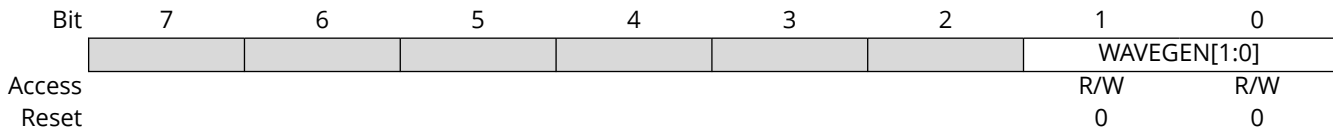
#### Bit 0 – STOP Stop Status Flag

This bit is set when the TC is disabled, on a Stop command, or on an overflow/underflow condition when the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) is '1'.

Value	Description
0	Counter is running.
1	Counter is stopped.

### 40.7.2.9 Waveform Generation Control

**Name:** WAVE  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected



#### Bits 1:0 - WAVEGEN[1:0] Waveform Generation Mode

These bits select the waveform generation operation. They affect the top value, as shown in Waveform Output Operations. They also control whether frequency or PWM waveform generation must be used. The waveform generation operations are explained in Waveform Output Operations. See *Waveform Output Operations* from Related Links.

These bits are not synchronized.

Value	Name	Operation	Top Value	Output Waveform on Match	Output Waveform on Wraparound
0x0	NFRQ	Normal frequency	PER <sup>1</sup> / Max	Toggle	No action
0x1	MFRQ	Match frequency	CC0	Toggle	No action
0x2	NPWM	Normal PWM	PER <sup>1</sup> / Max	Set	Clear
0x3	MPWM	Match PWM	CC0	Set	Clear

1. This depends on the TC mode: In 8-bit mode, the top value is the Period Value register (PER). In 16- and 32-bit mode, it is the respective MAX value.

#### Related Links

[40.6.2.6.1. Waveform Output Operations](#)

### 40.7.2.10 Driver Control

**Name:** DRVCTRL  
**Offset:** 0x0D  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
							INVEN1	INVEN0
Access							R/W	R/W
Reset							0	0

#### Bits 0, 1 – INVENx Output Waveform x Invert Enable [x=1..0]

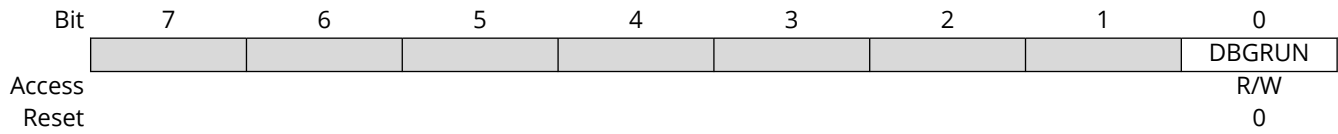
Bit x of INVEN[1:0] selects inversion of the output or capture trigger input of channel x.

Value	Description
0	Disable inversion of the WO[x] output and IO input pin.
1	Enable inversion of the WO[x] output and IO input pin.



### 40.7.2.11 Debug Control

**Name:** DBGCTRL  
**Offset:** 0x0F  
**Reset:** 0x00  
**Property:** PAC Write-Protection



#### Bit 0 - DBGRUN Run in Debug Mode

This bit is not affected by a software Reset, and must not be changed by software while the TC is enabled.

Value	Description
0	The TC is halted when the device is halted in debug mode.
1	The TC continues normal operation when the device is halted in debug mode.

#### 40.7.2.12 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x10  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
	CC1	CC0	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

##### Bits 6, 7 – CCx Compare/Capture Channel x Synchronization Busy [x=0..1]

For details on CC channels number, refer to each TC feature list.

This bit is set when the synchronization of CCx between clock domains is started.

This bit is also set when the CCBUFx is written, and cleared on update condition. The bit is automatically cleared when the STATUS.CCBUFx bit is cleared.

##### Bit 5 – PER PER Synchronization Busy

This bit is cleared when the synchronization of PER between the clock domains is complete.

This bit is set when the synchronization of PER between clock domains is started.

This bit is also set when the PER is written, and cleared on update condition. The bit is automatically cleared when the STATUS.PERBUF bit is cleared.

##### Bit 4 – COUNT COUNT Synchronization Busy

This bit is cleared when the synchronization of COUNT between the clock domains is complete.

This bit is set when the synchronization of COUNT between clock domains is started.

##### Bit 3 – STATUS STATUS Synchronization Busy

This bit is cleared when the synchronization of STATUS between the clock domains is complete.

This bit is set when a '1' is written to the Capture Channel Buffer Valid status flags (STATUS.CCBUFVx) and the synchronization of STATUS between clock domains is started.

##### Bit 2 – CTRLB CTRLB Synchronization Busy

This bit is cleared when the synchronization of CTRLB between the clock domains is complete.

This bit is set when the synchronization of CTRLB between clock domains is started.

##### Bit 1 – ENABLE ENABLE Synchronization Busy

This bit is cleared when the synchronization of ENABLE bit between the clock domains is complete.

This bit is set when the synchronization of ENABLE bit between clock domains is started.

##### Bit 0 – SWRST SWRST Synchronization Busy

This bit is cleared when the synchronization of SWRST bit between the clock domains is complete.

This bit is set when the synchronization of SWRST bit between clock domains is started.

**Note:** During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

### 40.7.2.13 Counter Value, 8-bit Mode

**Name:** COUNT  
**Offset:** 0x14  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

**Note:** Prior to any read access, this register must be synchronized by user by writing the according TC Command value to the Control B Set register (CTRLBSET.CMD=READSYNC).

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 7:0 - COUNT[7:0]** Counter Value  
 These bits contain the current counter value.

#### 40.7.2.14 Period Value, 8-bit Mode

**Name:** PER  
**Offset:** 0x1B  
**Reset:** 0xFF  
**Property:** Write-Synchronized

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	1

#### Bits 7:0 - PER[7:0] Period Value

These bits hold the value of the Period Buffer register PERBUF. The value is copied to PER register on UPDATE condition.

#### 40.7.2.15 Channel x Compare/Capture Value, 8-bit Mode

**Name:** CCx  
**Offset:** 0x1C + x\*0x01 [x=0..1]  
**Reset:** 0x00  
**Property:** Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	CC[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 - CC[7:0] Channel x Compare/Capture Value

These bits contain the compare/capture value in 8-bit TC mode. In Match frequency (MFRQ) or Match PWM (MPWM) waveform operation (WAVE.WAVEGEN), the CC0 register is used as a period register.

#### 40.7.2.16 Period Buffer Value, 8-bit Mode

**Name:** PERBUF  
**Offset:** 0x2F  
**Reset:** 0xFF  
**Property:** Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	PERBUF[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	1

#### Bits 7:0 - PERBUF[7:0] Period Buffer Value

These bits hold the value of the period buffer register. The value is copied to PER register on UPDATE condition.

#### 40.7.2.17 Channel x Compare Buffer Value, 8-bit Mode

**Name:** CCBUFx  
**Offset:** 0x30 + x\*0x01 [x=0..1]  
**Reset:** 0x00  
**Property:** Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	CCBUF[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 - CCBUF[7:0] Channel x Compare Buffer Value

These bits hold the value of the Channel x Compare Buffer Value. When the buffer valid flag is '1' and double buffering is enabled (CTRLBCLR.LUPD=1), the data from buffer registers will be copied into the corresponding CCx register under UPDATE condition (CTRLBSET.CMD=0x3), including the software update command.

### 40.7.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
		15:8	DMAOS				ALOCK	PRESCALER[2:0]		
		23:16			COPEN1	COPEN0			CAPTEN1	CAPTEN0
		31:24				CAPTMODE1[1:0]			CAPTMODE0[1:0]	
0x04	CTRLBCLR	7:0	CMD[2:0]				ONESHOT	LUPD	DIR	
0x05	CTRLBSET	7:0	CMD[2:0]				ONESHOT	LUPD	DIR	
0x06	EVCTRL	7:0			TCEI	TCINV		EVACT[2:0]		
		15:8			MCEO1	MCEO0			OVFEO	
0x08	INTENCLR	7:0			MC1	MC0		ERR	OVF	
0x09	INTENSET	7:0			MC1	MC0		ERR	OVF	
0x0A	INTFLAG	7:0			MC1	MC0		ERR	OVF	
0x0B	STATUS	7:0			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
0x0C	WAVE	7:0						WAVEGEN[1:0]		
0x0D	DRVCTRL	7:0						INVEN1	INVEN0	
0x0E	Reserved									
0x0F	DBGCTRL	7:0								DBGRUN
0x10	SYNCBUSY	7:0	CC1	CC0	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
0x11 ... 0x13	Reserved									
0x14	COUNT	7:0	COUNT[7:0]							
		15:8	COUNT[15:8]							
0x16 ... 0x1B	Reserved									
0x1C	CC0	7:0	CC[7:0]							
		15:8	CC[15:8]							
0x1E	CC1	7:0	CC[7:0]							
		15:8	CC[15:8]							
0x20 ... 0x2F	Reserved									
0x30	CCBUF0	7:0	CCBUF[7:0]							
		15:8	CCBUF[15:8]							
0x32	CCBUF1	7:0	CCBUF[7:0]							
		15:8	CCBUF[15:8]							

### 40.7.4 Register Description - 16-bit Mode



#### 40.7.4.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized, Enable-Protected

Bit	31	30	29	28	27	26	25	24
			CAPTMODE1[1:0]				CAPTMODE0[1:0]	
Access			R/W		R/W		R/W	
Reset			0		0		0	
Bit	23	22	21	20	19	18	17	16
			COPEN1	COPEN0			CAPTEN1	CAPTEN0
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0
Bit	15	14	13	12	11	10	9	8
	DMAOS					ALOCK	PRESCALER[2:0]	
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0
Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
Access	R/W	R/W	R/W		R/W		R/W	W
Reset	0	0	0		0		0	0

#### Bits 28:27 – CAPTMODE1[1:0] Capture mode Channel 1

These bits select the channel 1 capture mode.

Value	Name	Description
0x0	DEFAULT	Default capture
0x1	CAPTMIN	Minimum capture
0x2	CAPTMAX	Maximum capture
0x3		Reserved

#### Bits 25:24 – CAPTMODE0[1:0] Capture mode Channel 0

These bits select the channel 0 capture mode.

Value	Name	Description
0x0	DEFAULT	Default capture
0x1	CAPTMIN	Minimum capture
0x2	CAPTMAX	Maximum capture
0x3		Reserved

#### Bits 20, 21 – COPENx Capture On Pin x Enable [x=1..0]

Bit x of COPEN[1:0] selects the trigger source for capture operation, either events or I/O pin input. This bit is not synchronized.

Value	Description
0	Event from Event System is selected as trigger source for capture operation on channel x.
1	I/O pin is selected as trigger source for capture operation on channel x.

#### Bits 16, 17 – CAPTENx Capture Channel x Enable [x=1..0]

Bit x of CAPTEN[1:0] selects whether channel x is a capture or a compare channel. These bits are not synchronized.

Value	Description
0	CAPTEN disables capture on channel x.
1	CAPTEN enables capture on channel x.

**Bit 15 – DMAOS** DMA One-Shot Trigger Mode

This bit enables the DMA One-shot Trigger Mode.  
Writing a '1' to this bit will generate a DMA trigger on TC cycle following a TC\_CTRLBSET\_CMD\_DMAOS command.  
Writing a '0' to this bit will generate DMA triggers on each TC cycle.  
This bit is not synchronized.

**Bit 11 – ALOCK** Auto Lock

When this bit is set, Lock bit update (LUPD) is set to '1' on each overflow/underflow or re-trigger event.  
This bit is not synchronized.

Value	Description
0	The LUPD bit is not affected on overflow/underflow, and re-trigger event.
1	The LUPD bit is set on each overflow/underflow or re-trigger event.

**Bits 10:8 – PRESCALER[2:0]** Prescaler

These bits select the counter prescaler factor.  
These bits are not synchronized.

Value	Name	Description
0x0	DIV1	Prescaler: GCLK_TC
0x1	DIV2	Prescaler: GCLK_TC/2
0x2	DIV4	Prescaler: GCLK_TC/4
0x3	DIV8	Prescaler: GCLK_TC/8
0x4	DIV16	Prescaler: GCLK_TC/16
0x5	DIV64	Prescaler: GCLK_TC/64
0x6	DIV256	Prescaler: GCLK_TC/256
0x7	DIV1024	Prescaler: GCLK_TC/1024

**Bit 7 – ONDEMAND** Clock On Demand

This bit selects the clock requirements when the TC is stopped.  
In standby mode, if the Run in Standby bit (CTRLA.RUNSTDBY) is '0', ONDEMAND is forced to '0'.  
This bit is not synchronized.

Value	Description
0	The On Demand is disabled. If On Demand is disabled, the TC will continue to request the clock when its operation is stopped (STATUS.STOP=1).
1	The On Demand is enabled. When On Demand is enabled, the stopped TC will not request the clock. The clock is requested when a software re-trigger command is applied or when an event with start/re-trigger action is detected.

**Bit 6 – RUNSTDBY** Run in Standby

This bit is used to keep the TC running in standby mode.  
This bit is not synchronized.

Value	Description
0	The TC is halted in standby.
1	The TC continues to run in standby.

**Bits 5:4 – PRESCSYNC[1:0]** Prescaler and Counter Synchronization

These bits select whether the counter must wrap around on the next GCLK\_TCx clock or the next prescaled GCLK\_TCx clock. It also makes it possible to reset the prescaler.  
These bits are not synchronized.

Value	Name	Description
0x0	GCLK	Reload or reset the counter on next generic clock

Value	Name	Description
0x1	PRESC	Reload or reset the counter on next prescaler clock
0x2	RESYNC	Reload or reset the counter on next generic clock. Reset the prescaler counter
0x3	-	Reserved

**Bits 3:2 – MODE[1:0]** Timer Counter Mode

These bits select the counter mode.  
 These bits are not synchronized.

Value	Name	Description
0x0	COUNT16	Counter in 16-bit mode
0x1	COUNT8	Counter in 8-bit mode
0x2	COUNT32	Counter in 32-bit mode
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 Synchronization Busy bit in the SYNCBUSY 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.
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 TC, except DBGCTRL, to their initial state, and the TC will be disabled.  
 Writing a '1' to CTRLA.SWRST will always take precedence; all other writes in the same write-operation will be discarded.  
 This bit is not enable-protected.

#### 40.7.4.2 Control B Clear

**Name:** CTRLBCLR  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Read-Synchronized, Write-Synchronized

This register allows the user to clear bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set register (CTRLBSET).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

##### Bits 7:5 – CMD[2:0] Command

These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK\_TCx clock cycle. When a command has been executed, the CMD bit group will be read back as zero.

Writing 0x0 to these bits has no effect.

Writing a '1' to any of these bits will clear the pending command.

##### Bit 2 – ONESHOT One-Shot on Counter

This bit controls one-shot operation of the TC.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will disable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

##### Bit 1 – LUPD Lock Update

This bit controls the update operation of the TC buffered registers.

When CTRLB.LUPD is set, no any update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.

This bit has no effect when input capture operation is enabled.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the LUPD bit.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

##### Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

### 40.7.4.3 Control B Set

**Name:** CTRLBSET  
**Offset:** 0x05  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Read-Synchronized, Write-Synchronized

This register allows the user to set bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Clear register (CTRLBCLR).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

#### Bits 7:5 – CMD[2:0] Command

These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK\_TC clock cycle. When a command has been executed, the CMD bit group will be read back as zero.

Writing 0x0 to these bits has no effect.

Writing a value different from 0x0 from the following table will issue a command for execution.

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force a read synchronization of COUNT

#### Bit 2 – ONESHOT One-Shot on Counter

This bit controls one-shot operation of the TC.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will enable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

#### Bit 1 – LUPD Lock Update

This bit controls the update operation of the TC buffered registers.

When CTRLB.LUPD is set, no any update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the LUPD bit.

This bit has no effect when input capture operation is enabled.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

#### Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

#### 40.7.4.4 Event Control

**Name:** EVCTRL  
**Offset:** 0x06  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
			MCEO1	MCEO0				OVFEO
Access			R/W	R/W				R/W
Reset			0	0				0
Bit	7	6	5	4	3	2	1	0
			TCEI	TCINV		EVACT[2:0]		
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

#### Bits 12, 13 – MCEOx Match or Capture Channel x Event Output Enable [x = 1..0]

These bits enable the generation of an event for every match or capture on channel x.

Value	Description
0	Match/Capture event on channel x is disabled and will not be generated.
1	Match/Capture event on channel x is enabled and will be generated for every compare/capture.

#### Bit 8 – OVFEO Overflow/Underflow Event Output Enable

This bit enables the Overflow/Underflow event. When enabled, an event will be generated when the counter overflows/underflows.

Value	Description
0	Overflow/Underflow event is disabled and will not be generated.
1	Overflow/Underflow event is enabled and will be generated for every counter overflow/underflow.

#### Bit 5 – TCEI TC Event Enable

This bit is used to enable asynchronous input events to the TC.

Value	Description
0	Incoming events are disabled.
1	Incoming events are enabled.

#### Bit 4 – TCINV TC Inverted Event Input Polarity

This bit inverts the asynchronous input event source.

Value	Description
0	Input event source is not inverted.
1	Input event source is inverted.

#### Bits 2:0 – EVACT[2:0] Event Action

These bits define the event action the TC will perform on an event.

Value	Name	Description
0x0	OFF	Event action disabled
0x1	RETRIGGER	Start, restart or retrigger TC on event
0x2	COUNT	Count on event
0x3	START	Start TC on event
0x4	STAMP	Time stamp capture
0x5	PPW	Period captured in CC0, pulse width in CC1
0x6	PWP	Period captured in CC1, pulse width in CC0
0x7	PW	Pulse width capture

#### 40.7.4.5 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
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

##### Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable

Writing a '0' to these bits has no effect.

Writing a '1' to MCx will clear the corresponding Match or Capture Channel x Interrupt Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

##### Bit 1 – ERR Error Interrupt Disable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

##### Bit 0 – OVF Overflow Interrupt Disable

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 request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.



#### 40.7.4.6 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 register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

##### Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable

Writing a '0' to these bits has no effect.

Writing a '1' to MCx will clear the corresponding Match or Capture Channel x Interrupt Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

##### Bit 1 – ERR Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

##### Bit 0 – 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 request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

#### 40.7.4.7 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x0A  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

##### Bits 4, 5 – MCx Match or Capture Channel x

This flag is set on a comparison match, or when the corresponding CCx register contains a valid capture value. This flag is set on the next CLK\_TC\_CNT cycle, and will generate an interrupt request if the corresponding Match or Capture Channel x Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.MCx) is '1'.

Writing a '0' to one of these bits has no effect.

Writing a '1' to one of these bits will clear the corresponding Match or Capture Channel x interrupt flag

In capture operation, this flag is automatically cleared when CCx register is read.

##### Bit 1 – ERR Error Interrupt Flag

This flag is set when a new capture occurs on a channel while the corresponding Match or Capture Channel x interrupt flag is set, in which case there is nowhere to store the new capture.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Error interrupt flag.

##### Bit 0 – OVF Overflow Interrupt Flag

This flag is set on the next CLK\_TC\_CNT cycle after an overflow condition occurs, and will generate an interrupt request if INTENCLR.OVF or INTENSET.OVF is '1'.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Overflow interrupt flag.

#### 40.7.4.8 Status

**Name:** STATUS  
**Offset:** 0x0B  
**Reset:** 0x01  
**Property:** Read-Synchronized, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
Access			R/W	R/W	R/W		R	R
Reset			0	0	0		0	1

#### Bits 4, 5 – CCBUFVx Channel x Compare or Capture Buffer Valid

For a compare channel x, the bit x is set when a new value is written to the corresponding CCBUFx register.

The bit x is cleared by writing a '1' to it when CTRLB.LUPD is set, or it is cleared automatically by hardware on UPDATE condition.

For a capture channel x, the bit x is set when a valid capture value is stored in the CCBUFx register. The bit x is cleared automatically when the CCx register is read.

#### Bit 3 – PERBUFV Period Buffer Valid

This bit is set when a new value is written to the PERBUF register. The bit is cleared by writing '1' to the corresponding location when CTRLB.LUPD is set, or automatically cleared by hardware on UPDATE condition. This bit is available only in 8-bit mode and will always read zero in 16- and 32-bit modes.

#### Bit 1 – SLAVE Client Status Flag

This bit is only available in 32-bit mode on the client TC (i.e., TC1 and/or TC3). The bit is set when the associated host TC (TC0 and TC2, respectively) is set to run in 32-bit mode.

#### Bit 0 – STOP Stop Status Flag

This bit is set when the TC is disabled, on a Stop command, or on an overflow/underflow condition when the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) is '1'.

Value	Description
0	Counter is running.
1	Counter is stopped.

#### 40.7.4.9 Waveform Generation Control

**Name:** WAVE  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

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

##### Bits 1:0 - WAVEGEN[1:0] Waveform Generation Mode

These bits select the waveform generation operation. They affect the top value, as shown in Waveform Output Operations. They also control whether frequency or PWM waveform generation must be used. The waveform generation operations are explained in Waveform Output Operations. See *Waveform Output Operations* from Related Links.

These bits are not synchronized.

Value	Name	Operation	Top Value	Output Waveform on Match	Output Waveform on Wraparound
0x0	NFRQ	Normal frequency	PER <sup>1</sup> / Max	Toggle	No action
0x1	MFRQ	Match frequency	CC0	Toggle	No action
0x2	NPWM	Normal PWM	PER <sup>1</sup> / Max	Set	Clear
0x3	MPWM	Match PWM	CC0	Set	Clear

1. This depends on the TC mode: In 8-bit mode, the top value is the Period Value register (PER). In 16- and 32-bit mode, it is the respective MAX value.

##### Related Links

[40.6.2.6.1. Waveform Output Operations](#)

#### 40.7.4.10 Driver Control

**Name:** DRVCTRL  
**Offset:** 0x0D  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
							INVEN1	INVEN0
Access							R/W	R/W
Reset							0	0

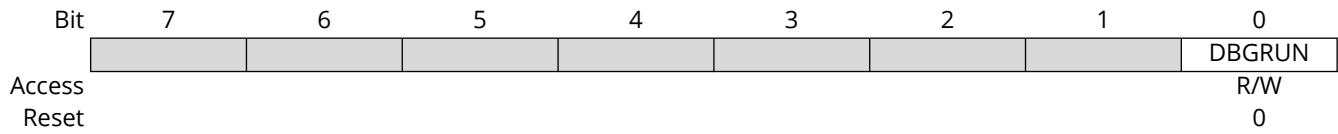
#### Bits 0, 1 – INVENx Output Waveform x Invert Enable

INVENx bit selects inversion of the output or capture trigger input of channel x.

Value	Description
0	Disable inversion of the WO[x] output and IO input pin.
1	Enable inversion of the WO[x] output and IO input pin.

#### 40.7.4.11 Debug Control

**Name:** DBGCTRL  
**Offset:** 0x0F  
**Reset:** 0x00  
**Property:** PAC Write-Protection



#### Bit 0 – DBGRUN Run in Debug Mode

This bit is not affected by a software Reset, and must not be changed by software while the TC is enabled.

Value	Description
0	The TC is halted when the device is halted in debug mode.
1	The TC continues normal operation when the device is halted in debug mode.

#### 40.7.4.12 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x10  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
	CC1	CC0	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

##### Bits 6, 7 – CCx Compare/Capture Channel x Synchronization Busy [x=0..1]

For details on CC channels number, refer to each TC feature list.

This bit is set when the synchronization of CCx between clock domains is started.

This bit is also set when the CCBUFx is written, and cleared on update condition. The bit is automatically cleared when the STATUS.CCBUFx bit is cleared.

##### Bit 5 – PER PER Synchronization Busy

This bit is cleared when the synchronization of PER between the clock domains is complete.

This bit is set when the synchronization of PER between clock domains is started.

This bit is also set when the PER is written, and cleared on update condition. The bit is automatically cleared when the STATUS.PERBUF bit is cleared.

##### Bit 4 – COUNT COUNT Synchronization Busy

This bit is cleared when the synchronization of COUNT between the clock domains is complete.

This bit is set when the synchronization of COUNT between clock domains is started.

##### Bit 3 – STATUS STATUS Synchronization Busy

This bit is cleared when the synchronization of STATUS between the clock domains is complete.

This bit is set when a '1' is written to the Capture Channel Buffer Valid status flags (STATUS.CCBUFVx) and the synchronization of STATUS between clock domains is started.

##### Bit 2 – CTRLB CTRLB Synchronization Busy

This bit is cleared when the synchronization of CTRLB between the clock domains is complete.

This bit is set when the synchronization of CTRLB between clock domains is started.

##### Bit 1 – ENABLE ENABLE Synchronization Busy

This bit is cleared when the synchronization of ENABLE bit between the clock domains is complete.

This bit is set when the synchronization of ENABLE bit between clock domains is started.

##### Bit 0 – SWRST SWRST Synchronization Busy

This bit is cleared when the synchronization of SWRST bit between the clock domains is complete.

This bit is set when the synchronization of SWRST bit between clock domains is started.

**Note:** During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

#### 40.7.4.13 Counter Value, 16-bit Mode

**Name:** COUNT  
**Offset:** 0x14  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

**Note:** Prior to any read access, this register must be synchronized by user by writing the according TC Command value to the Control B Set register (CTRLBSET.CMD=READSYNC).

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 contain the current counter value.



#### 40.7.4.14 Channel x Compare/Capture Value, 16-bit Mode

**Name:** CCx  
**Offset:** 0x1C + x\*0x02 [x=0..1]  
**Reset:** 0x0000  
**Property:** Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	CC[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
	CC[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 – CC[15:0] Channel x Compare/Capture Value

These bits contain the compare/capture value in 16-bit TC mode. In Match frequency (MFRQ) or Match PWM (MPWM) waveform operation (WAVE.WAVEGEN), the CC0 register is used as a period register.

#### 40.7.4.15 Channel x Compare Buffer Value, 16-bit Mode

**Name:** CCBUFx  
**Offset:** 0x30 + x\*0x02 [x=0..1]  
**Reset:** 0x0000  
**Property:** Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	CCBUF[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
	CCBUF[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 – CCBUF[15:0] Channel x Compare Buffer Value

These bits hold the value of the Channel x Compare Buffer Value. When the buffer valid flag is '1' and double buffering is enabled (CTRLBCLR.LUPD=1), the data from buffer registers will be copied into the corresponding CCx register under UPDATE condition (CTRLBSET.CMD=0x3), including the software update command.

## 40.7.5 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
		15:8	DMAOS				ALOCK	PRESCALER[2:0]		
		23:16			COPEN1	COPEN0			CAPTEN1	CAPTEN0
		31:24				CAPTMODE1[1:0]			CAPTMODE0[1:0]	
0x04	CTRLBCLR	7:0	CMD[2:0]					ONESHOT	LUPD	DIR
0x05	CTRLBSET	7:0	CMD[2:0]					ONESHOT	LUPD	DIR
0x06	EVCTRL	7:0			TCEI	TCINV		EVACT[2:0]		
		15:8			MCEO1	MCEO0				OVFEO
0x08	INTENCLR	7:0			MC1	MC0			ERR	OVF
0x09	INTENSET	7:0			MC1	MC0			ERR	OVF
0x0A	INTFLAG	7:0			MC1	MC0			ERR	OVF
0x0B	STATUS	7:0			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
0x0C	WAVE	7:0							WAVEGEN[1:0]	
0x0D	DRVCTRL	7:0							INVEN1	INVEN0
0x0E	Reserved									
0x0F	DBGCTRL	7:0								DBGRUN
0x10	SYNCBUSY	7:0	CC1	CC0	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
0x11 ... 0x13	Reserved									
0x14	COUNT	7:0	COUNT[7:0]							
		15:8	COUNT[15:8]							
		23:16	COUNT[23:16]							
		31:24	COUNT[31:24]							
0x18 ... 0x1B	Reserved									
0x1C	CC0	7:0	CC[7:0]							
		15:8	CC[15:8]							
		23:16	CC[23:16]							
		31:24	CC[31:24]							
0x20	CC1	7:0	CC[7:0]							
		15:8	CC[15:8]							
		23:16	CC[23:16]							
		31:24	CC[31:24]							
0x24 ... 0x2F	Reserved									
0x30	CCBUF0	7:0	CCBUF[7:0]							
		15:8	CCBUF[15:8]							
		23:16	CCBUF[23:16]							
		31:24	CCBUF[31:24]							
0x34	CCBUF1	7:0	CCBUF[7:0]							
		15:8	CCBUF[15:8]							
		23:16	CCBUF[23:16]							
		31:24	CCBUF[31:24]							

## 40.7.6 Register Description - 32-bit Mode

### 40.7.6.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized, Enable-Protected

Bit	31	30	29	28	27	26	25	24
			CAPTMODE1[1:0]				CAPTMODE0[1:0]	
Access			R/W		R/W		R/W	
Reset			0		0		0	
Bit	23	22	21	20	19	18	17	16
			COPEN1	COPEN0			CAPTEN1	CAPTEN0
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0
Bit	15	14	13	12	11	10	9	8
	DMAOS					ALOCK	PRESCALER[2:0]	
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0
Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
Access	R/W	R/W	R/W		R/W		R/W	W
Reset	0	0	0		0		0	0

#### Bits 28:27 – CAPTMODE1[1:0] Capture mode Channel 1

These bits select the channel 1 capture mode.

Value	Name	Description
0x0	DEFAULT	Default capture
0x1	CAPTMIN	Minimum capture
0x2	CAPTMAX	Maximum capture
0x3		Reserved

#### Bits 25:24 – CAPTMODE0[1:0] Capture mode Channel 0

These bits select the channel 0 capture mode.

Value	Name	Description
0x0	DEFAULT	Default capture
0x1	CAPTMIN	Minimum capture
0x2	CAPTMAX	Maximum capture
0x3		Reserved

#### Bits 20, 21 – COPENx Capture On Pin x Enable [x=1..0]

Bit x of COPEN[1:0] selects the trigger source for capture operation, either events or I/O pin input. This bit is not synchronized.

Value	Description
0	Event from Event System is selected as trigger source for capture operation on channel x.
1	I/O pin is selected as trigger source for capture operation on channel x.

#### Bits 16, 17 – CAPTENx Capture Channel x Enable [x=1..0]

Bit x of CAPTEN[1:0] selects whether channel x is a capture or a compare channel. These bits are not synchronized.

Value	Description
0	CAPTEN disables capture on channel x.
1	CAPTEN enables capture on channel x.

**Bit 15 – DMAOS** DMA One-Shot Trigger Mode

This bit enables the DMA One-shot Trigger Mode.  
Writing a '1' to this bit will generate a DMA trigger on TC cycle following a TC\_CTRLBSET\_CMD\_DMAOS command.  
Writing a '0' to this bit will generate DMA triggers on each TC cycle.  
This bit is not synchronized.

**Bit 11 – ALOCK** Auto Lock

When this bit is set, Lock bit update (LUPD) is set to '1' on each overflow/underflow or re-trigger event.  
This bit is not synchronized.

Value	Description
0	The LUPD bit is not affected on overflow/underflow, and re-trigger event.
1	The LUPD bit is set on each overflow/underflow or re-trigger event.

**Bits 10:8 – PRESCALER[2:0]** Prescaler

These bits select the counter prescaler factor.  
These bits are not synchronized.

Value	Name	Description
0x0	DIV1	Prescaler: GCLK_TC
0x1	DIV2	Prescaler: GCLK_TC/2
0x2	DIV4	Prescaler: GCLK_TC/4
0x3	DIV8	Prescaler: GCLK_TC/8
0x4	DIV16	Prescaler: GCLK_TC/16
0x5	DIV64	Prescaler: GCLK_TC/64
0x6	DIV256	Prescaler: GCLK_TC/256
0x7	DIV1024	Prescaler: GCLK_TC/1024

**Bit 7 – ONDEMAND** Clock On Demand

This bit selects the clock requirements when the TC is stopped.  
In standby mode, if the Run in Standby bit (CTRLA.RUNSTDBY) is '0', ONDEMAND is forced to '0'.  
This bit is not synchronized.

Value	Description
0	The On Demand is disabled. If On Demand is disabled, the TC will continue to request the clock when its operation is stopped (STATUS.STOP=1).
1	The On Demand is enabled. When On Demand is enabled, the stopped TC will not request the clock. The clock is requested when a software re-trigger command is applied or when an event with start/re-trigger action is detected.

**Bit 6 – RUNSTDBY** Run in Standby

This bit is used to keep the TC running in standby mode.  
This bit is not synchronized.

Value	Description
0	The TC is halted in standby.
1	The TC continues to run in standby.

**Bits 5:4 – PRESCSYNC[1:0]** Prescaler and Counter Synchronization

These bits select whether the counter must wrap around on the next GCLK\_TCx clock or the next prescaled GCLK\_TCx clock. It also makes it possible to reset the prescaler.  
These bits are not synchronized.

Value	Name	Description
0x0	GCLK	Reload or reset the counter on next generic clock

Value	Name	Description
0x1	PRESC	Reload or reset the counter on next prescaler clock
0x2	RESYNC	Reload or reset the counter on next generic clock. Reset the prescaler counter
0x3	-	Reserved

**Bits 3:2 – MODE[1:0]** Timer Counter Mode

These bits select the counter mode.  
 These bits are not synchronized.

Value	Name	Description
0x0	COUNT16	Counter in 16-bit mode
0x1	COUNT8	Counter in 8-bit mode
0x2	COUNT32	Counter in 32-bit mode
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 Synchronization Busy bit in the SYNCBUSY 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.
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 TC, except DBGCTRL, to their initial state, and the TC will be disabled.

Writing a '1' to CTRLA.SWRST will always take precedence; all other writes in the same write-operation will be discarded.

This bit is not enable-protected.

## 40.7.6.2 Control B Clear

**Name:** CTRLBCLR  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Read-Synchronized, Write-Synchronized

This register allows the user to clear bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set register (CTRLBSET).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

### Bits 7:5 – CMD[2:0] Command

These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK\_TCx clock cycle. When a command has been executed, the CMD bit group will be read back as zero.

Writing 0x0 to these bits has no effect.

Writing a '1' to any of these bits will clear the pending command.

### Bit 2 – ONESHOT One-Shot on Counter

This bit controls one-shot operation of the TC.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will disable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

### Bit 1 – LUPD Lock Update

This bit controls the update operation of the TC buffered registers.

When CTRLB.LUPD is set, no any update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.

This bit has no effect when input capture operation is enabled.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the LUPD bit.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

### Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

### 40.7.6.3 Control B Set

**Name:** CTRLBSET  
**Offset:** 0x05  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Read-Synchronized, Write-Synchronized

This register allows the user to set bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Clear register (CTRLBCLR).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

#### Bits 7:5 – CMD[2:0] Command

These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK\_TC clock cycle. When a command has been executed, the CMD bit group will be read back as zero.

Writing 0x0 to these bits has no effect.

Writing a value different from 0x0 from the following table will issue a command for execution.

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force a read synchronization of COUNT

#### Bit 2 – ONESHOT One-Shot on Counter

This bit controls one-shot operation of the TC.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will enable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

#### Bit 1 – LUPD Lock Update

This bit controls the update operation of the TC buffered registers.

When CTRLB.LUPD is set, no any update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the LUPD bit.

This bit has no effect when input capture operation is enabled.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

#### Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will clear the bit and make the counter count up.



Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

#### 40.7.6.4 Event Control

**Name:** EVCTRL  
**Offset:** 0x06  
**Reset:** 0x0000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
			MCEO1	MCEO0				OVFEO
Access			R/W	R/W				R/W
Reset			0	0				0
Bit	7	6	5	4	3	2	1	0
			TCEI	TCINV			EVACT[2:0]	
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

#### Bits 12, 13 – MCEOx Match or Capture Channel x Event Output Enable [x = 1..0]

These bits enable the generation of an event for every match or capture on channel x.

Value	Description
0	Match/Capture event on channel x is disabled and will not be generated.
1	Match/Capture event on channel x is enabled and will be generated for every compare/capture.

#### Bit 8 – OVFEO Overflow/Underflow Event Output Enable

This bit enables the Overflow/Underflow event. When enabled, an event will be generated when the counter overflows/underflows.

Value	Description
0	Overflow/Underflow event is disabled and will not be generated.
1	Overflow/Underflow event is enabled and will be generated for every counter overflow/underflow.

#### Bit 5 – TCEI TC Event Enable

This bit is used to enable asynchronous input events to the TC.

Value	Description
0	Incoming events are disabled.
1	Incoming events are enabled.

#### Bit 4 – TCINV TC Inverted Event Input Polarity

This bit inverts the asynchronous input event source.

Value	Description
0	Input event source is not inverted.
1	Input event source is inverted.

#### Bits 2:0 – EVACT[2:0] Event Action

These bits define the event action the TC will perform on an event.

Value	Name	Description
0x0	OFF	Event action disabled
0x1	RETRIGGER	Start, restart or retrigger TC on event
0x2	COUNT	Count on event
0x3	START	Start TC on event
0x4	STAMP	Time stamp capture
0x5	PPW	Period captured in CC0, pulse width in CC1
0x6	PWP	Period captured in CC1, pulse width in CC0
0x7	PW	Pulse width capture

### 40.7.6.5 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
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

#### Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable

Writing a '0' to these bits has no effect.

Writing a '1' to MCx will clear the corresponding Match or Capture Channel x Interrupt Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

#### Bit 1 – ERR Error Interrupt Disable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

#### Bit 0 – OVF Overflow Interrupt Disable

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 request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

### 40.7.6.6 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 register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

#### Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable

Writing a '0' to these bits has no effect.

Writing a '1' to MCx will clear the corresponding Match or Capture Channel x Interrupt Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

#### Bit 1 – ERR Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

#### Bit 0 – 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 request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

### 40.7.6.7 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x0A  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

#### Bits 4, 5 – MCx Match or Capture Channel x

This flag is set on a comparison match, or when the corresponding CCx register contains a valid capture value. This flag is set on the next CLK\_TC\_CNT cycle, and will generate an interrupt request if the corresponding Match or Capture Channel x Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.MCx) is '1'.

Writing a '0' to one of these bits has no effect.

Writing a '1' to one of these bits will clear the corresponding Match or Capture Channel x interrupt flag

In capture operation, this flag is automatically cleared when CCx register is read.

#### Bit 1 – ERR Error Interrupt Flag

This flag is set when a new capture occurs on a channel while the corresponding Match or Capture Channel x interrupt flag is set, in which case there is nowhere to store the new capture.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Error interrupt flag.

#### Bit 0 – OVF Overflow Interrupt Flag

This flag is set on the next CLK\_TC\_CNT cycle after an overflow condition occurs, and will generate an interrupt request if INTENCLR.OVF or INTENSET.OVF is '1'.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Overflow interrupt flag.

### 40.7.6.8 Status

**Name:** STATUS  
**Offset:** 0x0B  
**Reset:** 0x01  
**Property:** Read-Synchronized, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
Access			R/W	R/W	R/W		R	R
Reset			0	0	0		0	1

#### Bits 4, 5 – CCBUFVx Channel x Compare or Capture Buffer Valid

For a compare channel x, the bit x is set when a new value is written to the corresponding CCBUFx register.

The bit x is cleared by writing a '1' to it when CTRLB.LUPD is set, or it is cleared automatically by hardware on UPDATE condition.

For a capture channel x, the bit x is set when a valid capture value is stored in the CCBUFx register. The bit x is cleared automatically when the CCx register is read.

#### Bit 3 – PERBUFV Period Buffer Valid

This bit is set when a new value is written to the PERBUF register. The bit is cleared by writing '1' to the corresponding location when CTRLB.LUPD is set, or automatically cleared by hardware on UPDATE condition. This bit is available only in 8-bit mode and will always read zero in 16- and 32-bit modes.

#### Bit 1 – SLAVE Client Status Flag

This bit is only available in 32-bit mode on the client TC (i.e., TC1 and/or TC3). The bit is set when the associated host TC (TC0 and TC2, respectively) is set to run in 32-bit mode.

#### Bit 0 – STOP Stop Status Flag

This bit is set when the TC is disabled, on a Stop command, or on an overflow/underflow condition when the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) is '1'.

Value	Description
0	Counter is running.
1	Counter is stopped.

### 40.7.6.9 Waveform Generation Control

**Name:** WAVE  
**Offset:** 0x0C  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

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

#### Bits 1:0 - WAVEGEN[1:0] Waveform Generation Mode

These bits select the waveform generation operation. They affect the top value, as shown in Waveform Output Operations. They also control whether frequency or PWM waveform generation must be used. The waveform generation operations are explained in Waveform Output Operations. See *Waveform Output Operations* from Related Links.

These bits are not synchronized.

Value	Name	Operation	Top Value	Output Waveform on Match	Output Waveform on Wraparound
0x0	NFRQ	Normal frequency	PER <sup>1</sup> / Max	Toggle	No action
0x1	MFRQ	Match frequency	CC0	Toggle	No action
0x2	NPWM	Normal PWM	PER <sup>1</sup> / Max	Set	Clear
0x3	MPWM	Match PWM	CC0	Set	Clear

1. This depends on the TC mode: In 8-bit mode, the top value is the Period Value register (PER). In 16- and 32-bit mode, it is the respective MAX value.

#### Related Links

[40.6.2.6.1. Waveform Output Operations](#)

### 40.7.6.10 Driver Control

**Name:** DRVCTRL  
**Offset:** 0x0D  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
							INVEN1	INVEN0
Access							R/W	R/W
Reset							0	0

#### Bits 0, 1 – INVENx Output Waveform x Invert Enable

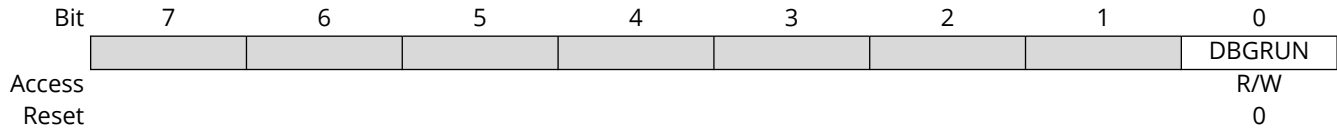
INVENx bit selects inversion of the output or capture trigger input of channel x.

Value	Description
0	Disable inversion of the WO[x] output and IO input pin.
1	Enable inversion of the WO[x] output and IO input pin.



### 40.7.6.11 Debug Control

**Name:** DBGCTRL  
**Offset:** 0x0F  
**Reset:** 0x00  
**Property:** PAC Write-Protection



#### Bit 0 - DBGRUN Run in Debug Mode

This bit is not affected by a software Reset, and must not be changed by software while the TC is enabled.

Value	Description
0	The TC is halted when the device is halted in debug mode.
1	The TC continues normal operation when the device is halted in debug mode.

#### 40.7.6.12 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x10  
**Reset:** 0x00  
**Property:** -

Bit	7	6	5	4	3	2	1	0
	CC1	CC0	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

##### Bits 6, 7 – CCx Compare/Capture Channel x Synchronization Busy [x=0..1]

For details on CC channels number, refer to each TC feature list.

This bit is set when the synchronization of CCx between clock domains is started.

This bit is also set when the CCBUFx is written, and cleared on update condition. The bit is automatically cleared when the STATUS.CCBUFx bit is cleared.

##### Bit 5 – PER PER Synchronization Busy

This bit is cleared when the synchronization of PER between the clock domains is complete.

This bit is set when the synchronization of PER between clock domains is started.

This bit is also set when the PER is written, and cleared on update condition. The bit is automatically cleared when the STATUS.PERBUF bit is cleared.

##### Bit 4 – COUNT COUNT Synchronization Busy

This bit is cleared when the synchronization of COUNT between the clock domains is complete.

This bit is set when the synchronization of COUNT between clock domains is started.

##### Bit 3 – STATUS STATUS Synchronization Busy

This bit is cleared when the synchronization of STATUS between the clock domains is complete.

This bit is set when a '1' is written to the Capture Channel Buffer Valid status flags (STATUS.CCBUFVx) and the synchronization of STATUS between clock domains is started.

##### Bit 2 – CTRLB CTRLB Synchronization Busy

This bit is cleared when the synchronization of CTRLB between the clock domains is complete.

This bit is set when the synchronization of CTRLB between clock domains is started.

##### Bit 1 – ENABLE ENABLE Synchronization Busy

This bit is cleared when the synchronization of ENABLE bit between the clock domains is complete.

This bit is set when the synchronization of ENABLE bit between clock domains is started.

##### Bit 0 – SWRST SWRST Synchronization Busy

This bit is cleared when the synchronization of SWRST bit between the clock domains is complete.

This bit is set when the synchronization of SWRST bit between clock domains is started.

**Note:** During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

### 40.7.6.13 Counter Value, 32-bit Mode

**Name:** COUNT  
**Offset:** 0x14  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

**Note:** Prior to any read access, this register must be synchronized by user by writing the according TC Command value to the Control B Set register (CTRLBSET.CMD=READSYNC).

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 contain the current counter value.

#### 40.7.6.14 Channel x Compare/Capture Value, 32-bit Mode

**Name:** CCx  
**Offset:** 0x1C + x\*0x04 [x=0..1]  
**Reset:** 0x00000000  
**Property:** Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	CC[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
	CC[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
	CC[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
	CC[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 – CC[31:0] Channel x Compare/Capture Value

These bits contain the compare/capture value in 32-bit TC mode. In Match frequency (MFRQ) or Match PWM (MPWM) waveform operation (WAVE.WAVEGEN), the CC0 register is used as a period register.

### 40.7.6.15 Channel x Compare Buffer Value, 32-bit Mode

**Name:** CCBUFx  
**Offset:** 0x30 + x\*0x04 [x=0..1]  
**Reset:** 0x00000000  
**Property:** Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	CCBUF[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
	CCBUF[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
	CCBUF[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
	CCBUF[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 – CCBUF[31:0] Channel x Compare Buffer Value

These bits hold the value of the Channel x Compare Buffer Value. When the buffer valid flag is '1' and double buffering is enabled (CTRLBCLR.LUPD=1), the data from buffer registers will be copied into the corresponding CCx register under UPDATE condition (CTRLBSET.CMD=0x3), including the software update command.

## 41. Timer/Counter for Control Applications (TCC)

### 41.1 Overview

The device provides three instances of the Timer/Counter for Control Applications (TCC).

Each TCC instance consists of a counter, a prescaler, compare/capture channels and control logic. The counter can be set to count events or clock pulses. The counter together with the compare/capture channels can be configured to time stamp input events, allowing capture of frequency and pulse-width. It can also perform waveform generation, such as frequency generation and pulse-width modulation.

Waveform extensions are featured for motor control, ballast, LED, H-bridge, power converters and other types of power control applications. They allow for low-side and high-side output with optional dead-time insertion. Waveform extensions can also generate a synchronized bit pattern across the waveform output pins. The fault options enable fault protection for safe and deterministic handling, disabling and/or shut-down of external drivers.

**Note:** The TCC configurations, such as channel numbers and features, may be reduced for some of the TCC instances.

**Table 41-1.** TCC Specific Configuration

TCC No.	Counter Size (SIZE)	Host Link (Host_Client_MODE) (0 = NA, 1 = Host, 2 = Client)	Channels (CC_NUM)	Pins WO_NUM (OW_NUM)	Extensions					
					Fault 1=YES	Dithering 1=YES	OutMatrix 1=YES (OTMX)	Dead Time Insertion 1=YES (DTI)	Swap 1=YES (SWAP)	Pattern Generation 1=YES (PG)
0	24	1	6	6	1	1	1	1	1	1
1	24	2	6	6	1	1	1	1	1	1
2	16	0	2	2	1	0	1	0	0	0

**Note:** Traditional Timer/Counter for Control Applications (TCC) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

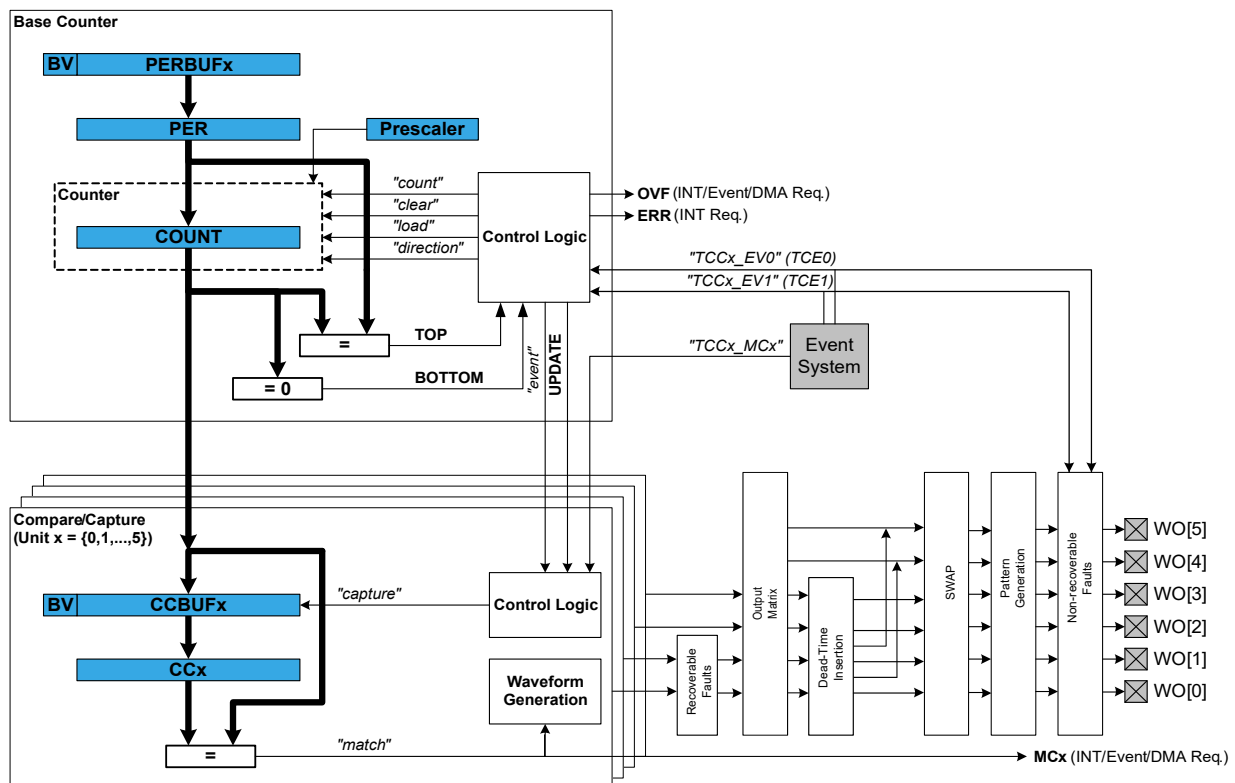
### 41.2 Features

- Up to six Compare/Capture Channels (CC) with:
  - Double buffered period setting
  - Double buffered compare or capture channel
  - Circular buffer on period and compare channel registers
- Waveform Generation:
  - Frequency generation
  - Single-slope pulse-width modulation (PWM)
  - Dual-slope PWM with half-cycle reload capability
- Input Capture:
  - Event capture
  - Frequency capture
  - Pulse-width capture
- Waveform Extensions:
  - Configurable distribution of compare channels outputs across port pins

- Low-side and high-side output with programmable dead-time insertion
- Waveform swap option with double buffer support
- Pattern generation with double buffer support
- Dithering support
- Fault Protection for Safe Disabling of Drivers:
  - Two recoverable fault sources
  - Two non-recoverable fault sources
  - Debugger can be a source of non-recoverable fault
- Input Events:
  - Two input events (EVx) for counter
  - One input event (MCx) for each channel
- Output Events:
  - Three output events (Count, re-trigger and overflow) are available for counter
  - One compare match/input capture event output for each channel
- Interrupts:
  - Overflow and re-trigger interrupt
  - Compare match/input capture interrupt
  - Interrupt on fault detection

### 41.3 Block Diagram

Figure 41-1. Timer/Counter for Control Applications - Block Diagram



## 41.4 Signal Description

Table 41-2. Signal Description

Pin Name	Type	Description
TCCx/WO[0]	Digital output	Compare channel 0 waveform output
TCCx/WO[1]	Digital output	Compare channel 1 waveform output
...	...	...
TCCx/WO[WO_NUM-1]	Digital output	Compare channel n waveform output

See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

### Related Links

[6. I/O Ports and Peripheral Pin Select \(PPS\)](#)

## 41.5 Product Dependencies

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

### 41.5.1 I/O Lines

In order to use the I/O lines of this peripheral, the I/O pins must be configured using the I/O Peripheral Pin Select (PPS).

### 41.5.2 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. Events connected to the event system can trigger other operations in the system without exiting Sleep modes.

### 41.5.3 Clocks

A generic clock (GCLK\_TCCx) is required to clock the TCC. This clock must be configured and enabled in the generic clock controller before using the TCC. Note that TCC1 and TCC2 share a single peripheral clock generator.

The generic clocks (GCLK\_TCCx) are asynchronous to the bus clock (PB1\_CLK). Due to this asynchronicity, writing certain registers will require synchronization between the clock domains.

### 41.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). In order 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 \(DMAC\)](#)

### 41.5.5 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 \(NVIC\)](#)

### 41.5.6 Events

The events of this peripheral are connected to the Event System.



## Related Links

[28. Event System \(EVSYS\)](#)

### 41.5.7 Debug Operation

When the CPU is halted in Debug mode, this peripheral will halt normal operation. This peripheral can be forced to continue operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

## Related Links

[41.8.8. DBGCTRL](#)

### 41.5.8 Register Access Protection

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

- Interrupt Flag register (INTFLAG)
- Status register (STATUS)
- Period and Period Buffer registers (PER, PERBUF)
- Compare/Capture and Compare/Capture Buffer registers (CCx, CCBUFx)
- Control Waveform register (WAVE)
- Pattern Generation Value and Pattern Generation Value Buffer registers (PATT, PATTBUF)

**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.

### 41.5.9 Analog Connections

Not applicable.

## 41.6 Functional Description

### 41.6.1 Principle of Operation

The following definitions are used throughout the documentation:

**Table 41-3.** Timer/Counter for Control Applications – Definitions

Name	Description
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence. The TOP value can be the same as Period (PER) or the Compare Channel 0 (CC0) register value depending on the Waveform Generator mode in Waveform Output Operations. See <i>Waveform Output Generation Operations</i> from Related Links.
ZERO	The counter reaches ZERO when it contains all zeros.
MAX	The counter reaches maximum when it contains all ones.
UPDATE	The timer/counter signals an update when it reaches ZERO or TOP, depending on the direction settings.
Timer	The timer/counter clock control is handled by an internal source.
Counter	The clock control is handled externally (e.g., counting external events).
CC	For compare operations, the CC are referred to as "compare channels." For capture operations, the CC are referred to as "capture channels."

Each TCC instance has up to six compare/capture channels (CCx).

The Counter register (COUNT), Period registers with Buffer (PER and PERBUF), and Compare and Capture registers with buffers (CCx and CCBUFx) are 16- or 24-bit registers, depending on each TCC

instance. Each Buffer register has a Buffer Valid (BUFV) flag that indicates when the buffer contains a new value.

Under normal operation, the counter value is continuously compared to the TOP or ZERO value to determine whether the counter has reached TOP or ZERO. In either case, the TCC can generate interrupt requests or generate events for the Event System. In Waveform Generator mode, these comparisons are used to set the waveform period or pulse width.

A prescaled generic clock (GCLK\_TCCx) and events from the event system can be used to control the counter. The event system is also used as a source to the input capture.

The Recoverable Fault Unit enables event-controlled waveforms by acting directly on the generated waveforms of the TCC compare channels output. These events can restart, halt the timer/counter period, shorten the output pulse active time, or disable waveform output as long as the fault condition is present. This can typically be used for current sensing regulation, and zero-crossing and demagnetization re-triggering.

The MCE0 and MCE1 asynchronous event sources are shared with the recoverable fault unit. Only asynchronous events are used internally when fault unit extension is enabled. See *Event System (EVSYS)* from Related Links for further details on how to configure asynchronous events routing.

Recoverable fault sources can be filtered and/or windowed to avoid false triggering, for example from I/O pin glitches, by using digital filtering, input blanking and qualification options. See *Recoverable Faults* from Related Links.

In order to support applications with different types of motor control, ballast, LED, H-bridge, power converter and other types of power switching applications, the following independent units are implemented in some of the TCC instances as optional and successive units:

- Recoverable faults and non-recoverable faults
- Output matrix
- Dead-time insertion
- Swap
- Pattern generation

See *Timer/Counter for Control Applications - Block Diagram* in the *Block Diagram* from Related Links.

The output matrix (OTMX) can distribute and route out the TCC waveform outputs across the port pins in different configurations, each optimized for different application types. The Dead-Time Insertion (DTI) unit splits the four lower OTMX outputs into two non-overlapping signals: the non-inverted Low Side (LS) and inverted High Side (HS) of the waveform output with optional dead-time insertion between LS and HS switching. The SWAP unit can swap the LS and HS pin outputs and can be used for fast decay motor control.

The pattern generation unit can be used to generate synchronized waveforms with constant logic level on TCC UPDATE conditions. This is useful for easy stepper motor and full bridge control.

The non-recoverable fault module enables event-controlled fault protection by acting directly on the generated waveforms of the timer/counter compare channel outputs. When a non-recoverable fault condition is detected, the output waveforms are forced to a preconfigured value that is safe for the application. This is typically used for instant and predictable shut-down and disabling high current or voltage drives.

The count event sources (TCE0 and TCE1) are shared with the non-recoverable fault extension. The events can be optionally filtered. If the filter options are not used, the non-recoverable faults provide an immediate asynchronous action on waveform output, even for cases where the clock is not present. See *Event System (EVSYS)* from Related Links for further details on how to configure asynchronous events routing.

## Related Links

- [28. Event System \(EVSYS\)](#)
- [40.6.2.6.1. Waveform Output Operations](#)
- [41.3. Block Diagram](#)
- [41.6.3.5. Recoverable Faults](#)

## 41.6.2 Basic Operation

### 41.6.2.1 Initialization

The following registers are enable-protected, meaning that they can only be written when the TCC is disabled (CTRLA.ENABLE=0):

- Control A (CTRLA) register, except Run Standby (RUNSTDBY), Enable (ENABLE) and Software Reset (SWRST) bits
- Recoverable Fault n Control registers (FCTRLA and FCTRLB)
- Waveform Extension Control register (WEXCTRL)
- Drive Control register (DRVCTRL)
- Event Control register (EVCTRL)

Enable-protected bits in the 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.

Before the TCC is enabled, it must be configured as outlined by the following steps:

1. Enable the TCC bus clock if not already enabled by default (PB1\_CLK).
2. If Capture mode is required, enable the channel in Capture mode by writing a '1' to the Capture Enable bit in the Control A register (CTRLA.CPTEN).

Optionally, the following configurations can be set before enabling TCC:

1. Select PRESCALER setting in the Control A register (CTRLA.PRESCALER).
2. Select Prescaler Synchronization setting in Control A register (CTRLA.PRESCSYNC).
3. If down-counting operation is desired, write the Counter Direction bit in the Control B Set register (CTRLBSET.DIR) to '1'.
4. Select the Waveform Generation operation in the WAVE register (WAVE.WAVEGEN).
5. Select the Waveform Output Polarity in the WAVE register (WAVE.POL).
6. The waveform output can be inverted for the individual channels using the Waveform Output Invert Enable bit group in the Driver register (DRVCTRL.INVEN).

### 41.6.2.2 Enabling, Disabling, and Resetting

The TCC is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The TCC is disabled by writing a zero to CTRLA.ENABLE.

The TCC is reset by writing '1' to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the TCC, except DBGCTRL, will be reset to their initial state, and the TCC will be disabled.

The TCC must be disabled before the TCC is reset to avoid undefined behavior.

### 41.6.2.3 Prescaler Selection

The GCLK\_TCCx clock is fed into the internal prescaler.

The prescaler consists of a counter that counts up to the selected prescaler value, whereupon the output of the prescaler toggles.

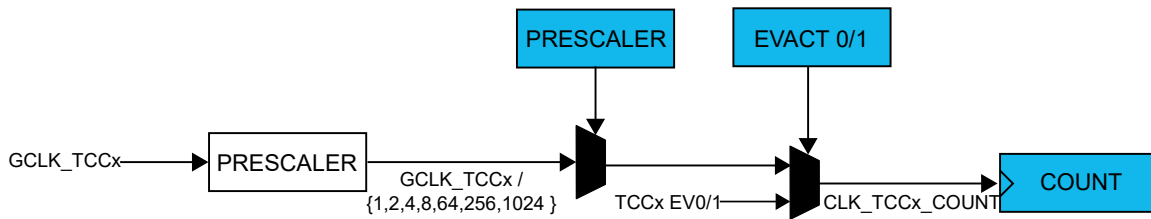
If the prescaler value is higher than one, the Counter Update condition can be optionally executed on the next GCLK\_TCCx clock pulse or the next prescaled clock pulse. For further details, refer to the Prescaler (CTRLA.PRESCALER) and Counter Synchronization (CTRLA.PRESYNC) descriptions.

Prescaler outputs from 1 to 1/1024 are available. For a complete list of available prescaler outputs, see the register description for the Prescaler bit group in the Control A register (CTRLA.PRESCALER).

**Note:** When counting events, the prescaler is bypassed.

The joint stream of prescaler ticks and event action ticks is called CLK\_TCCx\_COUNT.

**Figure 41-2.** Prescaler



#### 41.6.2.4 Counter Operation

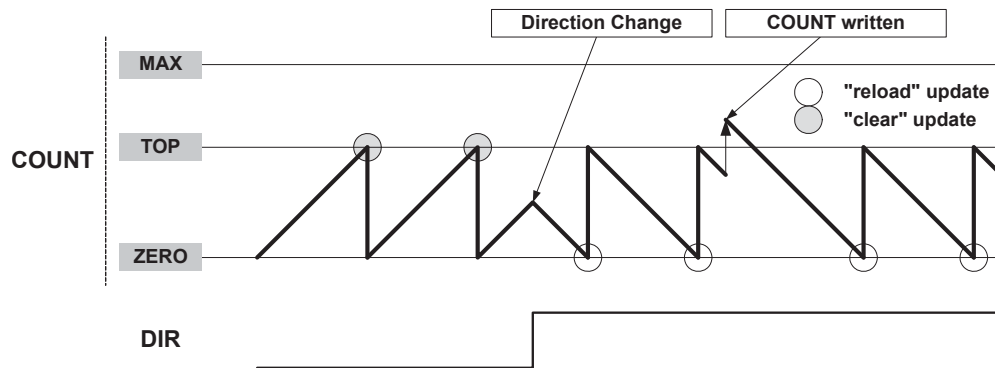
Depending on the mode of operation, the counter is cleared, reloaded, incremented, or decremented at each TCC clock input (CLK\_TCCx\_COUNT). A counter clear or reload mark the end of current counter cycle and the start of a new one.

The counting direction is set by the Direction bit in the Control B register (CTRLB.DIR). If the bit is zero, it's counting up and one if counting down.

The counter will count up or down for each tick (clock or event) until it reaches TOP or ZERO. When it's counting up and TOP is reached, the counter will be set to zero at the next tick (overflow) and the Overflow Interrupt Flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF) will be set. When down-counting, the counter is reloaded with the TOP value when ZERO is reached (underflow), and INTFLAG.OVF is set.

INTFLAG.OVF can be used to trigger an interrupt, or an event. An overflow/underflow occurrence (i.e. a compare match with TOP/ZERO) will stop counting if the One-Shot bit in the Control B register is set (CTRLBSET.ONESHOT). The One-Shot feature is explained in the [Additional Features](#) section.

**Figure 41-3.** Counter Operation



It is possible to change the counter value (by writing directly in the COUNT register) even when the counter is running. The COUNT value will always be ZERO or TOP, depending on direction set by CTRLBSET.DIR or CTRLBCLR.DIR, when starting the TCC, unless a different value has been written to it, or the TCC has been stopped at a value other than ZERO. The write access has higher priority than count, clear, or reload. The direction of the counter can also be changed during normal operation. See also [Figure 41-3](#).

#### Stop Command

A stop command can be issued from software by using TCC Command bits in Control B Set register (CTRLBSET.CMD=0x2, STOP).

When a stop is detected while the counter is running, the counter will maintain its current value. If the waveform generation (WG) is used, all waveforms are set to a state defined in Non-Recoverable State x Output Enable bit and Non-Recoverable State x Output Value bit in the Driver Control register (DRVCTRL.NREx and DRVCTRL.NRVx), and the Stop bit in the Status register is set (STATUS.STOP).

### Pause Event Action

A pause command can be issued when the stop event action is configured in the Input Event Action 1 bits in Event Control register (EVCTRL.EVACT1=0x3, STOP).

When a pause is detected, the counter can stop immediately maintaining its current value and all waveforms keep their current state, as long as a start event action is detected: Input Event Action 0 bits in Event Control register (EVCTRL.EVACT0=0x3, START).

### Re-Trigger Command and Event Action

A re-trigger command can be issued from software by using TCC Command bits in Control B Set register (CTRLBSET.CMD=0x1, RETRIGGER), or from event when the re-trigger event action is configured in the Input Event 0/1 Action bits in Event Control register (EVCTRL.EVACTn=0x1, RETRIGGER).

When the command is detected during counting operation, the counter will be reloaded or cleared, depending on the counting direction (CTRLBSET.DIR or CTRLBCLR.DIR). The Re-Trigger bit in the Interrupt Flag Status and Clear register will be set (INTFLAG.TRG). It is also possible to generate an event by writing a '1' to the Re-Trigger Event Output Enable bit in the Event Control register (EVCTRL.TRGE0). If the re-trigger command is detected when the counter is stopped, the counter will resume counting operation from the value in COUNT.

#### Note:

When a re-trigger event action is configured in the Event Action bits in the Event Control register (EVCTRL.EVACTn=0x1, RETRIGGER), enabling the counter will not start the counter. The counter will start on the next incoming event and restart on corresponding following event.

### Start Event Action

The start action can be selected in the Event Control register (EVCTRL.EVACT0=0x3, START) and can start the counting operation when previously stopped. The event has no effect if the counter is already counting. When the module is enabled, the counter operation starts when the event is received or when a re-trigger software command is applied.

#### Note:

When a start event action is configured in the Event Action bits in the Event Control register (EVCTRL.EVACT0=0x3, START), enabling the counter will not start the counter. The counter will start on the next incoming event, but it will not restart on subsequent events.

### Count Event Action

The TCC can count events. When an event is received, the counter increases or decreases the value, depending on direction settings (CTRLBSET.DIR or CTRLBCLR.DIR).

The count event action is selected by the Event Action 0 bit group in the Event Control register (EVCTRL.EVACT0=0x5, COUNT).

### Direction Event Action

The direction event action can be selected in the Event Control register (EVCTRL.EVACT1=0x2, DIR). When this event is used, the asynchronous event path specified in the event system must be configured or selected. The direction event action can be used to control the direction of the counter operation, depending on external events level. When received, the event level overrides

the Direction settings (CTRLBSET.DIR or CTRLBCLR.DIR) and the direction bit value is updated accordingly.

### Increment Event Action

The increment event action can be selected in the Event Control register (EVCTRL.EVACT0=0x4, INC) and can change the Counter state when an event is received. When the TCE0 event (TCCx\_EV0) is received, the counter increments, whatever the direction setting (CTRLBSET.DIR or CTRLBCLR.DIR) is.

### Decrement Event Action

The decrement event action can be selected in the Event Control register (EVCTRL.EVACT1=0x4, DEC) and can change the Counter state when an event is received. When the TCE1 (TCCx\_EV1) event is received, the counter decrements, whatever the direction setting (CTRLBSET.DIR or CTRLBCLR.DIR) is.

### Non-recoverable Fault Event Action

Non-recoverable fault actions can be selected in the Event Control register (EVCTRL.EVACTn=0x7, FAULT). When received, the counter will be stopped and the output of the compare channels is overridden according to the Driver Control register settings (DRVCTRL.NREx and DRVCTRL.NRVx). TCE0 and TCE1 must be configured as asynchronous events.

### Event Action Off

If the event action is disabled (EVCTRL.EVACTn=0x0, OFF), enabling the counter will also start the counter.

### Related Links

[41.6.3.1. One-Shot Operation](#)

## 41.6.2.5 Compare Operations

By default, the Compare/Capture channel is configured for compare operations. To perform capture operations, it must be re-configured.

When using the TCC with the Compare/Capture Value registers (CCx) for compare operations, the counter value is continuously compared to the values in the CCx registers. This can be used for timer or for waveform operation.

The Channel x Compare/Capture Buffer Value (CCBUFx) registers provide double buffer capability. The double buffering synchronizes the update of the CCx register with the buffer value at the UPDATE condition or a force update command (CTRLBSET.CMD=0x3, UPDATE). See *Double Buffering* from Related Links. The synchronization prevents the occurrence of odd-length, non-symmetrical pulses and ensures glitch-free output.

### Related Links

[41.6.2.6. Double Buffering](#)

### 41.6.2.5.1 Waveform Output Generation Operations

The compare channels can be used for waveform generation on output port pins. To make the waveform available on the connected pin, the following requirements must be fulfilled:

1. Choose a Waveform Generation mode in the Waveform Generation Operation bit in Waveform register (WAVE.WAVEGEN).
2. Optionally, invert the waveform output WO[x] by writing the corresponding Waveform Output x Inversion bit in the Driver Control register (DRVCTRL.INVENx).
3. Configure the pins with the I/O Pin Controller. See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links.

**Note:** Event must not be used when the compare channel is set in waveform output operating mode.

The counter value is continuously compared with each CCx value. On a comparison match, the Match or Capture Channel x bit in the Interrupt Flag Status and Clear register (INTFLAG.MCx) will be set on the next zero-to-one transition of CLK\_TCC\_COUNT (see Normal Frequency Operation). An interrupt and/or event can be generated on the same condition if Match/Capture occurs, i.e., INTENSET.MCx and/or EVCTRL.MCEOx is '1'. Both interrupt and event can be generated simultaneously.

There are seven waveform configurations for the Waveform Generation Operation bit group in the Waveform register (WAVE.WAVEGEN). This will influence how the waveform is generated and impose restrictions on the top value. The configurations are:

- Normal Frequency (NFRQ)
- Match Frequency (MFRQ)
- Normal Pulse-Width Modulation (NPWM)
- Dual-slope, interrupt/event at TOP (DSTOP)
- Dual-slope, interrupt/event at ZERO (DSBOTTOM)
- Dual-slope, interrupt/event at Top and ZERO (DSBOTH)
- Dual-slope, critical interrupt/event at ZERO (DSCRITICAL)

When using MFRQ configuration, the TOP value is defined by the CC0 register value. For the other waveform operations, the TOP value is defined by the Period (PER) register value.

For dual-slope waveform operations, the update time occurs when the counter reaches ZERO. For the other Waveforms Generation modes, the update time occurs on counter wraparound, on overflow, underflow or re-trigger.

The table below shows the update counter and overflow event/interrupt generation conditions in different operation modes.

**Table 41-4.** Counter Update and Overflow Event/interrupt Conditions

Name	Operation	TOP	Update	Output Waveform		OVFI/Event	
				On Match	On Update	Up	Down
NFRQ	Normal Frequency	PER	TOP/ ZERO	Toggle	Stable	TOP	ZERO
MFRQ	Match Frequency	CC0	TOP/ ZERO	Toggle	Stable	TOP	ZERO
NPWM	Single-slope PWM	PER	TOP/ ZERO	See section 'Output Polarity' below		TOP	ZERO
DSCRITICAL	Dual-slope PWM	PER	ZERO			—	ZERO
DSBOTTOM	Dual-slope PWM	PER	ZERO			—	ZERO
DSBOTH	Dual-slope PWM	PER	TOP <sup>(1)</sup> & ZERO			TOP	ZERO
DSTOP	Dual-slope PWM	PER	ZERO	TOP	—		

1. The UPDATE condition on TOP only will occur when circular buffer is enabled for the channel.

### Related Links

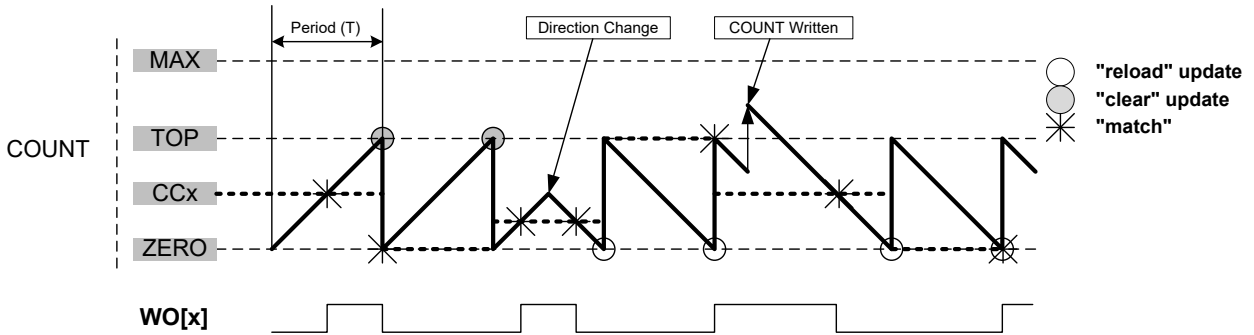
#### [6. I/O Ports and Peripheral Pin Select \(PPS\)](#)

#### 41.6.2.5.2 Normal Frequency (NFRQ)

For Normal Frequency generation, the period time (T) is controlled by the period register (PER). The waveform generation output (WO[x]) is toggled on each compare match between COUNT and CCx, and the corresponding Match or Capture Channel x Interrupt Flag (EVCTRL.MCEOx) will be set.



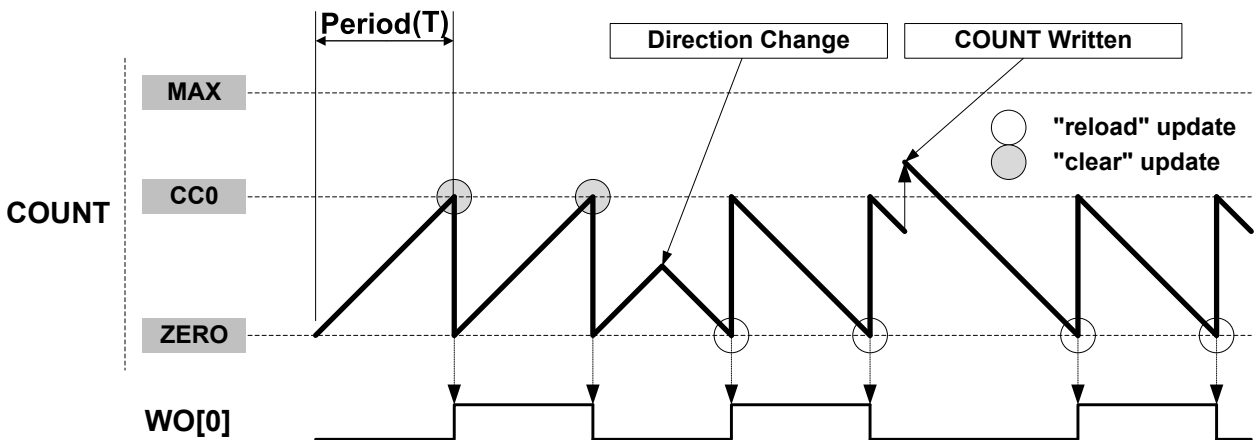
Figure 41-4. Normal Frequency Operation



#### 41.6.2.5.3 Match Frequency (MFRQ)

For Match Frequency generation, the period time (T) is controlled by CC0 register instead of PER. WO[0] toggles on each update condition.

Figure 41-5. Match Frequency Operation



#### 41.6.2.5.4 Normal Pulse-Width Modulation (NPWM)

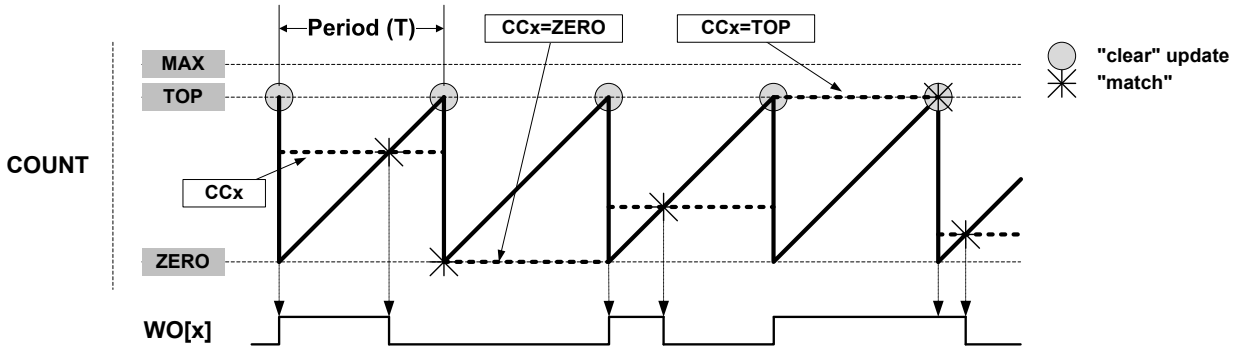
NPWM uses single-slope PWM generation.

#### 41.6.2.5.5 Single-Slope PWM Operation

For single-slope PWM generation, the period time (T) is controlled by Top value, and CCx controls the duty cycle of the generated waveform output. When up-counting, the WO[x] is set at start or compare match between the COUNT and TOP values, and cleared on compare match between COUNT and CCx register values. When down-counting, the WO[x] is cleared at start or compare match between the COUNT and ZERO values, and set on compare match between COUNT and CCx register values.



Figure 41-6. Single-Slope PWM Operation



The following equation calculates the exact resolution for a single-slope PWM ( $R_{PWM\_SS}$ ) waveform:

$$R_{PWM\_SS} = \frac{\log(TOP+1)}{\log(2)}$$

The PWM frequency depends on the Period register value (PER) and the peripheral clock frequency ( $f_{GCLK\_TCCx}$ ), and can be calculated by the following equation:

$$f_{PWM\_SS} = \frac{f_{GCLK\_TCCx}}{N(TOP+1)}$$

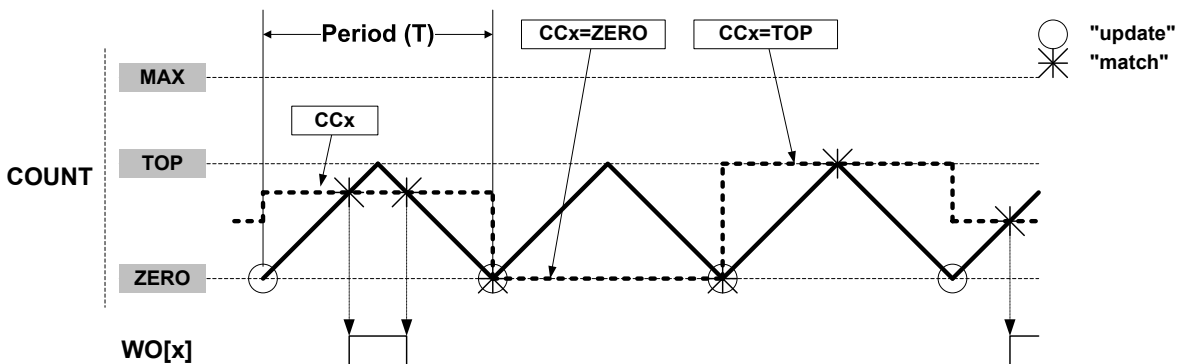
Where N represents the prescaler divider used (1, 2, 4, 8, 16, 64, 256, 1024).

#### 41.6.2.5.6 Dual-Slope PWM Generation

For dual-slope PWM generation, the period setting (TOP) is controlled by PER, while CCx control the duty cycle of the generated waveform output. The figure below shows how the counter repeatedly counts from ZERO to PER and then from PER to ZERO. The waveform generator output is set on compare match when up-counting, and cleared on compare match when down-counting. An interrupt and/or event is generated on TOP (when counting upwards) and/or ZERO (when counting up or down).

In DSBOTH operation, the circular buffer must be enabled to enable the update condition on TOP.

Figure 41-7. Dual-Slope Pulse Width Modulation



Using dual-slope PWM results in a lower maximum operation frequency compared to single-slope PWM generation. The period (TOP) defines the PWM resolution. The minimum resolution is 1 bit (TOP=0x00000001).

The following equation calculates the exact resolution for dual-slope PWM ( $R_{PWM\_DS}$ ):

$$R_{PWM\_DS} = \frac{\log(PER+1)}{\log(2)}$$

The PWM frequency  $f_{P_{PWM\_DS}}$  depends on the period setting (TOP) and the peripheral clock frequency  $f_{GCLK\_TCCx}$ , and can be calculated by the following equation:

$$f_{P_{PWM\_DS}} = \frac{f_{GCLK\_TCCx}}{2N \cdot PER}$$

$N$  represents the prescaler divider used. The waveform generated will have a maximum frequency of half of the TCC clock frequency ( $f_{GCLK\_TCCx}$ ) when TOP is set to 0x00000001 and no prescaling is used.

The pulse width ( $P_{P_{PWM\_DS}}$ ) depends on the compare channel (CCx) register value and the peripheral clock frequency ( $f_{GCLK\_TCCx}$ ), and can be calculated by the following equation:

$$P_{P_{PWM\_DS}} = \frac{2N \cdot (TOP - CCx)}{f_{GCLK\_TCCx}}$$

$N$  represents the prescaler divider used.

**Note:** In DSTOP, DSBOTTOM and DSBOTH operation, when TOP is lower than MAX/2, the CCx MSB bit defines the ramp on which the CCx Match interrupt or event is generated. (Rising if CCx[MSB] = 0, falling if CCx[MSB] = 1.)

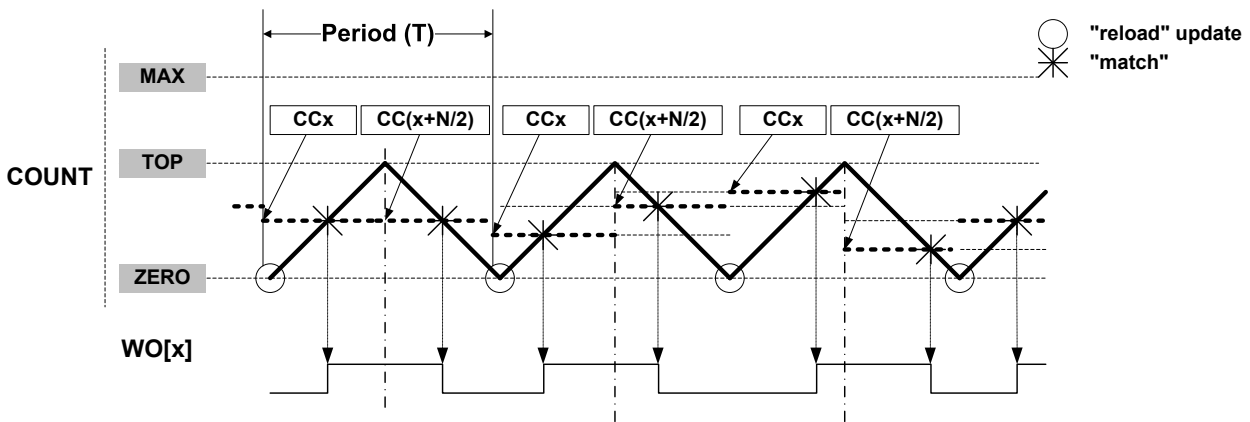
### Related Links

[41.6.3.2. Circular Buffer](#)

#### 41.6.2.5.7 Dual-Slope Critical PWM Generation

Critical mode generation allows generation of non-aligned centered pulses. In this mode, the period time is controlled by PER while CCx control the generated waveform output edge during up-counting and CC(x+CC\_NUM/2) control the generated waveform output edge during down-counting.

Figure 41-8. Dual-Slope Critical Pulse Width Modulation (N=CC\_NUM)



#### 41.6.2.5.8 Output Polarity

The polarity (WAVE.POLx) is available in all waveform output generation. In single-slope and dual-slope PWM operation, it is possible to invert the pulse edge alignment individually on start or end of a PWM cycle for each compare channels. The table below shows the waveform output set/clear conditions, depending on the settings of timer/counter, direction, and polarity.

**Table 41-5. Waveform Generation Set/Clear Conditions**

Waveform Generation Operation	DIR	POLx	Waveform Generation Output Update	
			Set	Clear
Single-Slope PWM	0	0	Timer/counter matches TOP	Timer/counter matches CCx
		1	Timer/counter matches CC	Timer/counter matches TOP
	1	0	Timer/counter matches CC	Timer/counter matches ZERO
		1	Timer/counter matches ZERO	Timer/counter matches CC
Dual-Slope PWM	x	0	Timer/counter matches CC when counting up	Timer/counter matches CC when counting down
		1	Timer/counter matches CC when counting down	Timer/counter matches CC when counting up

In Normal and Match Frequency, the WAVE.POLx value represents the initial state of the waveform output.

#### 41.6.2.6 Double Buffering

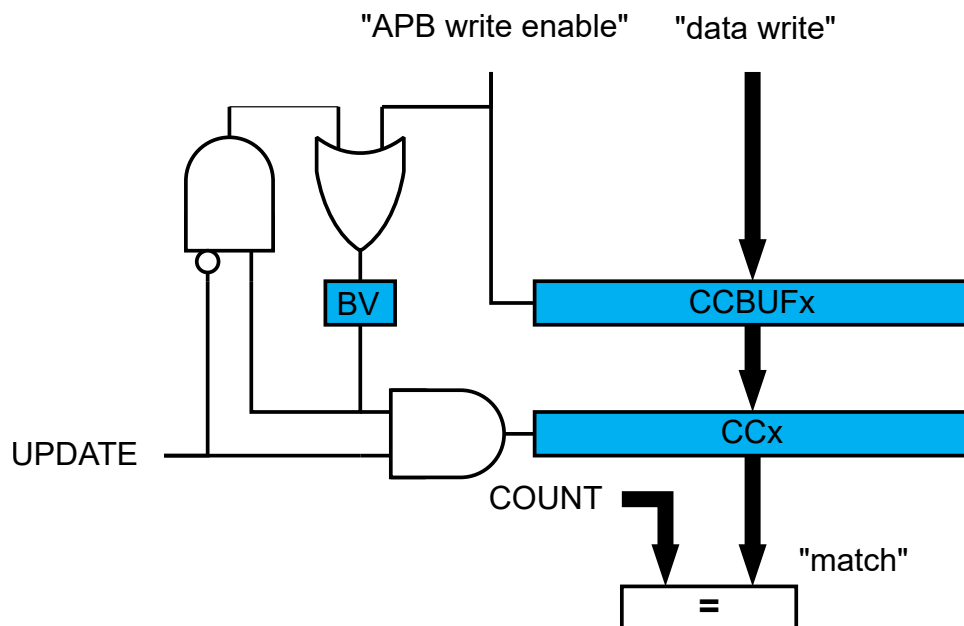
The Pattern (PATT), Period (PER) and Compare Channels (CCx) registers are all double buffered. Each buffer register has a buffer valid (PATTBUFV, PERBUFV and CCBUFVx) bit in the STATUS register that indicates that the Buffer register contains a valid value that can be copied into the corresponding register. As long as the respective Buffer Valid Status flags (PATTBUFV, PERBUFV or CCBUFVx) are set to '1', the related SYNCBUSY bits are set (SYNCBUSY.PATT, SYNCBUSY.PER or SYNCBUSY.CCx), a write to the respective PATT/PATTBUF, PER/PERBUF or CCx/CCBUFx registers will generate a PAC error, and read access to the respective PATT, PER or CCx register is invalid.

When the Buffer Valid Flag bit in the STATUS register is '1' and the Lock Update bit in the CTRLB register is set to '0' (writing CTRLBCLR.LUPD to '1'), double buffering is enabled: the data from the buffer registers will be copied into the corresponding register under the hardware UPDATE conditions, then the Buffer Valid flags bit in the STATUS register is automatically cleared by the hardware.

**Note:** The software update command (CTRLBSET.CMD=0x3) acts independently of the LUPD value.

A compare register is double buffered as in the following figure.

**Figure 41-9. Compare Channel Double Buffering**



Both the registers (PATT/PER/CCx) and corresponding Buffer registers (PATTBUFFERBUF/CCBUFx) are available in the I/O register map, and the double buffering feature is not mandatory. The double buffering is disabled by writing a '1' to CTRLSET.LUPD.

**Note:** In NFRQ, MFRQ or PWM Down-Counting Counter mode (CTRLBSET.DIR=1), when double buffering is enabled (CTRLBCLR.LUPD=1), the PERBUF register is continuously copied into the PER independently of the update conditions.

### Changing the Period

The counter period can be changed by writing a new Top value to the Period register (PER or CC0, depending on the Waveform Generation mode). Any period update on the registers (PER or CCx) is effective after the synchronization delay, whatever double buffering enabling is.

Figure 41-10. Unbuffered Single-Slope Up-Counting Operation

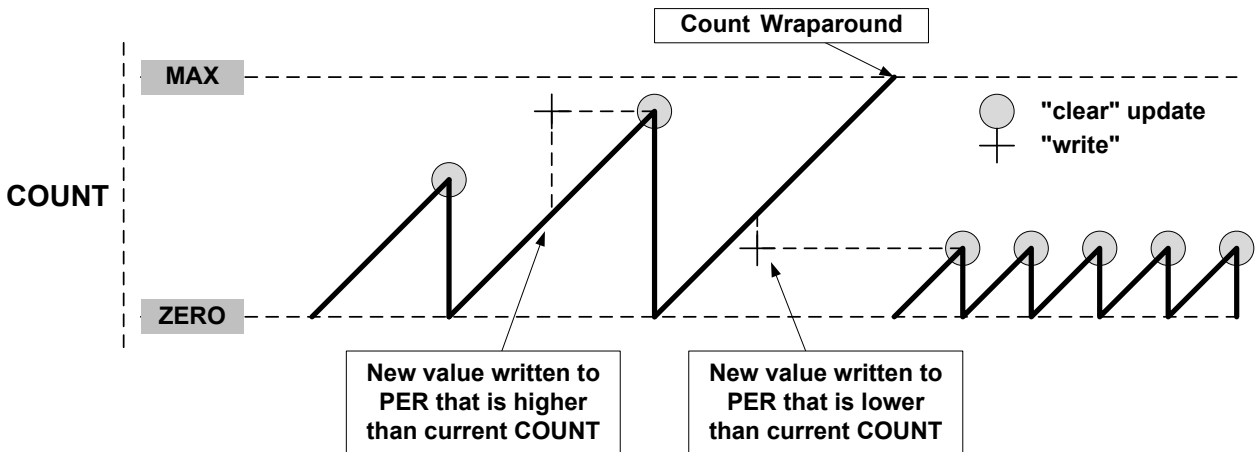
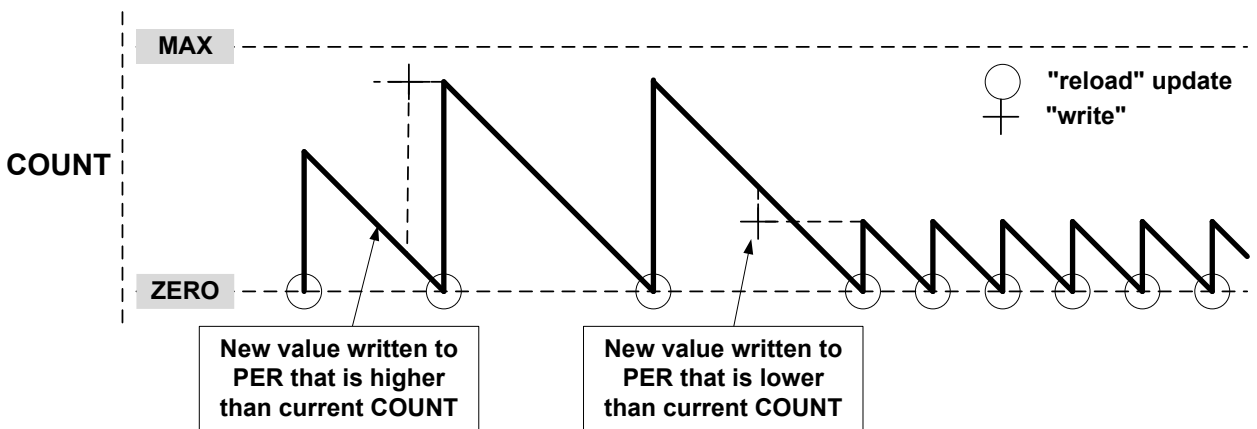
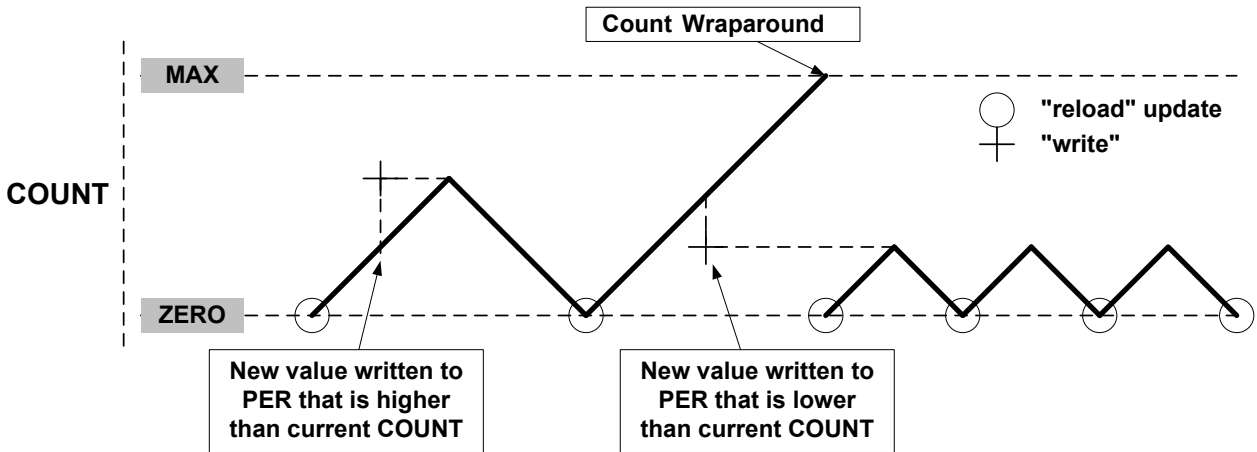


Figure 41-11. Unbuffered Single-Slope Down-Counting Operation



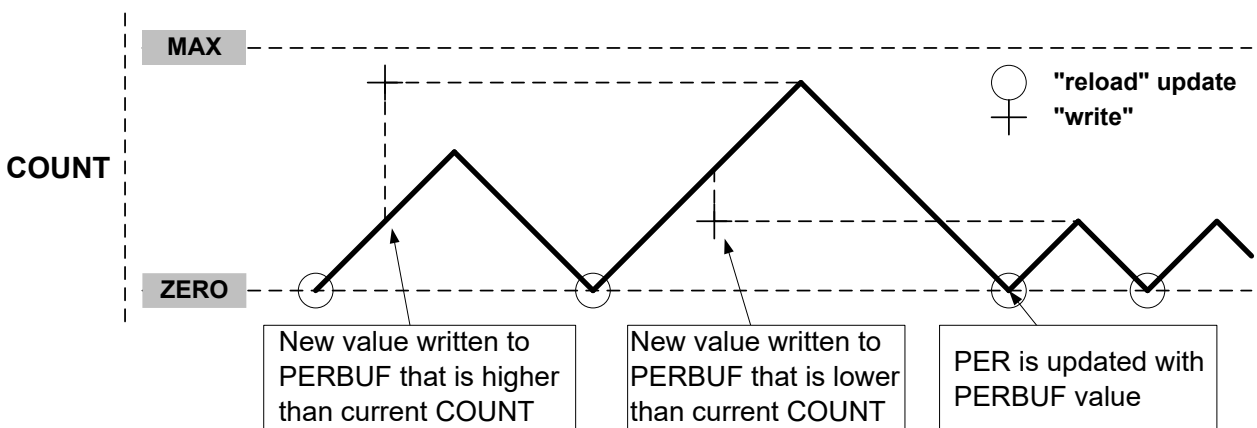
A counter wraparound can occur in any operation mode when up-counting without buffering (see [Figure 41-10](#)). COUNT and TOP are continuously compared, so when a new value that is lower than the current COUNT is written to TOP, COUNT will wrap before a compare match.

Figure 41-12. Unbuffered Dual-Slope Operation



When double buffering is used, the buffer can be written at any time and the counter will still maintain correct operation. The period register is always updated on the update condition, as shown in Figure 41-13. This prevents wraparound and the generation of odd waveforms.

Figure 41-13. Changing the Period Using Buffering



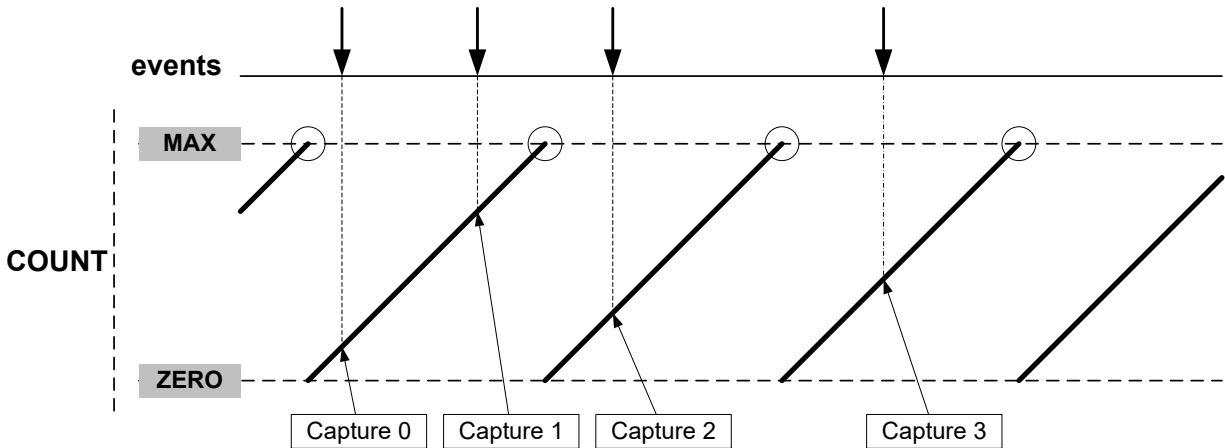
#### 41.6.2.7 Capture Operations

To enable and use capture operations, the Match or Capture Channel x Event Input Enable bit in the Event Control register (EVCTRL.MCEIx) must be written to '1'. The capture channels to be used must also be enabled in the Capture Channel x Enable bit in the Control A register (CTRLA.CPTENx) before capturing can be performed.

#### Event Capture Action

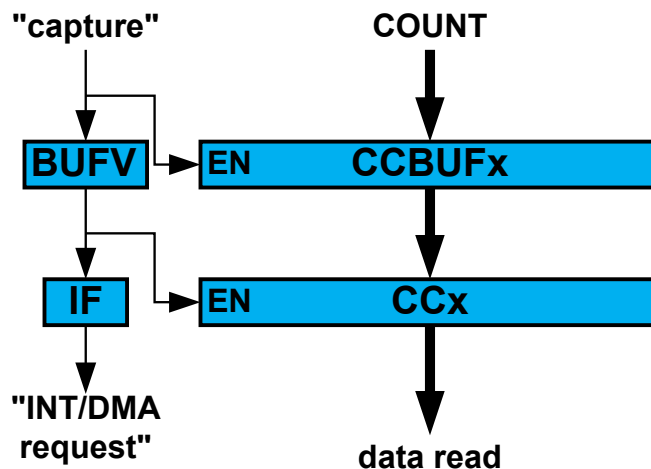
The compare/capture channels can be used as input capture channels to capture events from the Event System, and give them a timestamp. The following figure shows four capture events for one capture channel. Event system channels must be configured to operate in asynchronous mode when used for capture operations.

Figure 41-14. Input Capture Timing



For input capture, the Buffer register and the corresponding CCx act like a FIFO. When CCx is empty or read, any content in CCBUFx is transferred to CCx. The Buffer Valid flag is passed to set the CCx Interrupt flag (IF) and generate the optional interrupt, event, or DMA request. The CCBUFx register value cannot be read, all captured data must be read from the CCx register.

Figure 41-15. Capture Double Buffering



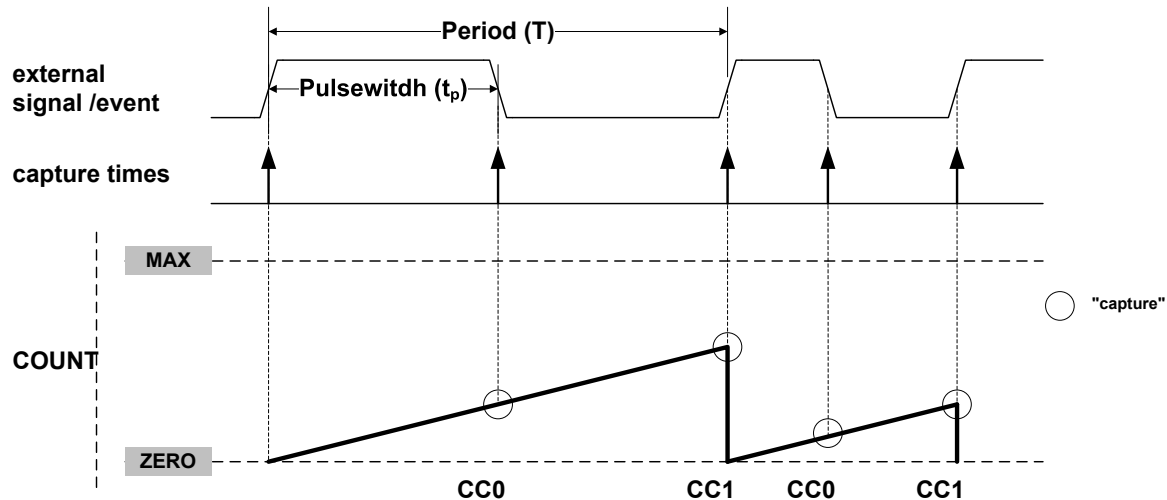
The TCC can detect capture overflow of the input capture channels. When a new capture event is detected while the Capture Buffer Valid flag (STATUS.CCBUFV) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

### Period and Pulse-Width (PPW) Capture Action

The TCC can perform two input captures and restart the counter on one of the edges. This enables the TCC to measure the pulse-width and period and to characterize the frequency  $f$  and  $dutyCycle$  of an input signal, as shown below:

$$f = \frac{1}{T} \quad , \quad dutyCycle = \frac{t_p}{T}$$

Figure 41-16. PWP Capture



Selecting PWP or PPW in the Timer/Counter Event Input 1 Action bit group in the Event Control register (EVCTRL.EVACT1) enables the TCC to perform one capture action on the rising edge and the other one on the falling edge. When using PPW event action, period  $T$  will be captured into CC0 and the pulse-width  $t_p$  into CC1. The PWP (Pulse-width and Period) event action offers the same functionality, but  $T$  will be captured into CC1 and  $t_p$  into CC0.

The Timer/Counter Event  $x$  Invert Enable bit in Event Control register (EVCTRL.TCEINV $x$ ) is used for event source  $x$  to select whether the wraparound must occur on the rising edge or the falling edge. If EVCTRL.TCEINV $x$ =1, the wraparound will happen on the falling edge.

The corresponding capture is done only if the channel is enabled in Capture mode (CTRLA.CPTEN $x$ =1). If not, the capture action will be ignored and the channel will be enabled in compare mode of operation. When only one of these channel is required, the other channel can be used for other purposes.

The TCC can detect capture overflow of the input capture channels. When a new capture event is detected while the INTFLAG.MC $x$  is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

**Note:** When up-counting (CTRLBSET.DIR=0), counter values lower than 1 cannot be captured in Capture Minimum mode (FCTRLn.CAPTURE=CAPTMIN). To capture the full range including value 0, the TCC must be configured in Down-counting mode (CTRLBSET.DIR=0).

**Note:** In dual-slope PWM operation, and when TOP is lower than MAX/2, the CC $x$  MSB captures the CTRLB.DIR state to identify the ramp on which the capture has been done. For rising ramps CC $x$ [MSB] is zero, for falling ramps CC $x$ [MSB]=1.

### 41.6.3 Additional Features

#### 41.6.3.1 One-Shot Operation

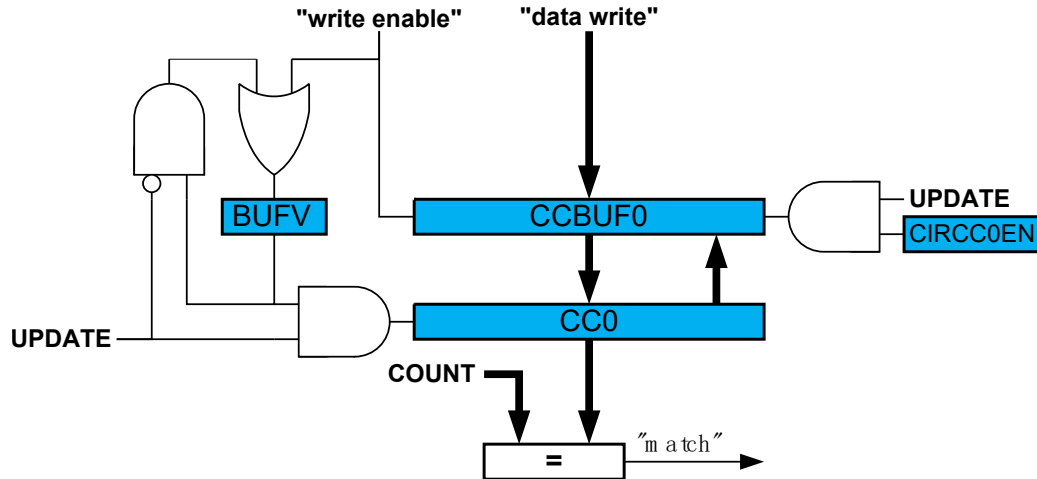
When one-shot is enabled, the counter automatically stops on the next Counter Overflow or Underflow condition. When the counter is stopped, the Stop bit in the Status register (STATUS.STOP) is set and the waveform outputs are set to the value defined by DRVCTRL.NRE $x$  and DRVCTRL.NRV $x$ .

One-shot operation can be enabled by writing a '1' to the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) and disabled by writing a '1' to CTRLBCLR.ONESHOT. When enabled, the TCC will count until an overflow or underflow occurs and stop counting. The one-shot operation can be restarted by a re-trigger software command, a re-trigger event or a start event. When the counter restarts its operation, STATUS.STOP is automatically cleared.

### 41.6.3.2 Circular Buffer

The Period register (PER) and the Compare Channels register (CC0 to CC5) support circular buffer operation. When circular buffer operation is enabled, the PER or CCx values are copied into the corresponding buffer registers at each update condition. Circular buffering is dedicated to RAMP2, RAMP2A, and DSBOTHS operations.

Figure 41-17. Circular Buffer on Channel 0



### 41.6.3.3 Dithering Operation

The TCC supports dithering on Pulse-width or Period on a 16, 32 or 64 PWM cycles frame.

Dithering consists in adding some extra clocks cycles in a frame of several PWM cycles, and can improve the accuracy of the *average* output pulse width and period. The extra clock cycles are added on some of the compare match signals, one at a time, through a "blue noise" process that minimizes the flickering on the resulting dither patterns.

Dithering is enabled by writing the corresponding configuration in the Enhanced Resolution bits in CTRLA register (CTRLA.RESOLUTION):

- DITH4 enable dithering every 16 PWM frames
- DITH5 enable dithering every 32 PWM frames
- DITH6 enable dithering every 64 PWM frames

The DITHERCY bits of COUNT, PER and CCx define the number of extra cycles to add into the frame (DITHERCY bits from the respective COUNT, PER or CCx registers). The remaining bits of COUNT, PER, CCx define the compare value itself.

The pseudo code, giving the extra cycles insertion regarding the cycle is:

```
int extra_cycle(resolution, dithercy, cycle){
    int MASK;
    int value
    switch (resolution){
        DITH4: MASK = 0x0f;
        DITH5: MASK = 0x1f;
        DITH6: MASK = 0x3f;
    }
    value = cycle * dithercy;
    if ((MASK & value) + dithercy) > MASK)
        return 1;
    return 0;
}
```

### Dithering on Period

Writing DITHERCY in PER will lead to an average PWM period configured by the following formulas.



DITH4 mode:

$$PwmPeriod = \left( \frac{DITHERCY}{16} + PER \right) \left( \frac{1}{f_{GCLK\_TCCx}} \right)$$

**Note:** If DITH4 mode is enabled, the last 4 significant bits from PER/CCx or COUNT register correspond to the DITHERCY value, rest of the bits corresponds to PER/CCx or COUNT value.

DITH5 mode:

$$PwmPeriod = \left( \frac{DITHERCY}{32} + PER \right) \left( \frac{1}{f_{GCLK\_TCCx}} \right)$$

DITH6 mode:

$$PwmPeriod = \left( \frac{DITHERCY}{64} + PER \right) \left( \frac{1}{f_{GCLK\_TCCx}} \right)$$

### Dithering on Pulse-Width

Writing DITHERCY in CCx will lead to an average PWM pulse width configured by the following formula.

DITH4 mode:

$$PwmPulseWidth = \left( \frac{DITHERCY}{16} + CCx \right) \left( \frac{1}{f_{GCLK\_TCCx}} \right)$$

DITH5 mode:

$$PwmPulseWidth = \left( \frac{DITHERCY}{32} + CCx \right) \left( \frac{1}{f_{GCLK\_TCCx}} \right)$$

DITH6 mode:

$$PwmPulseWidth = \left( \frac{DITHERCY}{64} + CCx \right) \left( \frac{1}{f_{GCLK\_TCCx}} \right)$$

**Note:** The PWM period will remain static in this case.

### 41.6.3.4 Ramp Operations

Three ramp operation modes are supported. All of them require the timer/counter running in single-slope PWM generation. The Ramp mode is selected by writing to the Ramp Mode bits in the Waveform Control register (WAVE.RAMP).

#### RAMP1 Operation

This is the default PWM operation.

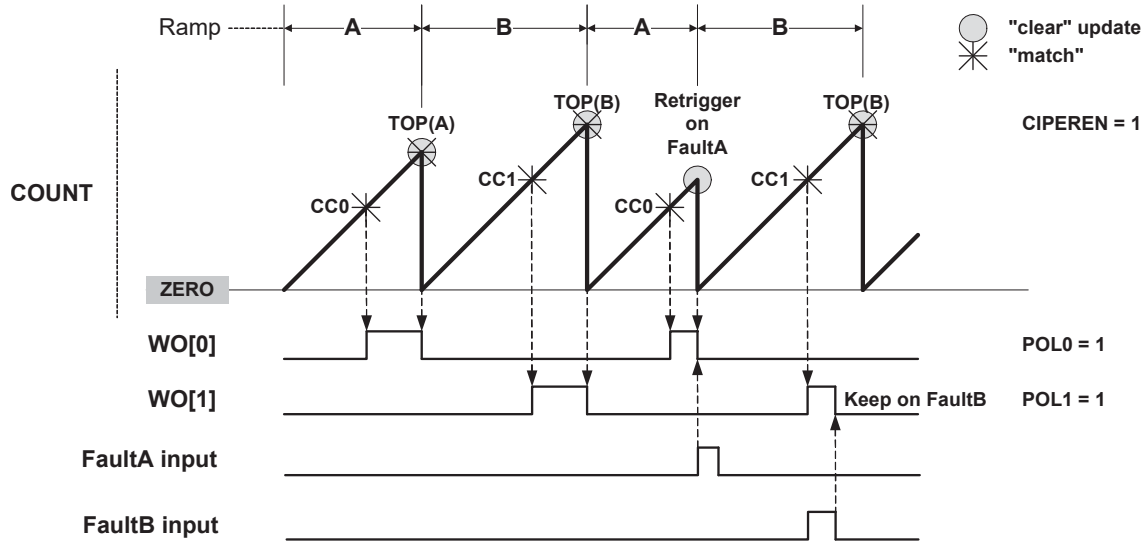
#### RAMP2 Operation

These operation modes are dedicated for power factor correction (PFC), Half-Bridge and Push-Pull SMPS topologies, where two consecutive timer/counter cycles are interleaved (see the following figure). In cycle A, odd channel output is disabled, and in cycle B, even channel output is disabled. The ramp index changes after each update, but can be software modified using the Ramp index command bits in the Control B Set register (CTRLBSET.IDXCMD).

#### Standard RAMP2 (RAMP2) Operation

Ramp A and B periods are controlled by the PER register value. The PER value can be different on each ramp by the Circular Period buffer option in the Wave register (WAVE.CIPEREN=1). This mode uses a two-channel TCC to generate two output signals, or one output signal with another CC channel enabled in Capture mode.

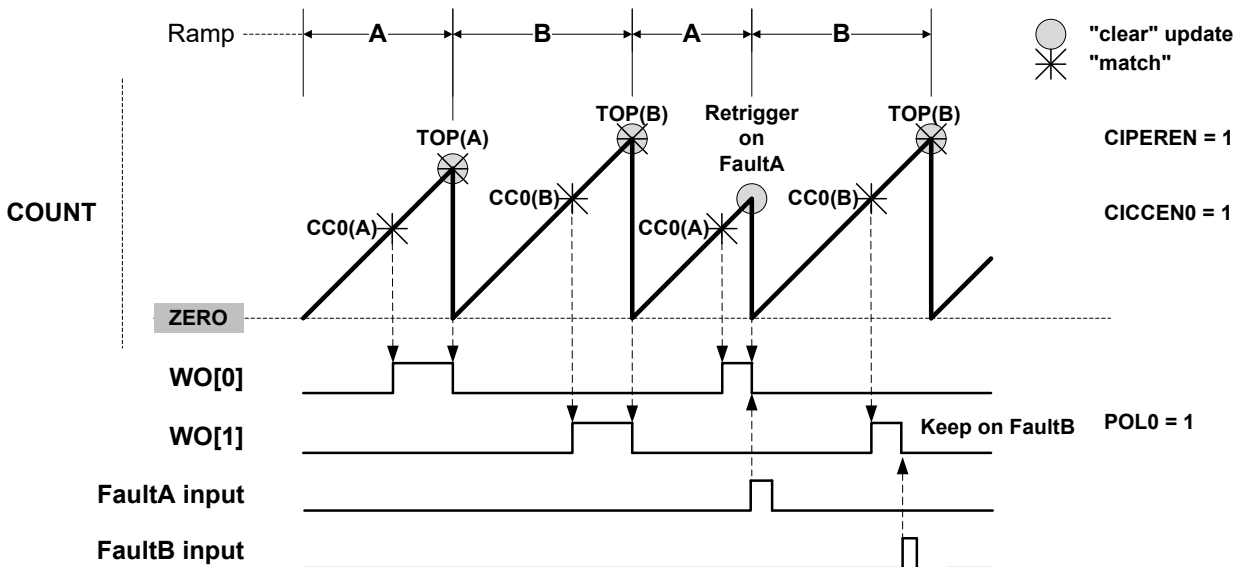
Figure 41-18. RAMP2 Standard Operation



### Alternate RAMP2 (RAMP2A) Operation

Alternate RAMP2 operation is similar to RAMP2, but CC0 controls both WO[0] and WO[1] waveforms when the corresponding circular buffer option is enabled (CIPEREN=1). The waveform polarity is the same on both outputs. Channel 1 can be used in capture mode.

Figure 41-19. RAMP2 Alternate Operation



### Critical RAMP2 (RAMP2C) Operation

Critical RAMP2 operation provides a way to cover RAMP2 operation requirements without the update constraint associated with the use of circular buffers. In this mode, CC0 is controlling the period of ramp A and PER is controlling the period of ramp B. When using more than two channels, WO[0] output is controlled by CC2 (HIGH) and CC0 (LOW). On TCC with 2 channels, a pulse on WO[0] will last the entire period of ramp A, if WAVE.POL0=0.

Figure 41-20. RAMP2 Critical Operation With More Than 2 Channels

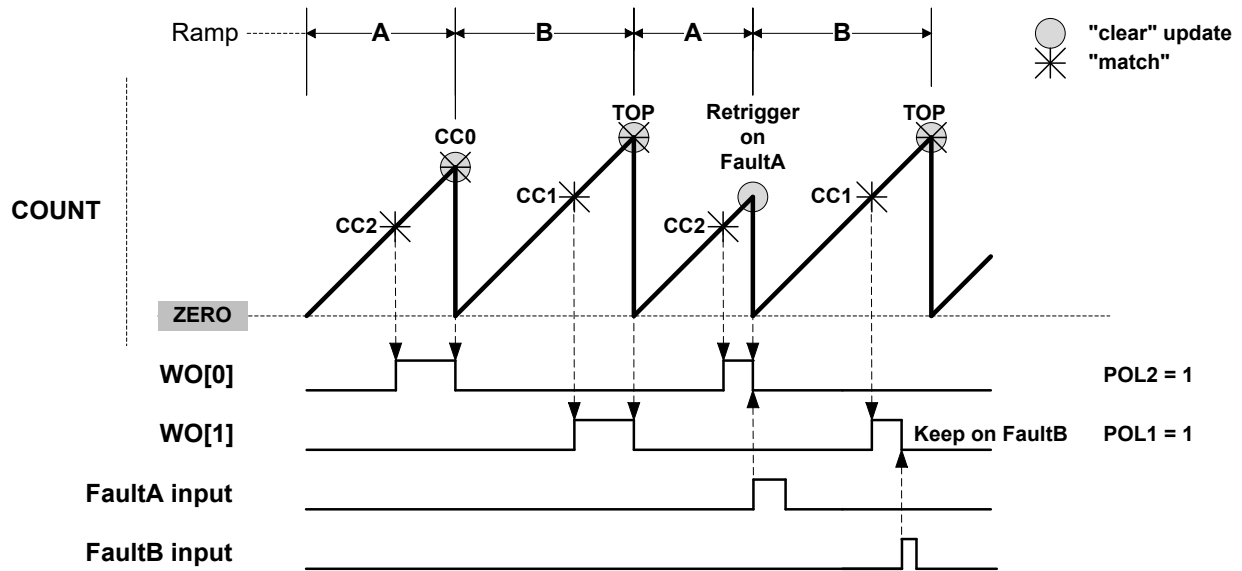
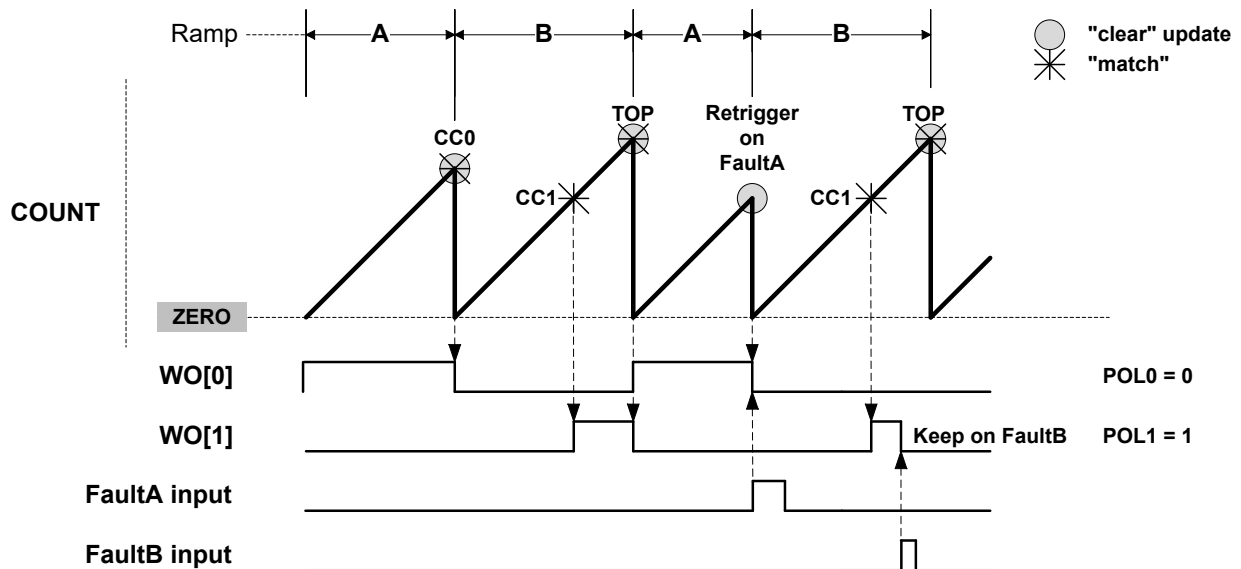


Figure 41-21. RAMP2 Critical Operation With 2 Channels



### 41.6.3.5 Recoverable Faults

Recoverable faults can restart or halt the timer/counter. Two faults, called Fault A and Fault B, can trigger recoverable fault actions on the compare channels CC0 and CC1 of the TCC. The compare channels' outputs can be clamped to inactive state either as long as the fault condition is present, or from the first valid fault condition detection on until the end of the timer/counter cycle.

#### Fault Inputs

The first two channel input events (TCCxMC0 and TCCxMC1) can be used as Fault A and Fault B inputs, respectively. Event system channels connected to these fault inputs must be configured as asynchronous. The TCC must work in a PWM mode.

## Fault Filtering

There are three filters available for each input Fault A and Fault B. They are configured by the corresponding Recoverable Fault n Configuration registers (FCTRLA and FCTRLB). The three filters can either be used independently or in any combination.

**Input Filtering** By default, the event detection is asynchronous. When the event occurs, the fault system will immediately and asynchronously perform the selected fault action on the compare channel output, also in device power modes where the clock is not available. To avoid false fault detection on external events (e.g. due to a glitch on an I/O port) a digital filter can be enabled and configured by the Fault B Filter Value bits in the Fault n Configuration registers (FCTRLn.FILTERVAL). If the event width is less than FILTERVAL (in clock cycles), the event will be discarded. A valid event will be delayed by FILTERVAL clock cycles.

**Fault Blanking** This ignores any fault input for a certain time just after a selected waveform output edge. This can be used to prevent false fault triggering due to signal bouncing, as shown in the figure below. Blanking can be enabled by writing an edge triggering configuration to the Fault n Blanking Mode bits in the Recoverable Fault n Configuration register (FCTRLn.BLANK). The desired duration of the blanking must be written to the Fault n Blanking Time bits (FCTRLn.BLANKVAL).

The blanking time  $t_b$  is calculated by

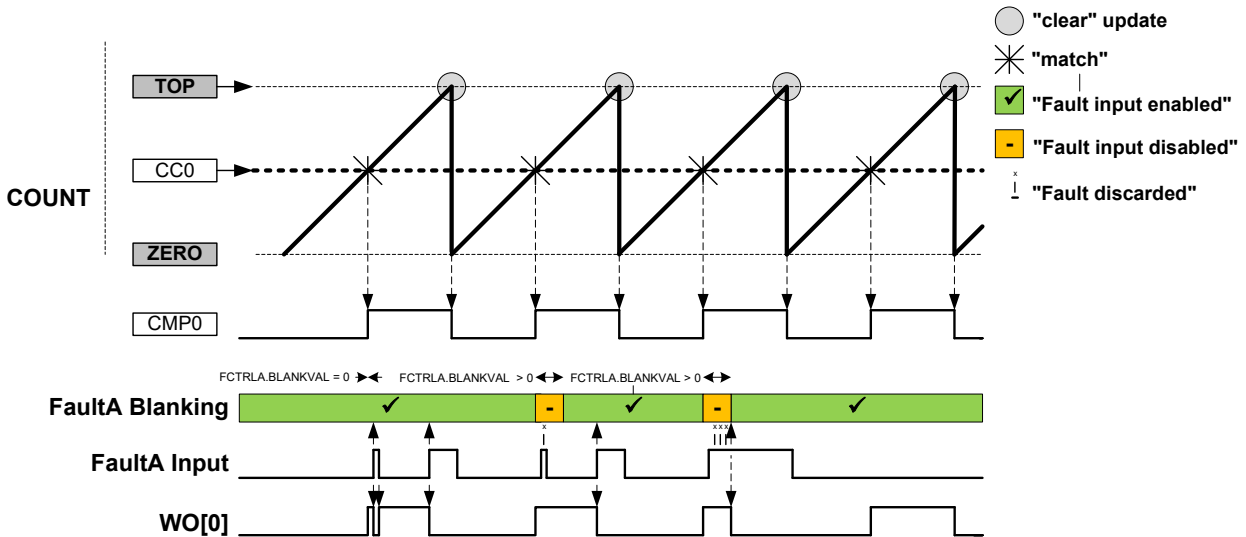
$$t_b = \frac{1 + \text{BLANKVAL}}{f_{\text{GCLK\_TCCx\_PRESC}}}$$

Here,  $f_{\text{GCLK\_TCCx\_PRESC}}$  is the frequency of the prescaled peripheral clock frequency  $f_{\text{GCLK\_TCCx}}$ .

The prescaler is enabled by writing '1' to the Fault n Blanking Prescaler bit (FCTRLn.BLANKPRESC). When disabled,  $f_{\text{GCLK\_TCCx\_PRESC}} = f_{\text{GCLK\_TCCx}}$ . When enabled,  $f_{\text{GCLK\_TCCx\_PRESC}} = f_{\text{GCLK\_TCCx}}/64$ .

The maximum blanking time (FCTRLn.BLANKVAL=255) at  $f_{\text{GCLK\_TCCx}}=96\text{MHz}$  is  $2.67\mu\text{s}$  (no prescaler) or  $170\mu\text{s}$  (prescaling). For  $f_{\text{GCLK\_TCCx}}=1\text{MHz}$ , the maximum blanking time is either  $170\mu\text{s}$  (no prescaling) or  $10.9\text{ms}$  (prescaling enabled).

Figure 41-22. Fault Blanking in RAMP1 Operation with Inverted Polarity



**Fault Qualification** This is enabled by writing a '1' to the Fault n Qualification bit in the Recoverable Fault n Configuration register (FCTRLn.QUAL). When the recoverable fault qualification is enabled (FCTRLn.QUAL=1), the fault input is disabled all the time the corresponding channel output has an inactive level, as shown in the figures below.

Figure 41-23. Fault Qualification in RAMP1 Operation

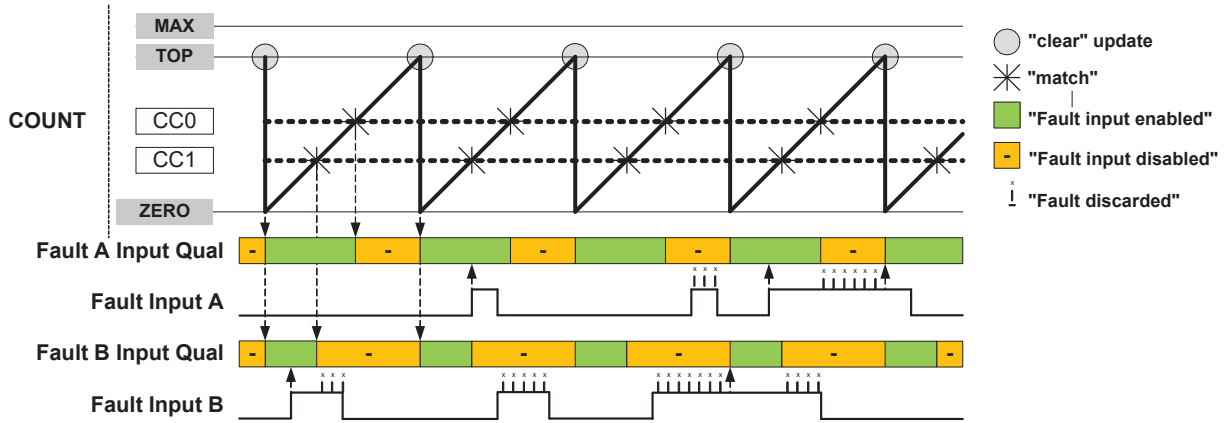
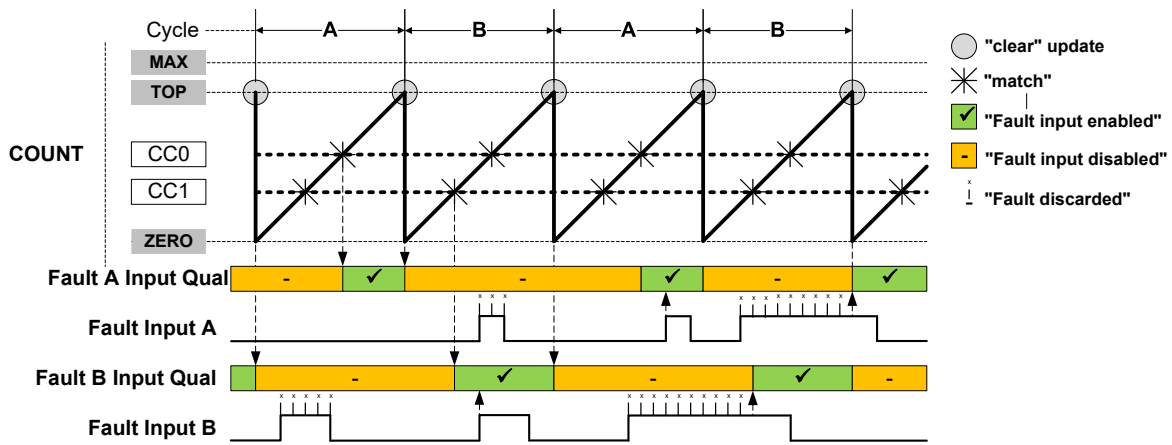


Figure 41-24. Fault Qualification in RAMP2 Operation with Inverted Polarity

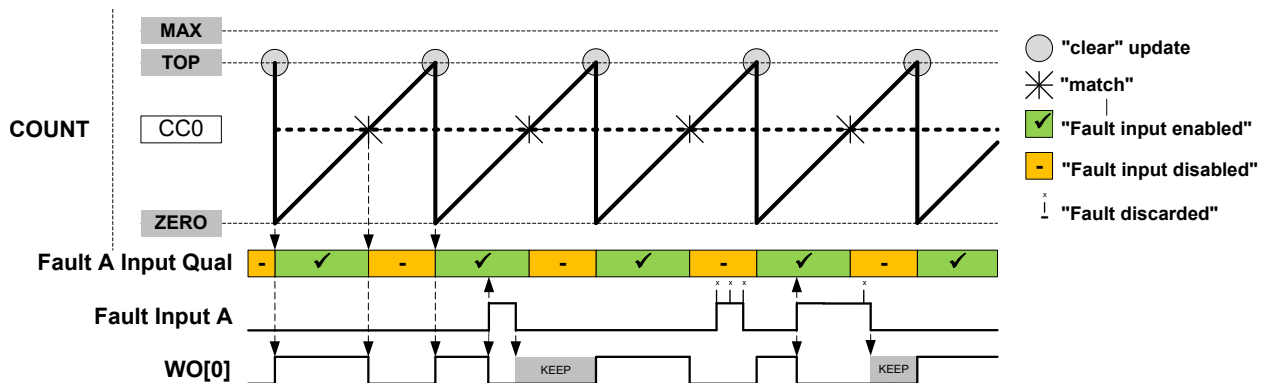


### Fault Actions

Different fault actions can be configured individually for Fault A and Fault B. Most fault actions are not mutually exclusive; hence two or more actions can be enabled at the same time to achieve a result that is a combination of fault actions.

**Keep Action** This is enabled by writing the Fault n Keeper bit in the Recoverable Fault n Configuration register (FCTRLn.KEEP) to '1'. When enabled, the corresponding channel output will be clamped to zero as long as the fault condition is present. The clamp will be released on the start of the first cycle after the fault condition is no longer present, see next Figure.

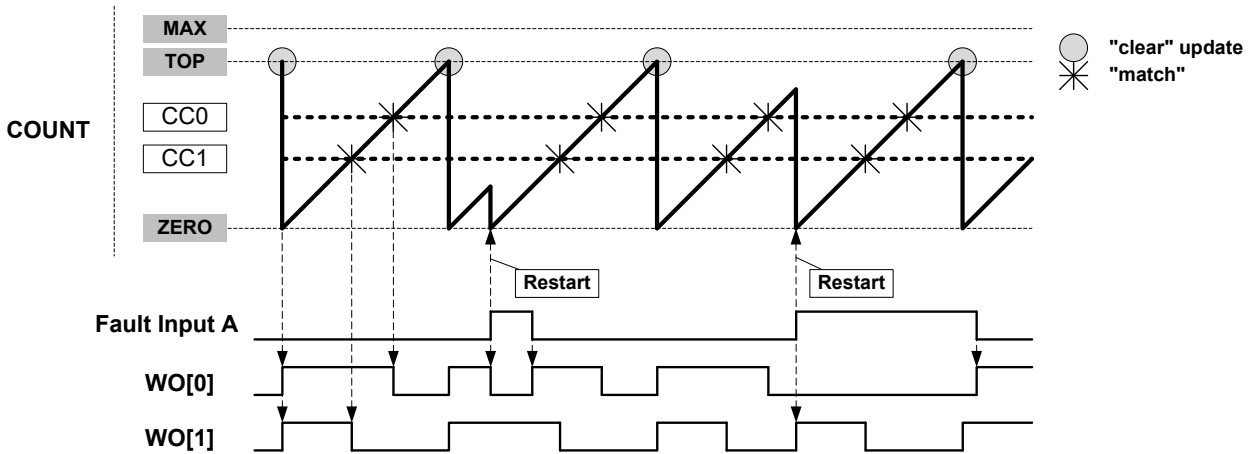
Figure 41-25. Waveform Generation with Fault Qualification and Keep Action



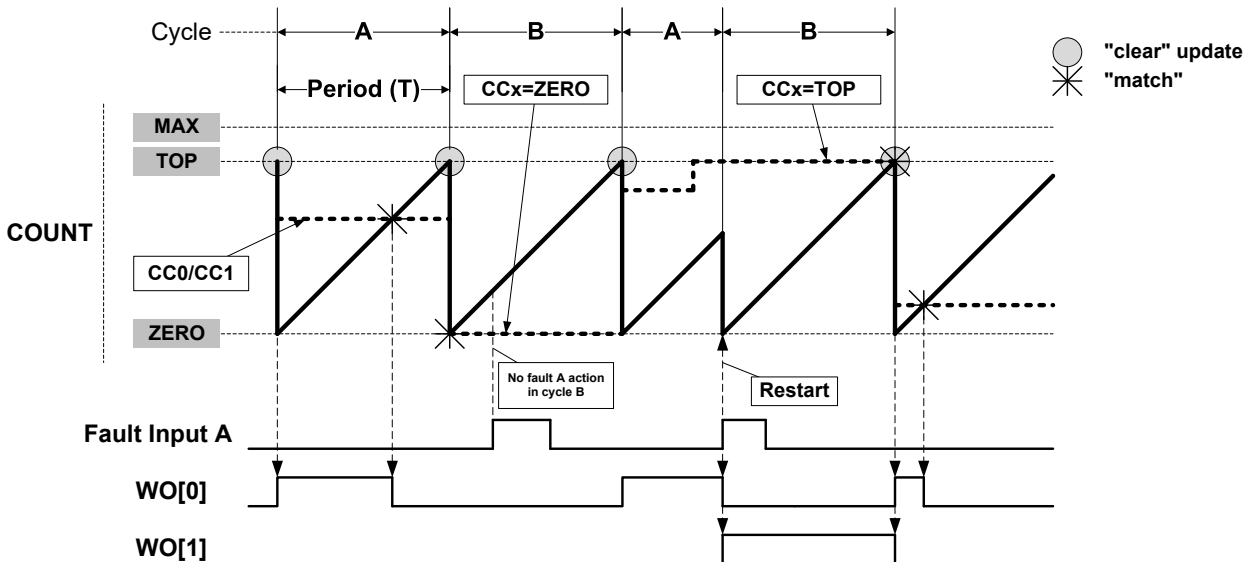
**Restart Action** This is enabled by writing the Fault n Restart bit in Recoverable Fault n Configuration register (FCTRLn.RESTART) to '1'. When enabled, the timer/counter will be restarted as soon as the corresponding fault condition is present. The ongoing cycle is stopped and the timer/counter starts a new cycle, see [Figure 41-26](#). In Ramp 1 mode, when the new cycle starts, the compare outputs will be clamped to inactive level as long as the fault condition is present.

**Note:** For RAMP2 operation, when a new timer/counter cycle starts the cycle index will change automatically, see [Figure 41-27](#). Fault A and Fault B are qualified only during the cycle A and cycle B respectively: Fault A is disabled during cycle B, and Fault B is disabled during cycle A.

**Figure 41-26.** Waveform Generation in RAMP1 mode with Restart Action



**Figure 41-27.** Waveform Generation in RAMP2 mode with Restart Action



**Capture Action** Several capture actions can be selected by writing the Fault n Capture Action bits in the Fault n Control register (FCTRLn.CAPTURE). When one of the capture operations is selected, the counter value is captured when the fault occurs. These capture operations are available:

- CAPT - the equivalent to a standard capture operation.
- CAPTMIN - gets the minimum time stamped value: on each new local minimum captured value, an event or interrupt is issued.
- CAPTMAX - gets the maximum time stamped value: on each new local maximum captured value, an event or interrupt (IT) is issued, see [Figure 41-28](#).
- LOCMIN - notifies by event or interrupt when a local minimum captured value is detected.

- LOCMAX - notifies by event or interrupt when a local maximum captured value is detected.
- DERIVO - notifies by event or interrupt when a local extreme captured value is detected, see Figure 41-29.

*CCx Content:*

In CAPTMIN and CAPTMAX operations, CCx keeps the respective extremum captured values, see Figure 41-28. In LOCMIN, LOCMAX or DERIVO operation, CCx follows the counter value at fault time, see Figure 41-29.

Before enabling CAPTMIN or CAPTMAX mode of capture, the user must initialize the corresponding CCx register value to a value different from zero (for CAPTMIN) top (for CAPTMAX). If the CCx register initial value is zero (for CAPTMIN) top (for CAPTMAX), no captures will be performed using the corresponding channel.

*MCx Behaviour:*

In LOCMIN and LOCMAX operation, capture is performed on each capture event. The MCx interrupt flag is set only when the captured value is above or equal (for LOCMIN) or below or equal (for LOCMAX) to the previous captured value. So interrupt flag is set when a new relative local Minimum (for CAPTMIN) or Maximum (for CAPTMAX) value has been detected. DERIVO is equivalent to an OR function of (LOCMIN, LOCMAX).

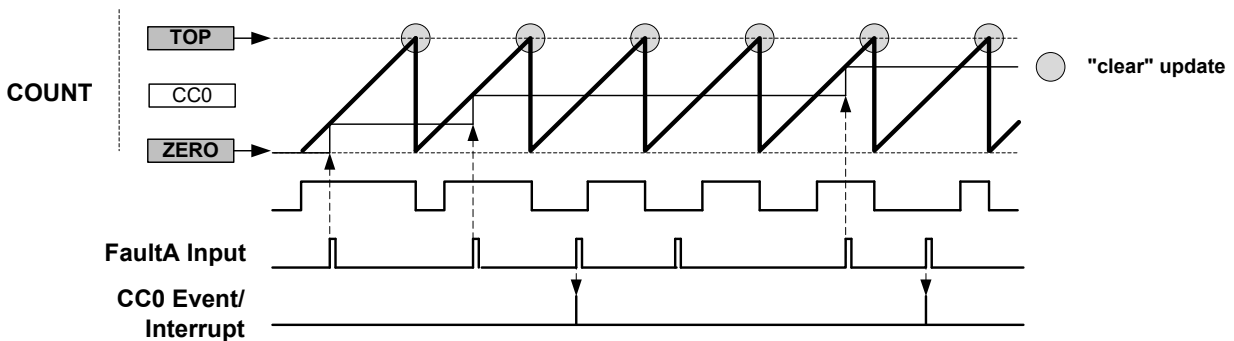
In CAPT operation, capture is performed on each capture event. The MCx interrupt flag is set on each new capture.

In CAPTMIN and CAPTMAX operation, capture is performed only when on capture event time, the counter value is lower (for CAPTMIN) or higher (for CAPMAX) than the last captured value. The MCx interrupt flag is set only when on capture event time, the counter value is higher or equal (for CAPTMIN) or lower or equal (for CAPTMAX) to the value captured on the previous event. So interrupt flag is set when a new absolute local Minimum (for CAPTMIN) or Maximum (for CAPTMAX) value has been detected.

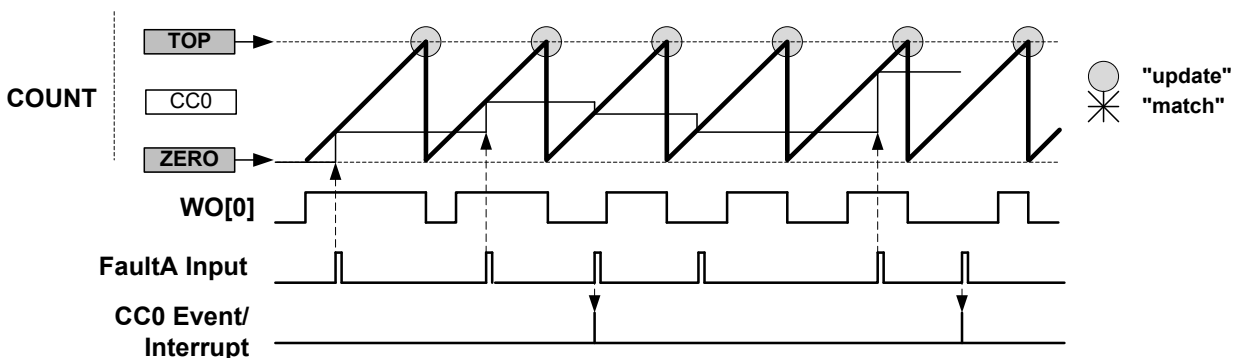
*Interrupt Generation*

In CAPT mode, an interrupt is generated on each filtered Fault n and each dedicated CCx channel capture counter value. In other modes, an interrupt is only generated on an extreme captured value.

**Figure 41-28. Capture Action "CAPTMAX"**



**Figure 41-29. Capture Action "DERIVO"**



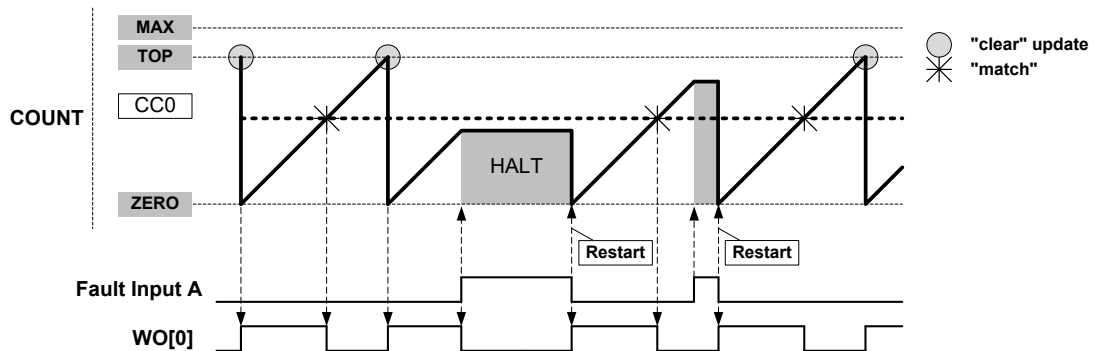
**Hardware Halt Action** This is configured by writing 0x1 to the Fault n Halt mode bits in the Recoverable Fault n Configuration register (FCTRLn.HALT). When enabled, the timer/counter is halted and the cycle is extended as long as the corresponding fault is present.

The next figure ('Waveform Generation with Halt and Restart Actions') shows an example where both restart action and hardware halt action are enabled for Fault A. The compare channel 0 output is clamped to inactive level as long as the timer/counter is halted. The timer/counter resumes the counting operation as soon as the fault condition is no longer present. As the restart action is enabled in this example, the timer/counter is restarted after the fault condition is no longer present.

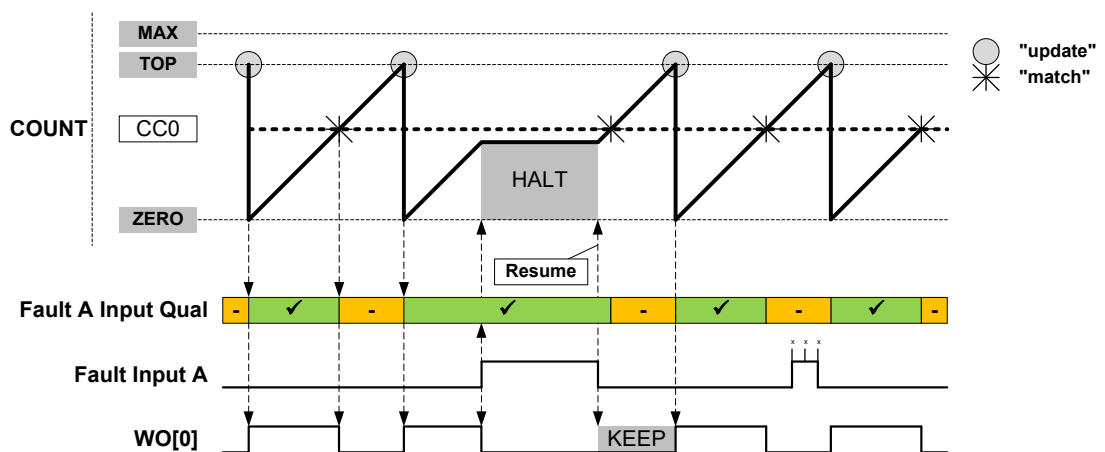
The figure after that ('Waveform Generation with Fault Qualification, Halt, and Restart Actions') shows a similar example, but with additionally enabled fault qualification. Here, counting is resumed after the fault condition is no longer present.

Note that in RAMP2 and RAMP2A operations, when a new timer/counter cycle starts, the cycle index will automatically change.

**Figure 41-30. Waveform Generation with Halt and Restart Actions**



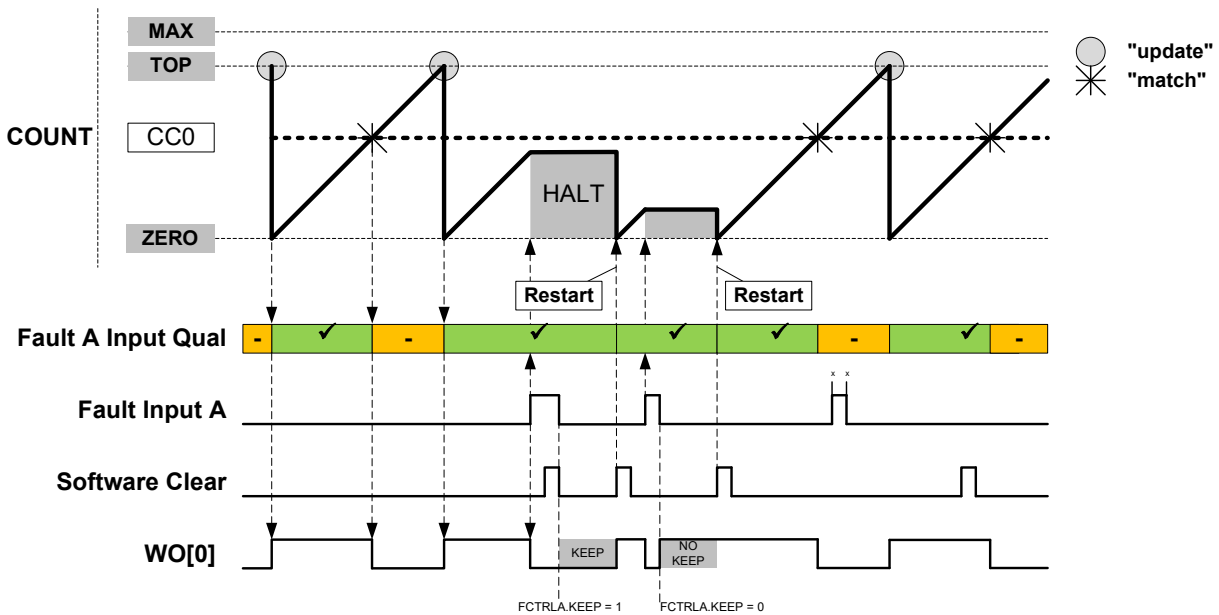
**Figure 41-31. Waveform Generation with Fault Qualification, Halt, and Restart Actions**



**Software Halt Action** This is configured by writing 0x2 to the Fault n Halt mode bits in the Recoverable Fault n configuration register (FCTRLn.HALT). Software halt action is similar to hardware halt action, but in order to restart the timer/counter, the corresponding fault condition must not be present anymore, and the corresponding FAULT n bit in the STATUS register must be cleared by software.



Figure 41-32. Waveform Generation with Software Halt, Fault Qualification, Keep and Restart Actions



#### 41.6.3.6 Non-Recoverable Faults

The non-recoverable fault action will force all the compare outputs to a pre-defined level programmed into the Driver Control register (DRVCTRL.NRE and DRVCTRL.NRV). The non-recoverable fault input (EV0 and EV1) actions are enabled in Event Control register (EVCTRL.EVACT0 and EVCTRL.EVACT1).

To avoid false fault detection on external events (e.g. a glitch on an I/O port) a digital filter can be enabled using Non-Recoverable Fault Input x Filter Value bits in the Driver Control register (DRVCTRL.FILTERVALn). Therefore, the event detection is synchronous, and event action is delayed by the selected digital filter value clock cycles.

When the Fault Detection on Debug Break Detection bit in Debug Control register (DGBCTRL.FDDBD) is written to '1', a non-recoverable Debug Faults State and an interrupt (DFS) is generated when the system goes in debug operation.

In RAMP2, RAMP2A, or DSBOTH operation, when the Lock Update bit in the Control B register is set by writing CTRLBSET.LUPD=1 and the ramp index or counter direction changes, a non-recoverable Update Fault State and the respective interrupt (UFS) are generated.

#### 41.6.3.7 Time-Stamp Capture

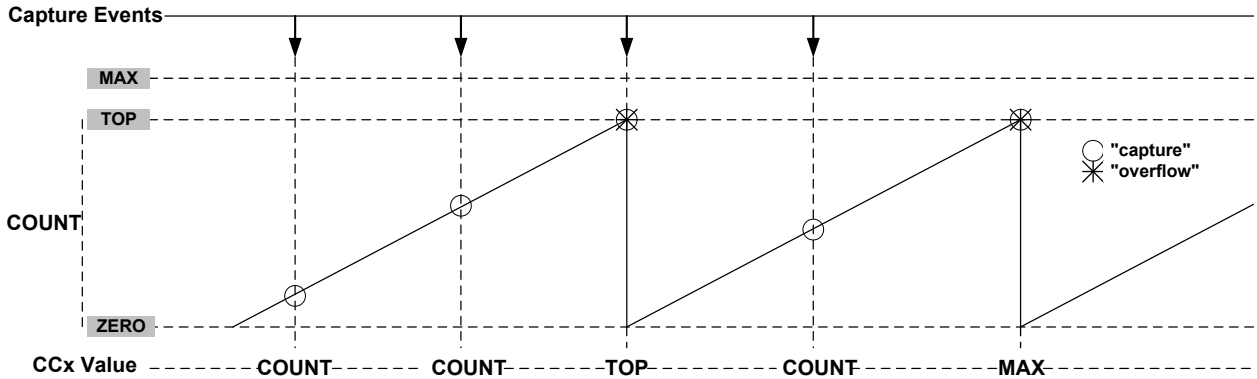
This feature is enabled when the Capture Time Stamp (STAMP) Event Action in Event Control register (EVCTRL.EVACT) is selected. The counter TOP value must be smaller than MAX.

When a capture event is detected, the COUNT value is copied into the corresponding Channel x Compare/Capture Value (CCx) register. In case of an overflow, the MAX value is copied into the corresponding CCx register.

When a valid captured value is present in the capture channel register, the corresponding Capture Channel x Interrupt Flag (INTFLAG.MCx) is set.

The timer/counter can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Channel interrupt flag (INTFLAG.MCx) is still set, the new time-stamp will not be stored and INTFLAG.ERR will be set.

Figure 41-33. Time-Stamp



#### 41.6.3.8 Waveform Extension

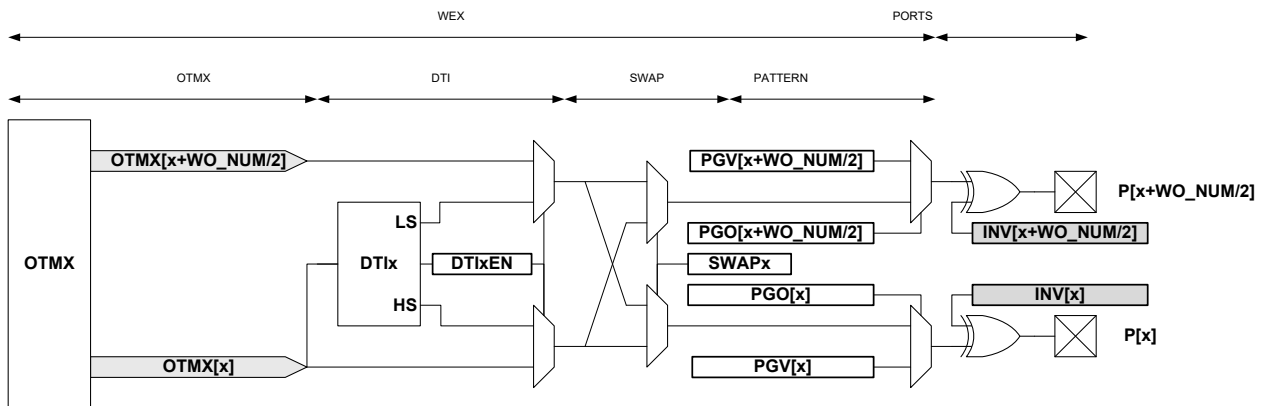
Figure 41-34 shows a schematic diagram of actions of the four optional units that follow the recoverable fault stage on a port pin pair: Output Matrix (OTMX), Dead-Time Insertion (DTI), SWAP and Pattern Generation. The DTI and SWAP units can be seen as a four port pair slices:

- Slice 0 DTI0 / SWAP0 acting on port pins (WO[0], WO[WO\_NUM/2 +0])
- Slice 1 DTI1 / SWAP1 acting on port pins (WO[1], WO[WO\_NUM/2 +1])

And generally:

- Slice n DTIx / SWAPx acting on port pins (WO[x], WO[WO\_NUM/2 +x])

Figure 41-34. Waveform Extension Stage Details



The **output matrix (OTMX)** unit distributes compare channels, according to the selectable configurations in the following table.

Table 41-6. Output Matrix Channel Pin Routing Configuration

Value	OTMX[7]	OTMX[6]	OTMX[5]	OTMX[4]	OTMX[3]	OTMX[2]	OTMX[1]	OTMX[0]
0x0	CC1	CC0	CC5	CC4	CC3	CC2	CC1	CC0
0x1	CC1	CC0	CC2	CC1	CC0	CC2	CC1	CC0
0x2	CC0	CC0	CC0	CC0	CC0	CC0	CC0	CC0
0x3	CC1	CC1	CC1	CC1	CC1	CC1	CC1	CC0

- Configuration 0x0 is the default configuration. The channel location is the default one and channels are distributed on outputs modulo the number of channels. Channel 0 is routed to the Output matrix output OTMX[0], and Channel 1 to OTMX[1]. If there are more outputs than channels, then channel 0 is duplicated to the Output matrix output OTMX[CC\_NUM], channel 1 to OTMX[CC\_NUM+1] and so on.

- Configuration 0x1 distributes the channels on output modulo half the number of channels. This assigns twice the number of output locations to the lower channels than the default configuration. This can be used, for example, to control the four transistors of a full bridge using only two compare channels. Using pattern generation, some of these four outputs can be overwritten by a constant level, enabling flexible drive of a full bridge in all quadrant configurations.
- Configuration 0x2 distributes compare channel 0 (CC0) to all port pins. With pattern generation, this configuration can control a stepper motor.
- Configuration 0x3 distributes the compare channel CC0 to the first output, and the channel CC1 to all other outputs. Together with pattern generation and the fault extension, this configuration can control up to seven LED strings, with a boost stage.

The table below is an example showing four compare channels on four outputs.

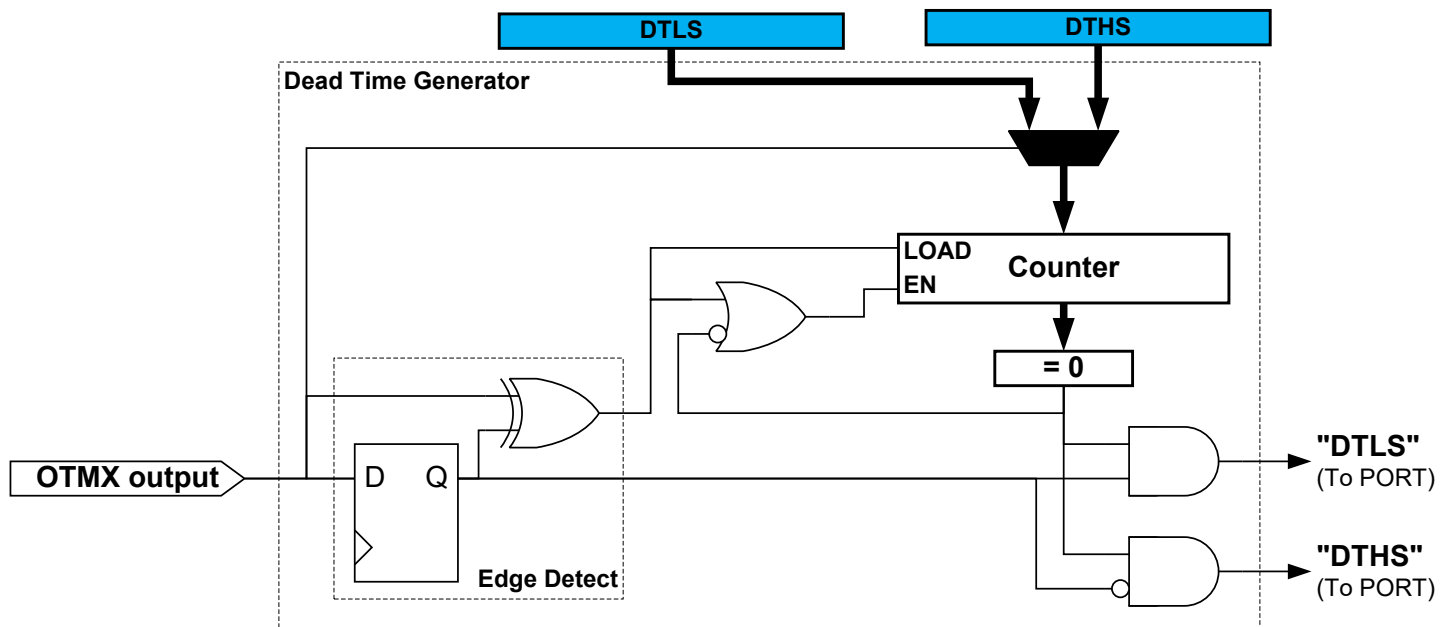
**Table 41-7.** Four Compare Channels on Four Outputs

Value	OTMX[3]	OTMX[2]	OTMX[1]	OTMX[0]
0x0	CC3	CC2	CC1	CC0
0x1	CC1	CC0	CC1	CC0
0x2	CC0	CC0	CC0	CC0
0x3	CC1	CC1	CC1	CC0

The **dead-time insertion (DTI)** unit generates OFF time with the non-inverted low side (LS) and inverted high side (HS) of the wave generator output forced at low level. This OFF time is called dead time. Dead-time insertion ensures that the LS and HS will never switch simultaneously.

The DTI stage consists of four equal dead-time insertion generators; one for each of the first four compare channels. Figure 41-35 shows the block diagram of one DTI generator. The four channels have a common register which controls the dead time, which is independent of high side and low side setting.

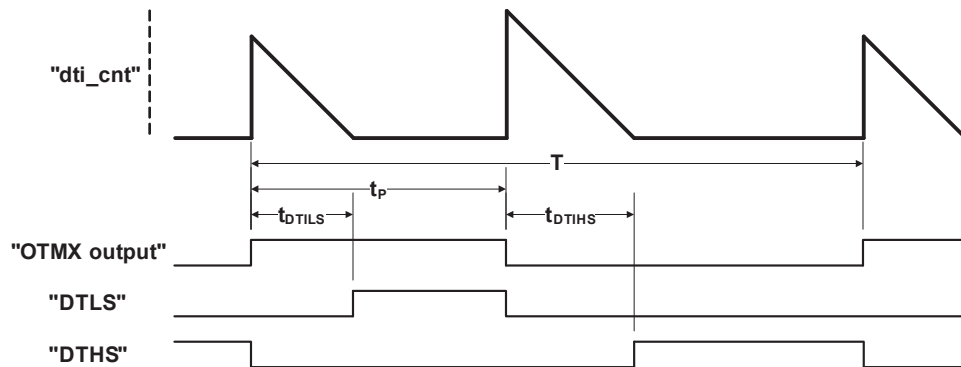
**Figure 41-35.** Dead-Time Generator Block Diagram



As shown in Figure 41-36, the 8-bit dead-time counter is decremented by one for each peripheral clock cycle until it reaches zero. A non-zero counter value will force both the low side and high side

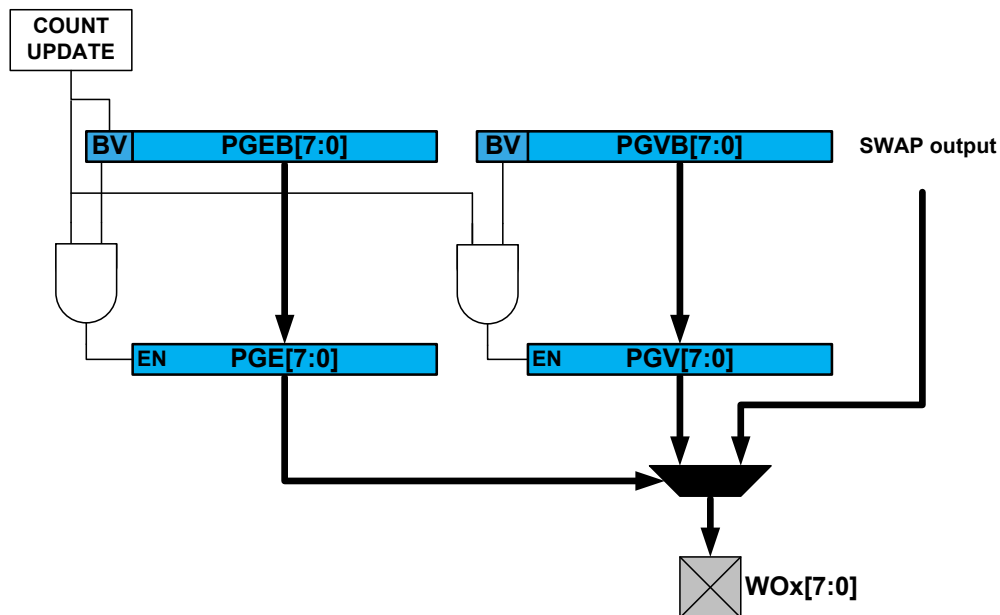
outputs into their OFF state. When the output matrix (OTMX) output changes, the dead-time counter is reloaded according to the edge of the input. When the output changes from low to high (positive edge) it initiates a counter reload of the DTLS register. When the output changes from high to low (negative edge) it reloads the DTHS register.

**Figure 41-36.** Dead-Time Generator Timing Diagram



The **pattern generator unit** produces a synchronized bit pattern across the port pins it is connected to. The pattern generation features are primarily intended for handling the commutation sequence in brushless DC motors (BLDC), stepper motors, and full bridge control. See also [Figure 41-37](#).

**Figure 41-37.** Pattern Generator Block Diagram



As with other double-buffered timer/counter registers, the register update is synchronized to the UPDATE condition set by the timer/counter waveform generation operation. If synchronization is not required by the application, the software can simply access directly the PATT.PGE, PATT.PGV bits registers.

#### 41.6.4 Host/Client Operation

Two TCC instances sharing the same GCLK\_TCC clock, can be linked to provide more synchronized CC channels. The operation is enabled by setting the Host Synchronization bit in Control A register

(CTRLA.MSYNC) in the Client instance. When the bit is set, the Client TCC instance will synchronize the CC channels to the Host counter.

### Related Links

[41.8.1. CTRLA](#)

## 41.6.5 DMA, Interrupts, and Events

**Table 41-8.** Module Requests for TCC

Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
Overflow / Underflow	Yes	Yes		Yes <sup>(1)</sup>	On DMA acknowledge
Channel Compare Match or Capture	Yes	Yes	Yes <sup>(2)</sup>	Yes <sup>(3)</sup>	For circular buffering: on DMA acknowledge For capture channel: when CCx register is read
Retrigger	Yes	Yes			
Count	Yes	Yes			
Capture Overflow Error	Yes				
Debug Fault State	Yes				
Recoverable Faults	Yes				
Non-Recoverable Faults	Yes				
TCCx Event 0 input			Yes <sup>(4)</sup>		
TCCx Event 1 input			Yes <sup>(5)</sup>		

Notes:

1. DMA request set on Overflow, Underflow or Re-trigger conditions.
2. Can perform capture or generate recoverable fault on an event input.
3. In Capture or Circular modes.
4. On event input, either action can be executed:
  - re-trigger counter
  - control counter direction
  - stop the counter
  - decrement the counter
  - perform period and pulse width capture
  - generate non-recoverable fault
5. On event input, either action can be executed:
  - re-trigger counter
  - increment or decrement counter depending on direction
  - start the counter
  - increment or decrement counter based on direction
  - increment counter regardless of direction
  - generate non-recoverable fault

### 41.6.5.1 DMA Operation

The TCC can generate the following DMA requests:

**Counter overflow (OVF)** If the One-shot Trigger mode in the control A register (CTRLA.DMAOS) is written to '0', the TCC generates a DMA request on each cycle when an update condition (Overflow, Underflow or Re-trigger) is detected. When an update condition (Overflow, Underflow or Re-trigger) is detected while CTRLA.DMAOS=1, the TCC generates a DMA trigger on the cycle following the DMA One-Shot Command written to the Control B register (CTRLBSET.CMD=DMAOS).

In both cases, the request is cleared by hardware on DMA acknowledge.

**Channel Match (MCx)** A DMA request is set only on a compare match if CTRLA.DMAOS=0. The request is cleared by hardware on DMA acknowledge. When CTRLA.DMAOS=1, the DMA requests are not generated.

**Channel Capture (MCx)** For a capture channel, the request is set when valid data is present in the CCx register, and cleared once the CCx register is read. In this operation mode, the CTRLA.DMAOS bit value is ignored.

### DMA Operation with Circular Buffer

When circular buffer operation is enabled, the Buffer registers must be written in a correct order and synchronized to the update times of the timer. The DMA triggers of the TCC provide a way to ensure a safe and correct update of circular buffers.

**Note:** Circular buffer are intended to be used with RAMP2, RAMP2A and DSBOTH operation only.

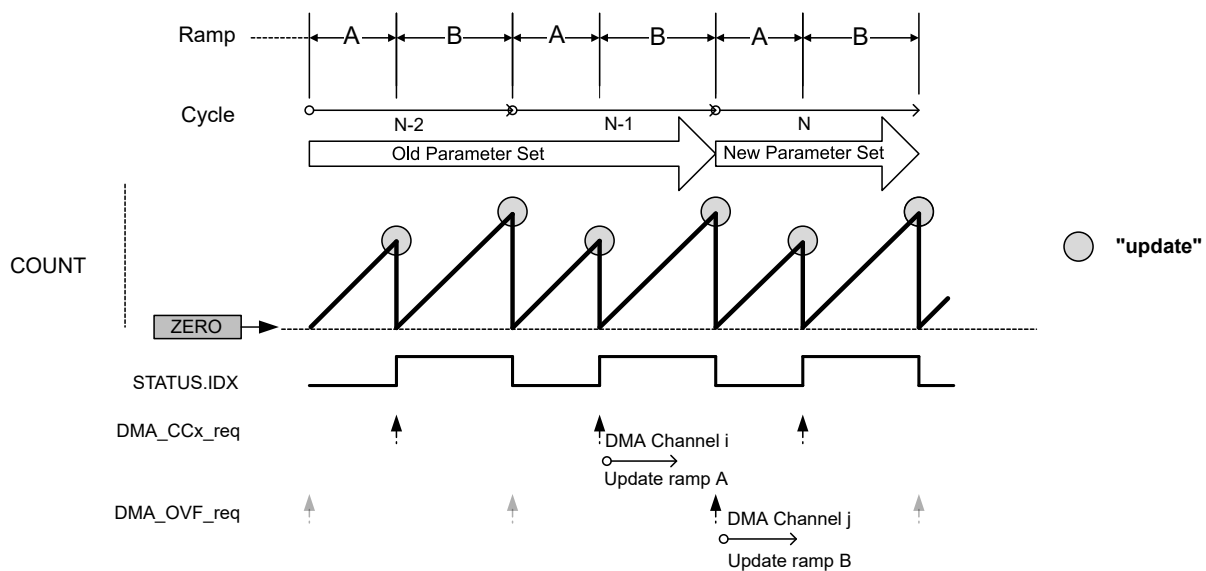
#### DMA Operation with Circular Buffer in RAMP2 and RAMP2A Mode

When a CCx channel is selected as a circular buffer, the related DMA request is not set on a compare match detection, but on start of ramp B.

If at least one circular buffer is enabled, the DMA overflow request is conditioned to the start of ramp A with an effective DMA transfer on previous ramp B (DMA acknowledge).

The update of all circular buffer values for ramp A can be done through a DMA channel triggered on a MC trigger. The update of all circular buffer values for ramp B, can be done through a second DMA channel triggered by the overflow DMA request.

**Figure 41-38.** DMA Triggers in RAMP and RAMP2 Operation Mode and Circular Buffer Enabled



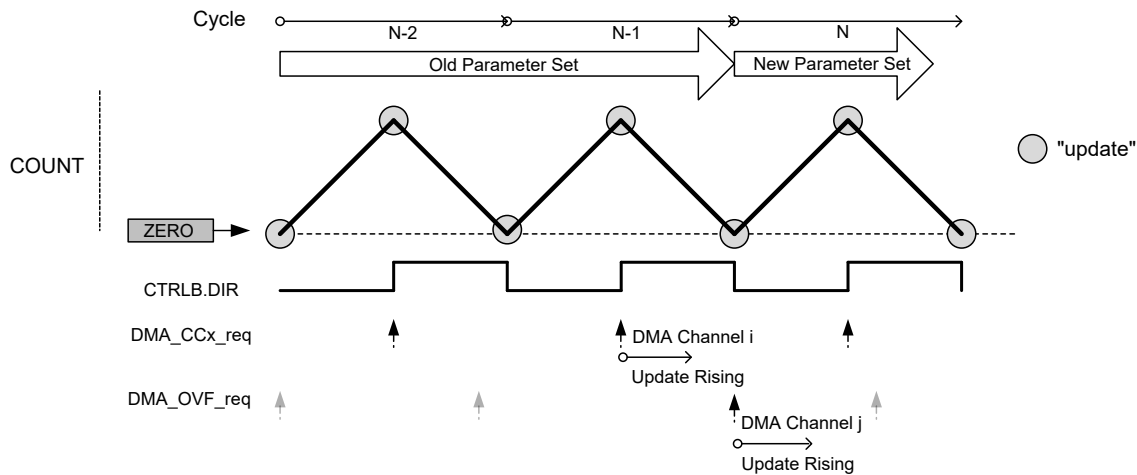
#### DMA Operation with Circular Buffer in DSBOTH Mode

When a CC channel is selected as a circular buffer, the related DMA request is not set on a compare match detection, but on start of down-counting phase.

If at least one circular buffer is enabled, the DMA overflow request is conditioned to the start of up-counting phase with an effective DMA transfer on previous down-counting phase (DMA acknowledge).

When up-counting, all circular buffer values can be updated through a DMA channel triggered by MC trigger. When down-counting, all circular buffer values can be updated through a second DMA channel, triggered by the OVF DMA request.

**Figure 41-39.** DMA Triggers in DSBOOTH Operation Mode and Circular Buffer Enabled



#### 41.6.5.2 Interrupts

The TCC has the following interrupt sources:

- Overflow/Underflow (OVF)
- Retrigger (TRG)
- Count (CNT) – Refer also to the description of EVCTRL.CNTSEL
- Capture Overflow Error (ERR)
- Non-Recoverable Update Fault (UFS)
- Debug Fault State (DFS)
- Recoverable Faults (FAULTn)
- Non-recoverable Faults (FAULTx)
- Compare Match or Capture Channels (MCx)

These interrupts are asynchronous wake-up sources.

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. 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 TCC is reset. See *INTFLAG* from Related Links for details on how to clear Interrupt flags. The TCC has one common interrupt request line for all the interrupt sources. The user must read the INTFLAG register to determine which Interrupt condition is present.

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 \(NVIC\)](#)

[41.8.12. INTFLAG](#)

#### 41.6.5.3 Events

The TCC can generate the following output events:

- Overflow/Underflow (OVF)
- Trigger (TRG)
- Counter (CNT) (For further details, refer to the EVCTRL.CNTSEL description.)
- Compare Match or Capture on compare/capture channels: MCx

Writing a '1' ('0') to an Event Output bit in the Event Control Register (EVCTRL.xxEO) enables (disables) the corresponding output event. See *Event System (EVSYS)* from Related Links.

The TCC can take the following actions on a channel input event (MCx):

- Capture event
- Generate a recoverable or non-recoverable fault

The TCC can take the following actions on counter Event 1 (TCCx EV1):

- Counter re-trigger
- Counter direction control
- Stop the counter
- Decrement the counter on event
- Period and pulse width capture
- Non-recoverable fault

The TCC can take the following actions on counter Event 0 (TCCx EV0):

- Counter re-trigger
- Count on event (increment or decrement, depending on counter direction)
- Counter start – Start counting on the event rising edge. Further events will not restart the counter; the counter will keep counting using prescaled GCLK\_TCCx, until it reaches TOP or ZERO, depending on the direction.
- Counter increment on event. This will increment the counter, irrespective of the counter direction.
- Count during active state of an asynchronous event (increment or decrement, depending on counter direction). In this case, the counter will be incremented or decremented on each cycle of the prescaled clock, as long as the event is active.
- Non-recoverable fault

The counter Event Actions are available in the Event Control registers (EVCTRL.EVACT0 and EVCTRL.EVACT1). See *EVCTRL* from Related Links.

Writing a '1' ('0') to an Event Input bit in the Event Control register (EVCTRL.MCEIx or EVCTRL.TCEIx) enables (disables) the corresponding action on input event.

**Note:** When several events are connected to the TCC, the enabled action will apply for each of the incoming events. See *Event System (EVSYS)* from Related Links for details on how to configure the Event System.



## Related Links

[28. Event System \(EVSYS\)](#)

[41.8.9. EVCTRL](#)

### 41.6.6 Sleep Mode Operation

The TCC can be configured to operate in any Sleep mode. To be able to run in standby the RUNSTDBY bit in the Control A register (CTRLA.RUNSTDBY) must be '1'. The MODULE can in any Sleep mode wake-up the device using interrupts or perform actions through the Event System.

### 41.6.7 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 and Enable bits in Control A register (CTRLA.SWRST and CTRLA.ENABLE)

The following registers are synchronized when written:

- Control B Clear and Control B Set registers (CTRLBCLR and CTRLBSET)
- Status register (STATUS)
- Pattern and Pattern Buffer registers (PATT and PATTBUF)
- Waveform register (WAVE)
- Count Value register (COUNT)
- Period Value and Period Buffer Value registers (PER and PERBUF)
- Compare/Capture Channel x and Channel x Compare/Capture Buffer Value registers (CCx and CCBUFx)

The following registers are synchronized when read:

- Control B Clear and Control B Set registers (CTRLBCLR and CTRLBSET)
- Count Value register (COUNT): synchronization is done on demand through READSYNC command (CTRLBSET.CMD)
- Pattern and Pattern Buffer registers (PATT and PATTBUF)
- Waveform register (WAVE)
- Period Value and Period Buffer Value registers (PER and PERBUF)
- Compare/Capture Channel x and Channel x Compare/Capture Buffer Value registers (CCx and CCBUFx)

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.

## 41.7 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
0x00	CTRLA	7:0	RESOLUTION[1:0]							ENABLE	SWRST
		15:8	MSYNC	ALOCK	PRESCYNC[1:0]		RUNSTDBY	PRESCALER[2:0]			
		23:16	DMAOS								
		31:24					CPTEN3	CPTEN2	CPTEN1	CPTEN0	
0x04	CTRLBCLR	7:0	CMD[2:0]		IDXCMD[1:0]		ONESHOT	LUPD	DIR		
0x05	CTRLBSET	7:0	CMD[2:0]		IDXCMD[1:0]		ONESHOT	LUPD	DIR		
0x06	Reserved										
...											
0x07											
0x08	SYNCBUSY	7:0	PER	WAVE	PATT	COUNT	STATUS	CTRLB	ENABLE	SWRST	
		15:8			CC5	CC4	CC3	CC2	CC1	CC0	
		23:16									
		31:24									
0x0C	FCTRLA	7:0	RESTART	BLANK[1:0]		QUAL	KEEP		SRC[1:0]		
		15:8	BLANKPRESC	CAPTURE[2:0]		CHSEL[1:0]			HALT[1:0]		
		23:16	BLANKVAL[7:0]								
		31:24								FILTERVAL[3:0]	
0x10	FCTRLB	7:0	RESTART	BLANK[1:0]		QUAL	KEEP		SRC[1:0]		
		15:8	BLANKPRESC	CAPTURE[2:0]		CHSEL[1:0]			HALT[1:0]		
		23:16	BLANKVAL[7:0]								
		31:24								FILTERVAL[3:0]	
0x14	WEXCTRL	7:0							OTMX[1:0]		
		15:8					DTIEN3	DTIEN2	DTIEN1	DTIEN0	
		23:16	DTLS[7:0]								
		31:24	DTHS[7:0]								
0x18	DRVCTRL	7:0	NRE7	NRE6	NRE5	NRE4	NRE3	NRE2	NRE1	NRE0	
		15:8	NRV7	NRV6	NRV5	NRV4	NRV3	NRV2	NRV1	NRV0	
		23:16	INVEN7	INVEN6	INVEN5	INVEN4	INVEN3	INVEN2	INVEN1	INVEN0	
		31:24	FILTERVAL1[3:0]			FILTERVAL0[3:0]					
0x1C	Reserved										
...											
0x1D											
0x1E	DBGCTRL	7:0					FDDBD			DBGRUN	
0x1F	Reserved										
0x20	EVCTRL	7:0	CNTSEL[1:0]		EVACT1[2:0]		EVACT0[2:0]				
		15:8	TCEI1	TCEI0	TCINV1	TCINV0		CNTEO	TRGEO	OVFEO	
		23:16					MCEI3	MCEI2	MCEI1	MCEI0	
		31:24					MCEO3	MCEO2	MCEO1	MCEO0	
0x24	INTENCLR	7:0					ERR	CNT	TRG	OVF	
		15:8	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS			
		23:16					MCx3	MCx2	MCx1	MCx0	
		31:24									
0x28	INTENSET	7:0					ERR	CNT	TRG	OVF	
		15:8	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS			
		23:16					MC3	MC2	MC1	MC0	
		31:24									
0x2C	INTFLAG	7:0					ERR	CNT	TRG	OVF	
		15:8	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS			
		23:16					MC3	MC2	MC1	MC0	
		31:24									
0x30	STATUS	7:0	PERBUFV		PATTBUFV	SLAVE	DFS	UFS	IDX	STOP	
		15:8	FAULT1	FAULT0	FAULTB	FAULTA	FAULT1IN	FAULT0IN	FAULTBIN	FAULTAIN	
		23:16					CCBUFV3	CCBUFV2	CCBUFV1	CCBUFV0	
		31:24					CMP3	CMP2	CMP1	CMP0	
0x34	COUNT	7:0	COUNT[7:0]								
		15:8	COUNT[15:8]								
		23:16	COUNT[23:16]								
		31:24									

.....continued										
Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x38	PATT	7:0	PGE7	PGE6	PGE5	PGE4	PGE3	PGE2	PGE1	PGE0
		15:8	PGV7	PGV6	PGV5	PGV4	PGV3	PGV2	PGV1	PGV0
0x3A ... 0x3B	Reserved									
0x3C	WAVE	7:0	CIPEREN		RAMP[1:0]			WAVEGEN[2:0]		
		15:8					CICCEN3	CICCEN2	CICCEN1	CICCEN0
		23:16			POL5	POL4	POL3	POL2	POL1	POL0
		31:24					SWAP3	SWAP2	SWAP1	SWAP0
0x40	PER	7:0	PER[1:0]		DITHER[5:0]					
		15:8	PER[9:2]							
		23:16	PER[17:10]							
		31:24								
0x44	CC0	7:0	CC[1:0]		DITHER[5:0]					
		15:8	CC[9:2]							
		23:16	CC[17:10]							
		31:24								
0x48	CC1	7:0	CC[1:0]		DITHER[5:0]					
		15:8	CC[9:2]							
		23:16	CC[17:10]							
		31:24								
0x4C	CC2	7:0	CC[1:0]		DITHER[5:0]					
		15:8	CC[9:2]							
		23:16	CC[17:10]							
		31:24								
0x50	CC3	7:0	CC[1:0]		DITHER[5:0]					
		15:8	CC[9:2]							
		23:16	CC[17:10]							
		31:24								
0x54	CC4	7:0	CC[1:0]		DITHER[5:0]					
		15:8	CC[9:2]							
		23:16	CC[17:10]							
		31:24								
0x58	CC5	7:0	CC[1:0]		DITHER[5:0]					
		15:8	CC[9:2]							
		23:16	CC[17:10]							
		31:24								
0x5C ... 0x63	Reserved									
0x64	PATTBUF	7:0	PGEB7	PGEB6	PGEB5	PGEB4	PGEB3	PGEB2	PGEB1	PGEB0
		15:8	PGVB7	PGVB6	PGVB5	PGVB4	PGVB3	PGVB2	PGVB1	PGVB0
0x66 ... 0x6B	Reserved									
0x6C	PERBUF	7:0	PERBUF[1:0]		DITHERBUF[5:0]					
		15:8	PERBUF[9:2]							
		23:16	PERBUF[17:10]							
		31:24								
0x70	CCBUF0	7:0	CCBUF[1:0]		DITHERBUF[5:0]					
		15:8	CCBUF[9:2]							
		23:16	CCBUF[17:10]							
		31:24								
0x74	CCBUF1	7:0	CCBUF[1:0]		DITHERBUF[5:0]					
		15:8	CCBUF[9:2]							
		23:16	CCBUF[17:10]							
		31:24								
0x78	CCBUF2	7:0	CCBUF[1:0]		DITHERBUF[5:0]					
		15:8	CCBUF[9:2]							
		23:16	CCBUF[17:10]							
		31:24								

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0	
0x7C	CCBUF3	7:0	CCBUF[1:0]			DITHERBUF[5:0]					
		15:8	CCBUF[9:2]								
		23:16	CCBUF[17:10]								
		31:24									
0x80	CCBUF4	7:0	CCBUF[1:0]			DITHERBUF[5:0]					
		15:8	CCBUF[9:2]								
		23:16	CCBUF[17:10]								
		31:24									
0x84	CCBUF5	7:0	CCBUF[1:0]			DITHERBUF[5:0]					
		15:8	CCBUF[9:2]								
		23:16	CCBUF[17:10]								
		31:24									

## 41.8 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.

### 41.8.1 Control A

**Name:** CTRLA  
**Offset:** 0x00  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected, Write-Synchronized (ENABLE, SWRST)

Bit	31	30	29	28	27	26	25	24
					CPTEN3	CPTEN2	CPTEN1	CPTEN0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
	DMAOS							
Access	R/W							
Reset	0							
Bit	15	14	13	12	11	10	9	8
	MSYNC	ALOCK	PRESCYNC[1:0]		RUNSTDBY	PRESCALER[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
	RESOLUTION[1:0]					ENABLE		SWRST
Access	R/W		R/W			R/W		R/W
Reset	0		0			0		0

#### Bits 24, 25, 26, 27 - CPTENx Capture Channel x Enable

These bits are used to select the capture or compare operation on channel x.  
 Writing a '1' to CPTENx enables capture on channel x.  
 Writing a '0' to CPTENx disables capture on channel x.

#### Bit 23 - DMAOS DMA One-Shot Trigger Mode

This bit enables the DMA One-shot Trigger Mode.  
 Writing a '1' to this bit will generate a DMA trigger on TCC cycle following a TCC\_CTRLBSET\_CMD\_DMAOS command.  
 Writing a '0' to this bit will generate DMA triggers on each TCC cycle.

#### Bit 15 - MSYNC Host Synchronization (only for TCC Client instance)

This bit must be set if the TCC counting operation must be synchronized on its Host TCC.  
 This bit is not synchronized.

Value	Description
0	The TCC controls its own counter.
1	The counter is controlled by its Host TCC.

#### Bit 14 - ALOCK Auto Lock

This bit is not synchronized.

Value	Description
0	The Lock Update bit in the Control B register (CTRLB.LUPD) is not affected by overflow/underflow, and re-trigger events
1	CTRLB.LUPD is set to '1' on each overflow/underflow or re-trigger event.

**Bits 13:12 – PRESCYNC[1:0]** Prescaler and Counter Synchronization

These bits select if on re-trigger event, the Counter is cleared or reloaded on either the next GCLK\_TCCx clock, or on the next prescaled GCLK\_TCCx clock. It is also possible to reset the prescaler on re-trigger event.

These bits are not synchronized.

Value	Name	Description	
		Counter Reloaded	Prescaler
0x0	GCLK	Reload or reset Counter on next GCLK	-
0x1	PRESC	Reload or reset Counter on next prescaler clock	-
0x2	RESYNC	Reload or reset Counter on next GCLK	Reset prescaler counter
0x3	Reserved		

**Bit 11 – RUNSTDBY** Run in Standby

This bit is used to keep the TCC running in Standby mode.

This bit is not synchronized.

Value	Description
0	The TCC is halted in standby.
1	The TCC continues to run in standby.

**Bits 10:8 – PRESCALER[2:0]** Prescaler

These bits select the Counter prescaler factor.

These bits are not synchronized.

Value	Name	Description
0x0	DIV1	Prescaler: GCLK_TCCx
0x1	DIV2	Prescaler: GCLK_TCCx/2
0x2	DIV4	Prescaler: GCLK_TCCx/4
0x3	DIV8	Prescaler: GCLK_TCCx/8
0x4	DIV16	Prescaler: GCLK_TCCx/16
0x5	DIV64	Prescaler: GCLK_TCCx/64
0x6	DIV256	Prescaler: GCLK_TCCx/256
0x7	DIV1024	Prescaler: GCLK_TCCx/1024

**Bits 6:5 – RESOLUTION[1:0]** Dithering Resolution

These bits increase the TCC resolution by enabling the dithering options.

These bits are not synchronized.

**Table 41-9.** Dithering

Value	Name	Description
0x0	NONE	The dithering is disabled.
0x1	DITH4	Dithering is done every 16 PWM frames. PER[3:0] and CCx[3:0] contain dithering pattern selection.
0x2	DITH5	Dithering is done every 32 PWM frames. PER[4:0] and CCx[4:0] contain dithering pattern selection.
0x3	DITH6	Dithering is done every 64 PWM frames. PER[5:0] and CCx[5:0] contain dithering pattern selection.

**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 SYNCBUSY 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 TCC (except DBGCTRL) to their initial state, and the TCC will be disabled.

Writing a '1' to CTRLA.SWRST will always take precedence; 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.

Value	Description
0	There is no Reset operation ongoing.
1	The Reset operation is ongoing.

## 41.8.2 Control B Clear

**Name:** CTRLBCLR  
**Offset:** 0x04  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

This register allows the user to change this register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set (CTRLBSET) register.

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]			IDXCMD[1:0]		ONESHOT	LUPD	DIR
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 7:5 – CMD[2:0] TCC Command

Writing zero to this bit group has no effect.

Writing a '1' to any of these bits will clear the pending command.

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Clear start, restart or retrigger
0x2	STOP	Force stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force COUNT read synchronization
0x5	DMAOS	One-shot DMA trigger

### Bits 4:3 – IDXCMD[1:0] Ramp Index Command

These bits can be used to force cycle A and cycle B changes in RAMP2 and RAMP2A operation. On timer/counter update condition, the command is executed, the IDX flag in STATUS register is updated and the IDXCMD command is cleared.

Writing zero to these bits has no effect.

Writing a '1' to any of these bits will clear the pending command.

Value	Name	Description
0x0	DISABLE	DISABLE Command disabled: IDX toggles between cycles A and B
0x1	SET	Set IDX: cycle B will be forced in the next cycle
0x2	CLEAR	Clear IDX: cycle A will be forced in next cycle
0x3	HOLD	Hold IDX: the next cycle will be the same as the current cycle.

### Bit 2 – ONESHOT One-Shot

This bit controls one-shot operation of the TCC. When one-shot operation is enabled, the TCC will stop counting on the next overflow/underflow condition or on a stop command.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will disable the one-shot operation.

Value	Description
0	The TCC will update the counter value on overflow/underflow condition and continue operation.
1	The TCC will stop counting on the next underflow/overflow condition.

### Bit 1 – LUPD Lock Update

This bit controls the update operation of the TCC buffered registers.

When CTRLB.LUPD is cleared, the hardware UPDATE registers with value from their buffered registers is enabled.

This bit has no effect when input capture operation is enabled.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will enable the registers updates on hardware UPDATE condition.



Value	Description
0	The CCBx, PERB, PGVB, and PGEB buffer registers values <i>are</i> copied into the corresponding CCx, PER, PGV, and PGE registers on hardware update condition.
1	The CCBx, PERB, PGVB, and PGEB buffer registers values are <i>not</i> copied into the corresponding CCx, PER, PGV, and PGE registers on hardware update condition.

**Bit 0 – DIR** Counter Direction

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

### 41.8.3 Control B Set

**Name:** CTRLBSET  
**Offset:** 0x05  
**Reset:** 0x00  
**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

This register allows the user to change this register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set (CTRLBCLR) register.

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]			IDXCMD[1:0]		ONESHOT	LUPD	DIR
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 7:5 – CMD[2:0] TCC Command

These bits can be used for software control of re-triggering and stop commands of the TCC. When a command has been executed, the CMD bit field will be read back as zero. The commands are executed on the next prescaled GCLK\_TCCx clock cycle.

Writing zero to this bit group has no effect

Writing a valid value to this bit group, as shown in the following table, will set the associated command.

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force start, restart or retrigger
0x2	STOP	Force stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force a read synchronization of COUNT
0x5	DMAOS	One-shot DMA trigger

#### Bits 4:3 – IDXCMD[1:0] Ramp Index Command

These bits can be used to force cycle A and cycle B changes in RAMP2 and RAMP2A operation. On timer/counter update condition, the command is executed, the IDX flag in STATUS register is updated and the IDXCMD command is cleared.

Writing a zero to these bits has no effect.

Writing a valid value to these bits will set a command.

Value	Name	Description
0x0	DISABLE	Command disabled: IDX toggles between cycles A and B
0x1	SET	Set IDX: cycle B will be forced in the next cycle
0x2	CLEAR	Clear IDX: cycle A will be forced in next cycle
0x3	HOLD	Hold IDX: the next cycle will be the same as the current cycle.

#### Bit 2 – ONESHOT One-Shot

This bit controls one-shot operation of the TCC. When in one-shot operation, the TCC will stop counting on the next overflow/underflow condition or a stop command.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will enable the one-shot operation.

Value	Description
0	The TCC will count continuously.
1	The TCC will stop counting on the next underflow/overflow condition.

#### Bit 1 – LUPD Lock Update

This bit controls the update operation of the TCC buffered registers.

When CTRLB.LUPD is set, the hardware UPDATE registers with value from their buffered registers is disabled. Disabling the update ensures that all buffer registers are valid before an hardware

update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.

This bit has no effect when input capture operation is enabled.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will disable the registers updates on hardware UPDATE condition.

Value	Description
0	The CCBx, PERB, PGVB, and PGEB buffer registers values <i>are</i> copied into the corresponding CCx, PER, PGV, and PGE registers on hardware update condition.
1	The CCBx, PERB, PGVB, and PGEB buffer registers values are <i>not</i> copied into CCx, PER, PGV, and PGE registers on hardware update condition.

#### Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.

Writing a '0' to this bit has no effect

Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

#### 41.8.4 Synchronization Busy

**Name:** SYNCBUSY  
**Offset:** 0x08  
**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			CC5	CC4	CC3	CC2	CC1	CC0
Reset			R	R	R	R	R	R
Reset			0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Access	PER	WAVE	PATT	COUNT	STATUS	CTRLB	ENABLE	SWRST
Reset	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

#### Bits 8, 9, 10, 11, 12, 13 – CC Compare/Capture Channel x Synchronization Busy

This bit is cleared when the synchronization of the Compare/Capture Channel x register between the clock domains is complete.

This bit is set when the synchronization of the Compare/Capture Channel x register between clock domains is started.

The CCx bit is available only for existing Compare/Capture Channels. For details on the CC channels number, refer to each TCC feature list.

This bit is set when the synchronization of the CCx register between clock domains is started.

#### Bit 7 – PER PER Synchronization Busy

This bit is cleared when the synchronization of the PER register between the clock domains is complete.

This bit is set when the synchronization of the PER register between clock domains is started.

#### Bit 6 – WAVE WAVE Synchronization Busy

This bit is cleared when the synchronization of the WAVE register between the clock domains is complete.

This bit is set when the synchronization of the WAVE register between clock domains is started.

#### Bit 5 – PATT PATT Synchronization Busy

This bit is cleared when the synchronization of the PATTERN register between the clock domains is complete.

This bit is set when the synchronization of the PATTERN register between clock domains is started.

**Bit 4 – COUNT** COUNT Synchronization Busy

This bit is cleared when the synchronization of the COUNT register between the clock domains is complete.

This bit is set when the synchronization of the COUNT register between clock domains is started.

**Bit 3 – STATUS** STATUS Synchronization Busy

This bit is cleared when the synchronization of the STATUS register between the clock domains is complete.

This bit is set when the synchronization of the STATUS register between clock domains is started.

**Bit 2 – CTRLB** CTRLB Synchronization Busy

This bit is cleared when the synchronization of the CTRLB register between the clock domains is complete.

This bit is set when the synchronization of the CTRLB register between clock domains is started.

**Bit 1 – ENABLE** ENABLE Synchronization Busy

This bit is cleared when the synchronization of the ENABLE bit between the clock domains is complete.

This bit is set when the synchronization of the ENABLE bit between clock domains is started.

**Bit 0 – SWRST** SWRST Synchronization Busy

This bit is cleared when the synchronization of the SWRST bit between the clock domains is complete.

This bit is set when the synchronization of the SWRST bit between clock domains is started.

**Note:** During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST is cleared by hardware.

### 41.8.5 Fault Control A and B

**Name:** FCTRLn  
**Offset:** 0x0C + n\*0x04 [n=0..1]  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	FILTERVAL[3:0]							
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
	BLANKVAL[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
	BLANKPRESC	CAPTURE[2:0]			CHSEL[1:0]		HALT[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
	RESTART	BLANK[1:0]		QUAL	KEEP		SRC[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

#### Bits 27:24 – FILTERVAL[3:0] Recoverable Fault n Filter Value

These bits define the filter value applied on MCE<sub>x</sub> (x=0,1) event input line. The value must be set to zero when MCE<sub>x</sub> event is used as synchronous event.

#### Bits 23:16 – BLANKVAL[7:0] Recoverable Fault n Blanking Value

These bits determine the duration of the blanking of the fault input source. Activation and edge selection of the blank filtering are done by the BLANK bits (FCTRLn.BLANK). When enabled, the fault input source is internally disabled for BLANKVAL\* prescaled GCLK\_TCC<sub>x</sub> periods after the detection of the waveform edge.

#### Bit 15 – BLANKPRESC Recoverable Fault n Blanking Value Prescaler

This bit enables a factor 64 prescaler factor on used as base frequency of the BLANKVAL value.

Value	Description
0	Blank time is BLANKVAL* prescaled GCLK_TCC <sub>x</sub> .
1	Blank time is BLANKVAL* 64 * prescaled GCLK_TCC <sub>x</sub> .

#### Bits 14:12 – CAPTURE[2:0] Recoverable Fault n Capture Action

These bits select the capture and Fault n interrupt/event conditions.

**Table 41-10.** Fault n Capture Action

Value	Name	Description
0x0	DISABLE	Capture on valid recoverable Fault n is disabled
0x1	CAPT	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0]. INTFLAG.FAULTn flag rises on each new captured value.
0x2	CAPTMIN	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0], if COUNT value is lower than the last stored capture value (CC). INTFLAG.FAULTn flag rises on each local minimum detection.

.....continued

Value	Name	Description
0x3	CAPTMAX	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0], if COUNT value is higher than the last stored capture value (CC). INTFLAG.FAULTn flag rises on each local maximum detection.
0x4	LOCMIN	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0]. INTFLAG.FAULTn flag rises on each local minimum value detection.
0x5	LOCMAX	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0]. INTFLAG.FAULTn flag rises on each local maximum detection.
0x6	DERIVO	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0]. INTFLAG.FAULTn flag rises on each local maximum or minimum detection.
0x7	CAPTMARK	Capture with ramp index as MSB value.

#### Bits 11:10 – CHSEL[1:0] Recoverable Fault n Capture Channel

These bits select the channel for capture operation triggered by recoverable Fault n.

Value	Name	Description
0x0	CC0	Capture value stored into CC0
0x1	CC1	Capture value stored into CC1
0x2	CC2	Capture value stored into CC2
0x3	CC3	Capture value stored into CC3

#### Bits 9:8 – HALT[1:0] Recoverable Fault n Halt Operation

These bits select the halt action for recoverable Fault n.

Value	Name	Description
0x0	DISABLE	Halt action disabled
0x1	HW	Hardware halt action
0x2	SW	Software halt action
0x3	NR	Non-recoverable fault

#### Bit 7 – RESTART Recoverable Fault n Restart

Setting this bit enables restart action for Fault n.

Value	Description
0	Fault n restart action is disabled.
1	Fault n restart action is enabled.

#### Bits 6:5 – BLANK[1:0] Recoverable Fault n Blanking Operation

These bits, select the blanking start point for recoverable Fault n.

Value	Name	Description
0x0	START	Blanking applied from start of the Ramp period
0x1	RISE	Blanking applied from rising edge of the waveform output
0x2	FALL	Blanking applied from falling edge of the waveform output
0x3	BOTH	Blanking applied from each toggle of the waveform output

#### Bit 4 – QUAL Recoverable Fault n Qualification

Setting this bit enables the recoverable Fault n input qualification.

Value	Description
0	The recoverable Fault n input is not disabled on CMPx value condition.
1	The recoverable Fault n input is disabled when output signal is at inactive level (CMPx == 0).

#### Bit 3 – KEEP Recoverable Fault n Keep

Setting this bit enables the Fault n keep action.

Value	Description
0	The Fault n state is released as soon as the recoverable Fault n is released.
1	The Fault n state is released at the end of TCC cycle.

#### Bits 1:0 – SRC[1:0] Recoverable Fault n Source

These bits select the TCC event input for recoverable Fault n.

Event system channel connected to MCE<sub>x</sub> event input, must be configured to route the event asynchronously, when used as a recoverable Fault n input.

Value	Name	Description
0x0	DISABLE	Fault input disabled
0x1	ENABLE	MCE <sub>x</sub> (x=0,1) event input
0x2	INVERT	Inverted MCE <sub>x</sub> (x=0,1) event input
0x3	ALTFAULT	Alternate fault (A or B) state at the end of the previous period.



## 41.8.6 Waveform Extension Control

**Name:** WEXCTRL  
**Offset:** 0x14  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	DTHS[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
	DTLS[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
					DTIEN3	DTIEN2	DTIEN1	DTIEN0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
							OTMX[1:0]	
Access							R/W	R/W
Reset							0	0

### Bits 31:24 – DTHS[7:0] Dead-Time High Side Outputs Value

This register holds the number of GCLK\_TCCx clock cycles for the dead-time high side.

### Bits 23:16 – DTLS[7:0] Dead-time Low Side Outputs Value

This register holds the number of GCLK\_TCCx clock cycles for the dead-time low side.

### Bits 8, 9, 10, 11 – DTIENx Dead-time Insertion Generator x Enable

Setting any of these bits enables the dead-time insertion generator for the corresponding output matrix. This will override the output matrix [x] and [x+WO\_NUM/2], with the low side and high side waveform respectively.

Value	Description
0	No dead-time insertion override.
1	Dead time insertion override on signal outputs[x] and [x+WO_NUM/2], from matrix outputs[x] signal.

### Bits 1:0 – OTMX[1:0] Output Matrix

These bits define the matrix routing of the TCC waveform generation outputs to the port pins, according to [41.6.3.8. Waveform Extension](#).

## 41.8.7 Driver Control

**Name:** DRVCTRL  
**Offset:** 0x18  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	FILTERVAL1[3:0]				FILTERVAL0[3:0]			
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
	INVEN7	INVEN6	INVEN5	INVEN4	INVEN3	INVEN2	INVEN1	INVEN0
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
	NRV7	NRV6	NRV5	NRV4	NRV3	NRV2	NRV1	NRV0
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
	NRE7	NRE6	NRE5	NRE4	NRE3	NRE2	NRE1	NRE0
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 31:28 – FILTERVAL1[3:0] Non-Recoverable Fault Input 1 Filter Value

These bits define the filter value applied on TCE1 event input line. When the TCE1 event input line is configured as a synchronous event, this value must be 0x0.

### Bits 27:24 – FILTERVAL0[3:0] Non-Recoverable Fault Input 0 Filter Value

These bits define the filter value applied on TCE0 event input line. When the TCE0 event input line is configured as a synchronous event, this value must be 0x0.

### Bits 16, 17, 18, 19, 20, 21, 22, 23 – INVENx Waveform Output x Inversion

These bits are used to select inversion on the output of channel x.  
 Writing a '1' to INVENx inverts output from WO[x].  
 Writing a '0' to INVENx disables inversion of output from WO[x].

### Bits 8, 9, 10, 11, 12, 13, 14, 15 – NRVx NRVx Non-Recoverable State x Output Value

These bits define the value of the enabled override outputs, under non-recoverable fault condition.

### Bits 0, 1, 2, 3, 4, 5, 6, 7 – NREx Non-Recoverable State x Output Enable

These bits enable the override of individual outputs by NRVx value, under non-recoverable fault condition.

Value	Description
0	Non-recoverable fault tri-state the output.
1	Non-recoverable faults set the output to NRVx level.

### 41.8.8 Debug control

**Name:** DBGCTRL  
**Offset:** 0x1E  
**Reset:** 0x00  
**Property:** PAC Write-Protection

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

#### Bit 2 – FDDBD Fault Detection on Debug Break Detection

This bit is not affected by software Reset and must not be changed by software while the TCC is enabled.

By default this bit is zero, and the on-chip debug (OCD) fault protection is disabled. When this bit is written to '1', OCD break request from the OCD system will trigger non-recoverable fault. When this bit is set, OCD fault protection is enabled and OCD break request from the OCD system will trigger a non-recoverable fault.

Value	Description
0	No faults are generated when TCC is halted in Debug mode.
1	A non recoverable fault is generated and FAULTD flag is set when TCC is halted in Debug mode.

#### Bit 0 – DBGRUN Debug Running State

This bit is not affected by software Reset and must not be changed by software while the TCC is enabled.

Value	Description
0	The TCC is halted when the device is halted in Debug mode.
1	The TCC continues normal operation when the device is halted in Debug mode.

### 41.8.9 Event Control

**Name:** EVCTRL  
**Offset:** 0x20  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
					MCEO3	MCEO2	MCEO1	MCEO0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
					MCEI3	MCEI2	MCEI1	MCEI0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	TCEI1	TCEI0	TCINV1	TCINV0		CNTE0	TRGEO	OVFEO
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
	CNTSEL[1:0]		EVACT1[2:0]			EVACT0[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

#### Bits 24, 25, 26, 27 – MCEO Match or Capture Channel x Event Output Enable

These bits control if the match/capture event on channel x is enabled and will be generated for every match or capture.

Value	Description
0	Match/capture x event is disabled and will not be generated.
1	Match/capture x event is enabled and will be generated for every compare/capture on channel x.

#### Bits 16, 17, 18, 19 – MCEI Match or Capture Channel x Event Input Enable

These bits indicate if the match/capture x incoming event is enabled. These bits are used to enable match or capture input events to the CCx channel of TCC.

Value	Description
0	Incoming events are disabled.
1	Incoming events are enabled.

#### Bits 14, 15 – TCEIx Timer/Counter Event Input x Enable

This bit is used to enable input event x to the TCC.

Value	Description
0	Incoming event x is disabled.
1	Incoming event x is enabled.

#### Bits 12, 13 – TCINVx Timer/Counter Event x Invert Enable

This bit inverts the event x input.

Value	Description
0	Input event source x is not inverted.
1	Input event source x is inverted.

**Bit 10 – CNTEO** Timer/Counter Event Output Enable

This bit is used to enable the counter cycle event. When enabled, an event will be generated on begin or end of counter cycle depending of CNTSEL[1:0] settings.

Value	Description
0	Counter cycle output event is disabled and will not be generated.
1	Counter cycle output event is enabled and will be generated depend of CNTSEL[1:0] value.

**Bit 9 – TRGEO** Retrigger Event Output Enable

This bit is used to enable the counter retrigger event. When enabled, an event will be generated when the counter retriggers operation.

Value	Description
0	Counter retrigger event is disabled and will not be generated.
1	Counter retrigger event is enabled and will be generated for every counter retrigger.

**Bit 8 – OVFE0** Overflow/Underflow Event Output Enable

This bit is used to enable the overflow/underflow event. When enabled an event will be generated when the counter reaches the TOP or the ZERO value.

Value	Description
0	Overflow/underflow counter event is disabled and will not be generated.
1	Overflow/underflow counter event is enabled and will be generated for every counter overflow/underflow.

**Bits 7:6 – CNTSEL[1:0]** Timer/Counter Interrupt and Event Output Selection

These bits define on which part of the counter cycle the counter event output is generated.

Value	Name	Description
0x0	BEGIN	An interrupt/event is generated at begin of each counter cycle
0x1	END	An interrupt/event is generated at end of each counter cycle
0x2	BETWEEN	An interrupt/event is generated between each counter cycle.
0x3	BOUNDARY	An interrupt/event is generated at begin of first counter cycle, and end of last counter cycle.

**Bits 5:3 – EVACT1[2:0]** Timer/Counter Event Input 1 Action

These bits define the action the TCC will perform on TCE1 event input.

Value	Name	Description
0x0	OFF	Event action disabled.
0x1	RETRIGGER	Start, restart or re-trigger TC on event
0x2	DIR (asynch)	Direction control
0x3	STOP	Stop TC on event
0x4	DEC	Decrement TC on event
0x5	PPW	Period captured into CC0 Pulse Width on CC1
0x6	PWP	Period captured into CC1 Pulse Width on CC0
0x7	FAULT	Non-recoverable Fault

**Bits 2:0 – EVACT0[2:0]** Timer/Counter Event Input 0 Action

These bits define the action the TCC will perform on TCE0 event input 0.

Value	Name	Description
0x0	OFF	Event action disabled.
0x1	RETRIGGER	Start, restart or re-trigger TC on event
0x2	COUNTEV	Count on event.
0x3	START	Start TC on event
0x4	INC	Increment TC on EVENT
0x5	COUNT (asynch)	Count on active state of asynchronous event
0x6	STAMP	Capture overflow times (Max value)
0x7	FAULT	Non-recoverable Fault

### 41.8.10 Interrupt Enable Clear

**Name:** INTENCLR  
**Offset:** 0x24  
**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 Set (INTENSET) register.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access					MCx3	MCx2	MCx1	MCx0
Reset					R/W	R/W	R/W	R/W
					0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
Reset	R/W	R/W	R/W	R/W	R/W	R/W		
	0	0	0	0	0	0		
Bit	7	6	5	4	3	2	1	0
Access					ERR	CNT	TRG	OVF
Reset					R/W	R/W	R/W	R/W
					0	0	0	0

#### Bits 16, 17, 18, 19 – MCx Match or Capture Channel x Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the corresponding Match or Capture Channel x Interrupt Disable/Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

#### Bit 15 – FAULT1 Non-Recoverable Fault x Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Non-Recoverable Fault x Interrupt Disable/Enable bit, which disables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

#### Bit 14 – FAULT0 Non-Recoverable Fault x Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Non-Recoverable Fault x Interrupt Disable/Enable bit, which disables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

**Bit 13 – FAULTB** Recoverable Fault B Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Recoverable Fault B Interrupt Disable/Enable bit, which disables the Recoverable Fault B interrupt.

Value	Description
0	The Recoverable Fault B interrupt is disabled.
1	The Recoverable Fault B interrupt is enabled.

**Bit 12 – FAULTA** Recoverable Fault A Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Recoverable Fault A Interrupt Disable/Enable bit, which disables the Recoverable Fault A interrupt.

Value	Description
0	The Recoverable Fault A interrupt is disabled.
1	The Recoverable Fault A interrupt is enabled.

**Bit 11 – DFS** Non-Recoverable Debug Fault Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Debug Fault State Interrupt Disable/Enable bit, which disables the Debug Fault State interrupt.

Value	Description
0	The Debug Fault State interrupt is disabled.
1	The Debug Fault State interrupt is enabled.

**Bit 10 – UFS** Non-Recoverable Update Fault Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Non-Recoverable Update Fault Interrupt Disable/Enable bit, which disables the Non-Recoverable Update Fault interrupt.

Value	Description
0	The Non-Recoverable Update Fault interrupt is disabled.
1	The Non-Recoverable Update Fault interrupt is enabled.

**Bit 3 – ERR** Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Error Interrupt Disable/Enable bit, which disables the Compare interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

**Bit 2 – CNT** Counter Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Counter Interrupt Disable/Enable bit, which disables the Counter interrupt.

Value	Description
0	The Counter interrupt is disabled.
1	The Counter interrupt is enabled.

**Bit 1 – TRG** Retrigger Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Retrigger Interrupt Disable/Enable bit, which disables the Retrigger interrupt.

Value	Description
0	The Retrigger interrupt is disabled.
1	The Retrigger interrupt is enabled.

**Bit 0 – OVF** Overflow Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Overflow Interrupt Disable/Enable bit, which disables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.



### 41.8.11 Interrupt Enable Set

**Name:** INTENSET  
**Offset:** 0x28  
**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					MC3	MC2	MC1	MC0
Reset					R/W	R/W	R/W	R/W
					0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
Reset	R/W	R/W	R/W	R/W	R/W	R/W		
	0	0	0	0	0	0		
Bit	7	6	5	4	3	2	1	0
Access					ERR	CNT	TRG	OVF
Reset					R/W	R/W	R/W	R/W
					0	0	0	0

#### Bits 16, 17, 18, 19 – MC Match or Capture Channel x Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the corresponding Match or Capture Channel x Interrupt Disable/Enable bit, which enables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

#### Bit 15 – FAULT1 Non-Recoverable Fault x Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Non-Recoverable Fault x Interrupt Disable/Enable bit, which enables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

#### Bit 14 – FAULT0 Non-Recoverable Fault x Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Non-Recoverable Fault x Interrupt Disable/Enable bit, which disables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

**Bit 13 – FAULTB** Recoverable Fault B Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Recoverable Fault B Interrupt Disable/Enable bit, which enables the Recoverable Fault B interrupt.

Value	Description
0	The Recoverable Fault B interrupt is disabled.
1	The Recoverable Fault B interrupt is enabled.

**Bit 12 – FAULTA** Recoverable Fault A Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Recoverable Fault A Interrupt Disable/Enable bit, which enables the Recoverable Fault A interrupt.

Value	Description
0	The Recoverable Fault A interrupt is disabled.
1	The Recoverable Fault A interrupt is enabled.

**Bit 11 – DFS** Non-Recoverable Debug Fault Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Debug Fault State Interrupt Disable/Enable bit, which enables the Debug Fault State interrupt.

Value	Description
0	The Debug Fault State interrupt is disabled.
1	The Debug Fault State interrupt is enabled.

**Bit 10 – UFS** Non-Recoverable Update Fault Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Non-Recoverable Update Fault Interrupt Disable/Enable bit, which disables the Non-Recoverable Update Fault interrupt.

Value	Description
0	The Non-Recoverable Update Fault interrupt is disabled.
1	The Non-Recoverable Update Fault interrupt is enabled.

**Bit 3 – ERR** Error Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Error Interrupt Disable/Enable bit, which enables the Compare interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

**Bit 2 – CNT** Counter Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Retrigger Interrupt Disable/Enable bit, which enables the Counter interrupt.

Value	Description
0	The Counter interrupt is disabled.
1	The Counter interrupt is enabled.

**Bit 1 – TRG** Retrigger Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Retrigger Interrupt Disable/Enable bit, which enables the Retrigger interrupt.

Value	Description
0	The Retrigger interrupt is disabled.
1	The Retrigger interrupt is enabled.

**Bit 0 – OVF** Overflow Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Overflow Interrupt Disable/Enable bit, which enables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

### 41.8.12 Interrupt Flag Status and Clear

**Name:** INTFLAG  
**Offset:** 0x2C  
**Reset:** 0x00000000  
**Property:** -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access					MC3	MC2	MC1	MC0
Reset					R/W	R/W	R/W	R/W
					0	0	0	0
Bit	15	14	13	12	11	10	9	8
Access	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
Reset	R/W	R/W	R/W	R/W	R/W	R/W		
	0	0	0	0	0	0		
Bit	7	6	5	4	3	2	1	0
Access					ERR	CNT	TRG	OVF
Reset					R/W	R/W	R/W	R/W
					0	0	0	0

#### Bits 16, 17, 18, 19 – MC Match or Capture Channel x Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a match with the compare condition or once the CCx register contains a valid capture value.

Writing a '0' to one of these bits has no effect.

Writing a '1' to one of these bits will clear the corresponding Match or Capture Channel x interrupt flag.

In the Capture operation, this flag is automatically cleared when the CCx register is read.

#### Bit 15 – FAULT1 Non-Recoverable Fault x Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a Non-Recoverable Fault x occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Non-Recoverable Fault x interrupt flag.

#### Bit 14 – FAULT0 Non-Recoverable Fault x Interrupt Enable

Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Non-Recoverable Fault x Interrupt Disable/Enable bit, which disables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

#### Bit 13 – FAULTB Recoverable Fault B Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a Recoverable Fault B occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Recoverable Fault B interrupt flag.

**Bit 12 – FAULTA** Recoverable Fault A Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a Recoverable Fault B occurs.  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit clears the Recoverable Fault B interrupt flag.

**Bit 11 – DFS** Non-Recoverable Debug Fault State Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a Debug Fault State occurs.  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit clears the Debug Fault State interrupt flag.

**Bit 10 – UFS** Non-Recoverable Update Fault Interrupt Enable

This flag is set when the RAMP index changes and the Lock Update bit is set (CTRLBSET.LUPD).  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit will clear the Non-Recoverable Update Fault Interrupt Disable/Enable bit, which disables the Non-Recoverable Update Fault interrupt.

Value	Description
0	The Non-Recoverable Update Fault interrupt is disabled.
1	The Non-Recoverable Update Fault interrupt is enabled.

**Bit 3 – ERR** Error Interrupt Flag

This flag is set if a new capture occurs on a channel when the corresponding Match or Capture Channel x interrupt flag is one. In which case, there is nowhere to store the new capture.  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit clears the error interrupt flag.

**Bit 2 – CNT** Counter Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a counter event occurs.  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit clears the CNT interrupt flag.

**Bit 1 – TRG** Retrigger Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after a counter retrigger occurs.  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit clears the re-trigger interrupt flag.

**Bit 0 – OVF** Overflow Interrupt Flag

This flag is set on the next CLK\_TCC\_COUNT cycle after an overflow condition occurs.  
 Writing a '0' to this bit has no effect.  
 Writing a '1' to this bit clears the Overflow interrupt flag.

### 41.8.13 Status

**Name:** STATUS  
**Offset:** 0x30  
**Reset:** 0x00000001  
**Property:** -

Bit	31	30	29	28	27	26	25	24
					CMP3	CMP2	CMP1	CMP0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
					CCBUFV3	CCBUFV2	CCBUFV1	CCBUFV0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	15	14	13	12	11	10	9	8
	FAULT1	FAULT0	FAULTB	FAULTA	FAULT1IN	FAULT0IN	FAULTBIN	FAULTAIN
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	PERBUFV		PATTBUFV	SLAVE	DFS	UFS	IDX	STOP
Access	R/W		R/W	R	R/W	R/W	R	R
Reset	0		0	0	0	0	0	1

#### Bits 24, 25, 26, 27 – CMP Channel x Compare Value

This bit reflects the channel x output compare value.

Value	Description
0	Channel compare output value is 0.
1	Channel compare output value is 1.

#### Bits 16, 17, 18, 19 – CCBUFV Channel x Compare or Capture Buffer Valid

For a compare channel, this bit is set when a new value is written to the corresponding CCBUFx register. The bit is cleared either by writing a '1' to the corresponding location when CTRLB.LUPD is set, or automatically on an UPDATE condition.

For a capture channel, the bit is set when a valid capture value is stored in the CCBUFx register. The bit is automatically cleared when the CCx register is read.

#### Bits 14, 15 – FAULT Non-recoverable Fault x State

This bit is set by hardware as soon as non-recoverable Fault x condition occurs.

This bit is cleared by writing a one to this bit and when the corresponding FAULTxIN status bit is low. Once this bit is clear, the timer/counter will restart from the last COUNT value. To restart the timer/counter from BOTTOM, the timer/counter restart command must be executed before clearing the corresponding STATEx bit. For further details on timer/counter commands, refer to the available commands description (CTRLBSET.CMD).

#### Bit 13 – FAULTB Recoverable Fault B State

This bit is set by hardware as soon as recoverable Fault B condition occurs.

This bit can be cleared by hardware when the Fault B action is resumed or by writing a '1' to this bit when the corresponding FAULTBIN bit is low. If the software halt command is enabled (FAULTB.HALT=SW), clearing this bit will release the timer/counter.

**Bit 12 – FAULTA** Recoverable Fault A State

This bit is set by hardware as soon as the recoverable Fault A condition occurs.  
 This bit can be cleared by hardware when the Fault A action is resumed or by writing a '1' to this bit when the corresponding FAULTAIN bit is low. If the software halt command is enabled (FAULTA.HALT=SW), clearing this bit will release the timer/counter.

**Bit 11 – FAULT1IN** Non-Recoverable Fault 1 Input

This bit is set while an active Non-Recoverable Fault 1 input is present.

**Bit 10 – FAULT0IN** Non-Recoverable Fault 0 Input

This bit is set while an active Non-Recoverable Fault 0 input is present.

**Bit 9 – FAULTBIN** Recoverable Fault B Input

This bit is set while an active Recoverable Fault B input is present.

**Bit 8 – FAULTAIN** Recoverable Fault A Input

This bit is set while an active Recoverable Fault A input is present.

**Bit 7 – PERBUFV** Period Buffer Valid

This bit is set when a new value is written to the PERBUF register. This bit is automatically cleared by hardware on the UPDATE condition when CTRLB.LUPD is set or by writing a '1' to this bit.

**Bit 5 – PATTBUFV** Pattern Generator Value Buffer Valid

This bit is set when a new value is written to the PATTBUF register. This bit is automatically cleared by hardware on the UPDATE condition when CTRLB.LUPD is set or by writing a '1' to this bit.

**Bit 4 – SLAVE** Client

This bit is set when TCC is set in Client mode. This bit follows the CTRLA.MSYNC bit state.

**Bit 3 – DFS** Debug Fault State

This bit is set by hardware in Debug mode when the DDBGCTRL.FDDBD bit is set. The bit is cleared by writing a '1' to this bit and when the TCC is not in Debug mode.  
 When the bit is set, the counter is halted and the Waveforms state depends on the DRVCTRL.NRE and DRVCTRL.NRV registers.

**Bit 2 – UFS** Non-recoverable Update Fault State

This bit is set by hardware when the RAMP index changes and the Lock Update bit is set (CTRLBSET.LUPD). The bit is cleared by writing a one to this bit.  
 When the bit is set, the waveforms state depends on the DRVCTRL.NRE and DRVCTRL.NRV registers.

**Bit 1 – IDX** Ramp Index

In RAMP2 and RAMP2A operation, the bit is cleared during the cycle A and set during the cycle B. In RAMP1 operation, the bit always reads zero. See *Ramp Operations* from Related Links..

**Bit 0 – STOP** Stop

This bit is set when the TCC is disabled either on a STOP command or on an UPDATE condition when One-Shot operation mode is enabled (CTRLBSET.ONESHOT=1).

This bit is clear on the next incoming counter increment or decrement.

Value	Description
0	Counter is running.
1	Counter is stopped.

**Related Links**

[41.6.3.4. Ramp Operations](#)

#### 41.8.14 Counter Value

**Name:** COUNT  
**Offset:** 0x34  
**Reset:** 0x00000000  
**Property:** PAC Write-Protection, Write-Synchronized, Read-Synchronized

**Note:** Prior to any read access, this register must be synchronized by user by writing the according TCC Command value to the Control B Set register (CTRLBSET.CMD=READSYNC).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
Access	COUNT[23:16]							
Reset	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	COUNT[15:8]							
Reset	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	COUNT[7:0]							
Reset	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 23:0 – COUNT[23:0] Counter Value

These bits hold the value of the Counter register.

**Note:** When the TCC is configured as 16-bit timer/counter, the excess bits are read zero.

**Note:** This bit field occupies the MSB of the register, [23:m]. m is dependent on the Resolution bit in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [23:m]
0x0 - NONE	23:0 (depicted)
0x1 - DITH4	23:4
0x2 - DITH5	23:5
0x3 - DITH6	23:6



### 41.8.15 Pattern

**Name:** PATT  
**Offset:** 0x38  
**Reset:** 0x0000  
**Property:** Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	PGV7	PGV6	PGV5	PGV4	PGV3	PGV2	PGV1	PGV0
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
	PGE7	PGE6	PGE5	PGE4	PGE3	PGE2	PGE1	PGE0
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 8, 9, 10, 11, 12, 13, 14, 15 – PGV** Pattern Generation Output Value  
 This register holds the values of pattern for each waveform output.

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – PGE** Pattern Generation Output Enable  
 This register holds the enable status of pattern generation for each waveform output. A bit written to '1' will override the corresponding SWAP output with the corresponding PGVn value.

## 41.8.16 Waveform

**Name:** WAVE  
**Offset:** 0x3C  
**Reset:** 0x00000000  
**Property:** Write-Synchronized

Bit	31	30	29	28	27	26	25	24
					SWAP3	SWAP2	SWAP1	SWAP0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	23	22	21	20	19	18	17	16
			POL5	POL4	POL3	POL2	POL1	POL0
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
					CICCEN3	CICCEN2	CICCEN1	CICCEN0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	7	6	5	4	3	2	1	0
	CIPEREN	RAMP[1:0]			WAVEGEN[2:0]			
Access	R/W	R/W			R/W			
Reset	0	0			0			

### Bits 24, 25, 26, 27 – SWAP Swap DTI Output Pair x

Setting these bits enables the output swap of DTI outputs [x] and [x+WO\_NUM/2]. Note, the DTIxEN settings will not affect the swap operation.

### Bits 16, 17, 18, 19, 20, 21 – POL Channel Polarity x

Setting these bits enables the output polarity in single-slope and dual-slope PWM operations.

Value	Name	Description
0	(single-slope PWM waveform generation)	Compare output is initialized to ~DIR and set to DIR when the TCC counter matches the CCx value
1	(single-slope PWM waveform generation)	Compare output is initialized to DIR and set to ~DIR when the TCC counter matches the CCx value.
0	(dual-slope PWM waveform generation)	Compare output is set to ~DIR when the TCC counter matches the CCx value
1	(dual-slope PWM waveform generation)	Compare output is set to DIR when the TCC counter matches the CCx value.

### Bits 8, 9, 10, 11 – CICCEN Circular CC Enable x

Setting this bit enables the compare circular buffer option on the first four Compare/Capture channels. When the bit is set, the CCx register value is copied-back into the CCx register on the UPDATE condition.

### Bit 7 – CIPEREN Circular Period Enable

Setting this bit enables the period circular buffer option. When the bit is set, the PER register value is copied-back into the PERB register on UPDATE condition.

### Bits 5:4 – RAMP[1:0] Ramp Operation

These bits select the Ramp operation (RAMP). These bits are not synchronized.

Value	Name	Description
0x0	RAMP1	RAMP1 operation
0x1	RAMP2A	Alternative RAMP2 operation
0x2	RAMP2	RAMP2 operation
0x3	RAMP2C	Critical RAMP2 operation

**Bits 2:0 – WAVEGEN[2:0]** Waveform Generation Operation

These bits select the waveform generation operation. The settings impact the top value and control if the frequency or PWM waveform generation must be used. These bits are not synchronized.

Value	Name	Description						
		Operation	Top	Update	Waveform Output On Match	Waveform Output On Update	OVFIF/Event Up Down	
0x0	NFRQ	Normal Frequency	PER	TOP/Zero	Toggle	Stable	TOP	Zero
0x1	MFRQ	Match Frequency	CC0	TOP/Zero	Toggle	Stable	TOP	Zero
0x2	NPWM	Normal PWM	PER	TOP/Zero	Set	Clear	TOP	Zero
0x3	Reserved	—	—	—	—	—	—	—
0x4	DSCRITICAL	Dual-slope PWM	PER	Zero	~DIR	Stable	—	Zero
0x5	DSBOTTOM	Dual-slope PWM	PER	Zero	~DIR	Stable	—	Zero
0x6	DSBOTH	Dual-slope PWM	PER	TOP & Zero	~DIR	Stable	TOP	Zero
0x7	DSTOP	Dual-slope PWM	PER	Zero	~DIR	Stable	TOP	—

### 41.8.17 Period Value

**Name:** PER  
**Offset:** 0x40  
**Reset:** 0xFFFFFFFF  
**Property:** Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	PER[17:10]							
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
	PER[9:2]							
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
	PER[1:0]		DITHER[5:0]					
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

#### Bits 23:6 – PER[17:0] Period Value

These bits hold the value of the TCC period count.

**Note:** When the TCC is configured as 16-bit timer/counter, the excess bits are read zero.

**Note:** This bit field occupies the MSB of the register, [23:m]. m is dependent on the Resolution bit in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [23:m]
0x0 - NONE	23:0
0x1 - DITH4	23:4
0x2 - DITH5	23:5
0x3 - DITH6	23:6 (depicted)

#### Bits 5:0 – DITHER[5:0] Dithering Cycle Number

These bits hold the number of extra cycles that are added on the PWM pulse period every 64 PWM frames.

**Note:** This bit field consists of the n LSB of the register. n is dependent on the value of the Resolution bits in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [n:0]
0x0 - NONE	-
0x1 - DITH4	3:0
0x2 - DITH5	4:0
0x3 - DITH6	5:0 (depicted)

### 41.8.18 Compare/Capture Channel x

**Name:** CC  
**Offset:** 0x44 + n\*0x04 [n=0..5]  
**Reset:** 0x00000000  
**Property:** Write-Synchronized, Read-Synchronized

The CCx register represents the 16-, 24- bit value, CCx. The register has two functions, depending of the mode of operation.

For capture operation, this register represents the second buffer level and access point for the CPU and DMA.

For compare operation, this register is continuously compared to the counter value. Normally, the output from the comparator is then used for generating waveforms.

CCx register is updated with the buffer value from their corresponding CCBUFx register when an UPDATE condition occurs.

In addition, in match frequency operation, the CC0 register controls the counter period.

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

#### Bits 23:6 – CC[17:0] Channel x Compare/Capture Value

These bits hold the value of the Channel x compare/capture register.

**Notes:**

1. When the TCC is configured as a 16-bit timer/counter, the excess bits are read as zero.
2. This bit field occupies the MSB of the register, [23:m]. m is dependent on the Resolution bit in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [23:m]
0x0 - NONE	23:0
0x1 - DITH4	23:4
0x2 - DITH5	23:5
0x3 - DITH6	23:6 (depicted)

**Bits 5:0 – DITHER[5:0]** Dithering Cycle Number

These bits hold the number of extra cycles that are added on the PWM pulse width every 64 PWM frames.

**Note:** This bit field consists of the n LSB of the register. n is dependent on the value of the Resolution bits in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [n:0]
0x0 - NONE	-
0x1 - DITH4	3:0
0x2 - DITH5	4:0
0x3 - DITH6	5:0 (depicted)

### 41.8.19 Pattern Buffer

**Name:** PATTBUF  
**Offset:** 0x64  
**Reset:** 0x0000  
**Property:** Write-Synchronized, Read-Synchronized

Bit	15	14	13	12	11	10	9	8
	PGVB7	PGVB6	PGVB5	PGVB4	PGVB3	PGVB2	PGVB1	PGVB0
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
	PGEB7	PGEB6	PGEB5	PGEB4	PGEB3	PGEB2	PGEB1	PGEB0
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 8, 9, 10, 11, 12, 13, 14, 15 – PGVB** Pattern Generation Output Value Buffer

This register is the buffer for the PGV register. If double buffering is used, valid content in this register is copied to the PGV register on an UPDATE condition.

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – PGEB** Pattern Generation Output Enable Buffer

This register is the buffer of the PGE register. If double buffering is used, valid content in this register is copied into the PGE register at an UPDATE condition.

## 41.8.20 Period Buffer Value

**Name:** PERBUF  
**Offset:** 0x6C  
**Reset:** 0xFFFFFFFF  
**Property:** Write-Synchronized, Read-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	PERBUF[17:10]							
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
	PERBUF[9:2]							
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
	PERBUF[1:0]		DITHERBUF[5:0]					
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

### Bits 23:6 – PERBUF[17:0] Period Buffer Value

These bits hold the value of the Period Buffer register. The value is copied to PER register on UPDATE condition.

**Note:** When the TCC is configured as 16-bit timer/counter, the excess bits are read zero.

**Note:** This bit field occupies the MSB of the register, [23:m]. m is dependent on the Resolution bit in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [23:m]
0x0 - NONE	23:0
0x1 - DITH4	23:4
0x2 - DITH5	23:5
0x3 - DITH6	23:6 (depicted)

### Bits 5:0 – DITHERBUF[5:0] Dithering Buffer Cycle Number

These bits represent the PER.DITHER bits buffer. When the double buffering is enabled, the value of this bit field is copied to the PER.DITHER bits on an UPDATE condition.

**Note:** This bit field consists of the n LSB of the register. n is dependent on the value of the Resolution bits in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [n:0]
0x0 - NONE	-
0x1 - DITH4	3:0
0x2 - DITH5	4:0
0x3 - DITH6	5:0 (depicted)



#### 41.8.21 Channel x Compare/Capture Buffer Value

**Name:** CCBUF  
**Offset:** 0x70 + n\*0x04 [n=0..5]  
**Reset:** 0x00000000  
**Property:** Write-Synchronized, Read-Synchronized

CCBUFx is copied into CCx at TCC update time

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	CCBUF[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
	CCBUF[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
	CCBUF[1:0]		DITHERBUF[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 23:6 – CCBUF[17:0] Channel x Compare/Capture Buffer Value

These bits hold the value of the Channel x Compare/Capture Buffer Value register. The register serves as the buffer for the associated compare or capture registers (CCx). Accessing this register using the CPU or DMA will affect the corresponding CCBUFVx status bit.

**Notes:**

1. When the TCC is configured as a 16-bit timer/counter, the excess bits are read as zero.
2. This bit field occupies the MSB of the register, [23:m]. m is dependent on the Resolution bit in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [23:m]
0x0 - NONE	23:0
0x1 - DITH4	23:4
0x2 - DITH5	23:5
0x3 - DITH6	23:6 (depicted)

#### Bits 5:0 – DITHERBUF[5:0] Dithering Buffer Cycle Number

These bits represent the CCx.DITHER bits buffer. When the double buffering is enable, DITHERBUF bits value is copied to the CCx.DITHER bits on an UPDATE condition.

**Note:** This bit field consists of the n LSB of the register. n is dependent on the value of the Resolution bits in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [n:0]
0x0 - NONE	-
0x1 - DITH4	3:0
0x2 - DITH5	4:0
0x3 - DITH6	5:0 (depicted)

## 42. Zigbee Bluetooth Radio Subsystem (ZBT)

### 42.1 Overview

The PIC32CX-BZ2 and the WBZ45 support on-chip IEEE 802.15.4 and 802.15.3 compliant Zigbee, Bluetooth® Low Energy 5.2 interface with integrated transceivers. The Wireless Subsystem block is comprised of on-chip Zigbee and Bluetooth 5.0 BBP/MAC, shared RF transceiver and Radio Arbiter. This section provides key features of the on-chip Wireless modules.

With integrated Ultra Low Power 2.4 GHz ISM band single transceiver, dual modem and dual MAC, the Radio supports dual Zigbee and Bluetooth 5.2 link protocols. An onboard intelligent Radio Arbiter HW module establishes both the links simultaneously using a single Radio Transmit/Receive with programmable QoS. The RF transceiver includes dual Power Amplifiers architecture and TR Switch. Therefore, medium to high power application use cases are supported without external FEM.

Arbitration between Application, Bluetooth link stack, Zigbee link stack and miscellaneous maintenance tasks are handled on the Cortex M4F (w/ DSP, FPU) CPU using available on-chip system memory resources via RTOS.

**Note:** Unique Bluetooth and 802.15.4 MAC address are available in OTP memory region. The software SDK and operational stacks provided by Microchip provide API's to access these addresses.

### 42.2 Features

#### 2.4 GHz RF Transceiver

- Integrated 2.4 GHz Ultra Low Power RF Transceiver shared between Bluetooth and Zigbee Modems and Link (MAC) Controllers
- Integrated 16 MHz  $\pm$ 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: up to 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:
  - -97 dBm for Bluetooth Low Energy 1 Mbps
  - -93 dBm for Bluetooth Low Energy 2 Mbps
  - -104 dBm for Bluetooth Low Energy 125 Kbps
  - -101 dBm for Bluetooth Low Energy 500 Kbps
  - Digital RSSI indicator (-90 dBm ~ -50 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
- HCI Interface via UART
- 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
  - 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**

- 802.15.4/Zigbee PSDU data rate: 250 Kbps
- Programmable RX Mode:
  - -103 dBm RX sensitivity in the Continuous mode
  - -98 dBm sensitivity in the RPC mode
  - RPC mode provides lower power consumption in RX mode to support California Green Energy Specification at the system level
- 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)
- Zigbee Stack Support:
  - Zigbee 3.0 ready

- Zigbee Pro 2015
- Zigbee green power support (proxy, sink and multi-sensor)

**Proprietary<sup>(1)</sup>**

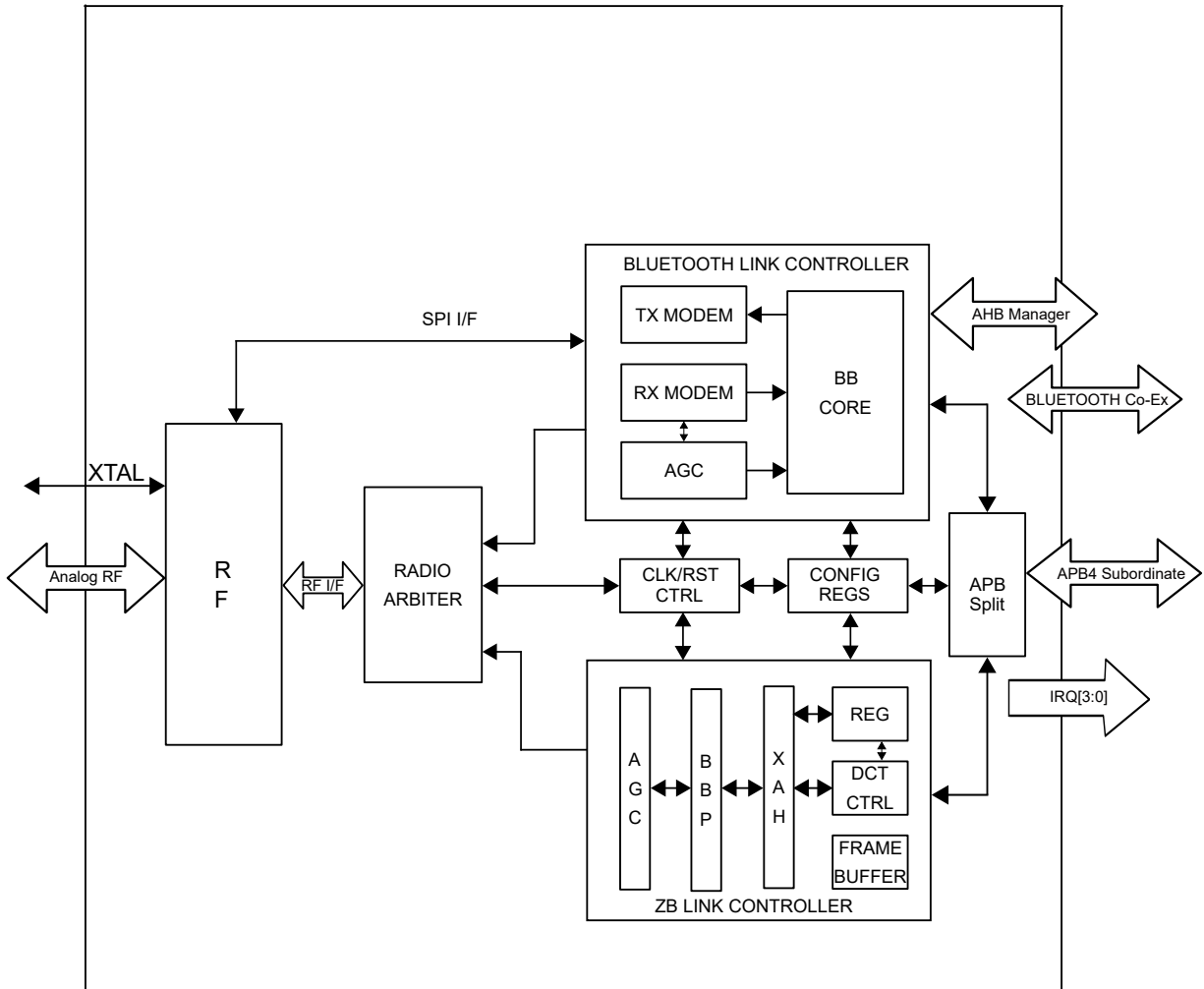
- 500 kbps and 1 Mbps are 2.4 GHz Proprietary with DSSS
- 2 Mbps are 2.4 GHz Proprietary without DSSS
- TX Output Power up to +12 dBm
- Receiver Sensitivity up to -96 dBm
- Hardware Assisted MAC:
  - Auto acknowledge
  - Auto retry
  - Channel access back-off
- SFD Detection, Spreading, De-spreading, 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)

**Note:**

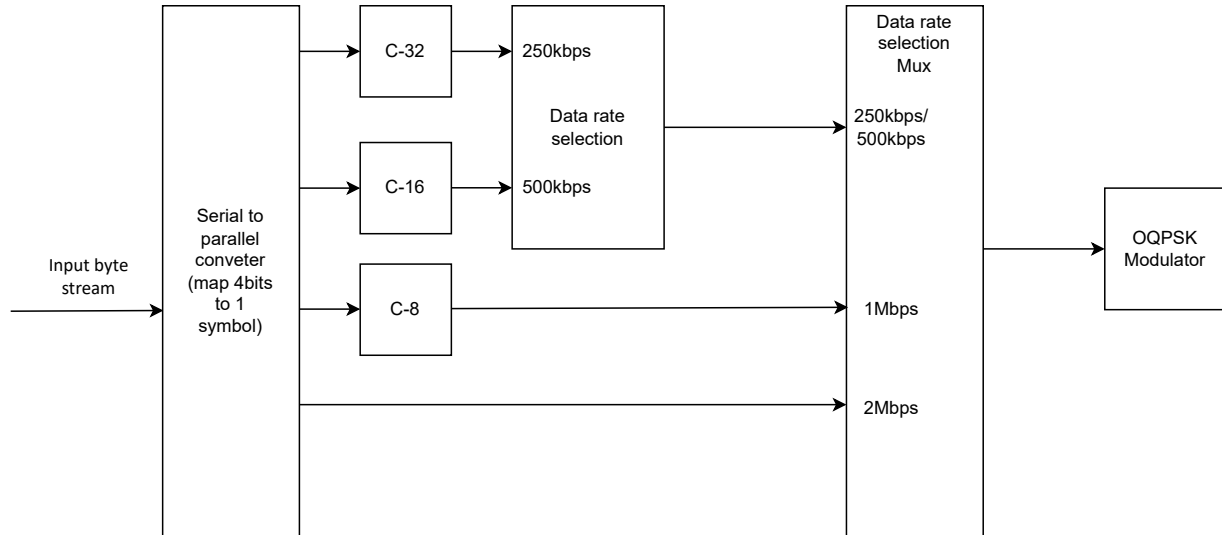
1. Proprietary modes are compatible to AT86RF233 modes of similar data rate.

### 42.3 Wireless Subsystem Top Level Diagram

Figure 42-1. Wireless Subsystem Top Level Diagram



**Figure 42-2.** Zigbee Baseband Processor Block Diagram



## 42.4 Bluetooth Link Controller

The APB interface provides access to internal register banks of the macro. These registers are programmed by the host (firmware) to set various configurations, trigger commands, read status, service interrupts and other functions.

All Bluetooth operations are carried out as transmit or receive tasks within the Link controller. There are several task controllers:

- Firmware – Firmware can trigger tasks by writing the task controller registers.
- Hardware Schedule Controller – This is Bluetooth 5.2 Bluetooth Low Energy advertisement scheduler controller (Adv. role).
- Hardware Scanner – This is Bluetooth 5.2 Bluetooth Low Energy advertisement scanner (Central role).
- Bluetooth Low Energy advertisement controller – This is Bluetooth 5.2 advertisement controller (Adv. role).

The requests from the task controllers are arbitrated and is carried out by the task controller.

### 42.4.1 Hardware Schedule Controller

Hardware Schedule controller is a hardware accelerator that offloads the Bluetooth 5.2 advertising task from the firmware and executes it in hardware completely.

### 42.4.2 Hardware Scanner

Hardware Scanner is another hardware accelerator to be used in the Bluetooth Low Energy/central mode. This offloads the Bluetooth Low Energy task of scanning for advertisements from the firmware. This enables fast scanning and quick connections.

### 42.4.3 Bluetooth Low Energy Advertisement Controller

Bluetooth Low Energy advertisement controller is a hardware controller that supports offloading of Bluetooth 5.2 advertisements. This is a simpler advertising scenario requiring much smaller memory.

The task controller manages overall TX/RX functionality and starts TX control FSM / RX control FSM in the baseband as per requirement to transmit/receive a packet. The task controller also manages the coexistence interface with Wi-Fi. The task controller can also be used directly by the firmware to

measure the RSSI across various Bluetooth channels, which can, then, be used to generate an AFH channel map.

#### 42.4.4 Bluetooth Clock Controller

The Bluetooth clock controller/Slot timer provides information on Bluetooth slots and the slot boundary. The Bluetooth clock synchronization and clock adjustment due to RX window uncertainty is also handled by this controller. It controls the timings for Bluetooth Low Energy non-TIFS tasks.

TX/RX control FSM (State machines) provide sequencing and control of various phases of a task. These control various elements in the TX and RX data path.

The Crypto engine is used for AES encryption and decryption of transmit and receive data. This performs inline encryption as the data passes through the data path in the BB link controller. This controller is also used to perform offline CMAC operation directly under application firmware control. This controller is also used by the Bluetooth Low Energy privacy controller for the generation and decoding of hash needed for resolving private addresses.

#### 42.4.5 Bluetooth Low Energy Privacy Controller

Bluetooth Low Energy Privacy controller is to implement the Bluetooth Low Energy privacy feature. It can be used in offline mode by the firmware to generate a Resolvable Private Address (RPA) to be used for a transmission. And it is also used inline by the RX Control FSM to resolve the RPA received in the packet before implementing the device filtering using the white list.

The Channel Hop Controller implements the logic needed to compute the next RF channel frequency to be used.

A dedicated math unit for big number (128-bit) operations is available for firmware to handle large number operations. This is programmed and used by the firmware directly. There is no direct use of this block by any other hardware block.

For simple, secure pairing, the P256 ECC module is available to perform computationally intensive key-matching procedures in hardware.

The transmit memory read controller reads the data from the common memory and feeds them to the transmit data path for Bluetooth PDU creations and transmit. The receive memory write controller receives data from the RX data path and writes them to the common memory.

### 42.5 Zigbee/Proprietary Data Rate Link Controller

The Zigbee Link Controller implements Zigbee 3.0 (802.15.4-2011) MAC and baseband functionality in hardware.

The CPU interface (APB Subordinate) provides access to internal registers of the macro. These registers are programmed by the host (firmware) to kick off transmit/receive functionality along with the programming of other features. The APB Subordinate contains shadow registers for the status registers in the design for single-cycle, contention-free read accesses. Interrupt controller logic also resides in the APB subordinate module.

Separate 128-byte frame buffers are provided for TX and RX.

The XAH module implements basic MAC functionality as well as a hardware accelerator for features such as automatic acknowledgement, CSMA-CA and retransmission, automatic FCS check and so on.

The BBP module implements Offset-QPSK PHY with 250 kb/s PSDU data rate. It also supports proprietary 500 kbps OQPSK PHY, 1 Mbps MSK PHY, and a 2 Mbps MSK non-spreading PHY.

#### 42.5.1 Transmit Operation

A frame transmission comprises of two actions: a write to Frame Buffer and the transmission of its contents. Both actions can be run in parallel if required by critical protocol timing.

## 42.5.2 Receive Operation

A frame reception comprises of two actions: the transceiver listens for, receives and demodulates the frame to the Frame Buffer and signals the reception to the microcontroller. After that process, the microcontroller can read the available frame data from the Frame Buffer via the APB interface.

## 42.6 Radio Arbiter

The Radio Arbiter determines the ownership of the Radio between Zigbee and Bluetooth link controllers. It also generates control to select between Zigbee/Bluetooth to drive the controls and data to or from the RF.

By default, the Arbiter is IDLE and no Link controller has the ownership. The general intention of the design is to provide a radio arbiter that is flexible enough that the arbitration schemes can be tuned by the firmware with real application Bluetooth/Zigbee scenarios.

All design and intellectual property parts of this arbiter design are property of Microchip Technology Inc., released under appropriate NDA.

### 42.6.1 Arbiter Modes

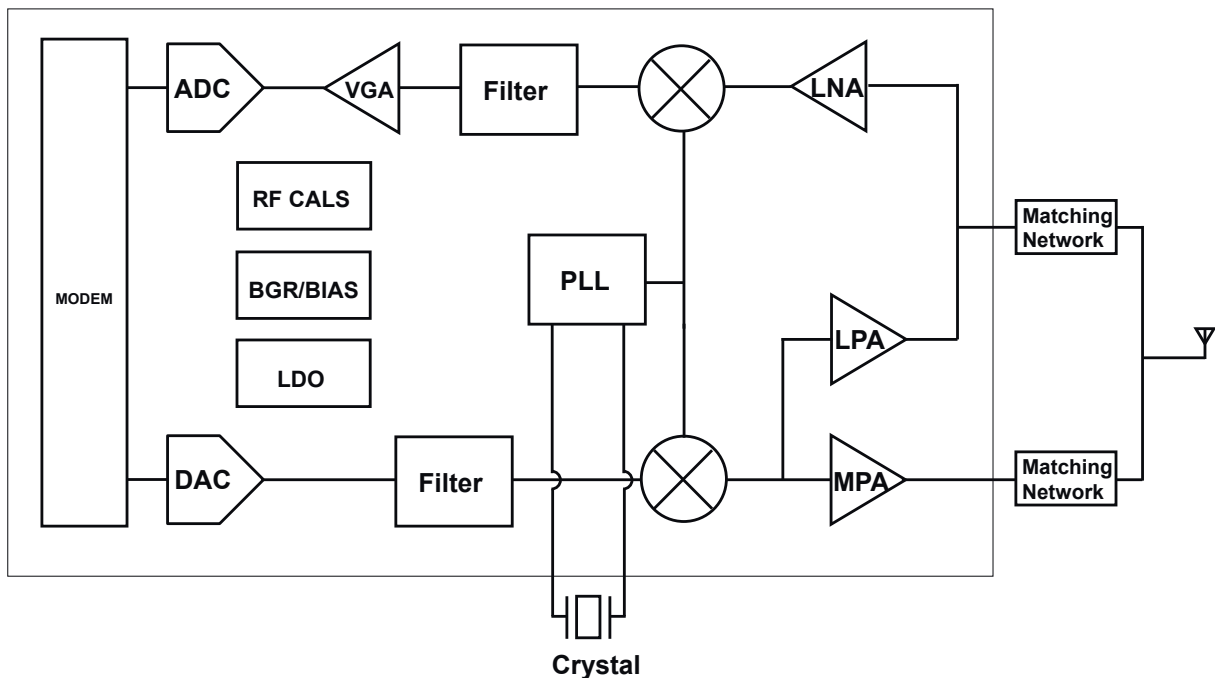
The following are the modes supported by the arbiter:

- Bluetooth static – Radio ownership is with the Bluetooth Link controller
- Zigbee static – Radio ownership is with the Zigbee Link controller
- Dynamic – Radio ownership decided dynamically at every arbitration event

## 42.7 RF Physical Layer

The top level block diagram of the transceiver architecture is as follows.

Figure 42-3. Transceiver Architecture





## 42.8 Frequency Synthesizer

The PIC32CX-BZ2 and the WBZ45 require 16 MHz for generating the local oscillator frequency between the 2.4 GHz to 2.48 GHz for the transmitter and receiver. The frequency synthesizer integrates all the passives for the loop filter inside the chip.

### 42.8.1 Transmit Mixer and Power Amplifier

The PIC32CX-BZ2 and WBZ45 implement a direct conversion I/Q transmitter. The I/Q from the low pass filter of the TX path is converted to the desired output frequency through an I/Q mixer that operates on the LO frequency from the Synthesizer. The VCO operates at 4x the LO frequency.

The supply for LPA and MPA is through separated power supply pins that can be connected to the output from the PMU with necessary decoupling.

The LPA is a single stage amplifier that provides ~5.5 dBm output power in standalone mode and ~4 dBm in LPA/MPA shared mode. Both LPA and MPA implement an on-chip BALUN. The LPA pin also acts as the input to the RX section.

The MPA is a single stage power amplifier designed to provide a maximum output power of ~12 dBm. MPA also implements an on-chip BALUN to get a single-ended TX output. MPA is NOT internally connected to the receiver.

It is mandatory that the LPA pin be used for the Receiver path. On a typical application, the LPA and MPA can be connected directly, eliminating the need for an external switch.

### 42.8.2 Receiver

The receiver path starts with the LPA pin, which is shared with the LPA path and the LNA. PIC32CX-BZ2 implements a low-IF differential receiver. The synthesizer generates the LO for down-conversion on the mixer, which goes through a BPF filter to the ADC. The Receiver implements an AGC to adjust the LNA gain depending upon the input signal level.

## 42.9 RFLDO

The RF section has multiple LDOs to power the various power domains of the RF subsystem, All these LDOs can be powered up from the MLDO/DC-DC mode of the PMU and can get 1.35V and generate 1.2V internally to feed the corresponding RF sections. These LDOs need a bypass capacitor on the output for their operation. See *Power Subsystem* from Related Links.

- RF-PLL
- BUCK-LPA
- BUCK-MPA
- BUCK-BB

### Related Links

[7. Power Subsystem](#)

## 43. Electrical Characteristics

This chapter provides the Electrical Specification and Characteristic of PIC32CX-BZ2 and WBZ45 Module across the operating temperature range of the product.

### 43.1 Absolute Maximum Electrical Characteristics

Exposure to these maximum rating conditions for extended periods may affect device reliability. Functional operation of the device at these or any other conditions above the parameters indicated in the operation listings of this specification is not implied.

**Table 43-1.** Absolute Maximum Ratings

Parameter	Value
Ambient temperature under bias	-40°C to +125°C
Storage temperature	-65°C to +150°C
Voltage on $V_{DD}/V_{DDIO}$ with respect to GND	-0.3V to +4.0V
Voltage on any Non-5V tolerant pin(s), with respect to GND	-0.3V to ( $V_{DD} + 0.3V$ )
Maximum current out of GND pins	200 mA
Maximum current into $V_{DD}$ pins	200 mA
Maximum output current sourced/sunk by any Low Current Mode I/O pin (8x drive strength)	15 mA
Maximum output current sourced/sunk by any Low Current Mode I/O pin (4x drive strength)	10 mA
Maximum output current sourced/sunk by any High Current Mode I/O pin (8x drive strength)	10 mA
Maximum output current sourced/sunk by any High Current Mode I/O pin (4x drive strength)	7 mA
Maximum current sink by all ports	120 mA
Maximum current sourced by all ports	120 mA
<b>ESD Qualification</b>	
Human Body Model (HBM) per JESD22-A114	2000V
Charged Device Model (CDM) (ANSI/ESD STM 5.3.1)...(All pins / Corner pins)	+500V/- 500V
<b>Note:</b>	
1. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.	

#### Related Links

[43.3. Thermal Specifications](#)

### 43.2 DC Electrical Characteristics

**Table 43-2.** Operating Frequency VS. Voltage

Param. No.	$V_{DDIO}, V_{DDANA}$ Range	Temp. Range (in °C)	Max. MCU Frequency	Comments
DC_5	1.9V to 3.6V	-40°C to +85°C	64 MHz	Industrial
DC_7	1.9V to 3.6V	-40°C to +125°C	64 MHz	Extended
<b>Note:</b> The same voltage must be applied to $V_{DDIN}$ and $V_{DDIO}$ .				

## 43.3 Thermal Specifications

**Table 43-3.** Thermal Operating Conditions

Rating	Symbol	Min.	Typ	Max.	Unit
<b>Industrial Temperature Devices:</b>					
Operating ambient temperature range	$T_A$	-40	—	+85	°C
Operating junction temperature range	$T_J$	-40	—	+105	°C
<b>Extended Temperature Range:</b>					
Operating ambient temperature range	$T_A$	-40	—	+125	°C
Operating junction temperature range	$T_J$	-40	—	+145	°C
Maximum allowed power dissipation	$P_{DMAX}$	$(T_J - T_A)/\theta_{JA}$			W

**Table 43-4.** Thermal Packaging Characteristics

Characteristics	Symbol	Typ	Max.	Unit
Thermal Resistance, 48-pin VQFN (7 mm x 7 mm x 0.9 mm) Package	$\theta_{JA}$	31.6	—	°C/W <sup>(1)</sup>
Thermal Resistance, 32-pin VQFN (5 mm x 5 mm x 1 mm) Package	$\theta_{JA}$	35.1	—	°C/W <sup>(1)</sup>
<b>Note:</b>				
1. The junction-to-ambient thermal resistance, $\theta_{JA}$ , numbers are achieved by package simulations. JEDEC standard.				

## 43.4 Power Supply DC Module Electrical Specifications

**Table 43-5.** Power Supply DC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9-3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
REG_1	VDDCORE_CIN	VDDCORE (CLDO_OUT) input bypass parallel capacitor pair <sup>(5)</sup>	—	1	—	μF	Bulk ceramic or solid tantalum with ESR <0.5Ω. Min and max represent absolute values including cap tolerances
			—	100	—	nF	Ceramic XR7/X5R with ESR <0.5Ω depending on temperature
REG_5	VDD33	VDD33 input bypass parallel capacitor pair <sup>(5)</sup>	—	10	—	μF	Bulk ceramic or solid tantalum with ESR <0.5Ω <sup>(5)</sup>
			—	100	—	nF	Ceramic XR7/X5R with ESR <0.5Ω depending on temperature on all VDDIO pins <sup>(5)</sup>
REG_6	PMU_VDDIO	Input bypass parallel capacitor pair for the PMU power section <sup>(5)</sup>	—	4.7	—	μF	Bulk Ceramic or solid Tantalum with ESR <0.5Ω <sup>(5)</sup>
REG_7	PMU_VDDP	Input bypass parallel capacitor pair for the PMU power section <sup>(5)</sup>	—	1	—	μF	Bulk ceramic or solid tantalum with ESR <0.5Ω <sup>(5)</sup>
REG_9	VDDFLASH_CIN	VDD_FLASH bypass parallel capacitor pair <sup>(5)</sup>	—	10	—	μF	Bulk ceramic or solid tantalum with ESR <0.5Ω <sup>(5)</sup>
			—	100	—	nF	Ceramic XR7/X5R with ESR <0.5Ω depending on temperature on all VDDFLASH pins <sup>(5)</sup>

.....continued

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9-3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
REG_17	VDDANA_CIN	VDDANA input bypass parallel capacitor pair <sup>(5)</sup>	—	10	—	μF	Bulk ceramic or solid tantalum with ESR <0.5Ω <sup>(5)</sup>
			—	0.1	—	nF	Ceramic XR7/X5R with ESR <0.5Ω
REG_18	VDDANA_LEXT	VDDANA series ferrite bead DCR (DC resistance)	—	—	0.1	Ω	≥600Ω at 100 MHz
REG_19		Ferrite bead current Rating <sup>(1)</sup>	100	—	—	mA	—
REG_20	BUCK_PLL_CIN	VDD bypass capacitor on the BUCK_PLL input	—	1	—	μF	Ceramic XR7/X5R with ESR <0.5Ω
REG_21	BUCK_BB_CIN	VDD bypass capacitor on the BUCK_BB input	—	1	—	μF	Ceramic XR7 with ESR <0.5Ω
REG_22	BUCK_MPA_CIN	VDD bypass capacitor on the BUCK_LPA input	—	1	—	μF	Ceramic XR7 with ESR <0.5Ω
REG_23	BUCK_LPA_CIN	VDD bypass capacitor on the BUCK_MPA input	—	1	—	μF	Ceramic XR7 with ESR <0.5Ω
REG_24	BUCK_CLDO_CIN	VDD bypass capacitor on the BUCK_CLDO input	—	1	—	μF	Ceramic XR7 with ESR <0.5Ω
REG_25	R_EXT	Bias for reference current generation	—	30	—	kΩ	—
REG_27	VSW_L <sub>EXT</sub> <sup>(2,3)</sup>	Buck Switch mode regulator inductor inductance	—	4.7	—	μH	Shielded inductor only
REG_29		Inductor DCR (DC resistance)	—	—	0.22	Ω	—
REG_31		Inductor I <sub>SAT</sub> rating <sup>(2,6)</sup>	250	—	—	mA	—
REG_32	VSW_CAPEXT	Buck Switch mode regulator bulk capacitor capacitance	10	—	—	μF	—
REG_32A		Buck Switch mode regulator filtering capacitor capacitance	100	—	—	nF	—
REG_36	VDDCORE	VDDCORE voltage range	1.14	1.2	1.26	V	MCU Active, cache and prefetch disabled while executing from Flash
REG_37	VDD33 <sup>(4)</sup>	VDD33 input voltage range	1.9	3.3	3.6	V	—
REG_39	VDDANA <sup>(4)</sup>	VDDANA input voltage range	1.9	3.3	3.6	V	—
REG_40	VDD_PMU	PMU output voltage	1.30	1.35	1.40	V	PMU output voltage
REG_43	SVDDIO_R	VDDIO rise ramp rate to ensure internal Power-on Reset signal	0.03	—	0.11	V/ms	Failure to meet this specification may lead to start-up or unexpected behaviors
REG_44	SVDDIO_F	VDDIO falling ramp rate to ensure internal Power-on Reset signal	—	—	1.39	V/ms	Failure to meet this specification may cause the device to not detect reset

.....continued

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA} 1.9-3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial Temp $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
REG_45	VP0R+	Power-on Reset	—	1.59	—	V	VDDIO power up/Down (See Param REG43, VDDIO Ramp Rate)
REG_45_A	VP0R-	Power-on Reset	—	1.56	—	V	VDDIO Power up/Down (See Param REG43, VDDIO Ramp Rate)
REG_47	VBOR33 <sup>(4)</sup>	VDDIO BOD	—	1.8	—	V	—
REG_48	VBOR12	BOR of the 1.2V regulator	—	1.1	—	V	—
REG_48A	VZPBOR33	Zero power BOR	—	1.8	—	V	—
REG_49	VBOR33L2H	BOR 3.3V low to high switch point	—	1.84	—	V	—
REG_50	VBOR1P2L2H	BOR 1.2V low to high switch point	—	1.1	—	V	—
REG_51	VBOD12 <sub>HYST_STEP</sub>	VBOD12 Hysteresis step size, HYST[3:0]	—	10.5	—	mv	—
REG_52	VBOD33 <sub>HYST_STEP</sub>	VBOD33 Hysteresis step size, HYST[3:0]	—	51.6	—	mv	—
REG_53	TRST <sup>(5)</sup>	External RESET valid active pulse width	—	11	—	$\mu s$	Minimum Reset active time to guarantee MCU Reset for the module. Reset filter circuit inside Module
			—	2.7	—	$\mu s$	Minimum Reset active time to guarantee MCU Reset for SoC with no Reset filter circuit

**Notes:**

- Ferrite Bead ISAT(min)  $\geq$  (IDDANA(max) \* 1.15).
- Buck Inductor ISAT(min)  $\geq$  ((ICAPACITOR + IVDDCORE\_MAX) \* 1.2) when the BUCK mode is enabled (shielded inductor only).
- User must select either LDO or BUCK Mode. The modes are exclusive to each other.
- VDD33 and VDDANA must be at the same voltage level.
- All bypass caps must be located immediately adjacent to pin(s) and on the same side of the PCB as the MCU. Each primary power supply group VDDIO, VDDANA, VDDCORE must have one bulk capacitor and all power pins with a 100 nF bypass cap.
- The RESET pulse width is the minimum pulse width required on the I/O pin after any filtering on the  $\overline{MCLR}$  pin.
- Keep the DCR as low as possible to improve efficiency.
- These parameters are characterized but not tested in manufacturing.

## 43.5 Active Current Consumption DC Electrical Specifications (85°C)

**Table 43-6.** Active Current Consumption DC Electrical Specifications

DC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp			
Param. No.	Symbol	Characteristics	Clock/Freq	Typ. <sup>(1)</sup>	Max.	Units	Conditions
APWR_1	$I_{DD\_ACTIVE}^{(2,3)}$	MCU $I_{DD}$ in Active mode w/LDO mode selected	PLL 64 MHz	105	291	$\mu\text{A}/\text{MHz}$	$V_{DD} = V_{DDANA} = 3.3\text{V}$
			PLL 48 MHz	89	274	$\mu\text{A}/\text{MHz}$	$V_{DD} = V_{DDANA} = 3.3\text{V}$
APWR_13		MCU $I_{DD}$ in Active mode w/BUCK mode selected	PLL 64 MHz	70	209	$\mu\text{A}/\text{MHz}$	$V_{DD} = V_{DDANA} = 3.3\text{V}$
			PLL 48 MHz	62	197	$\mu\text{A}/\text{MHz}$	$V_{DD} = V_{DDANA} = 1.8\text{V}$

**Notes:**

- Typical value measured during characterization across voltage and temperature.
- The test conditions are as follows:
  - All GPIO are input and pulled up.
  - RF System ON with default settings.
  - All Peripherals disabled with PMD bits.
  - All PB clocks are divided by 16.
  - LPRC is set as LPCLK.
  - SOSC is disabled.
  - PMU 1 MHz clock is derived from FRC. FRC is divided by 8.
  - Cache is enabled and configured wait time as 0xF.
  - WCM memories configured in Retention + NAP mode.
  - DSU is disconnected.
- MCU running while(1) loop with 50 NOP instructions.
- These parameters are characterized but not tested in manufacturing.

## 43.6 Active Current Consumption DC Electrical Specifications (125°C)

**Table 43-7.** Active Current Consumption DC Electrical Specifications

DC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp			
Param. No.	Symbol	Characteristics	Clock/Freq	Typ. <sup>(1)</sup>	Max.	Units	Conditions
APWR_1	$I_{DD\_ACTIVE}^{(2,3)}$	MCU $I_{DD}$ in Active mode w/LDO mode selected	PLL 64 MHz	105	411	$\mu\text{A}/\text{MHz}$	$V_{DD} = V_{DDANA} = 3.3\text{V}$
			PLL 48 MHz	89	395	$\mu\text{A}/\text{MHz}$	$V_{DD} = V_{DDANA} = 3.3\text{V}$
APWR_13		MCU $I_{DD}$ in Active mode w/BUCK mode selected	PLL 64 MHz	70	291	$\mu\text{A}/\text{MHz}$	$V_{DD} = V_{DDANA} = 3.3\text{V}$
			PLL 48 MHz	62	279	$\mu\text{A}/\text{MHz}$	$V_{DD} = V_{DDANA} = 1.8\text{V}$

.....continued

DC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp			
Param. No.	Symbol	Characteristics	Clock/Freq	Typ. <sup>(1)</sup>	Max.	Units	Conditions

**Notes:**

1. Typical value measured during characterization across voltage and temperature.
2. The test conditions are as follows:
  - All GPIO are input and pulled up.
  - RF System ON with default settings.
  - All Peripherals disabled with PMD bits.
  - All PB clocks are divided by 16.
  - LPRC is set as LPCLK.
  - SOSOC is disabled.
  - PMU 1 MHz clock is derived from FRC. FRC is divided by 8.
  - Cache is enabled and configured wait time as 0xF.
  - WCM memories configured in Retention + NAP mode.
  - DSU is disconnected.
3. MCU running while(1) loop with 50 NOP instructions.
4. These parameters are characterized but not tested in manufacturing.

**Figure 43-1. Run Mode Current Consumption in Buck Mode at PLL 64 MHz**

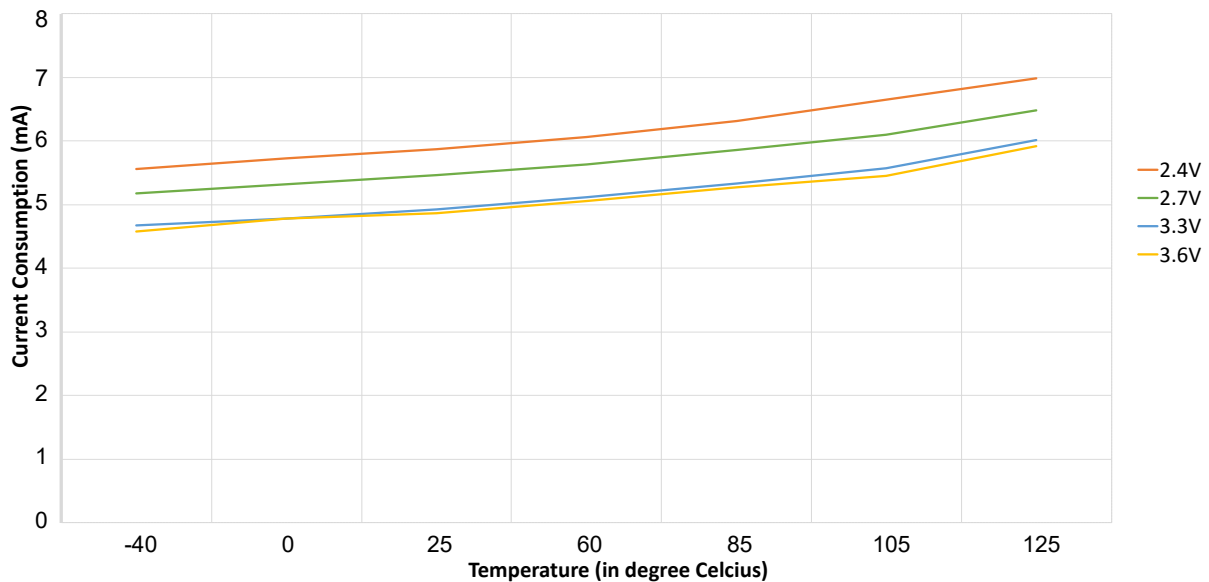
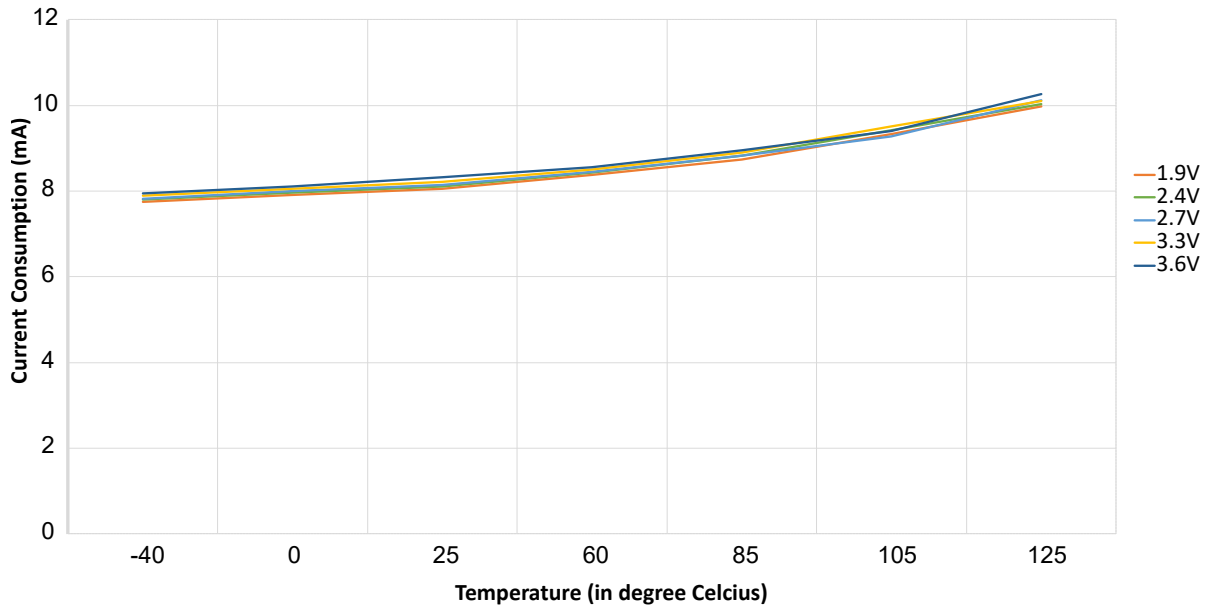


Figure 43-2. Run Mode Current Consumption in MLDO Mode at PLL 64 MHz



## 43.7 Idle Current Consumption DC Electrical Specifications (85°C)

Table 43-8. Idle Current Consumption DC Electrical Specifications

DC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp			
Param. No.	Symbol	Characteristics	Clock/Freq	Typ. <sup>(1)</sup>	Max.	Units	Conditions
IPWR_1	$I_{DD\_IDLE}^{(2)}$	MCU $I_{DD}$ in IDLE mode w/LDO mode selected	PLL 64 MHz	54	184	$\mu\text{A}/\text{MHz}$	$V_{DD} = V_{DDANA} = 3.3\text{V}$
			PLL 48 MHz	68	249		
IPWR_3		MCU $I_{DD}$ in IDLE mode w/BUCK mode selected	PLL 64 MHz	77	258		
			PLL 48 MHz	49	178		

**Notes:**

- Typical value measured during characterization across voltage and temperature.
- The test conditions are as follows:
  - All GPIO are input and pulled up.
  - All Peripherals disabled with PMD bits.
  - All PB clocks are divided by 16.
  - LPRC is set as LPCLK.
  - SOSC is disabled.
  - PMU 1 MHz clock is derived from FRC. FRC is divided by 8.
  - Cache is enabled and configured to wait time as 0xF.
  - WCM memories configured in Retention + NAP mode.
  - DSU is disconnected.
  - RF system OFF.
  - Entry to the Sleep mode is disabled and WFI instruction is executed.
  - On exit by External interrupt, verified RCON status to ensure system was in the IDLE mode.
- These parameters are characterized but not tested in manufacturing.



## 43.8 Idle Current Consumption DC Electrical Specifications (125°C)

**Table 43-9.** Idle Current Consumption DC Electrical Specifications

DC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp			
Param. No.	Symbol	Characteristics	Clock/Freq	Typ. <sup>(1)</sup>	Max.	Units	Conditions
IPWR_1	$I_{DD\_IDLE}^{(2)}$	MCU $I_{DD}$ in IDLE mode w/LDO mode selected	PLL 64 MHz	77	377	$\mu\text{A}/\text{MHz}$	$V_{DD} = V_{DDANA} = 3.3\text{V}$
			PLL 48 MHz	68	369		
IPWR_3		MCU $I_{DD}$ in IDLE mode w/BUCK mode selected	PLL 64 MHz	54	266		
			PLL 48 MHz	49	259		

**Notes:**

- Typical value measured during characterization across voltage and temperature.
- The test conditions are as follows:
  - All GPIO are input and pulled up.
  - All peripherals disabled with PMD bits.
  - All PB clocks are divided by 16.
  - LPRC is set as LPCLK.
  - SOSC is disabled.
  - PMU 1 MHz clock is derived from FRC. FRC is divided by 8.
  - Cache is enabled and configured to wait time as 0xF.
  - WCM memories configured in Retention + NAP mode.
  - DSU is disconnected.
  - RF system OFF.
  - Entry to the Sleep mode is disabled and WFI instruction is executed.
  - On exit by External interrupt, verified RCON status to ensure system was in the IDLE mode.
- These parameters are characterized but not tested in manufacturing.

**Figure 43-3.** Idle Mode Current Consumption in Buck Mode at PLL 64 MHz

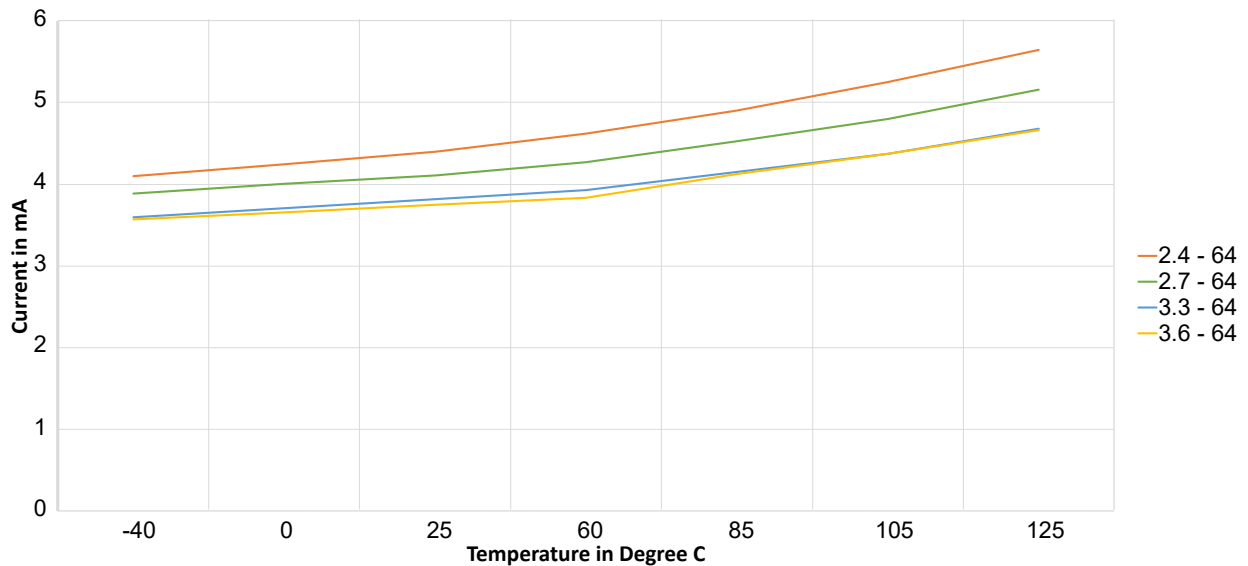
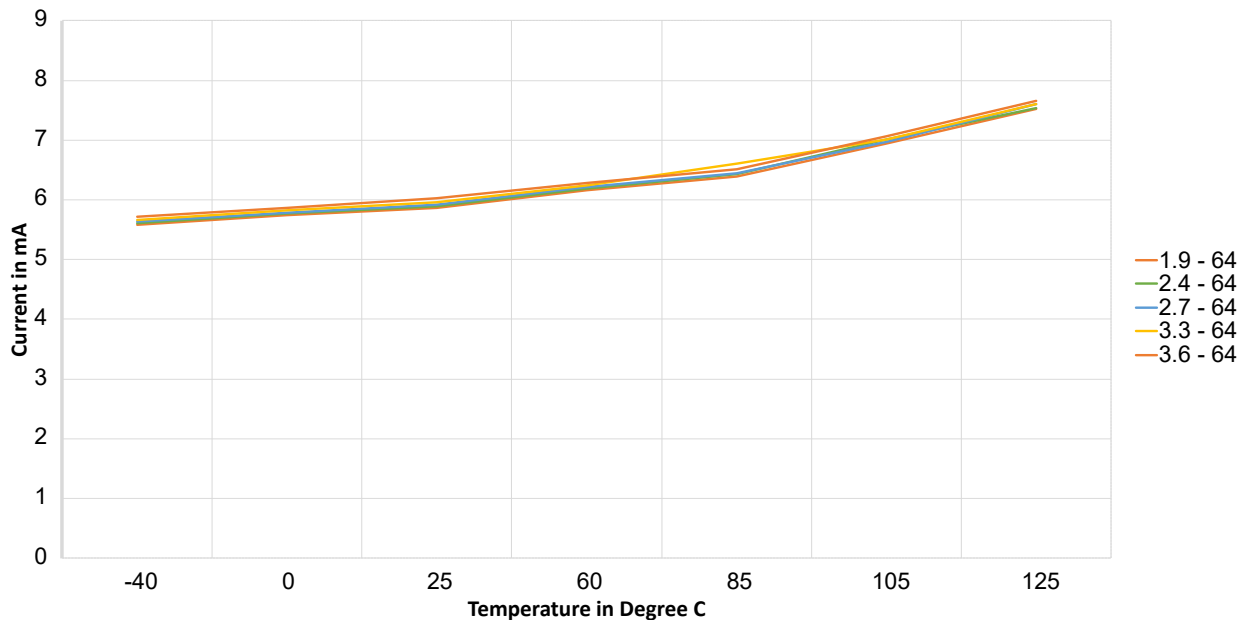


Figure 43-4. Idle Mode Current Consumption in MLDO Mode at PLL 64 MHz



## 43.9 Sleep Current Consumption DC Electrical Specifications (85°C)

Table 43-10. Sleep Current Consumption DC Electrical Specifications

DC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp				
Param. No.	Symbol	Characteristics	$V_{DDIO}$	Typ. <sup>(1)</sup>	Max.	Units	Conditions
SPWR_1	$I_{DD\_SLEEP}$	MCU $I_{DD}$ in Sleep mode w/LDO mode selected	3.3V	0.58	10.6	mA	XTAL = OFF
SPWR_5			3.3V	0.88	10.1	mA	XTAL = ON
SPWR_29		MCU $I_{DD}$ in Sleep mode w/BUCK mode selected	3.3V	0.67	10.7	mA	XTAL = ON
SPWR_33			3.3V	0.49	11.0	mA	XTAL = OFF

.....continued

DC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp				
Param. No.	Symbol	Characteristics	$V_{DDIO}$	Typ. <sup>(1)</sup>	Max.	Units	Conditions
<b>Notes:</b>							
1. Typical value measured during characterization across voltage and temperature.							
2. The test conditions are as follows:							
<ul style="list-style-type: none"> <li>- All GPIO are input and pulled up.</li> <li>- All peripherals disabled with PMD bits.</li> <li>- All PB clocks are divided by 16.</li> <li>- LPRC is set as LPCLK.</li> <li>- SOSOC is disabled.</li> <li>- CLDO is configured at lowest possible voltage (VREG Trim = 0x07).</li> <li>- PMU is configured to the Buck PSM mode on the Sleep Mode Entry.</li> <li>- Cache is enabled and configured wait time as 0xF.</li> <li>- WCM memories configured in Retention + NAP mode.</li> <li>- DSU is disconnected.</li> <li>- RF system is in low power configuration.</li> <li>- Entry to the Sleep mode is enabled and WFI instruction is executed.</li> <li>- On exit by External interrupt, verified RCON status to ensure system was in Sleep mode.</li> <li>- In XTAL ON mode - PMU Buck clock is derived from POSC 16 MHz and scaled to 1 MHz. FRC is OFF.</li> <li>- In XTAL OFF mode - PMU is clocked from FRC and XTAL 16 MHz clock is disabled and clock configuration in CRU changed to FRC.</li> </ul>							
3. These parameters are characterized but not tested in manufacturing.							

## 43.10 Sleep Current Consumption DC Electrical Specifications (125°C)

**Table 43-11.** Sleep Current Consumption DC Electrical Specifications

DC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	$V_{DDIO}$	Typ. <sup>(1)</sup>	Max.	Units	Conditions
SPWR_1	$I_{DD\_SLEEP}$	MCU $I_{DD}$ in Sleep mode w/LDO mode selected	3.3V	0.58	16.3	mA	XTAL = OFF, $T_A = 25^{\circ}\text{C}$
SPWR_5			3.3V	0.88	17.3	mA	XTAL = ON, $T_A = 25^{\circ}\text{C}$
SPWR_29		MCU $I_{DD}$ in Sleep mode w/BUCK mode selected	3.3V	0.67	13.0	mA	XTAL = ON, $T_A = 25^{\circ}\text{C}$
SPWR_33			3.3V	0.49	12.6	mA	XTAL = OFF, $T_A = 25^{\circ}\text{C}$

.....continued

DC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
--------------------	--	--	--	--	--	--	--

Param. No.	Symbol	Characteristics	$V_{DDIO}$	Typ. <sup>(1)</sup>	Max.	Units	Conditions
------------	--------	-----------------	------------	---------------------	------	-------	------------

**Notes:**

- Typical value measured during characterization across voltage and temperature.
- The test conditions are as follows:
  - All GPIO are input and pulled up.
  - All peripherals disabled with PMD bits.
  - All PB clocks are divided by 16.
  - LPRC is set as LPCLK.
  - SOSC is disabled.
  - CLDO is configured at lowest possible voltage (VREG Trim = 0x07).
  - PMU is configured to the Buck PSM mode on the Sleep Mode Entry.
  - Cache is enabled and configured wait time as 0xF.
  - WCM memories configured in Retention + NAP mode.
  - DSU is disconnected.
  - RF system is in low power configuration.
  - Entry to the Sleep mode is enabled and WFI instruction is executed.
  - On exit by External interrupt, verified RCON status to ensure system was in Sleep mode
  - In XTAL ON mode - PMU Buck clock is derived from POSC 16 MHz and scaled to 1 MHz. FRC is OFF.
  - In XTAL OFF mode - PMU is clocked from FRC and XTAL 16 MHz clock is disabled and clock configuration in CRU changed to FRC.
- These parameters are characterized but not tested in manufacturing.

**Figure 43-5.** Sleep Current with XTAL\_OFF\_MLDO

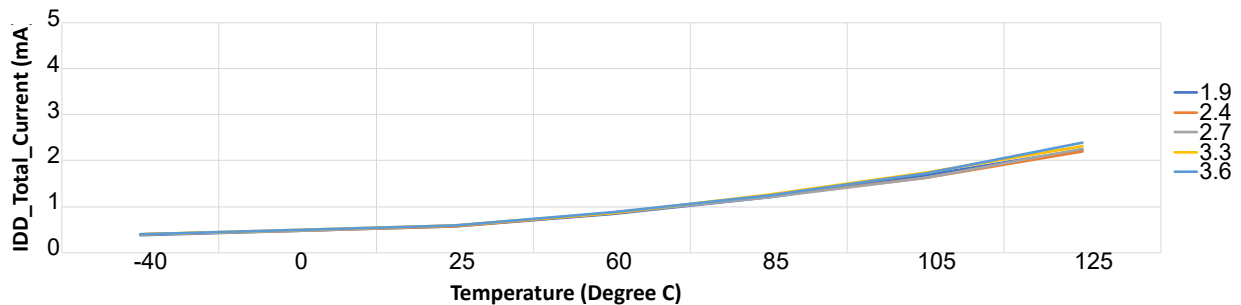


Figure 43-6. Sleep Current with XTAL\_OFF\_DC\_DC

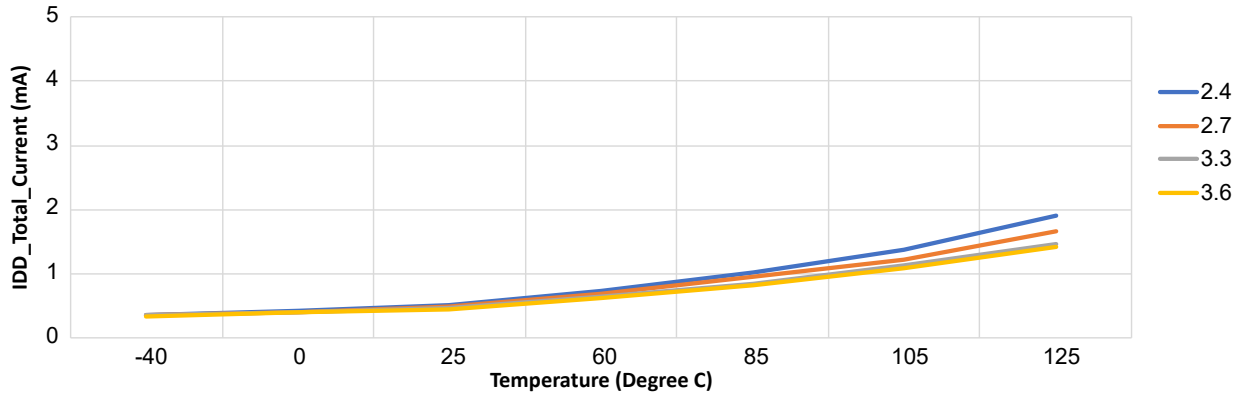


Figure 43-7. Sleep Current with XTAL\_ON\_MLDO

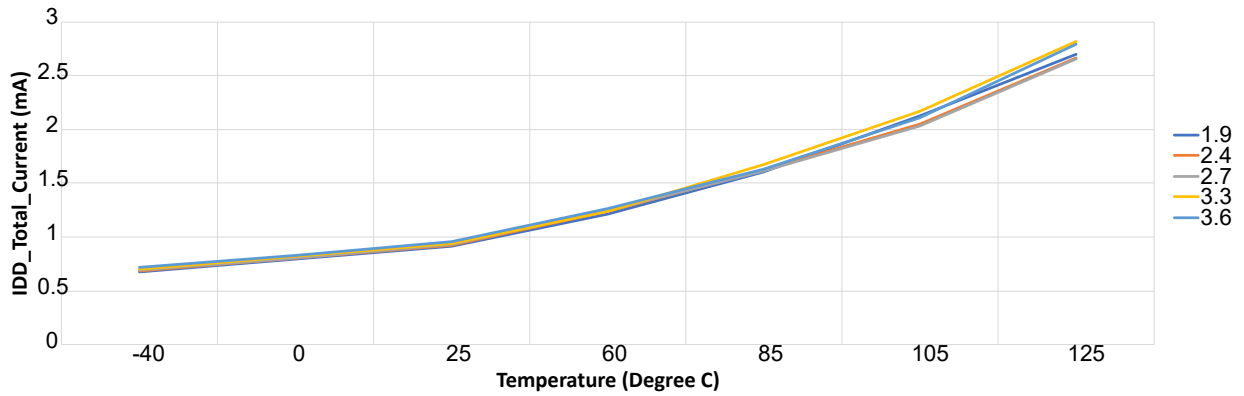
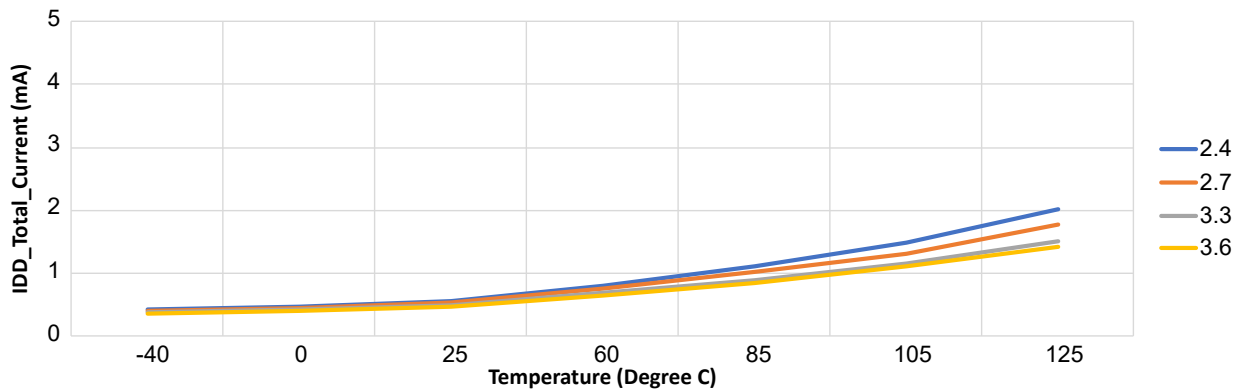


Figure 43-8. Sleep Current with XTAL\_ON\_DC\_DC



## 43.11 Deep Sleep Current Consumption DC Electrical Specifications (85°C)

**Table 43-12.** Deep Sleep Current Consumption DC Electrical Specifications

DC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp				
Param. No.	Symbol	Characteristics	$V_{DDIO}$	Typ. <sup>(1)</sup>	Max.	Units	Conditions
BPWR_1	$I_{DD\_BACKUP}^{(2)}$	MCU $I_{DD}$ in Deep Sleep mode powered from $V_{DDIO}$	3.3V	1.5	7.1	$\mu\text{A}$	No backup RAM retained
BPWR_3			1.9V	1.1	4.3	$\mu\text{A}$	
BPWR_9			3.3V	1.7	11.7	$\mu\text{A}$	8 KB backup RAM retained
BPWR_11			1.9V	1.4	8.8	$\mu\text{A}$	8 KB backup RAM retained

**Notes:**

- Typical value measured during characterization across voltage and temperature.
- The test conditions are as follows:
  - All GPIO are input and pulled up.
  - All peripherals disabled with PMD bits.
  - All PB clocks are divided by 16.
  - LPRC is set as LPCLK.
  - SOSC and POSC is disabled.
  - CLDO configured at lowest possible voltage (VREG Trim = 0x07).
  - DSU is disconnected.
  - RF system is in low power configuration.
  - DSWDT is enabled and configured for wake-up.
  - Deep sleep entry is configured and WFI instruction is executed.
  - 8 KB RAM WCM memory ON using WCMCFG.WCM1CFG[2:0] = 1 ; WCMCFG.WCM2CFG[2:0] = 1 ; For powered ON, RET + NAP Mode.
  - 8 KB RAM WCM memory OFF using WCMCFG.WCM1CFG[2:0] = 0 ; WCMCFG.WCM2CFG[2:0] = 0 ; For powered OFF.
- These parameters are characterized but not tested in manufacturing.

## 43.12 Deep Sleep Current Consumption DC Electrical Specifications (125°C)

**Table 43-13.** Deep Sleep Current Consumption DC Electrical Specifications

DC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	$V_{DDIO}$	Typ. <sup>(1)</sup>	Max.	Units	Conditions
BPWR_1	$I_{DD\_BACKUP}^{(2)}$	MCU $I_{DD}$ in Deep Sleep mode powered from $V_{DDIO}$	3.3V	1.46	18.68	$\mu\text{A}$	No backup RAM retained
BPWR_3			1.9V	1.09	13.12	$\mu\text{A}$	
BPWR_9			3.3V	1.7	30.9	$\mu\text{A}$	8 KB backup RAM retained
BPWR_11			1.9V	1.34	25.17	$\mu\text{A}$	8 KB backup RAM retained

.....continued

DC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	$V_{DDIO}$	Typ. <sup>(1)</sup>	Max.	Units	Conditions

**Notes:**

1. Typical value measured during characterization across voltage and temperature.
2. The test conditions are as follows:
  - All GPIO are input and pulled up.
  - All peripherals disabled with PMD bits.
  - All PB clocks are divided by 16.
  - LPRC is set as LPCLK.
  - SOSC and POSC is disabled.
  - CLDO configured at lowest possible voltage (VREG Trim = 0x07).
  - DSU is disconnected.
  - RF system is in low power configuration.
  - DSWDT is enabled and configured for wake-up.
  - Deep sleep entry is configured and WFI instruction is executed.
  - 8 KB RAM WCM memory ON using WCMCFG.WCM1CFG[2:0] = 1 ; WCMCFG.WCM2CFG[2:0] = 1 ; For powered ON, RET + NAP Mode.
  - 8 KB RAM WCM memory OFF using WCMCFG.WCM1CFG[2:0] = 0 ; WCMCFG.WCM2CFG[2:0] = 0 ; For powered OFF.
3. These parameters are characterized but not tested in manufacturing.

**Figure 43-9. Deep Sleep Current RAM ON**

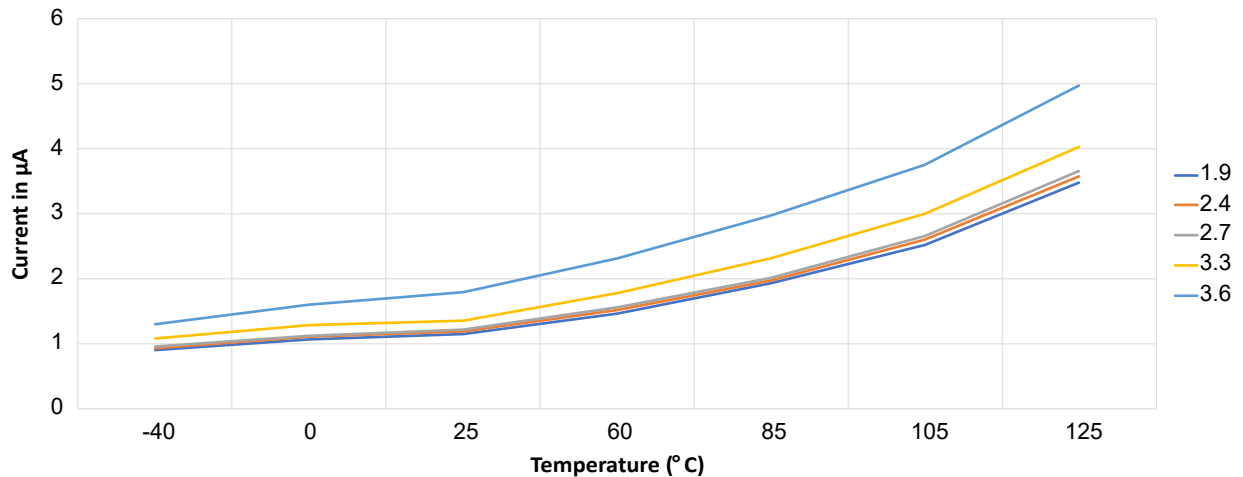
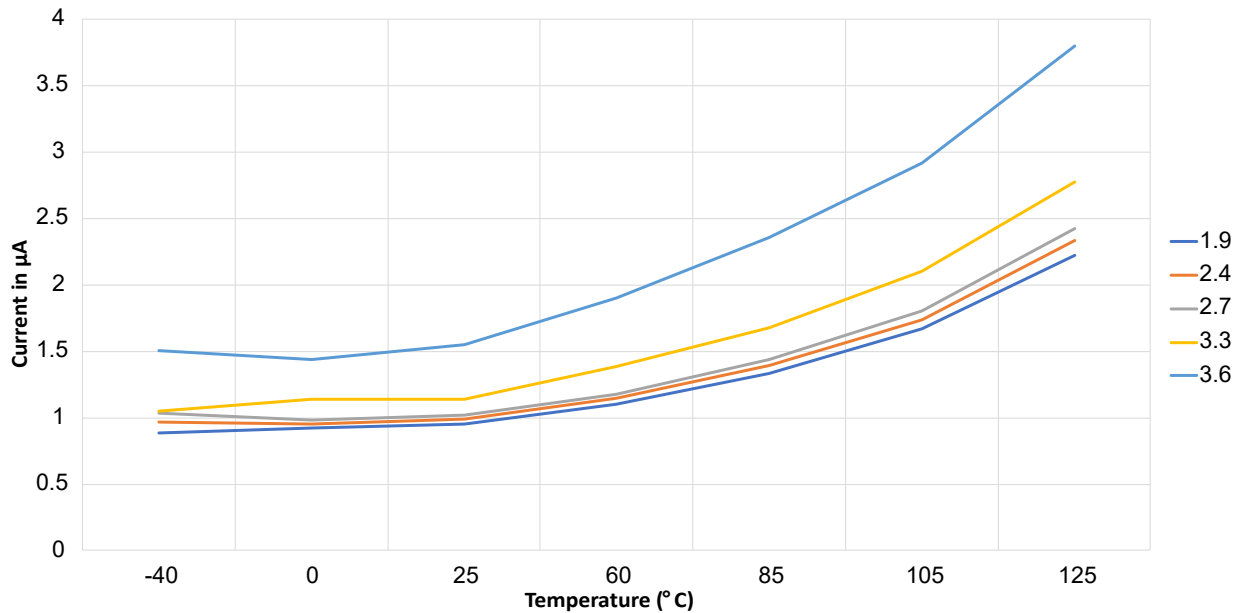


Figure 43-10. Deep Sleep Current RAM OFF



### 43.13 XDS (Extreme Deep Sleep) Current Consumption DC Electrical Specifications (85°C)

Table 43-14. XDS (Extreme Deep Sleep) Current Consumption DC Electrical Specifications

DC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp				
Param. No.	Symbol	Characteristics	$V_{DDIO}$	Typ. <sup>(1)</sup>	Max.	Units	Conditions
OPWR_1	$I_{DD\_OFF}^{(2)}$	MCU $I_{DD}$ in OFF mode powered from $V_{DDIO}$	3.3V	0.36	3.99	µA	In OFF mode, the device is entirely powered-off. <b>Note:</b> This mode is left by pulling the RESET pin low, or when a power Reset is done.
OPWR_3			1.9V	0.11	1.39		

**Notes:**

- Typical value measured during characterization across voltage and temperature.
- The test conditions are as follows:
  - All GPIO are input and pulled up.
  - All peripherals disabled with PMD bits.
  - DSU is disconnected.
  - RF system is in low power configuration.
  - DSWDT is disabled.
  - RTCC and POSC is disabled.
  - Deep sleep entry is configured and WFI instruction is executed.
- These parameters are characterized but not tested in manufacturing.



### 43.14 XDS (Extreme Deep Sleep) Current Consumption DC Electrical Specifications (125°C)

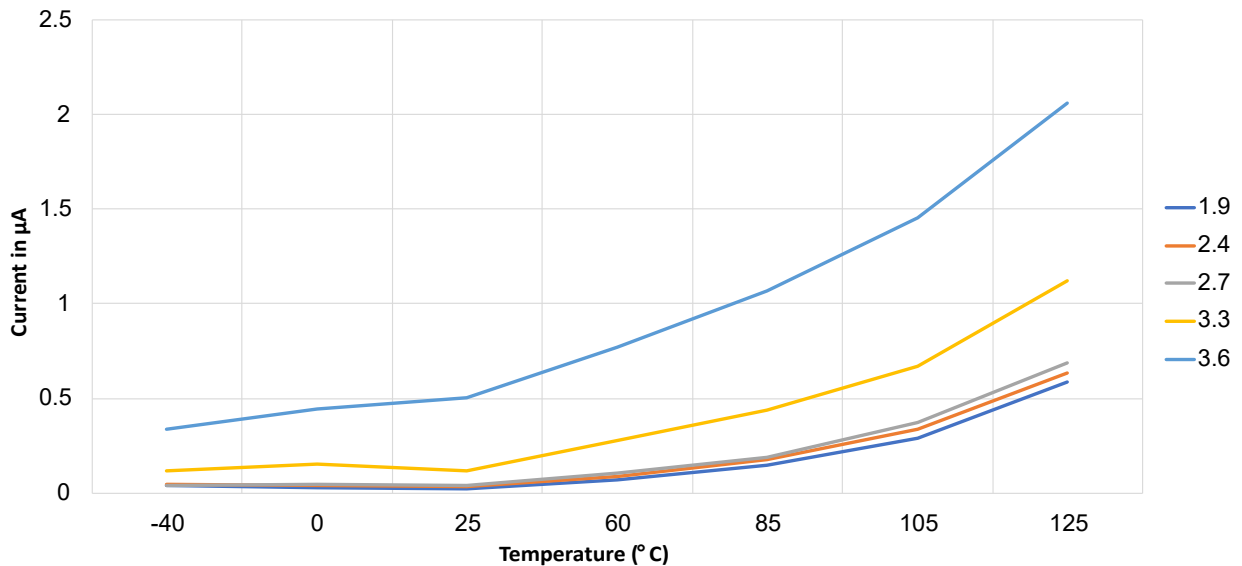
**Table 43-15.** XDS (Extreme Deep Sleep) Current Consumption DC Electrical Specifications

DC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	$V_{DDIO}$	Typ. <sup>(1)</sup>	Max.	Units	Conditions
OPWR_1	$I_{DD\_OFF}^{(2)}$	MCU $I_{DD}$ in OFF mode powered from $V_{DDIO}$	3.3V	0.36	12.8	$\mu\text{A}$	In OFF mode, the device is entirely powered-off. <b>Note:</b> This mode is left by pulling the RESET pin low, or when a power Reset is done.
OPWR_3			1.9V	0.11	7.66		

**Notes:**

- Typical value measured during characterization across voltage and temperature.
- The test conditions are as follows:
  - All GPIO are input and pulled up.
  - All peripherals disabled with PMD bits.
  - DSU is disconnected.
  - RF system is in low power configuration.
  - DSWDT is disabled.
  - RTCC and POSC is disabled.
  - Deep sleep entry is configured and WFI instruction is executed.
- These parameters are characterized but not tested in manufacturing.

**Figure 43-11.** Extreme Deep Sleep Current



## 43.15 I/O PIN AC/DC Electrical Specifications

Table 43-16. I/O PIN AC/DC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA} 1.9-3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial Temp $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ. <sup>(1)</sup>	Max.	Units	Conditions
DI_1	V <sub>OL</sub>	Output voltage low (Drive strength, 8x)	—	0.25	0.53	V	V <sub>DDIO</sub> = 3.3V at I <sub>OL</sub> = 10 mA
		Output voltage low (Drive strength, 4x)	—	0.32	0.57	V	V <sub>DDIO</sub> = 3.3V at I <sub>OL</sub> = 10 mA
DI_2	V <sub>OL</sub>	Output voltage low (Drive strength, 8x)	—	0.29	0.61	V	V <sub>DDIO</sub> = 3.3V at I <sub>OL</sub> = 13 mA
		Output voltage low (Drive strength, 4x)	—	0.40	0.72	V	V <sub>DDIO</sub> = 3.3V at I <sub>OL</sub> = 13 mA
DI_3	V <sub>OL</sub>	Output voltage low (Drive strength, 8x)	—	0.32	0.67	V	V <sub>DDIO</sub> = 3.3V at I <sub>OL</sub> = 15 mA
DI_4	V <sub>OH</sub>	Output voltage low (Drive strength, 8x)	2.67	3.09	3.50	V	V <sub>DDIO</sub> = 3.3V at I <sub>OH</sub> = 5 mA
		Output voltage low (Drive strength, 4x)	2.76	3.05	3.33	V	V <sub>DDIO</sub> = 3.3V at I <sub>OH</sub> = 5 mA
DI_5	V <sub>OH</sub>	Output voltage low (Drive strength, 8x)	2.54	3.04	3.54	V	V <sub>DDIO</sub> = 3.3V at I <sub>OH</sub> = 7 mA
		Output voltage low (Drive strength, 4x)	2.54	2.95	3.36	V	V <sub>DDIO</sub> = 3.3V at I <sub>OH</sub> = 7 mA
DI_6	V <sub>OH</sub>	Output voltage High (Drive strength, 8x)	2.319	2.96	3.598	V	V <sub>DDIO</sub> = 3.3V at I <sub>OH</sub> = 10 mA
DI_7	V <sub>IL</sub>	Input voltage low (Drive strength, 8x)	1.35	1.46	1.56	V	V <sub>DDIO</sub> = 3.3V
		Input voltage low (Drive strength, 4x)	1.20	1.34	1.46	V	V <sub>DDIO</sub> = 3.3V
DI_8	V <sub>IH</sub>	Input voltage high (Drive strength, 8x)	1.78	1.91	2.07	V	V <sub>DDIO</sub> = 3.3V
		Input voltage high (Drive strength, 4x)	1.85	1.95	2.06	V	V <sub>DDIO</sub> = 3.3V
DI_13	I <sub>IL</sub>	Input pin leakage current	—	71.8	—	nA	With drive strength, 8x
DI_14	I <sub>IL</sub>	Input pin leakage current	—	35.3	—	nA	With drive strength, 4x
DI_15	R <sub>PDWN</sub>	Internal pull-down (Drive strength, 8x)	20.6	21.2	22	k $\Omega$	V <sub>DDIO</sub> = 3.3V
		Internal pull-down (Drive strength, 4x)	—	3.9	—	k $\Omega$	
DI_17	R <sub>PUP</sub>	Internal pull-up (Drive strength, 8x)	—	18.6	—	k $\Omega$	V <sub>DDIO</sub> = 3.3V
		Internal pull-up (Drive strength, 4x)	—	3.3	—	k $\Omega$	
DI_19	I <sub>ICL</sub>	Input low injection current	0	—	-5	ma	This parameter applies to all I/O pins. Except AV <sub>DD</sub> , MCLR <sup>(1,4)</sup>
DI_21	I <sub>ICH</sub>	Input high injection current	0	—	5	ma	This parameter applies to all pins, with the exception of 5V tolerant I/O pins, AV <sub>DD</sub> , MCLR <sup>(2,3,4)</sup>
DI_23	$\Sigma I_{ICT}$	Total input injection current (sum of all I/O and control pins) Absolute value of $ \Sigma I_{ICT} $	-20	—	20	ma	Absolute instantaneous sum of all $\pm$ input injection currents from all I/O pins. ( $ I_{ICL}  +  I_{ICH} $ ) $\leq \Sigma I_{ICT}$
DI_25	T <sub>RISE</sub>	I/O pin rise time (High drive strength, high load)	—	2.0	7.5	ns	V <sub>DDIO</sub> = 3.3V, C <sub>LOAD</sub> , typical = 50 pF <sub>(MIN)</sub>
		I/O pin rise time (Low drive strength, high load)	—	10.7	19.1	ns	—
		I/O pin rise time (High drive strength, standard load)	—	2.0	5.1	ns	—
		I/O pin rise time (Low drive strength, standard load)	—	7.7	13.0	ns	—

.....continued

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA} 1.9-3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial Temp $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ. <sup>(1)</sup>	Max.	Units	Conditions
DI_27	$T_{FALL}$	I/O pin fall time (High drive strength, high load)	—	1.6	7.07	ns	—
		I/O pin fall time (Low drive strength, high load)	—	8.1	14.8	ns	—
		I/O pin fall time (High drive strength, standard load)	—	1.6	4.98	ns	—
		I/O pin fall time (Low drive strength, standard load)	—	5.1	9.9	ns	—
<b>Notes:</b>							
1. $V_{IL}$ source < (GND - 0.3). Characterized but not tested.							
2. $V_{IH}$ source > ( $V_{DDIO} + 0.3$ ) for non-5V tolerant pins only.							
3. Digital 5V tolerant pins do not have an internal high side diode to $V_{DDIO}$ , and therefore, cannot tolerate any positive input injection current.							
4. Any number and/or combination of I/O pins not excluded under $I_{ICL}$ or $I_{ICH}$ conditions are permitted provided the absolute instantaneous sum of the input injection currents from all pins do not exceed the specified $\sum I_{ICT}$ limit. To limit the injection current, the user must insert a resistor in series $R_{SERIES}$ (in other words, $R_S$ ), between the input source voltage and device pin. The resistor value is calculated according to: <ul style="list-style-type: none"> <li>- For negative Input voltages less than (GND - 0.3): <math>R_S \geq \text{absolute value of }  (V_{IL} \text{ source} - (GND - 0.3)) / I_{ICL} </math></li> <li>- For positive input voltages greater than (<math>V_{DDIO} + 0.3</math>): <math>R_S \geq (V_{IH} \text{ source} - (V_{DDIO} + 0.3)) / I_{ICH}</math></li> <li>- For Vpin voltages greater than <math>V_{DDIO} + 0.3</math> and less than GND - 0.3: <math>R_S = \text{the larger of the values calculated above}</math></li> </ul>							
5. High load = 50 pF and Standard load = 30 pF.							

## 43.16 External XTAL and Clock AC Electrical Specifications

**Table 43-17.** External XTAL and Clock AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA} 1.9V \text{ to } 3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial Temp $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions <sup>(1)</sup>
XOSC_1	$F_{OSC\_XOSC}$	XOSC crystal frequency	—	16	—	MHz	$X_{IN}, X_{OUT}$ primary oscillator
XOSC_1A	TOSC	$TOSC = 1/FOSC\_XOSC$	—	0.0625	—	$\mu s$	See parameter XOSC1 for FOSC_XOSC value
XOSC_2	$XOSC\_S T^{(2)}$	XOSC crystal start-up time	—	1.25	2.5	ms	Crystal stabilization time only not oscillator ready
XOSC_3	$C_{XIN}$	XOSC $X_{IN}$ parasitic pin capacitance	—	0.35	—	pF	With default crystal trim settings
XOSC_5	$C_{XOUT}$	XOSC $X_{OUT}$ parasitic pin capacitance	—	0.35	—	pF	With default crystal trim settings
XOSC_11	$C_{LOAD}^{(3)}$	XOSC crystal FOSC = 16 MHz	—	9	—	pF	—
XOSC_21	ESR	XOSC crystal FOSC = 16 MHz	—	—	100	$\Omega$	—
XOSC_33	$D_{LEVEL}$	MCU crystal oscillator power drive level	—	—	100	$\mu W$	—

.....continued

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA} 1.9V$ to $3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial Temp $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions <sup>(1)</sup>

**Notes:**

- $V_{DDIO} = V_{DDANA} = 3.3V$ .
- This is for guidance only. A major component of crystal start-up time is based on the second party crystal MFG parasitic that are outside the scope of this specification.
- The test conditions for crystal load capacitor calculation are as follows:
  - Standard PCB trace capacitance = 1.5 pF per 12.5 mm (0.5 inches) (in other words, PCB STD TRACE W = 0.175 mm, H = 36  $\mu$ m, T = 113  $\mu$ m).
  - Xtal PCB capacitance typical; therefore,  $\sim$  2.5 pF for a tight PCB xtal layout.
  - For CXIN and CXOUT within 4 pF of each other, assume  $CXTAL\_EFF = ((CXIN+CXOUT) / 2)$ .
  - Note:** Averaging CXIN and CXOUT will affect the final calculated CLOAD value by less than 0.25 pF.

**Equation 43-1. Equation 1:**

$$MFG \ CLOAD \ Spec = \{([CXIN + C1] * [CXOUT + C2]) / [CXIN + C1 + C2 + CXOUT]\} + \text{estimated oscillator PCB stray capacitance}$$

Assuming  $C1 = C2$  and  $CXin \sim CXout$ , the formula can be further simplified and restated to solve for  $C1$  and  $C2$  by:

**Equation 43-2. Equation 2 (In other words: Simplified Equation 1)**

$$C1 = C2 = ((2 * MFG \ CLOAD \ Spec) - CXTAL\_EFF - (2 * PCB \ capacitance))$$

Example:

- XTAL Mfg CLOAD Data Sheet Spec = 12 pF
- PCB XTAL trace Capacitance = 2.5 pF
- CXIN pin = 6.5 pF, CXOUT pin = 4.5 pF; therefore,  $CXTAL\_EFF = ((CXIN+CXOUT) / 2)$

$$CXTAL\_EFF = ((6.5 + 4.5) / 2) = 5.5 \text{ pF}$$

$$C1 = C2 = ((2 * MFG \ Cload \ spec) - CXTAL\_EFF - (2 * PCB \ capacitance))$$

$$C1 = C2 = (24 - 5.5 - (2 * 2.5))$$

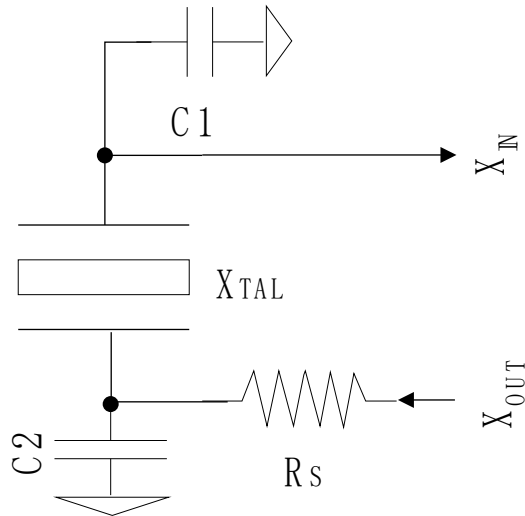
$$C1 = C2 = 13.5 \text{ pF (Always rounded down)}$$

$$C1 = C2 = 13 \text{ pF (in other words, for hypothetical example crystal external load capacitors)}$$

$$\text{User } C1 = C2 = 13 \text{ pF CLOAD (max) spec}$$

- Maximum start-up time user selectable in XOSCCTRL.STARTUP.
- These parameters are characterized but not tested in manufacturing.

Figure 43-12. External XTAL and Clock Diagram



### 43.17 XOSC32 AC Electrical Specifications

Table 43-18. XOSC32 AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions <sup>(1)</sup>
XOSC32_1	$F_{OSC\_XOSC32}$	XOSC32 oscillator crystal frequency	—	32.79	—	kHz	XIN32, XOUT32 secondary oscillator
XOSC32_3	$C_{XIN32}$	XOSC32 XIN32 parasitic pin capacitance	—	0.4 2.4	—	pF	0.4 pF at the SOC pins and 2.4 pF on the module
XOSC32_5	$C_{XOUT32}$	XOSC32 XOUT32 parasitic pin capacitance	—	0.4 2.4	—	pF	0.4 pF at the SOC pins and 2.4 pF on the module
XOSC32_11	$C_{LOAD\_X32}^{(3)}$	32.768 kHz crystal load capacitance	—	11	—	pF	—
XOSC32_12			—	—	—	pF	—
XOSC32_13	$ESR\_X32$	32.768 kHz crystal ESR	—	75	100	K $\Omega$	—
XOSC32_14			—	—	—	$\Omega$	—
XOSC32_15	TOSC32	$TOSC32 = 1/FOSC\_XOSC32$	—	30.5	—	$\mu\text{s}$	See parameter XOSC32_1 for FOSC_XOSC32 value
XOSC32_17	$XOSC32\_ST^{(2)}$	XOSC32 crystal start-up time	—	1024	—	TOSC	Crystal stabilization time only not oscillator ready

.....continued

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions <sup>(1)</sup>

**Notes:**

- $V_{DDIO} = V_{DDANA} = 3.3\text{V}$ .
- This is for guidance only. A major component of crystal start-up time is based on the second party crystal MFG parasitic that is outside the scope of this specification. If this is a major concern, the customer might need to characterize this based on their design choices.
- The test conditions for crystal load capacitor calculation are as follows:
  - Standard PCB trace capacitance = 1.5 pF per 12.5 mm (0.5 inches) (in other words, PCB STD TRACE W = 0.175 mm, H = 36  $\mu\text{m}$ , T = 113  $\mu\text{m}$ )
  - Xtal PCB capacitance typical; therefore,  $\approx 2.5$  pF for a tight PCB xtal layout
  - For CXIN and CXOUT within 4 pF of each other, assume  $\text{CXTAL\_EFF} = ((\text{CXIN} / 2))$
  - Note:** Averaging CXIN and CXOUT will affect the final calculated CLOAD value by less than the tolerance of the capacitor selection.

**Equation 1:**

$\text{MFG CLOAD Spec} = \{([\text{CXIN} + \text{C1}] * [\text{CXOUT} + \text{C2}]) / [\text{CXIN} + \text{C1} + \text{C2} + \text{CXOUT}]\} + \text{estimated oscillator PCB stray capacitance}$

Assuming  $\text{C1} = \text{C2}$  and  $\text{CXin} \approx \text{CXout}$ , the formula can be further simplified and restated to solve for C1 and C2 by:

**Equation 43-3. Equation 2 (In other words: Simplified Equation 1)**

$$\text{C1} = \text{C2} = ((2 * \text{MFG CLOAD Spec}) - \text{CXTAL\_EFF} - (2 * \text{PCB capacitance}))$$

Example:

- XTAL Mfg CLOAD Data Sheet Spec = 12 pF
- PCB XTAL trace Capacitance = 2.5 pF
- CXIN pin = 6.5 pF, CXOUT pin = 4.5 pF therefore  $\text{CXTAL\_EFF} = ((\text{CXIN} / 2))$

$$\text{CXTAL\_EFF} = ((6.5 + 4.5) / 2) = 5.5 \text{ pF}$$

$$\text{C1} = \text{C2} = ((2 * \text{MFG Cload spec}) - \text{CXTAL\_EFF} - (2 * \text{PCB capacitance}))$$

$$\text{C1} = \text{C2} = (24 - 5.5 - (2 * 2.5))$$

$$\text{C1} = \text{C2} = (24 - 5.5 - 5)$$

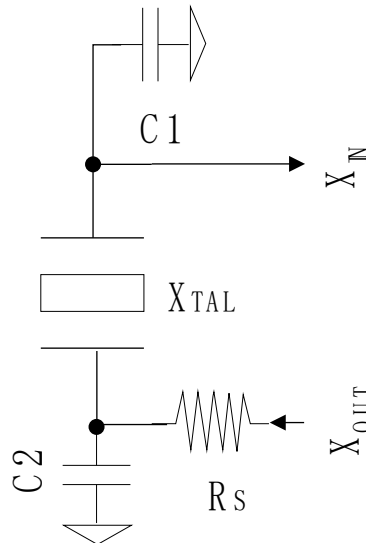
$$\text{C1} = \text{C2} = 13.5 \text{ pF (Always rounded down)}$$

$$\text{C1} = \text{C2} = 13 \text{ pF (in other words, for hypothetical example crystal external load capacitors)}$$

$$\text{User C1} = \text{C2} = 13 \text{ pF} \leq \text{CLOAD\_X32 (max.) spec}$$

- User selectable in OSC32KCTRL.STARTUP.
- These parameters are characterized but not tested in manufacturing.

Figure 43-13. XOSC32 Block Diagram



## 43.18 Low Power Internal 32 kHz RC Oscillator AC Electrical Specifications

Table 43-19. Low Power Internal 32 kHz RC Oscillator AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
LP32K_1	FOSC_LPRC32K	Output frequency	23.5	—	40.3	kHz	$V_{DDIO} = V_{DDANA} \geq V_{DDANA}(\text{min})$ $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$
LP32K_2	FOSC_LPRC32K	Output frequency	26.1	32.7	36.6	kHz	$V_{DDIO} = V_{DDANA} \geq V_{DDANA}(\text{min})$ $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ (in other words, Factory default calibration)
LP32K_9	RC32K_Duty	LPRC32K OSC duty cycle	—	50	—	%	$V_{DDIO} = V_{DDANA} \geq V_{DDANA}(\text{min})$

**Note:** These parameters are characterized but not tested in manufacturing.

## 43.19 FRC AC Electrical Specifications

Table 43-20. FRC AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
FRC1	FINTFREQ	Internal FRC frequency	—	8	—	MHz	Factory calibrated with tune bits set to 0x0
FRC2	TSUFRC	Start-up time of internal FRC	—	15	—	$\mu\text{s}$	—
FRC3	FACCU	FRC accuracy	—	5	—	%	—
FRC5	DUTY_CYCLE	FRC duty cycle	45	50	55	%	—

.....continued

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
FRC6	C_USE	FRC user tune	-1.625	6.299	1.615	%	Frequency drift = [ (Maximum frequency measured - Minimum frequency measured)/(Default frequency of 8 MHz) ] * 100; Maximum frequency drift possible by using the frequency trim bits
FRC7	C_USER_STEP_SIZE	FRC user tune step size	0.0467	0.046	0.0472	%	Step size = (% Drift across Osc tune)/(Total number of trim bit combinations); Change in frequency with incremental trim bit

**Note:** These parameters are characterized but not tested in manufacturing.

## 43.20 Frequency AC Electrical Specifications

**Table 43-21.** Frequency AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
MCU_1	FCY	MCU frequency of operation	—	—	64	MHz	$V_{DDIO(\min)}$ to $V_{DDIO(\max)}$
MCU_3	TCY	MCU clock period	1/FCY			ns	

**Note:** These parameters are characterized but not tested in manufacturing.

## PLL ( Phase Locked Loop) AC Electrical Specifications

**Table 43-22.** PLL MHz (Phase Locked Loop)

AC Characteristics FDPLLxxMHz (Fractional Digital Phase Locked Loop)			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
PLL_1	PLL_F <sub>IN</sub>	PLL input frequency	—	16	—	MHz	Over full voltage and temperature operating ranges
PLL_3	PLL_F <sub>OUT</sub>	PLL output clock frequency	64	96	96	MHz	—
PLL_4	PLL_VCO_FREQ	PLL VCO frequency	—	480	—	MHz	—

**Notes:**

- PLL input clock is 16 MHz XTAL.
- These parameters are characterized but not tested in manufacturing.



## 43.21 ADC Electrical Specifications

**Table 43-23.** ADC AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
<b>Device Supply</b>							
ADC_1	$V_{DDANA}$	ADC module supply	$V_{DDANA(\text{min})}$	—	$V_{DDANA(\text{max})}$	V	$V_{DDIO} = V_{DDANA}$
<b>Reference Inputs</b>							
ADC_3	$V_{\text{REF}}^{(4)}$	Reference voltage high (external reference buffers)	—	—	$V_{DDANA}$	V	ADC reference voltage
<b>Analog Input Range</b>							
ADC_7	$A_{\text{FS}}$	Full-scale analog input signal range (Single-ended)	0	—	$V_{DDANA}$	V	$V_{\text{REF}} = V_{DDANA(\text{max})}$

**Table 43-24.** ADC Single-Ended Mode AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
<b>Single-Ended Mode ADC Accuracy</b>							
SADC_11	Res	Resolution	6	—	12	bits	Selectable 8, 10, 12 bit resolution ranges
SADC_13	$\text{ENOB}^{(3)}$	Effective number of bits	6.081	9.849	10.795	bits	2 Msps, Internal $V_{\text{REF}}$ , $V_{DDANA} = V_{DDIO} = 3.3\text{V}$
TBD							
SADC_19	$\text{INL}^{(3)}$	Integral non linearity	-2.94	-1.18	1.96	LSb	2 Msps, Internal $V_{\text{REF}}$ , $V_{DDANA} = V_{DDIO} = 3.3\text{V}$
SADC_25	$\text{DNL}^{(3)}$	Differential non linearity	-1.42	1.2977	2.59	LSb	2 Msps, Internal $V_{\text{REF}}$ , $V_{DDANA} = V_{DDIO} = 3.3\text{V}$
SADC_31	$\text{GERR}^{(3)}$	Gain error	11.90	18.93	25.00	LSb	2 Msps, Internal $V_{\text{REF}}$ , $V_{DDANA} = V_{DDIO} = 3.3\text{V}$
SADC_37	$\text{E0FF}^{(3)}$	Offset error	-8.0837	-4.001	-0.3793	LSb	2 Msps, Internal $V_{\text{REF}}$ , $V_{DDANA} = V_{DDIO} = 3.3\text{V}$
<b>Single-Ended Mode ADC Dynamic Performance</b>							
SADC_49	$\text{SINAD}^{(1,2,3)}$	Signal to noise and distortion	38.4	61.1	66.8	dB	$V_{\text{REF}} = V_{DDANA} = V_{DDIO} = 3.3\text{V}$ at 12-bit resolution, max sampling rate <sup>(1,2)</sup>
SADC_51	$\text{SNR}^{(1,2,3)}$	Signal to noise ratio	38.4	61.8	67.3		
SADC_53	$\text{SFDR}^{(1,2,3)}$	Spurious free dynamic range	61.5	70.3	81.3		
SADC_55	$\text{THD}^{(1,2,3)}$	Total harmonic distortion	-84.0	-70.1	-59.8		

.....continued

AC Characteristics		Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp					
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
<b>Notes:</b>							
1. Characterized with an analog input sine wave = (FTP(max)/100). Example: FTP(max) = 1 Msps/100 = 10 kHz sine wave.							
2. Sine wave peak amplitude = 96% ADC_ Full Scale amplitude input with 12-bit resolution.							
3. Spec values collected under the following additional conditions:							
a. 12-bit resolution mode							
b. All registers at reset default value otherwise not mentioned							
4. ADC measurements done with 3.3V $V_{REF}$ voltage.							
5. Value taken over 7 harmonics.							
6. Referred to as AVDD in the pinout.							

**Table 43-25.** ADC Conversion AC Electrical Requirements

AC Characteristics		Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp					
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
<b>ADC Clock Requirements</b>							
ADC_57	TAD	ADC clock period	—	20.83	—	ns	$V_{REF} = V_{DDANA} = 3.3\text{V}$
<b>ADC Single-Ended Throughput Rates</b>							
ADC_59	$F_{TPR}$ (Single-ended mode)	Throughput rate <sup>(3)</sup> (Single-ended)	0.01	—	2	Msp	12-bit resolution, DIV_SHR = 2
<b>Note:</b>							
1. Throughput Rate = $1/((\text{SAMC\_SHR} + 15) * \text{TADX})$ TADX = $1/(\text{ADC Control Clock} / \text{DIV\_SHR})$ , SAMC_SHR = sampling rate							

**Table 43-26.** ADC Sample AC Electrical Requirements

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp							
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions			
ADC_63	$T_{\text{SAMP}}$	ADC sample time <sup>(1,2,3,4)</sup>	1 <sup>(1,4)</sup>	—	—	TAD	12-bit TAD(min), Ext Analog Input Rsource $\leq 147\Omega$			
							10-bit TAD(min), Ext Analog Input Rsource $\leq 504\Omega$			
							8-bit TAD(min), Ext Analog Input Rsource $\leq 1,000\Omega$			
							2 <sup>(1,4)</sup>	12-bit TAD(min), Ext Analog Input Rsource $\leq 2,272\Omega$		
								10-bit TAD(min), Ext Analog Input Rsource $\leq 3,008\Omega$		
								8-bit TAD(min), Ext Analog Input Rsource $\leq 4,000\Omega$		
			3 <sup>(1,4)</sup>	12-bit TAD(min), Ext Analog Input Rsource $\leq 4,416\Omega$						
				10-bit TAD(min), Ext Analog Input Rsource $\leq 5,504\Omega$						
				8-bit TAD(min), Ext Analog Input Rsource $\leq 6,976\Omega$						
			4 <sup>(1,2,4)</sup>	12-bit TAD(min), Ext Analog Input Rsource $\leq 6,560\Omega$						
				10-bit TAD(min), Ext Analog Input Rsource $\leq 8,000\Omega$						
				8-bit TAD(min), Ext Analog Input Rsource $\leq 9,984\Omega$						
			5 <sup>(1,4)</sup>	12-bit TAD(min), Ext Analog Input Rsource $\leq 8,704\Omega$						
				10-bit TAD(min), Ext Analog Input Rsource $\leq 10,496\Omega$						
				8-bit TAD(min), Ext Analog Input Rsource $\leq 12,992\Omega$						
			6 <sup>(1,4,5)</sup>	12-bit TAD(min), Ext Analog Input Rsource $\leq 10,880\Omega$						
				10-bit TAD(min), Ext Analog Input Rsource $\leq 12,992\Omega$						
				8-bit TAD(min), Ext Analog Input Rsource $\leq 16,000\Omega$						
			ADC_65	$T_{\text{CNV}}$	Conversion time <sup>(3)</sup> (Single-Ended mode)	14			TAD	12-bit resolution
						12				10-bit resolution
						10				8-bit resolution
			ADC_69	$C_{\text{SAMPLE}}$	ADC internal sample cap	4	5	6	pf	—
			ADC_71	$R_{\text{SAMPLE}}$	ADC internal impedance	—	—	200	$\Omega$	—

.....continued

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
<b>Notes:</b>							
1. When SAMPCTRL.OFFCOMP = 0:							
– $TSAMP = (((RSAMPLE + RSOURCE) * CSAMPLE * (\#Bits Resolution + 2) * \ln(2))/TAD) + 1$ rounded down to nearest whole integer							
– User SAMPCTRL.SAMPLEN = (TSAMP – 1)							
2. When SAMPCTRL.OFFCOMP = 1:							
– TSAMP = 4 (Forced by HDW)							
– User SAMPCTRL.SAMPLEN = (n/a, Ignored by HDW)							
3. ADC Throughput Rate FTP = $((1/((TSAMP + TCNV) * TAD))/(\# \text{ of user active analog inputs in use on specific target ADC module}))$							
<b>Note:</b> Specification values assume only one AINx channel in use.							
4. $TSAMP \geq (\text{INT}(((RSAMPLE + RSOURCE) * CSAMPLE * (\#Bits Resolution + 2) * \ln(2))/TAD) + 1)$ .							
5. For RSOURCE values exceeding TSAMP = 6 sample time condition, use the formula in note 5 to calculate.							

## 43.22 Comparator AC Electrical Specifications

**Table 43-27.** Comparator AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ. <sup>(1)</sup>	Max.	Units	Conditions
CMP_1	VIOFF_00	Input offset voltage	-16	—	16	mV	$V_{DDANA} = V_{DDIO}$ (min-to-max)
CMP_3	VICM	Input Common mode voltage	0	—	$AV_{DD}$	V	$V_{DDANA} = V_{DDIO}$ (min-to-max)
CMP_4	VIN	Input voltage range	$AV_{SS}$	—	$AV_{DD}$	V	With respect to GND and $V_{DDANA}$
CMP_15	TRESP <sub>SS</sub>	Small signal response time	—	100	160	ns	$V_{DDANA} = V_{DDIO}$ (min-to-max), Comparator ref voltage = $V_{DDANA}/2$ , Input overdrive = $\pm 100$ mV
CMP_19	C <sub>OUTVAL</sub>	Comparator enabled to output valid	—	31	—	$\mu\text{s}$	Comparator module is configured before enabling it
CMP_21	AC <sub>IREF</sub>	Comparator internal band gap voltage reference	1.164	—	1.236	V	$V_{DDANA} = 3.3\text{V}$ , $T = 25^{\circ}\text{C}$ ( $\pm 3\%$ ) - Uncalibrated
CMP_22			1.194	—	1.206	V	$V_{DDANA} = 3.3\text{V}$ , $T = 25^{\circ}\text{C}$ ( $\pm 0.5\%$ ) - Calibrated
CMP_23	C <sub>VREFRNG</sub>	Comparator voltage reference input range	$AV_{SS}$	—	$AV_{DD}$	V	—
CMP_25	F <sub>GCLK_AC</sub>	Analog comparator peripheral module clock freq	—	—	64	MHz	$V_{DDANA} = V_{DDIO}$ (min-to-max)

.....continued

AC Characteristics		Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp					
Param. No.	Symbol	Characteristics	Min.	Typ. <sup>(1)</sup>	Max.	Units	Conditions
<b>Notes:</b>							
1. These parameters are characterized but not tested in manufacturing.							
2. Values in typical column area taken at 25°C.							
3. Comparator Ref voltage cannot exceed: $(V_{IN}(\text{max}) - V_{IOFF}(\text{max}) - \text{CMP}_5(\text{max}) - 50 \text{ mV}) \geq \text{CMP VREF} \geq (V_{IN}(\text{min}) + V_{IOFF}(\text{min}) + \text{CMP}_5(\text{max}) + 50 \text{ mV})$							

## 43.23 SPI Module Electrical Specifications

**Table 43-28.** SPI Module Host Mode Electrical Specifications

AC Characteristics		Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9-3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp					
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
MSP_1	FSCK	SCK frequency	—	—	32	MHz	Fixed pins
			—	—	24		Remappable pins
MSP_2	TSCK	SCK time period	31.25	—	—	μs	Fixed pins
			41.66	—	—		Remappable pins
MSP_3	TSCL	SCK output low time	—	tsck/2	—	ns	—
MSP_5	TSCH	SCK output high time	—	tsck/2	—	ns	—
MSP_7	TSCF	SCK and MOSI output fall time	—	—	8.0	ns	$C_{LOAD} = 50 \text{ pF}$ See parameter DI27 I/O spec
			—	—	6.0		$C_{LOAD} = 20 \text{ pF}$ See parameter DI27 I/O spec
MSP_9	TSCR	SCK and MOSI output rise time	—	—	8.0	ns	$C_{LOAD} = 50 \text{ pF}$ See parameter DI27 I/O spec
			—	—	6.0		$C_{LOAD} = 20 \text{ pF}$ See parameter DI27 I/O spec
MSP_11	TMOV	MOSI Data output valid after SCK	—	11	17	ns	$V_{DDIO}(\text{Min}), C_{LOAD} = 30 \text{ pF}(\text{MIN})$
MSP_13	TMOH	MOSI hold after SCK	5	—	—	ns	
MSP_15	TMIS	MISO setup time of data input to SCK	5	—	—	ns	
MSP_17	TMIH	MISO hold time of data input to SCK	5	—	—	ns	
			5	—	—		

Figure 43-14. SPI Host Module CPHA=0 Timing Diagrams

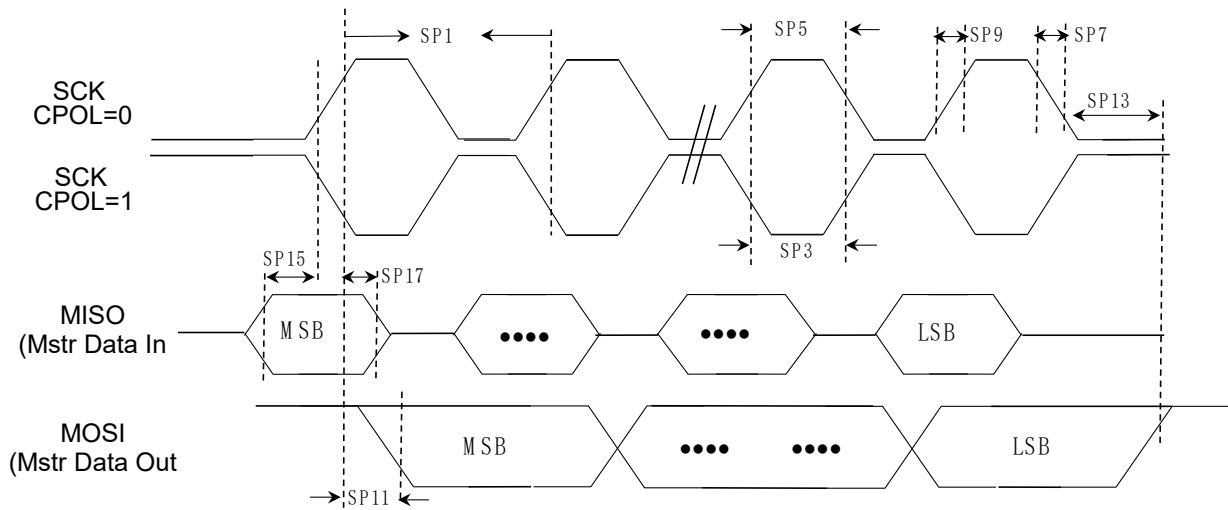
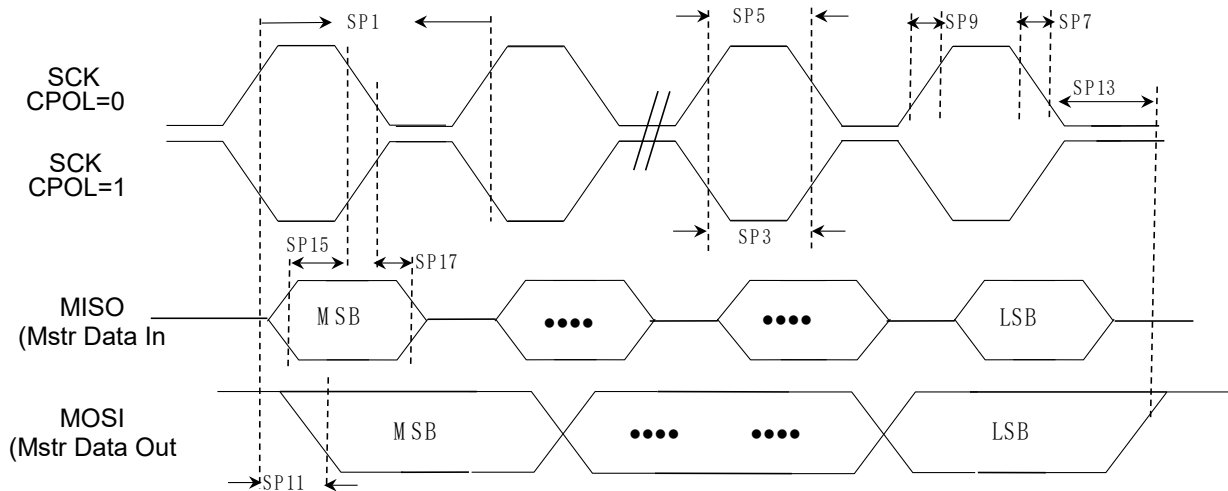


Figure 43-15. SPI Host Module CPHA=1 Timing Diagrams



**Note:**

1. Assumes  $V_{DDIO(min)}$  and 30 pF external load on all SPIx pins unless otherwise noted.

Table 43-29. SPI Module Client Mode Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9-3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ	Max.	Units	Conditions
SSP_1	FCK	SCK frequency	—	—	16	MHz	$V_{DDIO} = 1.9\text{V}$ or $V_{DDIO(min)}$ whichever is greater, $C_{LOAD} = 30\text{ pF}_{(MIN)}$ fixed pins
					12		Remappable pins
SSP_2	TCK	SCK time period	62.5	—	—	$\mu\text{s}$	$V_{DDIO} = 1.9\text{V}$ or $V_{DDIO(min)}$ whichever is greater, $C_{LOAD} = 30\text{ pF}_{(MIN)}$
			83.3	—	—		—

.....continued

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9-3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ	Max.	Units	Conditions
SSP_3	TSCL	SCK output low time	—	tsck/2	—	ns	—
SSP_5	TSCH	SCK output high time	—	tsck/2	—	ns	—
SSP_7	TSCF	SCK and MOSI output fall time	—	—	8.0	ns	$C_{LOAD} = 50$ pF $C_{LOAD} = 20$ pF See parameter DI27 I/O spec
					6.0		
SSP_9	TSCR	SCK and MOSI output rise time	—	—	8.0	ns	$C_{LOAD} = 50$ pF $C_{LOAD} = 20$ pF See parameter DI27 I/O spec
					6.0		
SSP_11	TSOV	MOSI data output valid after SCK	—	—	17	ns	$V_{DDIO} = 3.3\text{V}$ , $C_{LOAD} = 30$ pF <sub>MIN</sub> )
SSP_15	TSIS	MISO setup time of data input to SCK	5	—	—	ns	
SSP_17	TSIH	MISO hold time of data input to SCK	0	—	—	ns	
SSP_23	SPI_GCLK	SERCOM SPI input clock frequency, GCLK_SPI	—	—	64	MHz	—

Figure 43-16. SPI Client Module CPHA=0 Timing Diagrams

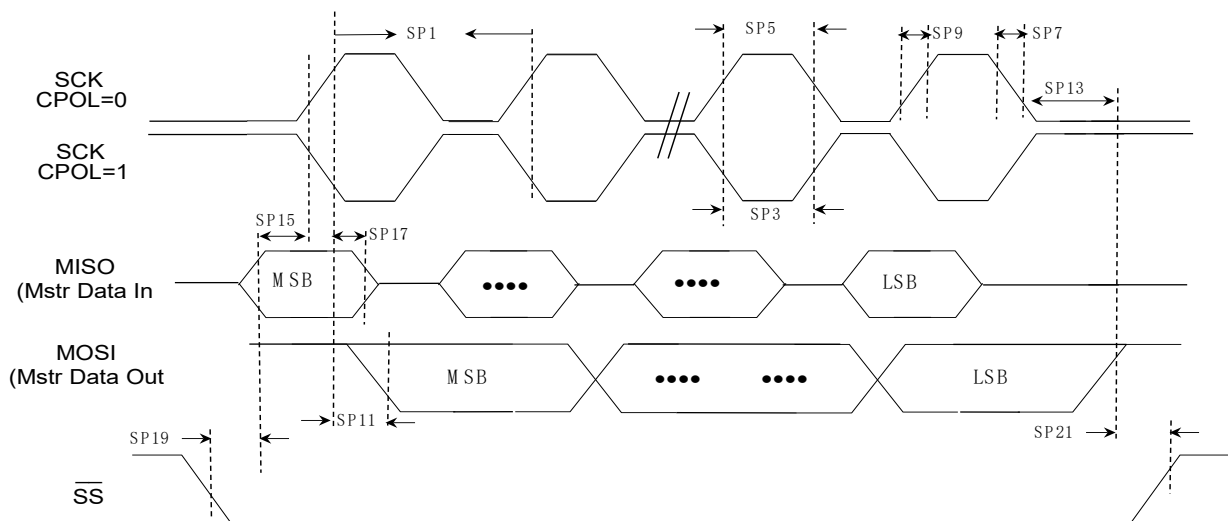
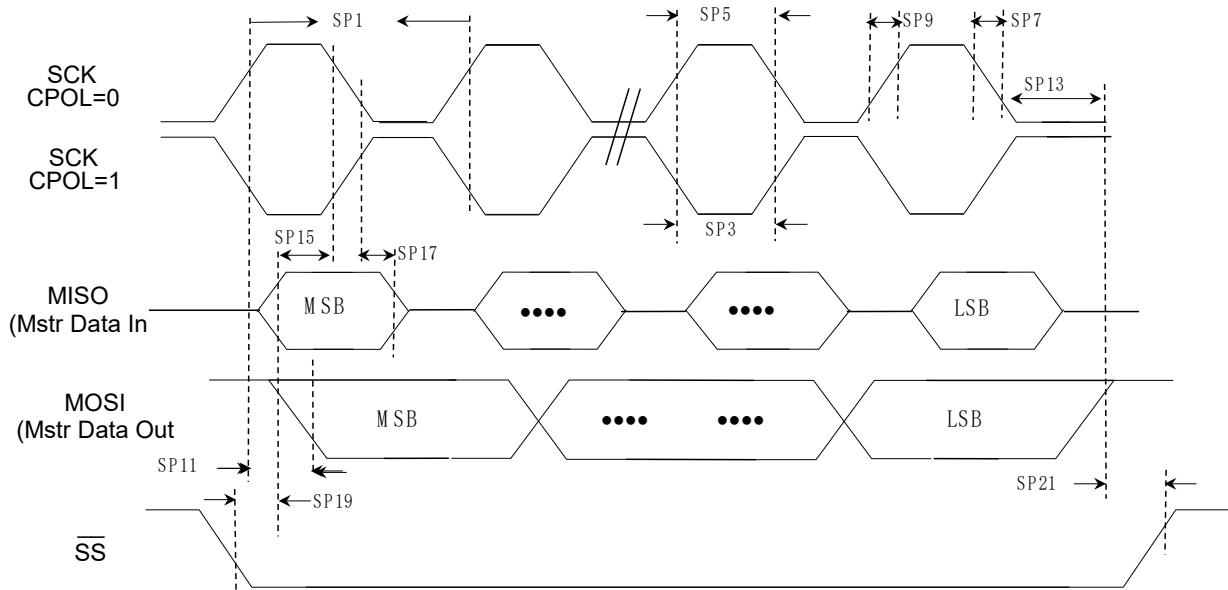


Figure 43-17. SPI Client Module CPHA=1 Timing Diagrams



## 43.24 UART AC Electrical Specifications

Table 43-30. UART AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
UT_1	$F_{\text{BRATE}}$	Baud rate	—	—	Asynchronous arithmetic formula <sup>(2)</sup>	Mbps	$V_{\text{DDIO}} = 3.3\text{V}$
UT_3		Asynchronous SAMPR = 16x mode					
UT_5		Asynchronous SAMPR = 8x mode					
UT_19		Asynchronous SAMPR = 3x mode					
UT_19		Synchronous mode	—	—	Synchronous formula <sup>(2)</sup>	Mbps	—
UT_23	$F_{\text{USART}}$	USART max GCLK_SERCOM	—	—	64	MHz	—
UT_25	$F_{\text{XCK}}$	USART external clock input	—	—	32	MHz	—

**Notes:**

1. These parameters are characterized, but not tested in manufacturing.

Operating Mode	Condition	Baud Rate (Bits Per Second)	Baud Register Value Calculation
Asynchronous arithmetic	$f_{\text{BAUD}} \leq \frac{f_{\text{ref}}}{S}$	$f_{\text{BAUD}} = \frac{f_{\text{ref}}}{S} \left( 1 - \frac{\text{BAUD}}{65536} \right)$	$\text{BAUD} = 65536 \cdot \left( 1 - S \cdot \frac{f_{\text{BAUD}}}{f_{\text{ref}}} \right)$
Synchronous	$f_{\text{BAUD}} \leq \frac{f_{\text{ref}}}{2}$	$f_{\text{BAUD}} \leq \frac{f_{\text{ref}}}{2 \cdot (\text{BAUD} + 1)}$	$\text{BAUD} = \frac{f_{\text{ref}}}{2 \cdot f_{\text{BAUD}}} - 1$



43.25 I<sup>2</sup>C Module Electrical SpecificationsTable 43-31. I<sup>2</sup>C Module Host Mode Electrical Specifications

AC Characteristics				Standard Operating Conditions: V <sub>DDIO</sub> = V <sub>DDANA</sub> 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T <sub>A</sub> ≤ +85°C for Industrial -40°C ≤ T <sub>A</sub> ≤ +125°C for Extended Temp				
Param. No.	Symbol	Characteristics		Min.	Typ.	Max.	Units	Conditions
I2CM_1	TL0:SCL	Host clock low time	100 kHz mode	4700	—	—	ns	V <sub>DDIO</sub> = 3.3V, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	1300	—	—	ns	
			1 MHz mode	500	—	—	ns	
I2CM_3	THI:SCL	Host clock high time	100 kHz mode	450	—	—	ns	V <sub>DDIO</sub> = 3.3V, I <sub>PULL-UP</sub> = 3 mA, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	650	—	—	ns	
			1 MHz mode	130	—	—	ns	
I2CM_5	TF:SCL	SDAx and SCLx fall time	100 kHz mode	—	—	250	ns	V <sub>DDIO</sub> = 3.3V, I <sub>PULL-UP</sub> = 3 mA, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	20 + (0.1 * C <sub>LOAD</sub> )	—	250	ns	
			1 MHz mode	20 + (0.1 * C <sub>LOAD</sub> )	—	250	ns	
I2CM_7	TR:SCL	SDAx and SCLx rise time	100 kHz mode	—	—	100	ns	V <sub>DDIO</sub> = 3.3V, I <sub>PULL-UP</sub> = 3 mA, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	20 + (0.1 * C <sub>LOAD</sub> )	—	300	ns	
			1 MHz mode	—	—	120	ns	
I2CM_9	TSU:DAT	Data input setup time	100 kHz mode	51	—	—	ns	V <sub>DDIO</sub> = 3.3V, I <sub>PULL-UP</sub> = 3 mA, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	51	—	—	ns	
			1 MHz mode	51	—	—	ns	
I2CM_11	THD:DAT	Data input hold time	100 kHz mode	71	—	—	ns	V <sub>DDIO</sub> = 3.3V, I <sub>PULL-UP</sub> = 3 mA, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	71	—	—	ns	
			1 MHz mode	71	—	—	ns	
I2CM_13	TSU:STA	Start condition setup time	100 kHz mode	t <sub>low</sub> +7	—	—	ns	V <sub>DDIO</sub> = 3.3V, I <sub>PULL-UP</sub> = 3 mA, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	t <sub>low</sub> +7	—	—	ns	
			1 MHz mode	t <sub>low</sub> +7	—	—	ns	

.....continued

AC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA} 1.9V$ to $3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp				
Param. No.	Symbol	Characteristics		Min.	Typ.	Max.	Units	Conditions
I2CM_15	THD:STA	Start condition hold time	100 kHz mode	tlow -9	—	—	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	tlow -9	—	—	ns	
			1 MHz mode	tlow -9	—	—	ns	
I2CM_17	TSU:STO	Stop condition setup time	100 kHz mode	Tlow +9	—	—	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	Tlow +9	—	—	ns	
			1 MHz mode	Tlow +9	—	—	ns	
I2CM_19	THD:STO	Stop condition hold time	100 kHz mode	tlow -9	—	—	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	tlow -9	—	—	ns	
			1 MHz mode	tlow -9	—	—	ns	
I2CM_21	TAA:SCL	Output valid from clock	100 kHz mode	—	—	280	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	—	—	280	ns	
			1 MHz mode	—	—	280	ns	
I2CM_23	TBF:SDA	Bus free time <sup>(1)</sup>	100 kHz mode	tlow	—	—	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	tlow	—	—	ns	
			1 MHz mode	tlow	—	—	ns	

**Note:**

1. The amount of time the bus must be free before a new transmission can start (STOP condition to START condition).

**Figure 43-18.** I<sup>2</sup>C Start/Stop Bits Host Mode AC Timing Diagram

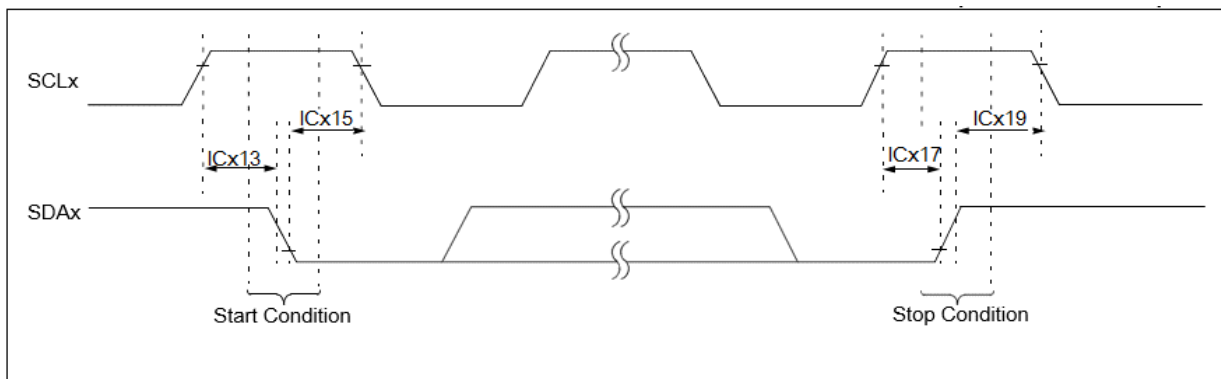


Figure 43-19. I<sup>2</sup>C Bus Data Host Mode AC Timing Diagram

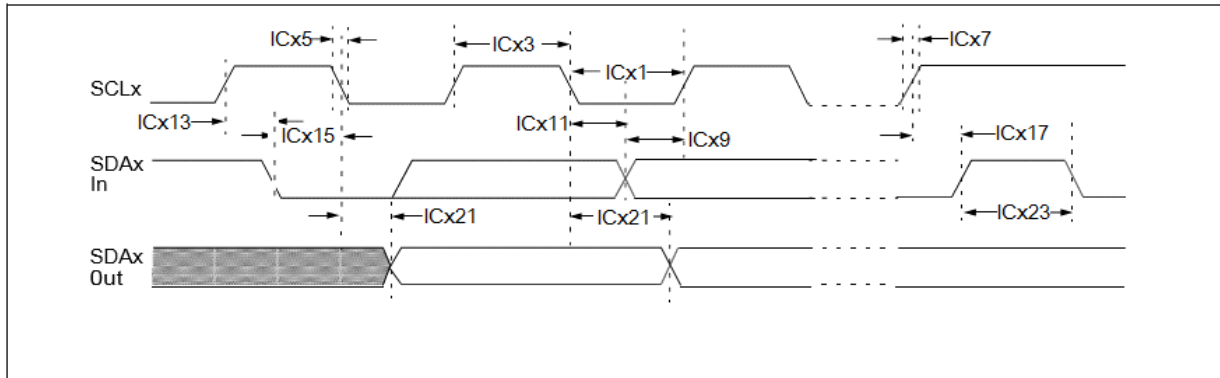


Table 43-32. I<sup>2</sup>C Module Client Mode Electrical Specifications

AC Characteristics				Standard Operating Conditions: V <sub>DDIO</sub> = V <sub>DDANA</sub> 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T <sub>A</sub> ≤ +85°C for Industrial -40°C ≤ T <sub>A</sub> ≤ +125°C for Extended Temp			
Param. No.	Symbol	Characteristics		Min.	Max.	Units	Conditions
I2CS_1	TL0:SCL	Client clock low time	100 kHz mode	4700	—	ns	V <sub>DDIO</sub> = 3.3V, I <sub>PULL-UP</sub> = 3 mA, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	1300	—	ns	
			1 MHz mode	500	—	ns	
I2CS_3	THI:SCL	Client clock high time	100 kHz mode	4050	—	ns	V <sub>DDIO</sub> = 3.3V, I <sub>PULL-UP</sub> = 3 mA, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	650	—	ns	
			1 MHz mode	130	—	ns	
I2CS_5	TF:SCL	SDAx and SCLx fall time	100 kHz mode	—	—	ns	V <sub>DDIO</sub> = 3.3V, I <sub>PULL-UP</sub> = 3 mA, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	20 + (0.1 * C <sub>Load</sub> )	—	ns	
			1 MHz mode	20 + (0.1 * C <sub>Load</sub> )	—	ns	
I2CS_7	TR:SCL	SDAx and SCLx rise time	100 kHz mode	—	—	ns	V <sub>DDIO</sub> = 3.3V, I <sub>PULL-UP</sub> = 3 mA, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	20 + (0.1 * C <sub>Load</sub> )	—	ns	
			1 MHz mode	—	—	ns	
I2CS_9	TSU:DAT	Data input setup time	100 kHz mode	51	—	ns	V <sub>DDIO</sub> = 3.3V, I <sub>PULL-UP</sub> = 3 mA, C <sub>LOAD</sub> = 400 pF
			400 kHz mode	51	—	ns	
			1 MHz mode	51	—	ns	

.....continued

AC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA} 1.9V$ to $3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp			
Param. No.	Symbol	Characteristics		Min.	Max.	Units	Conditions
I2CS_11	THD:DAT	Data input hold time	100 kHz mode	71	—	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	71	—	ns	
			1 MHz mode	71	—	ns	
I2CS_13	TSU:STA	Start condition setup time	100 kHz mode	tlow +7	—	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	tlow +7	—	ns	
			1 MHz mode	tlow +7	—	ns	
I2CS_15	THD:STA	Start condition hold time	100 kHz mode	tlow -9	—	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	tlow -9	—	ns	
			1 MHz mode	tlow -9	—	ns	
I2CS_17	TSU:ST0	Stop condition setup time	100 kHz mode	Tlow +9	—	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	Tlow +9	—	ns	
			1 MHz mode	Tlow +9	—	ns	
I2CS_19	THD:ST0	Stop condition hold time	100 kHz mode	tlow -9	—	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	tlow -9	—	ns	
			1 MHz mode	tlow -9	—	ns	
I2CS_21	TAA:SCL	Output valid from clock	100 kHz mode	—	220	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	—	220	ns	
			1 MHz mode	—	220	ns	
I2CS_23	TBF:SDA	Bus free time <sup>(1)</sup>	100 kHz mode	tlow	—	ns	$V_{DDIO} = 3.3V, I_{PULL-UP} = 3 mA, C_{LOAD} = 400 pF$
			400 kHz mode	tlow	—	ns	
			1 MHz mode	tlow	—	ns	

**Note:**

1. The amount of time the bus must be free before a new transmission can start (STOP condition to START condition).

Figure 43-20. I<sup>2</sup>C Start/Stop Bits Client Mode AC Timing Diagram

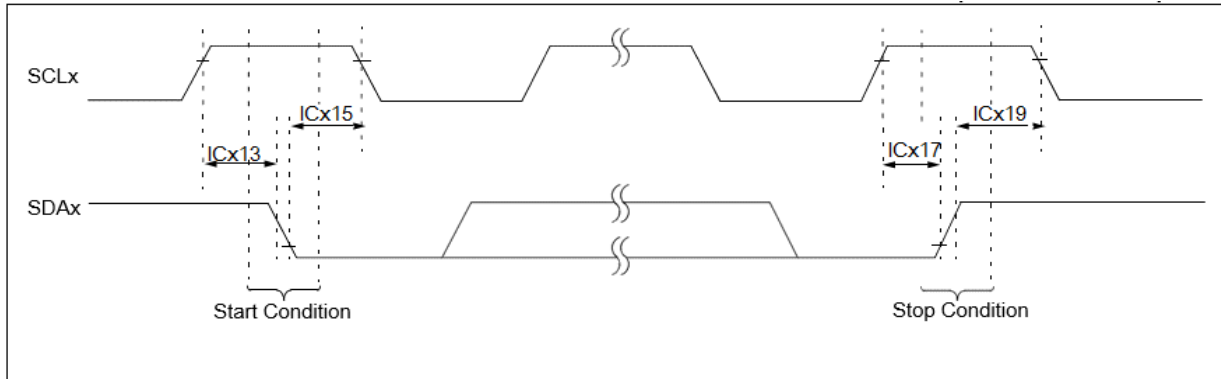
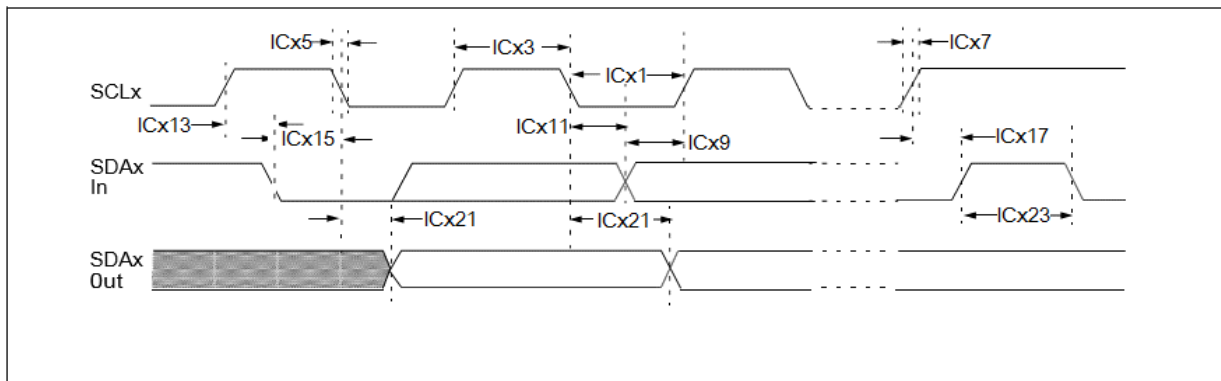


Figure 43-21. I<sup>2</sup>C Bus Data Client Mode AC Timing Diagram



## 43.26 QSPI Module Electrical Specifications

Table 43-33. QSPI Module Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9-3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ. <sup>(1)</sup>	Max.	Units	Conditions
QSPI_1	FCLK	QSPI serial clock frequency • SDR Host mode 0 and 2	—	—	32	MHz	$V_{DDIO} = 3.3\text{V}$ , $C_{LOAD} = 30\text{ pF}_{(MIN)}$
QSPI_2	FCLK	QSPI serial clock frequency • SDR Host mode 1 and 3	—	—	32	MHz	$V_{DDIO} = 3.3\text{V}$ , $C_{LOAD} = 30\text{ pF}_{(MIN)}$
QSPI_3	TSCKH	Serial clock high time	—	8.9	—	ns	—
QSPI_5	TSCKL	Serial clock low time	—	7.8	—	ns	—
QSPI_7	TSCKR	Serial clock rise time	—	6.9	—	ns	See parameter DI27 I/O spec
QSPI_9	TSCKF	Serial clock fall time	—	7.6	—	ns	See parameter DI25 I/O spec
QSPI_11	TCSS	CS active setup time	—	7.3	—	ns	—
QSPI_13	TCSH	CS active hold time	—	10.4	—	ns	—

.....continued

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9-3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ. <sup>(1)</sup>	Max.	Units	Conditions
QSPI_19	TDIS	Data in setup time	—	2.4	—	ns	—
QSPI_21	TDIH	Data in hold time	—	1.023	—	ns	—
QSPI_23	TDOH	Data out hold	—	1.17	—	ns	—
QSPI_25	TDOV	Data out valid	—	1.48	—	ns	—
QSPI_27	QSPI_GCLK	QSPI peripheral clock frequency	—	—	64	MHz	—

**Notes:**

- Assumes  $V_{DDIO}(\text{typ})$  and typical external load on all SQI pins unless otherwise noted.
- These parameters are characterized but not tested in manufacturing.

Figure 43-22. QSPI SDR Host Mode 0,1,2,3 Module Timing Diagram

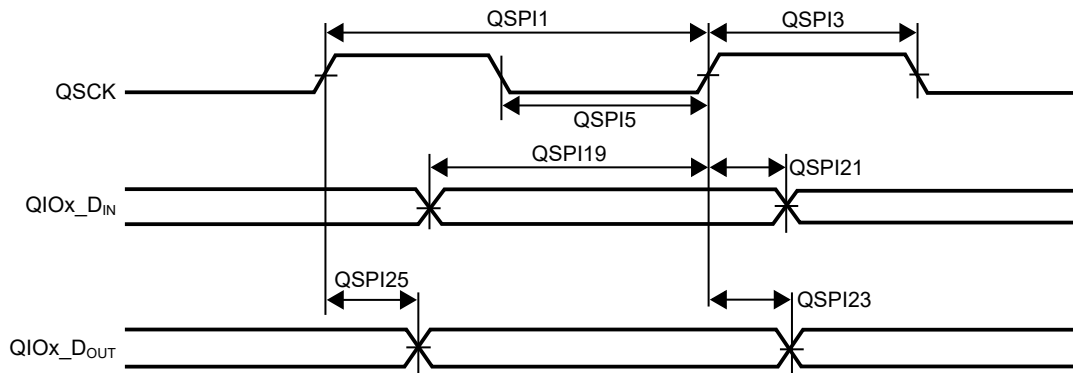


Figure 43-23. QSPI DDR Mode 0 Write Timing Diagram

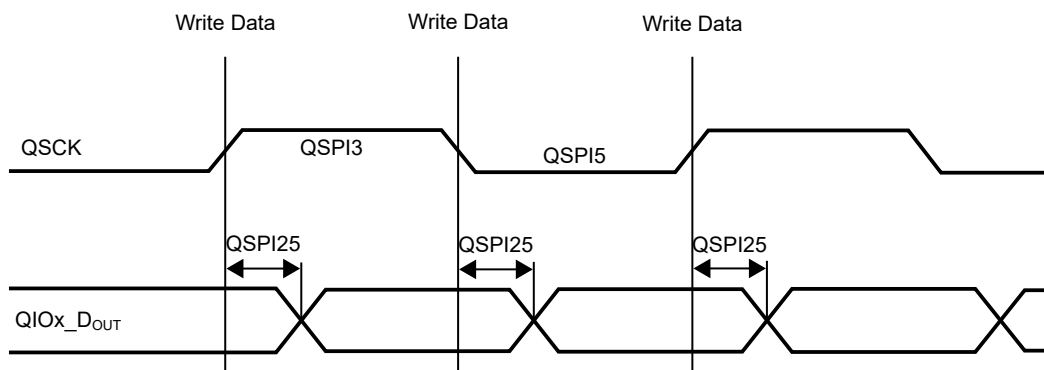
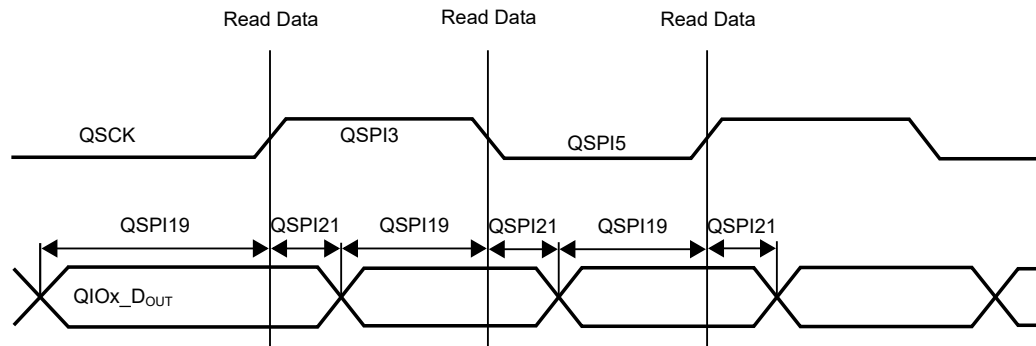


Figure 43-24. QSPI DDR Mode 0 Read Timing Diagram



## 43.27 TCx Timer Capture Module AC Electrical Specifications

Table 43-34. TCx Timer Capture Module AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DD} = 1.9V$ to $3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial Temp $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
TC_1	$TC_{INLOW}$	Capture TCx input low time	$2/f_{GLK\_TCx}$	—	—	ns	$V_{DDIO} = 3.3V$ and meet TC_5 spec
TC_3	$TC_{INHIGH}$	Capture TCx input high time	$2/f_{GLK\_TCx}$	—	—	ns	$V_{DDIO} = 3.3V$ and meet TC_5 spec
TC_5	$TC_{INPERIOD}$	Capture input period	$4/f_{GLK\_TCx}$	—	—	ns	$V_{DDIO} = 3.3V$
TC_7	$TC_{OUTLOW}$	Compare TCx output low time	$1/f_{GCLK\_TCx}$	—	—	ns	$V_{DDIO} = 3.3V$ and meet TC_11 spec
TC_9	$TC_{OUTHIGH}$	Compare TCx output high time	$1/f_{GCLK\_TCx}$	—	—	ns	$V_{DDIO} = 3.3V$ and meet TC_11 spec
TC_11	$TC_{OUTPERIOD}$	Compare output period	$2/f_{GCLK\_TCx}$	—	—	ns	$V_{DDIO} = 3.3V$
TC_13	$f_{GCLK\_TCx}$	TC peripheral module clock frequency	—	—	64	MHz	$V_{DDIO(min)}$

**Note:** These parameters are characterized but not tested in manufacturing.

Figure 43-25. TCx Timer Capture Input Module AC Timing Diagram

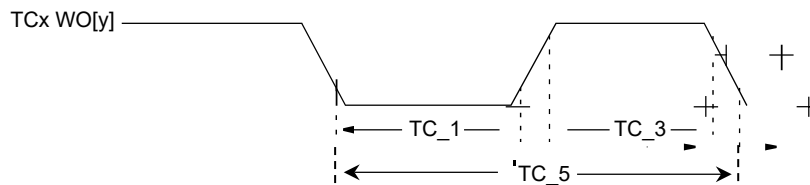
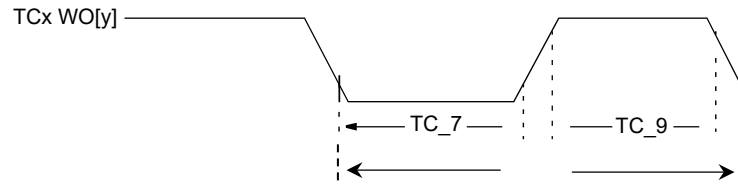


Figure 43-26. TCx Timer Capture Output Module AC Timing Diagram



## 43.28 TCCx Timer Capture Module AC Electrical Specifications

Table 43-35. TCCx Timer Capture Module AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DD} = 1.9V$ to $3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial Temp $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
TCC_1	TCC <sub>INLOW</sub>	Capture TCCx input low time	$2/f_{GLK\_TCCx}$	—	—	ns	$V_{DDIO(min)}$ and meet TCC_5 spec
TCC_3	TCC <sub>INHIGH</sub>	Capture TCCx input high time	$2/f_{GLK\_TCCx}$	—	—	ns	$V_{DDIO(min)}$ and meet TCC_5 spec
TCC_5	TCC <sub>INPERIOD</sub>	Capture input period	$4/f_{GLK\_TCCx}$	—	—	ns	$V_{DDIO(min)}$
TCC_7	TCC <sub>OUTLOW</sub>	Compare TCCx output low time	$1/f_{GCLK\_TCx}$	—	—	ns	$V_{DDIO(min)}$ and meet TCC_11 spec
TCC_9	TCC <sub>OUTHIGH</sub>	Compare TCCx output high time	$1/f_{GCLK\_TCx}$	—	—	ns	$V_{DDIO(min)}$ and meet TCC_11 spec
TCC_11	TCC <sub>OUTPERIOD</sub>	Compare output period	$2/f_{GCLK\_TCx}$	—	—	ns	$V_{DDIO(min)}$
TCC_13	f <sub>GCLK_TCCx</sub>	TCC peripheral module clock frequency	—	—	64	MHz	
TCC_15	TCC <sub>FD</sub>	Fault input to I/O pin change	—	—	63	ns	
TCC_17	TCC <sub>FLT</sub>	Fault input pulse width	10	—	—	ns	

Figure 43-27. TCCx Timer Capture Input Module AC Timing Diagram

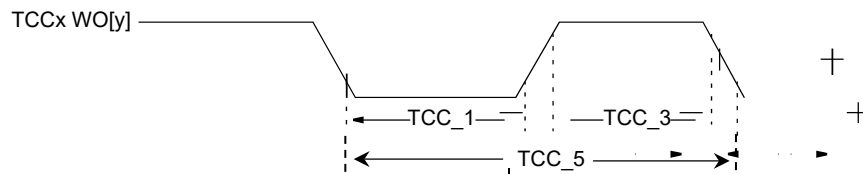


Figure 43-28. TCCx Timer Capture Output Module AC Timing Diagram

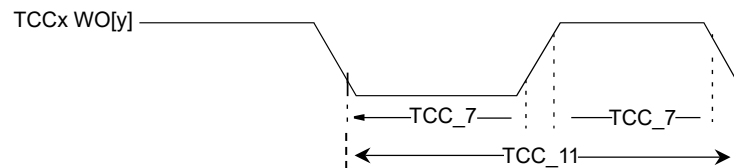
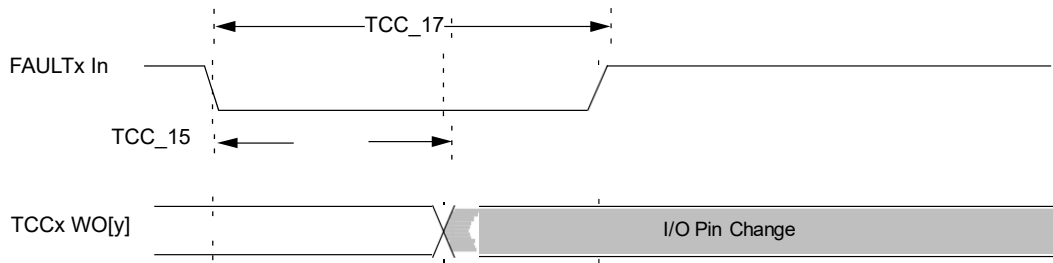




Figure 43-29. TCC\_x Timer Compare Fault Output Module AC Timing Diagram



## 43.29 FLASH NVM AC Electrical Specifications

Table 43-36. FLASH NVM AC Electrical Specifications

AC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics		Min.	Typ.	Max.	Units	Conditions
NVM_1	$F_{RETEN}$	Flash data retention		—	15	—	Yrs	Under all conditions less than absolute maximum ratings specifications
NVM_3	EP	Cell endurance (Flash erase and Write operation)		—	100k	—	Cycles	
NVM_5	$F_{READ}$	Flash read	0 wait states	—	—	16	MHz	$V_{DDIO} = 3.3\text{V}$
			1 wait states	—	—	32		
			2 wait states	—	—	64		
			3 wait states	—	—	64		
			4 wait states	—	—	64		
NVM_7	$T_{FPW}$	Program cycle time	Row write	—	2.5	—	ms	—
NVM_9	$T_{CE}$		Erase chip	—	20	—	ms	

**Note:**

- The maximum FLASH operating frequencies are given in the table above but are limited by the Embedded Flash access time when the processor is fetching code out of it. This field defines the number of Wait states required to access the Embedded Flash Memory.

## 43.30 Frequency Meter AC Electrical Specifications

Table 43-37. Frequency Meter AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DD} = 1.9\text{V}$ to $3.6\text{V}$ (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
FM_1	$FM_{LOW}$	GCLK_IOx input low time	—	15.625	—	ns	$V_{DDIO}(\text{min})$ and meet FM5 spec
FM_3	$FM_{HIGH}$	GCLK_IOx input high time	—	15.625	—	ns	
FM_5	$FM_{PERIOD}$	GCLK_IOx input period	—	31.25	—	ns	$V_{DDIO}(\text{min})$
FM_7	$f_{GCLK\_FREQM\_REF}$	FREQM reference	—	—	64	MHz	
FM_9	$f_{GCLK\_FREQM\_MSR}$	FREQM measure	—	—	64	MHz	

### 43.31 Bluetooth Low Energy RF Characteristics

Table 43-38. Bluetooth Low Energy RF Characteristics

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA} 1.9-3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial Temp $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
BTG1	FREQ	Frequency range of operation	2402	—	2480	MHz	—
BTTX1	TXPWR:MPA	Bluetooth transmit power MPA	—	11.67	—	dBm	—
BTTX2	TXPWR:LPA	Bluetooth transmit power LPA	—	3.95	—	dBm	—
BTX3	TXIB:1 Mbps	In-band emission for FTX $\pm 2$ MHz	—	-32	—	dBm	—
		In-band emission for FTX $\pm -(3+N)$ MHz	—	-45	—	dBm	—
BTX4	TXIB:2MBPS	In-band emission for FTX $\pm -4$ MHz	—	-43	—	dBm	—
		In-band emission for FTX $\pm -5$ MHz	—	-48	—	dBm	—
		In-band emission for FTX $\pm -(6+N)$ MHz	—	-51	—	dBm	—
BTRX1	RXSENSE	Receiver sensitivity at 1 Mbps	—	-95.6	—	dBm	—
		Receiver sensitivity at 2 Mbps	—	-92.45	—	dBm	—
		Receiver sensitivity at S = 8	—	-102.28	—	dBm	—
		Receiver sensitivity at S = 2	—	-98.9	—	dBm	—
BTRX2	MAXINSIG	Maximum input signal level at 1 Mbps	—	0	—	dBm	—
		Maximum input signal level at 2 Mbps	—	0	—	dBm	—
		Maximum input signal level at S = 2	—	0	—	dBm	—
		Maximum input signal level at S = 8	—	0	—	dBm	—
BTRX3	CI1M:COCH	C/I Co channel rejection	—	13	—	dB	—
	CI1M: $\pm -1$ MHz	C/I adjacent channel rejection	—	14	—	dB	—
	CI1M: $\pm -2$ MHz	C/I adjacent channel rejection	—	13	—	dB	—
	CI1M:ADJ(3+n)	C/I alternate channel rejection	—	21	—	dB	—
	CI1M:IMG	C/I image frequency rejection	—	15	—	dB	—
	CI1M:IMG $\pm -1$ MHz	C/I adjacent channel to image freq rejection	—	16	—	dB	—

.....continued

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA} 1.9-3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial Temp $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions
BTRX4	CIS2:COCH	C/I Co channel rejection	—	11	—	dB	—
	CIS2: $\pm 1$ MHz	C/I adjacent channel rejection	—	17	—	dB	—
	CIS2: $\pm 2$ MHz	C/I adjacent channel rejection	—	17	—	dB	—
	CIS2:ADJ(3+n)	C/I alternate channel rejection	—	19	—	dB	—
	CIS2:IMG	C/I image frequency rejection	—	14	—	dB	—
	CIS2:IMG $\pm 1$ MHz	C/I adjacent channel to image freq rejection	—	18	—	dB	—
BTRX5	CIS8:COCH	C/I Co channel rejection	—	5	—	dB	—
	CIS8: $\pm 1$ MHz	C/I adjacent channel rejection	—	13	—	dB	—
	CIS8: $\pm 2$ MHz	C/I adjacent channel rejection	—	12	—	dB	—
	CIS8:ADJ(3+n)	C/I alternate channel rejection	—	12	—	dB	—
	CIS8:IMG	C/I image frequency rejection	—	9	—	dB	—
	CIS2:IMG $\pm 1$ MHz	C/I adjacent channel to image freq rejection	—	17	—	dB	—
BTRX6	CI2M:COCH	C/I Co channel rejection	—	13	—	dB	—
	CI2M: $\pm 2$ MHz	C/I adjacent channel rejection	—	17	—	dB	—
	CI2M: $\pm 4$ MHz	C/I adjacent channel rejection	—	19	—	dB	—
	CI2M:ADJ(6+2n)	C/I alternate channel rejection	—	19	—	dB	—
	CI2M:IMG	C/I image frequency rejection	—	16	—	dB	—
	CI2M:IMG $\pm 2$ MHz	C/I adjacent channel to image freq rejection	—	18	—	dB	—
BTRX7	BLOCK1M:<2 GHz	Blocking performance from 30-2 GHz	—	20	—	dB	—
	BLOCK1M:2 GHz<SIG<2399 MHz	Blocking performance from 2003-2399 MHz	—	14	—	dB	—
	BLOCK1M:2484 MHz<SIG<2977 MHz	Blocking performance between 2484-2997 MHz	—	20	—	dB	—
	BLOCK1M:3 GHz<SIG<12.75 GHz	Blocking performance between 3-12.5 GHz	—	20	—	dB	—
BTRX8	BLE1M:INTERMOD	Inter modulation performance for BLEM	—	14.5	—	dB	—
	BLE2M:INTERMOD	Inter modulation performance for BLEM	—	21.5	—	dB	—

.....continued

AC Characteristics		Standard Operating Conditions: $V_{DDIO} = V_{DDANA} 1.9-3.6V$ (unless otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ for Industrial Temp $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp					
Param. No.	Symbol	Characteristics	Min.	Typ.	Max.	Units	Conditions

**Notes:**

1. Measured at 25°C, averaged across all voltages and channels.
2. Measured on a board with the reference schematic.
3. All measurement across voltage based on the SIG specifications.
4. The specified value is the limit above the SIG specifications.
5. PDU length = 37, channels = 2402/2426/2440/2480 MHz.

**Table 43-39.** Bluetooth Low Energy LPA RF Characteristics

AC Characteristics			Standard Operating Conditions: Industrial $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp	
Param. No.	Symbol	Characteristics	Min.	Typ.
BTLG1	FREQ	Frequency range of operation	2400	—
BTLTX2	TXPWR:LPA	Bluetooth transmit power LPA	—	5.47
BTLX3	TXIB:1MBPS	In-band emission for FTX $\pm 2$ MHz	—	-40
		In-band emission for FTX $\pm (3+N)$ MHz	—	-49
BTX4	TXIB:2MBPS	In-band emission for FTX $\pm 4$ MHz	—	-46
		In-band emission for FTX $\pm 5$ MHz	—	-51
		In-band emission for FTX $\pm (6+N)$ MHz	—	-54
BTLRX1	RXSENSE	Receiver sensitivity at 1 Mbps	—	-97
		Receiver sensitivity at 2 Mbps	—	-94
		Receiver sensitivity at S = 8	—	-104
		Receiver sensitivity at S = 2	—	-100
BTLRX2	MAXINSIG	Maximum Input signal level at 1 Mbps	—	0
		Maximum Input signal level at 2 Mbps	—	0
		Maximum Input signal level at S = 2	—	0
		Maximum Input signal level at S = 8	—	0

**Notes:**

1. Measured at 25°C, averaged across all voltages and channels.
2. Measured on a board with the reference schematic.
3. All measurement across voltage based on the SIG specifications.
4. The specified value is the limit above the SIG specifications.
5. PDU length = 37, channels = 2402/2426/2440/2480 MHz.

**Table 43-40.** Bluetooth Low Energy RF Current Characteristics

AC Characteristics					Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9-3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	RF Power	CPU Frequency	Min.	Typ.	Max.	Units	Conditions
IBLETX1	IDDTXMPA	Current consumption with output power in DC-DC mode 1 Mbps	+12 dBm	64 MHz	—	42.8	—	mA	—
IBLETX4		Current consumption at +12 dBm output power in MLDO mode	+12 dBm	64 MHz	—	96.7	—	mA	—
IBLETX7	IDDTXLPA	Current consumption at +4 dBm output power in DC-DC mode 1 Mbps	4 dBm	64 MHz	—	24.9	—	mA	—
IBLETX10		Current consumption at +4 dBm output power in MLDO mode	4 dBm	64 MHz	—	55.5	—	mA	—
IBLETX7	IDDTXLP0	Current consumption at +0 dBm output power in DC-DC mode 1 Mbps	0 dBm	64 MHz	—	22.7	—	mA	—
IBLETX10		Current consumption at 0 dBm output power in MLDO mode	0 dBm	64 MHz	—	47.6	—	mA	—
IBLERX1	IDDRXBLE1M	Current consumption at RX signal level -80 dBm in DC-DC mode	-80 dBm	64 MHz	—	20.6	—	mA	—
IBLERX4		Current consumption at RX signal level -80 dBm in MLDO mode	-80 dBm	64 MHz	—	40.6	—	mA	—

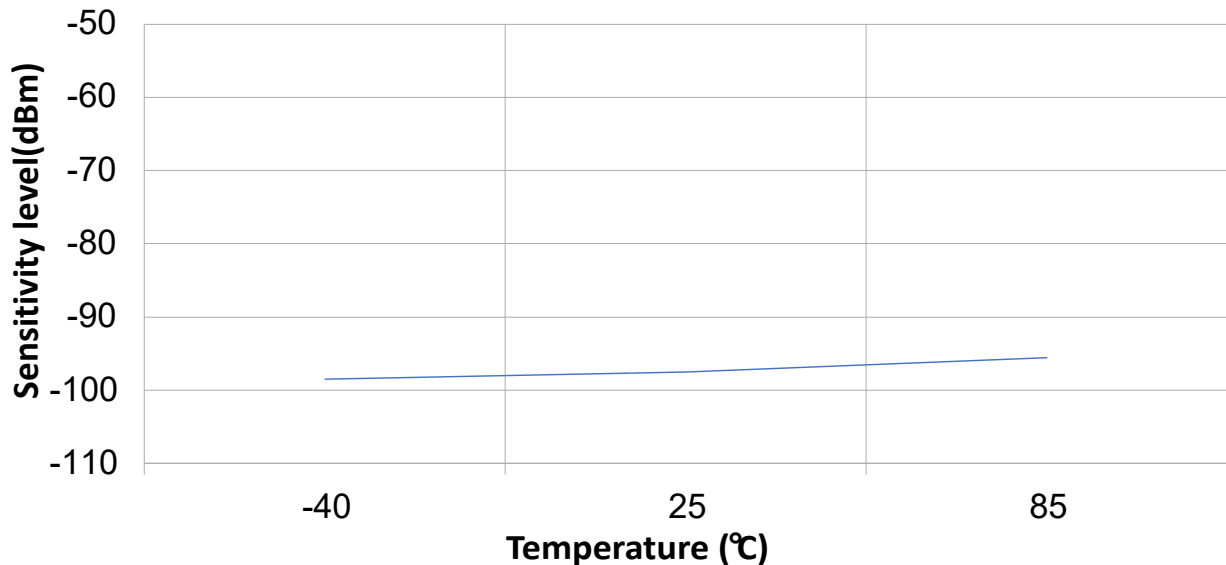
**Notes:**

- Current consumption is measured on a board based upon the Microchip Technology Reference Design.
- Current consumption is for the entire SoC (including the MCU), measured at the input power rail.
- Current reported is the average of the current during the transmit or receive burst (exclude off cycle of the transmit/receive operation).

**Table 43-41.** Bluetooth Low Energy RF Current LPA Characteristics

AC Characteristics					Standard Operating Conditions: $V_{DDIO} = V_{DD}$ (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	RF Power	CPU Frequency	Min.	Typ.	Max.	Units	Con
IBLETX1	IDDTXMPA	Current consumption with output power 5.5 dBm in DC-DC mode 1 Mbps	+5.5 dBm	64 MHz	—	24.1	—	mA	$V_{DD}$
IBLETX4		Current consumption at +5.5 dBm output power in MLDO mode	+5.5 dBm	64 MHz	—	46.9	—	mA	$V_{DD}$
IBLETX7	IDDTXLPA	Current consumption at +0 dBm output power in DC-DC mode 1 Mbps	0 dBm	64 MHz	—	20.6	—	mA	$V_{DD}$
IBLETX10		Current consumption at +0 dBm output power in MLDO mode 1 Mbps	0 dBm	64 MHz	—	40.4	—	mA	$V_{DD}$
IBLETX7	IDDTXLPA0	Current consumption at -5 dBm output power in DC-DC mode 1 Mbps	-5 dBm	64 MHz	—	19.2	—	mA	$V_{DD}$
IBLETX10		Current consumption at -5 dBm output power in MLDO mode 1 Mbps	-5 dBm	64 MHz	—	37.5	—	mA	$V_{DD}$
IBLERX1	IDDRXBLE1M	Current consumption at RX signal level -80 dBm in DC-DC mode	-80 dBm	64 MHz	—	21.2	—	mA	$V_{DD}$
IBLERX4		Current consumption at RX signal level -80 dBm in MLDO mode	-80 dBm	64 MHz	—	39.9	—	mA	$V_{DD}$

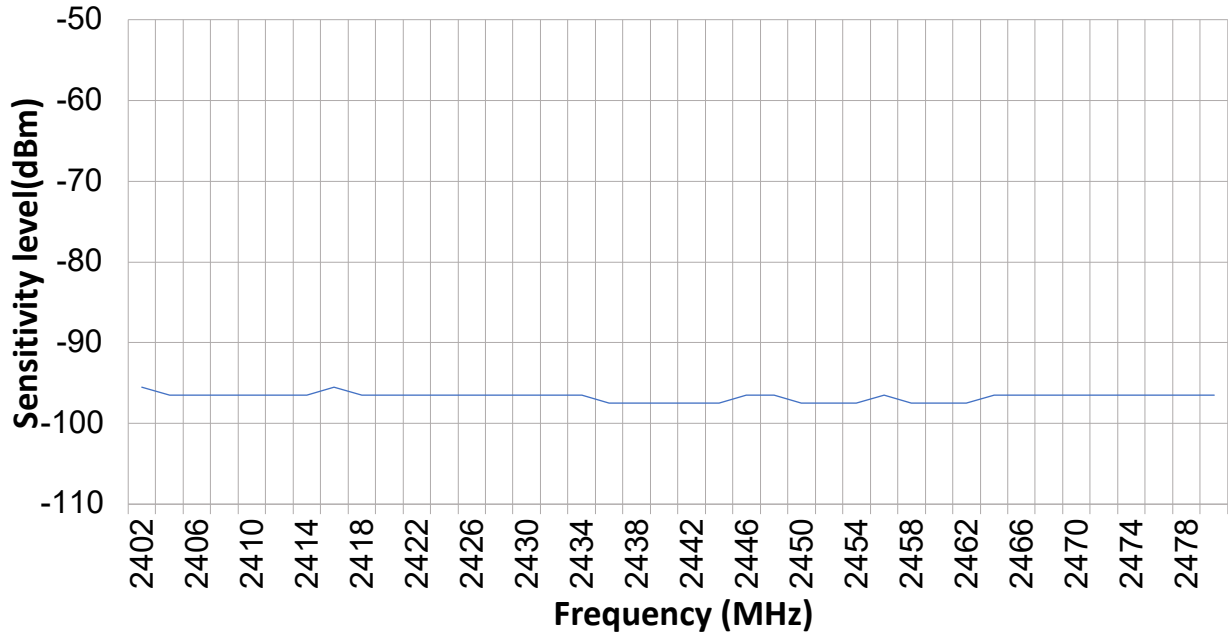
**Figure 43-30.** Module Bluetooth Low Energy Receive Sensitivity vs Temperature



**Notes:**

- Bluetooth Low Energy receive sensitivity is measured across temperature at 3.6V, 2440 MHz, uncoded data at 1 Ms/s.
- PDU length = 37.
- Sensitivity is measured according to the SIG specifications.

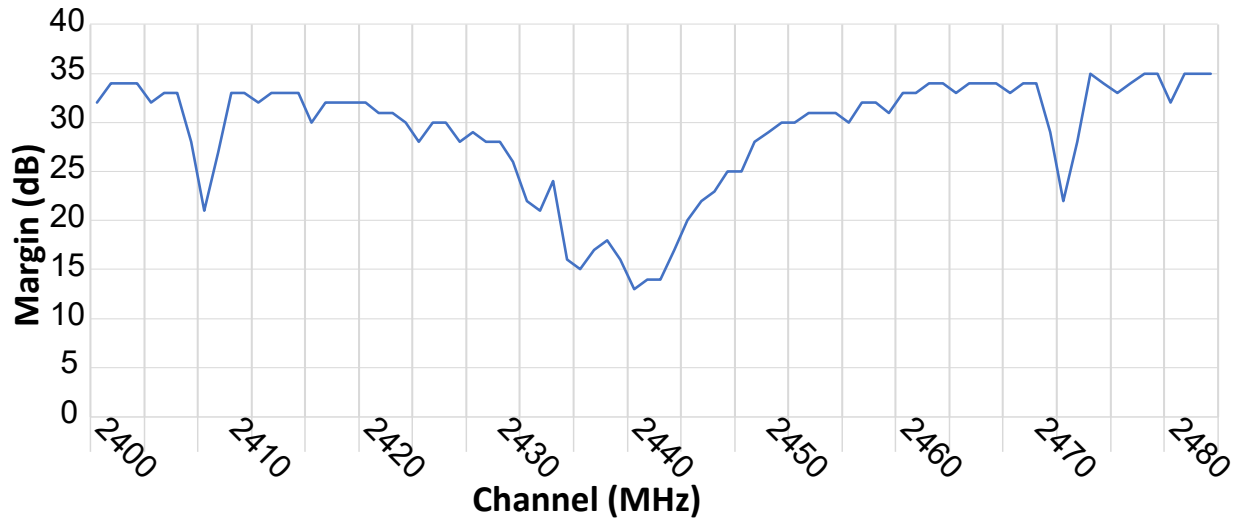
**Figure 43-31.** Module Bluetooth Low Energy Receive Sensitivity vs Frequency



**Notes:**

- Bluetooth Low Energy sensitivity is measured across channels at 3.6V at 25°C, uncoded data at 1 Ms/s.
- PDU length = 37.
- Sensitivity is measured according to the SIG specifications.

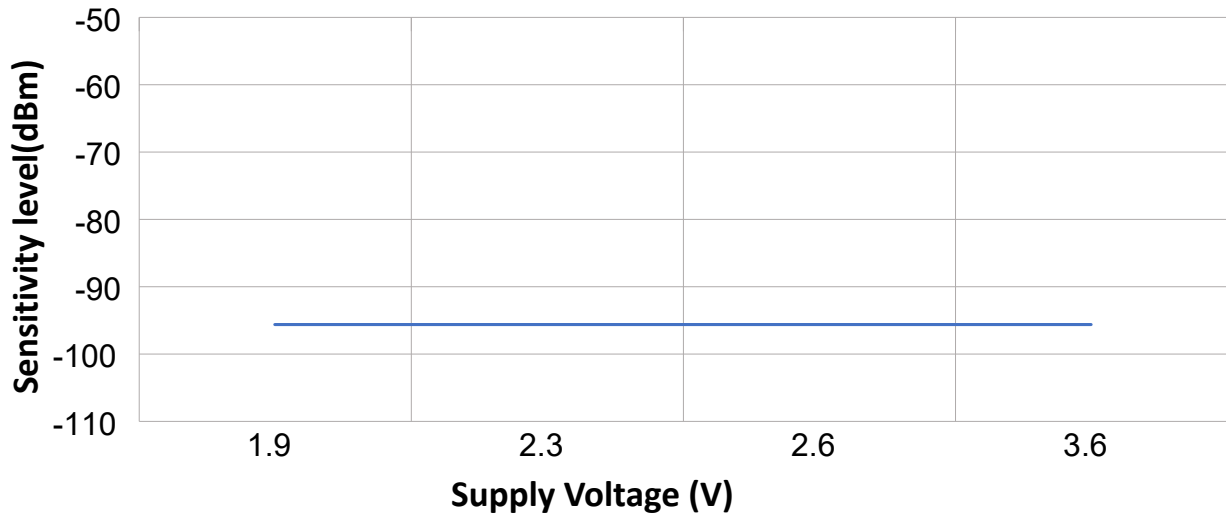
Figure 43-32. Bluetooth Low Energy 1M CI Margin



**Notes:**

- Bluetooth Low Energy 1M C/I Margin is measured at 2440 MHz at 25°C, 3.6V, uncoded data at 1 Ms/s.
- Reported C/I margin is the margin above the C/I specifications from SIG.

Figure 43-33. Bluetooth Low Energy Receive Sensitivity vs Voltage

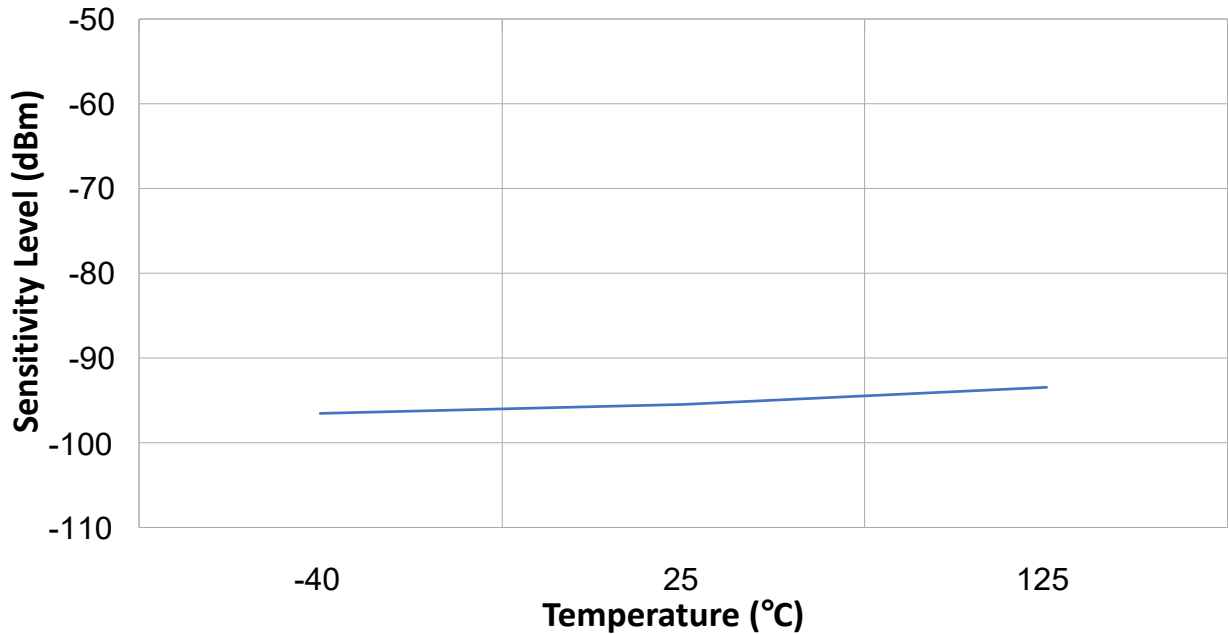


**Notes:**

- Bluetooth Low Energy receive sensitivity is measured at 2440 MHz at 25°C, uncoded data at 1 Ms/s.
- PDU length = 37.
- Sensitivity is measured according to the SIG specifications.



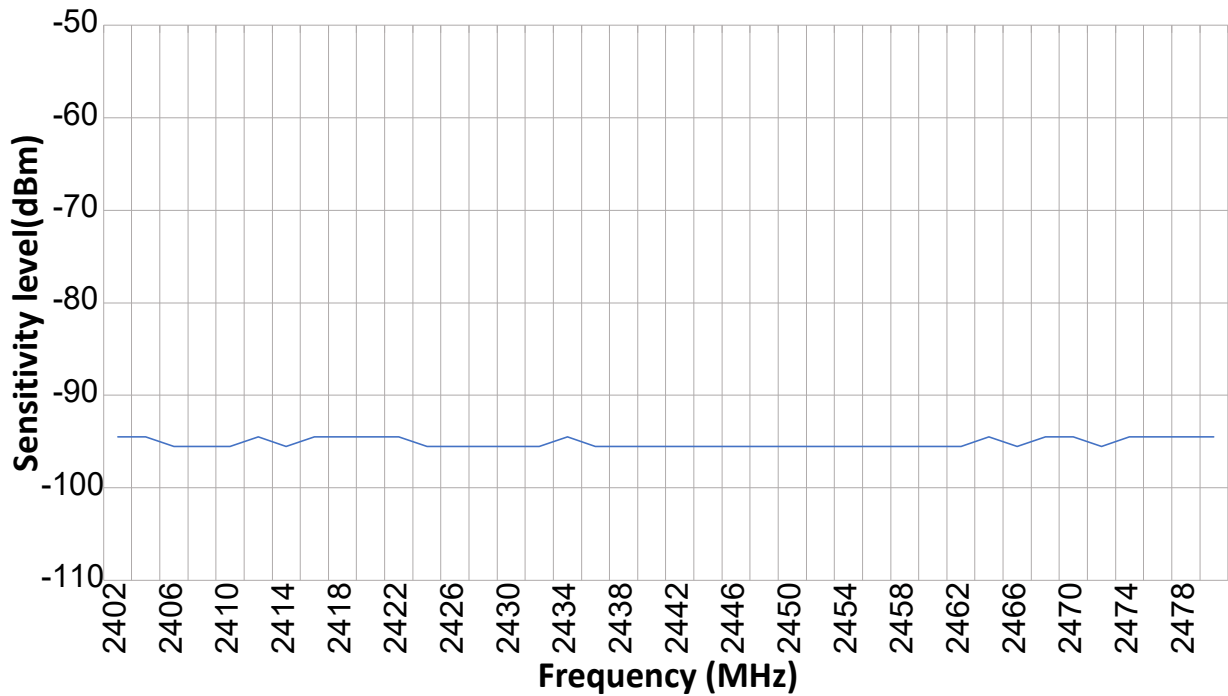
Figure 43-34. Bluetooth Low Energy Receive Sensitivity vs Temperature



**Notes:**

- Bluetooth Low Energy receive sensitivity is measured across channels at 3.6V, 2440 MHz, uncoded data at 1 Ms/s.
- PDU length = 37.
- Sensitivity is measured according to the SIG specifications.

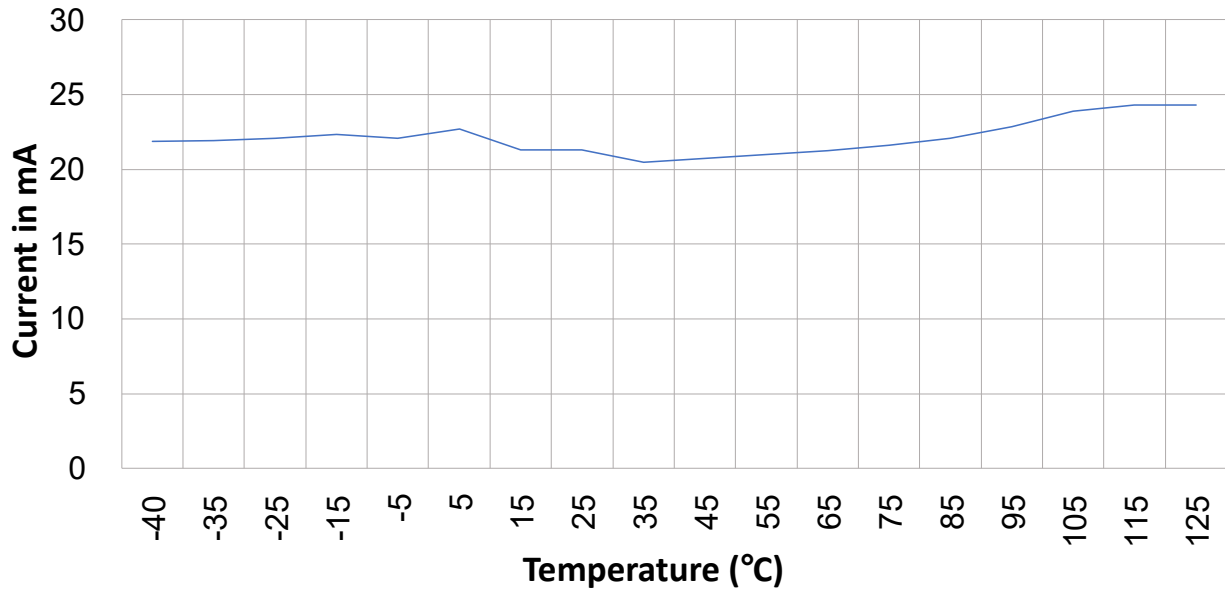
Figure 43-35. Bluetooth Low Energy Receive Sensitivity vs Frequency



**Notes:**

- Bluetooth Low Energy receiver sensitivity is measured across channels at 3.6V at 25°C, uncoded data at 1 Ms/s.
- PDU length = 37.
- Sensitivity is measured according to the SIG specifications.

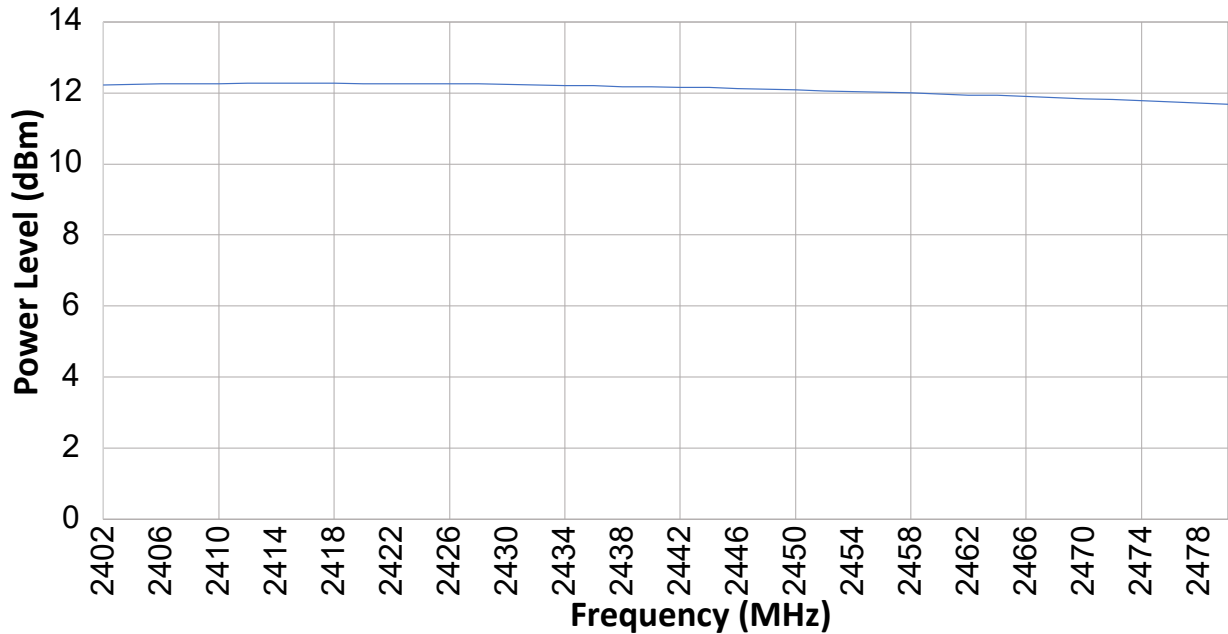
**Figure 43-36.** Bluetooth Low Energy Receive Current vs Temperature



**Notes:**

- Bluetooth Low Energy receive current is measured at 3.3V (Buck mode), uncoded data at 1 Ms/s with LNA configured at maximum gain.
- PDU length = 37.
- Current is measured on input power rail to SoC (includes processor current as well).

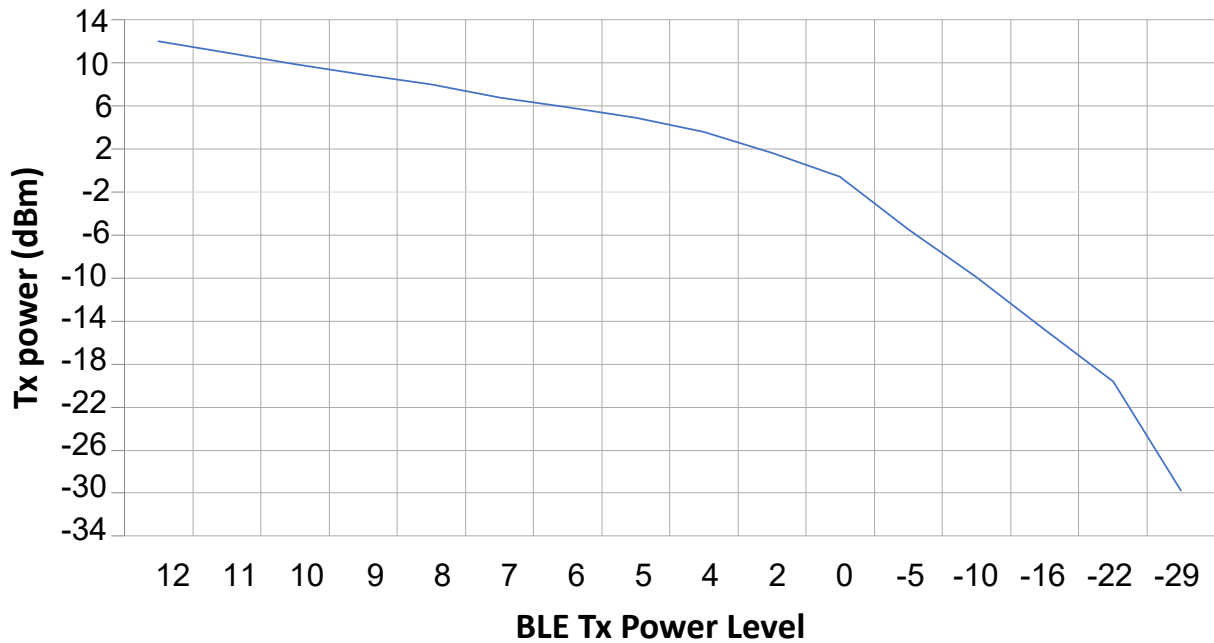
Figure 43-37. Bluetooth Low Energy Transmit Power vs Frequency



**Notes:**

- Bluetooth Low Energy transmit power is measured across frequency after transmit power calibration at 3.3V (Buck mode).
- Transmit power is measured after the PA matching and LPF.

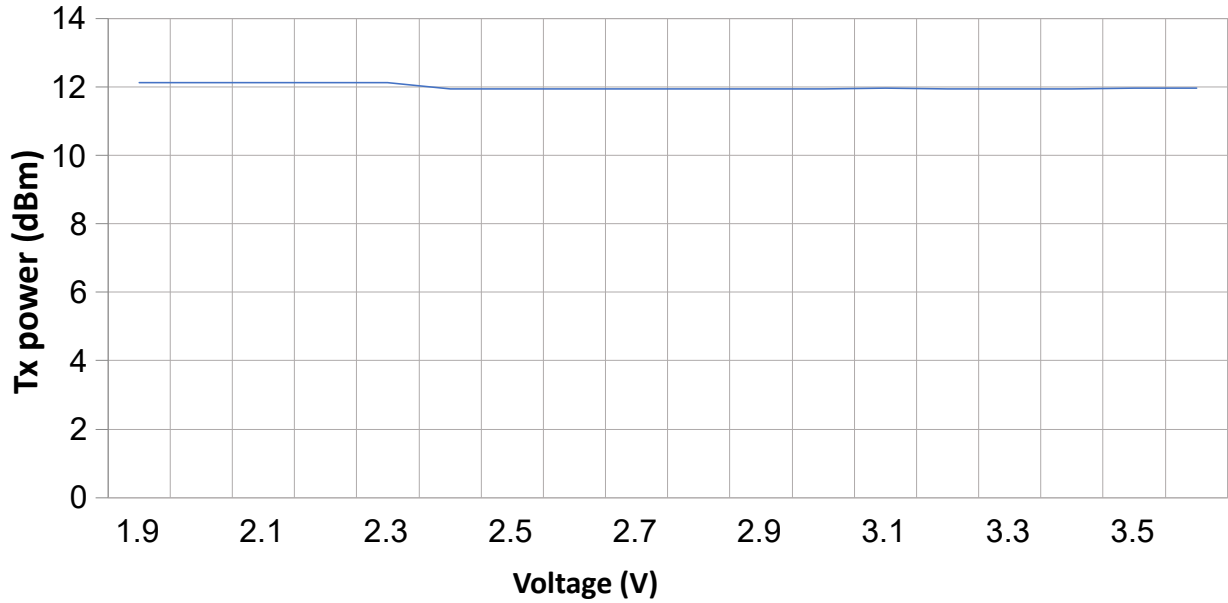
Figure 43-38. Bluetooth Low Energy Transmit Power vs Transmit Power Level



**Notes:**

- Bluetooth Low Energy transmit power is measured at 2440 MHz after transmit power calibration.
- Transmit power is measured on board based on Microchip Technology Reference Design.
- Transmit power is measured after PA match and LPF.

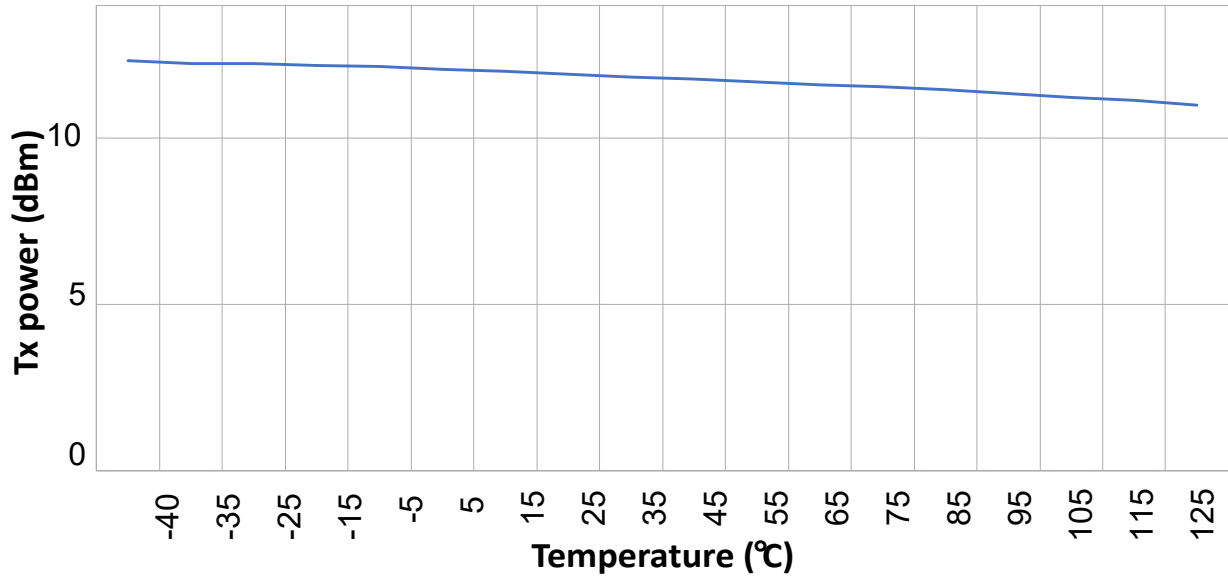
**Figure 43-39.** Bluetooth Low Energy Transmit Power vs VDD Supply Voltage



**Notes:**

- Bluetooth Low Energy transmit power is measured across voltage after transmit power calibration.
- Transmit power is measured after calibration at +12 dBm ( $\pm 0.5$  dBm).
- Transmit power is measured on board based on the Microchip Reference Design.
- Transmit power is measured after the LPA and PA match section.

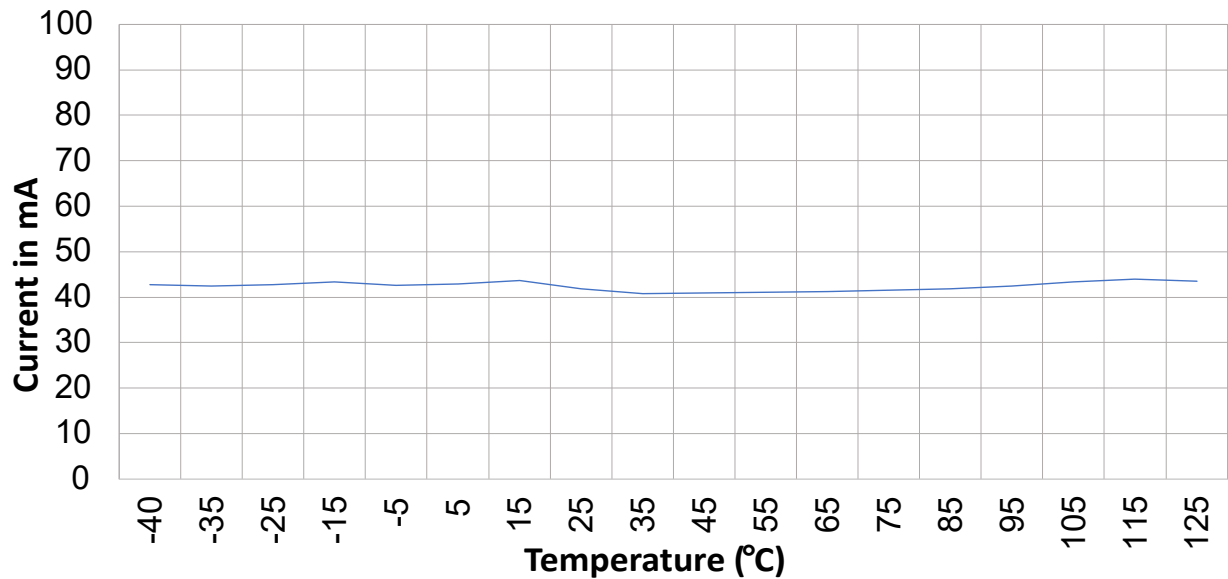
Figure 43-40. Bluetooth Low Energy Transmit Power vs. Temperature



**Notes:**

- Bluetooth Low Energy transmit power is measured across temperature after transmit power calibration at 3.6V and 2440 MHz.
- Temperature power compensation is triggered before power measurement.
- Transmit power is measured after the PA matching and LPF.

Figure 43-41. Bluetooth Low Energy Transmit Current vs Temperature



**Notes:**

- Bluetooth Low Energy transmit current is measured at 3.3V (Buck mode) at 2440 MHz across temperature.
- Transmit current is measured after calibration at +12 dBm ( $\pm 0.5$  dBm).
- Current is measured on input power rail to SoC.

### 43.32 Zigbee RF Characteristics

**Table 43-42.** Zigbee RF Characteristics

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ. <sup>(1,2)</sup>	Max.	Units	Conditions
ZBG1	FREQ	Frequency range	2405	—	2480	MHz	—
ZBG2	FCH	Channel spacing	—	5	—	MHz	—
ZBG3	PSDU	Bit rate	—	250	—	kbps	—
ZBT1	TXOPMPA	Transmit output power MPA	—	11.67	—	dBm	—
ZBT2	TXOPLPA	Transmit output power LPA	—	3.95	—	dBm	—
ZBT3	POWERRANGE	Output power range	-14	—	12	dB	TX power on ZB power levels from (-14 to 12 dBm)
ZBT4	EVM	Error vector magnitude	—	10	—	%RMS	—
ZBT5	MPA2HAR	Second harmonic from MPA	—	-47.2	—	dBm	—
ZBT6	MPA3HAR	Third harmonic from MPA	—	-45.1	—	dBm	—
ZBT7	LPA2HAR	Second harmonic from LPA	—	-57.8	—	dBm	—
ZBT8	LPA3HAR	Third harmonic from LPA	—	-52.9	—	dBm	—
ZBRX1	SENS250	Receiver sensitivity in 250 kbps	—	-99	—	dBm	—
	SENS500	Receiver sensitivity in 500 kbps	—	-96	—	dBm	—
	SENS1M	Receiver sensitivity in 1 Mbps	—	-94	—	dBm	—
	SENS2M	Receiver sensitivity in 2 Mbps	—	-88	—	dBm	—
ZBRX2	PMAX	Maximum input level	—	0	—	dBm	—
ZBRX3	PACRP	Adjacent channel rejection +5 MHz	—	35	—	dB	—
ZBRX4	PACRN	Adjacent channel rejection -5 MHz	—	31	—	dB	—
ZBRX5	PALRP	Alternate channel rejection +10 MHz	—	47	—	dB	—
ZBRX6	PALRN	Alternate channel rejection -10 MHz	—	47	—	dB	—
ZBRX7	LOLEAK	LO leakage	—	-28/-34	—	dB	—
ZBRX8	RSSIRANGE	Dynamic range of RSSI	—	40	—	dB	—
ZBRX9	RSSIRES	Resolution of RSSI	—	1	—	dB	—

.....continued

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	Min.	Typ. <sup>(1,2)</sup>	Max.	Units	Conditions
ZBRX10	RSSIBASEVAL	Minimum value of RSSI	—	-103	—	dBm	—

**Notes:**

1. Measured on a board based on the Microchip Technology reference design.
2. Measured across channels at 3.3V and according to the 802.15.4 standard specifications.
3. Specified value is the margin above the 802.15.4 standard limits.
4. Measured across channels and voltages.
5. LO leakage on LPA mode, measured across voltage.
6. All results are based on measurement conditions as per the 802.15.4 standards.

**Table 43-43.** Zigbee LPA RF Characteristics

AC Characteristics			Standard Operating Conditions: $V_{DDIO} =$ Industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp	
Param. No.	Symbol	Characteristics	Min.	Typ
ZBLG1	FREQ	Frequency range	2405	—
ZBLG2	FCH	Channel spacing	—	5
ZBLG3	PSDU	Bitrate	—	250
ZBLT2	TXOPLPA	Transmit output power LPA	—	5.38
ZBLT3	POWERRANGE	Output power range	-16	—
ZBLT4	EVM	Error vector magnitude	—	10
ZBLT7	LPA2HAR	Second harmonic from LPA	—	-53
ZBLT8	LPA3HAR	Third harmonic from LPA	—	-49
ZBLRX1	SENS	Receiver sensitivity in 250 kbps	—	-101
		Receiver sensitivity in 500 kbps	—	-97
		Receiver sensitivity in 1 Mbps	—	-95
		Receiver sensitivity in 2 Mbps	—	-89
ZBLRX2	PMAX	Maximum input level	—	0
ZBLRX7	LOLEAK	LO leakage	—	-34
ZBLRX8	RSSIRANGE	Dynamic range of RSSI	—	40

**Notes:**

1. Measured on a board based on the Microchip Technology reference design.
2. Measured across channels at 3.3V and according to the 802.15.4 standard specifications.
3. Specified value is the margin above the 802.15.4 standard limits.
4. Measured across channels and voltages.
5. LO leakage on LPA mode, measured across voltage.
6. All results are based on measurement conditions as per the 802.15.4 standards.

**Table 43-44.** Zigbee RF Current Characteristics

AC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	CPU Frequency	Min.	Typ.	Max.	Units	Conditions
IZBTX1	IDDTXMPA	Current consumption at +12 dBm output power in DC-DC mode 250 kbps	64 MHz	—	43.3	—	—	—
IZBTX4		Current consumption at +12 dBm output power in MLDO mode	64 MHz	—	96.4	—	—	—
IZBTX7	IDDTXLPA	Current consumption at +4 dBm output power in DC-DC mode 250 kbps	64 MHz	—	27.3	—	—	—
IZBTX10		Current consumption at +12 dBm output power in MLDO mode	64 MHz	—	51.7	—	—	—
IZBRX1	IDDRXZB	Current consumption at RX signal level -95 dBm in DC-DC mode	64 MHz	—	19.4	—	—	—
IZBRX4		Current consumption at RX signal level -95 dBm in MLDO mode	64 MHz	—	38.5	—	—	—
IZBRX1	IDDRXZBRPC	Current consumption at RX signal level -95 dBm in DC-DC mode	64 MHz	—	13.6	—	—	—
IZBRX4		Current consumption at RX signal level -95 dBm in MLDO mode	64 MHz	—	29	—	—	—

**Notes:**

- Current consumption is measured on a board based upon the Microchip Technology Reference Design.
- Current consumption is for the entire SoC (including the MCU).
- Current reported is the average of the current during the transmit burst (exclude off cycle of the transmission).

**Table 43-45.** Zigbee RF Current LPA Characteristics

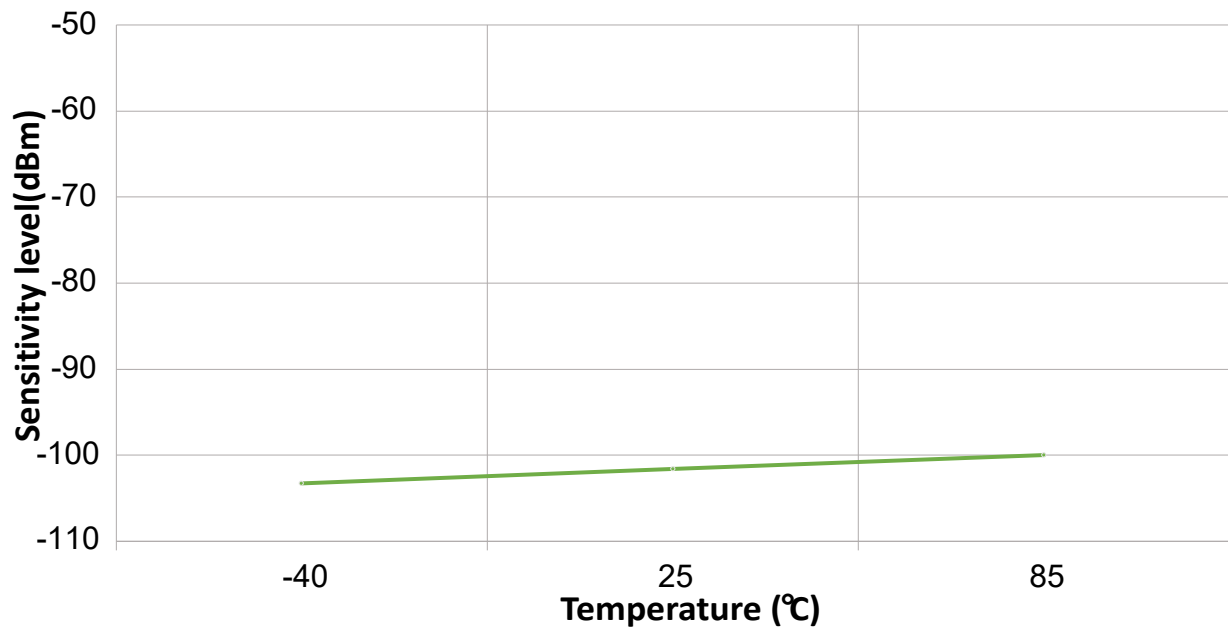
AC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V (unless otherwise stated) Operating Temperature: $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial Temp $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for Extended Temp				
Param. No.	Symbol	Characteristics	CPU Frequency	Min.	Typ.	Max.	Units	Conditions
IZBTX1	IDDTXMPA	Current consumption at +5.5 dBm output power in DC-DC mode 250 kbps	64 MHz	—	24.5	—	mA	—
IZBTX4		Current consumption at +5.5 dBm output power in MLDO mode	64 MHz	—	47.4	—	mA	—
IZBTX7	IDDTXLPA	Current consumption at 0 dBm output power in DC-DC mode 250 kbps	64 MHz	—	22.3	—	mA	—
IZBTX10		Current consumption at 0 dBm output power in MLDO mode	64 MHz	—	39.9	—	mA	—



.....continued

AC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA} = 1.9V$ (otherwise stated) Operating Temperature: $-40^{\circ}C \leq T_A \leq 125^{\circ}C$ Industrial Temp $-40^{\circ}C \leq T_A \leq +125^{\circ}C$ for Extended Temp			
Param. No.	Symbol	Characteristics	CPU Frequency	Min.	Typ.	Max.	Units
IZBRX1	IDDRXZB	Current consumption at Rx signal level -95 dBm in DC-DC mode	64 MHz	—	21.3	—	mA
IZBRX4	IDDRXZB	Current consumption at Rx signal level -95 dBm in MLDO mode	64 MHz	—	39.5	—	mA
IZBRX1	IDDRXZBRPC	Current consumption at Rx signal level -95 dBm in DC-DC mode	64 MHz	—	14.8	—	mA
IZBRX4	IDDRXZBRPC	Current consumption at Rx signal level -95 dBm in MLDO mode	64 MHz	—	29.7	—	mA

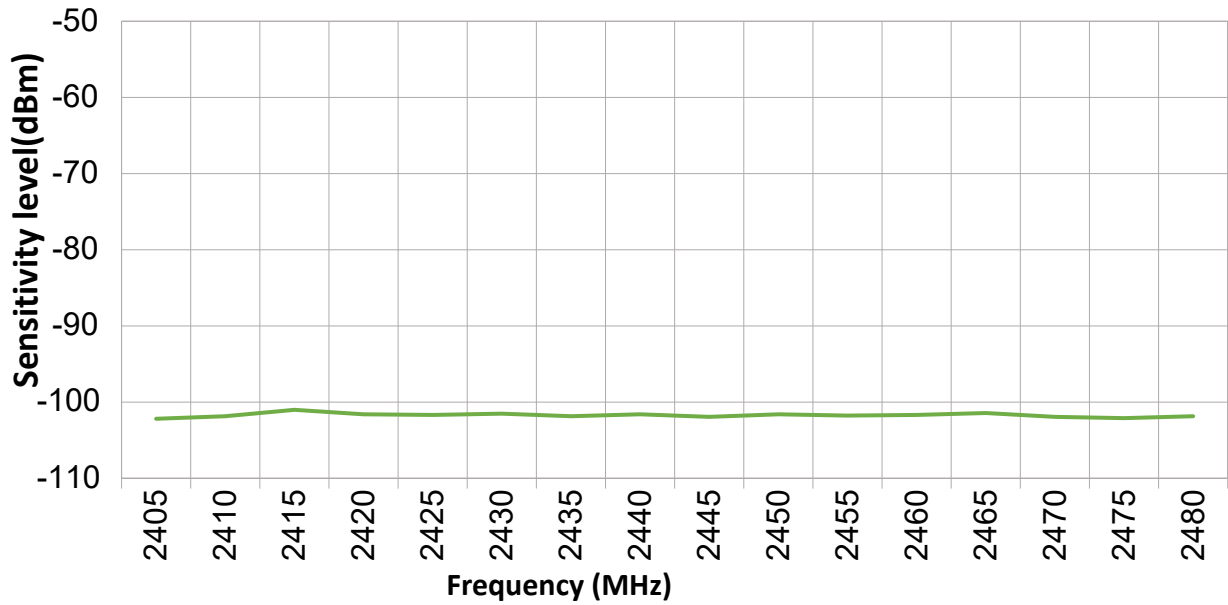
Figure 43-42. Module Zigbee Receive Sensitivity vs. Temperature



**Notes:**

- Receiver sensitivity is measured based on the 802.15.4 specifications.
- Receiver sensitivity is measured at 2440 MHz at 3.6V, 250 kbps.
- Measured after receiver calibration.

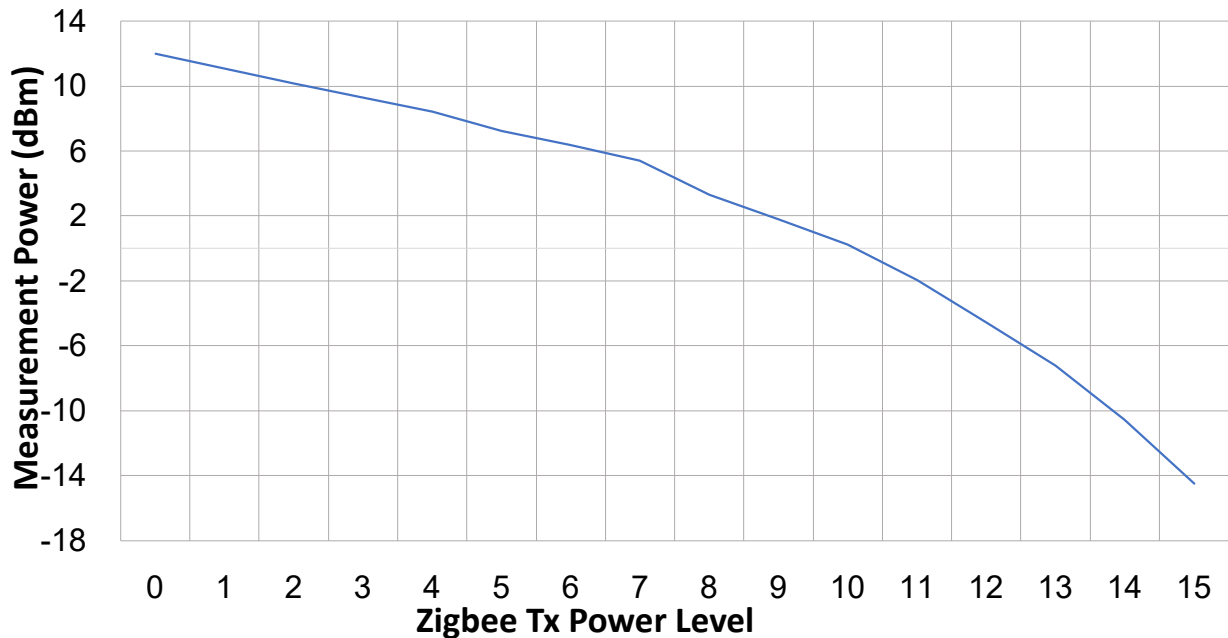
Figure 43-43. Module Zigbee Receive Sensitivity vs. Frequency



**Notes:**

- RX sensitivity across channels is measured at 3.6V at 25°C, 250 kbps.
- Sensitivity is measured according to the 802.15.4 specifications.

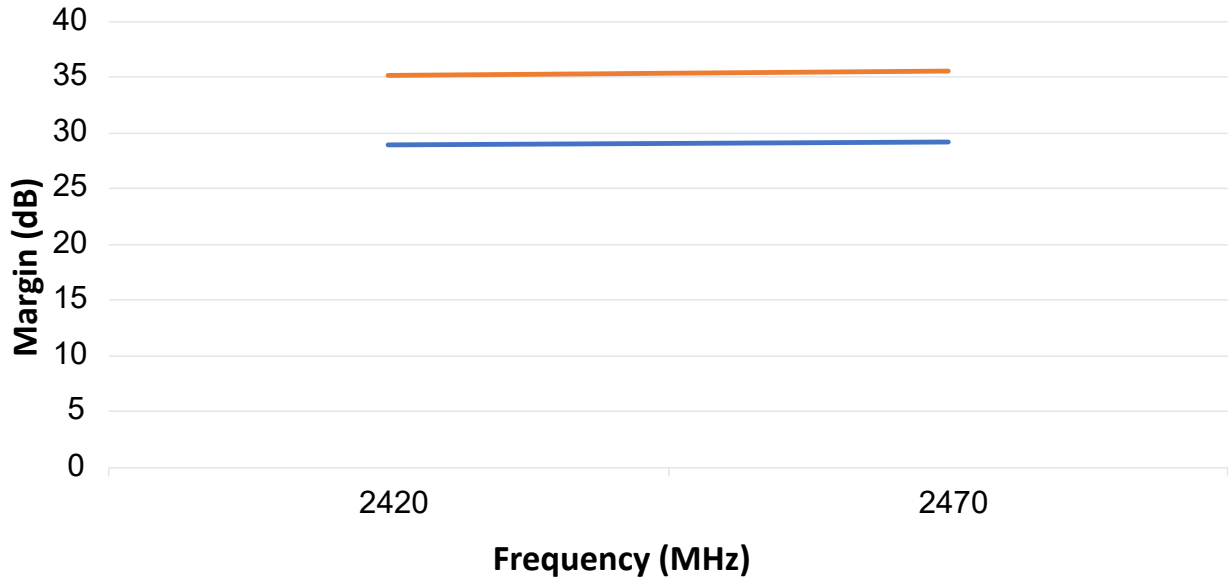
Figure 43-44. Zigbee TX Setting vs. Measurement Power



**Notes:**

- Transmit power is measured after calibration.
- Transmit power is measured across power levels at 2440 MHz at 3.6V, 25°C.
- Transmit power is measured after the PA matching and LPF.

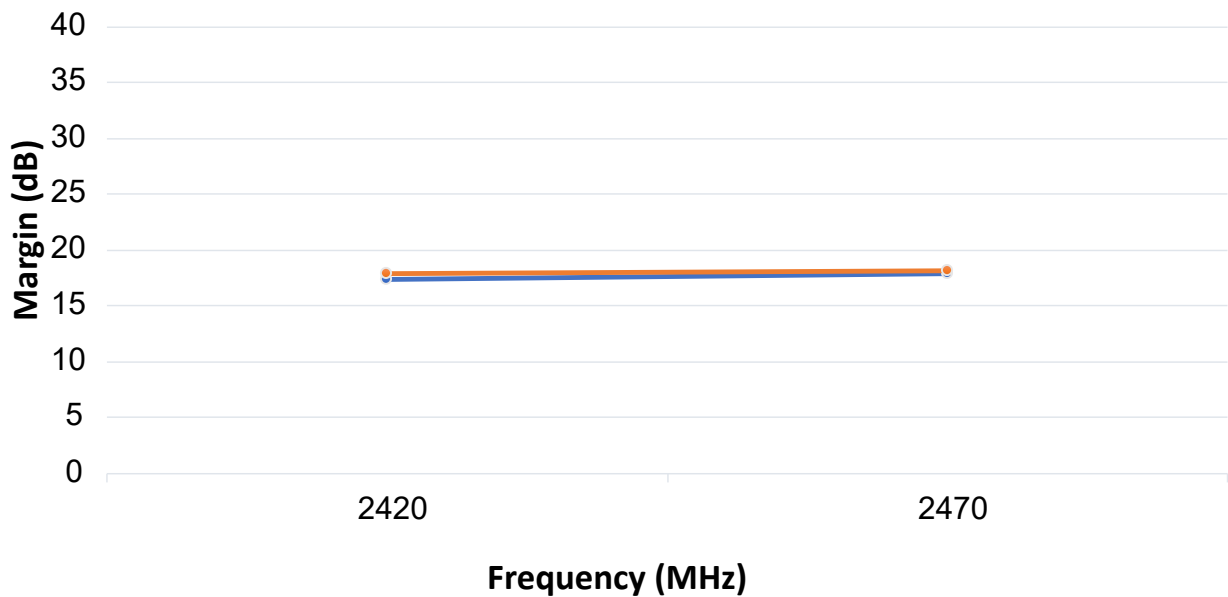
Figure 43-45. Zigbee ACR +-5M Margin



**Notes:**

- Adjacent channel rejection is measured at 3.6V at 25°C, 250 kbps.
- Measured based on the 802.15.4 adjacent channel relative jamming specification.
- Margin specified is the margin above the 802.15.4 specifications.
- Measured after receiver calibration.

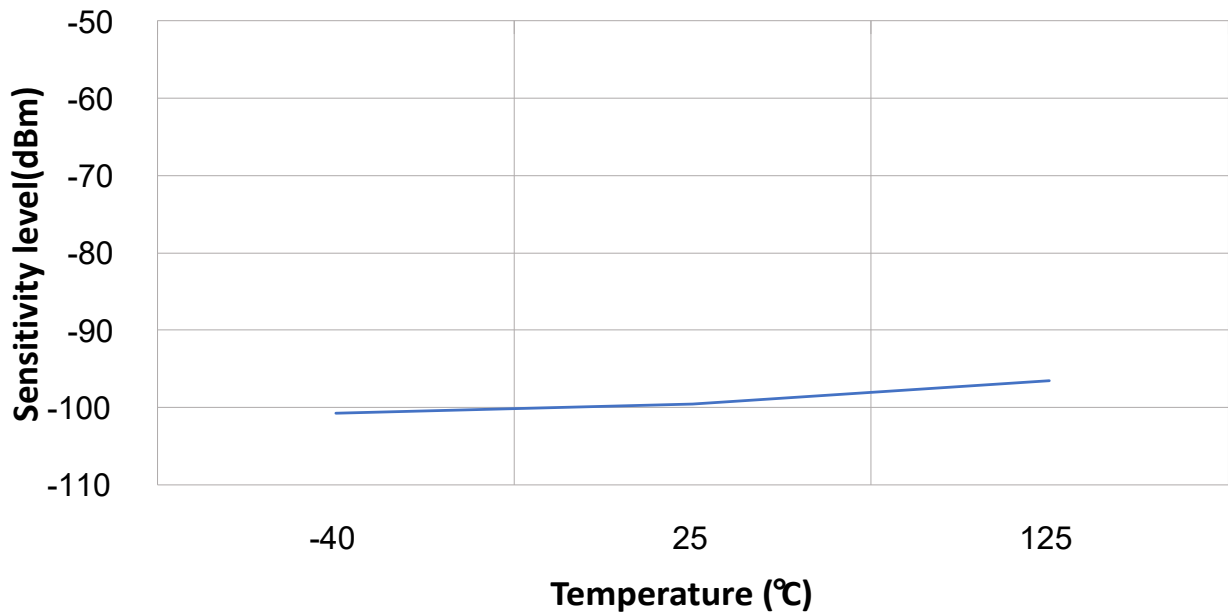
Figure 43-46. Zigbee ACR +-10M Margin



**Notes:**

- Adjacent channel rejection is measured at 3.6V at 25°C, 250 kbps.
- Measured based on the 802.15.4 adjacent channel relative jamming specification.
- Margin specified is the margin above the 802.15.4 specifications.
- Measured after receiver calibration.

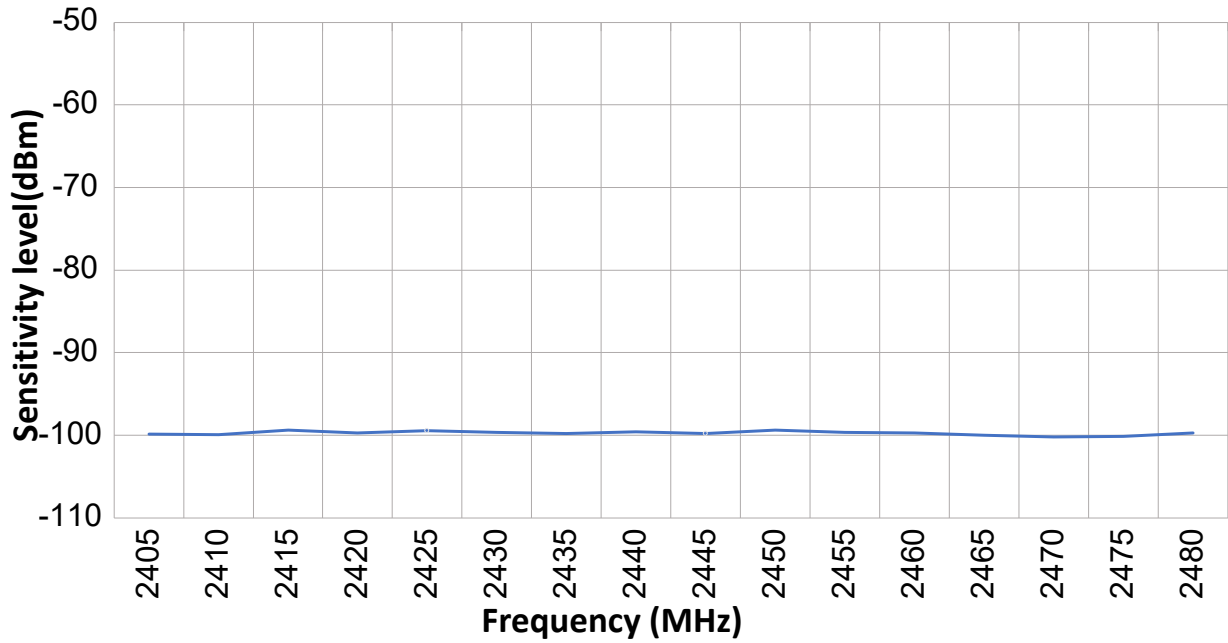
**Figure 43-47.** Zigbee Receive Sensitivity vs. Temperature



**Notes:**

- Receiver sensitivity is measured based on the 802.15.4 specifications.
- Sensitivity measured at 3.6V, 25°C on 2440 MHz across temperature, 250 kbps.
- Measured after receiver calibration.

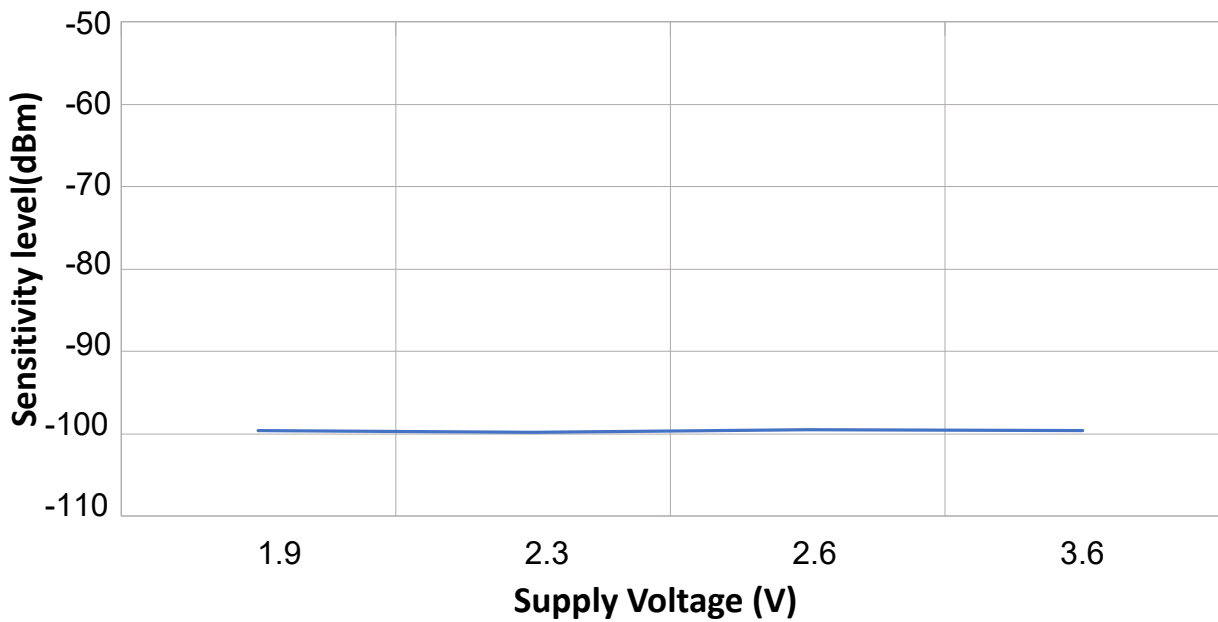
Figure 43-48. Zigbee Receive Sensitivity vs. Frequency



**Notes:**

- RX sensitivity is measured across channels at 3.6V at 25°C, 250 kbps.
- Sensitivity is measured according to the 802.15.4 specifications.
- Sensitivity is measured after RX calibration.

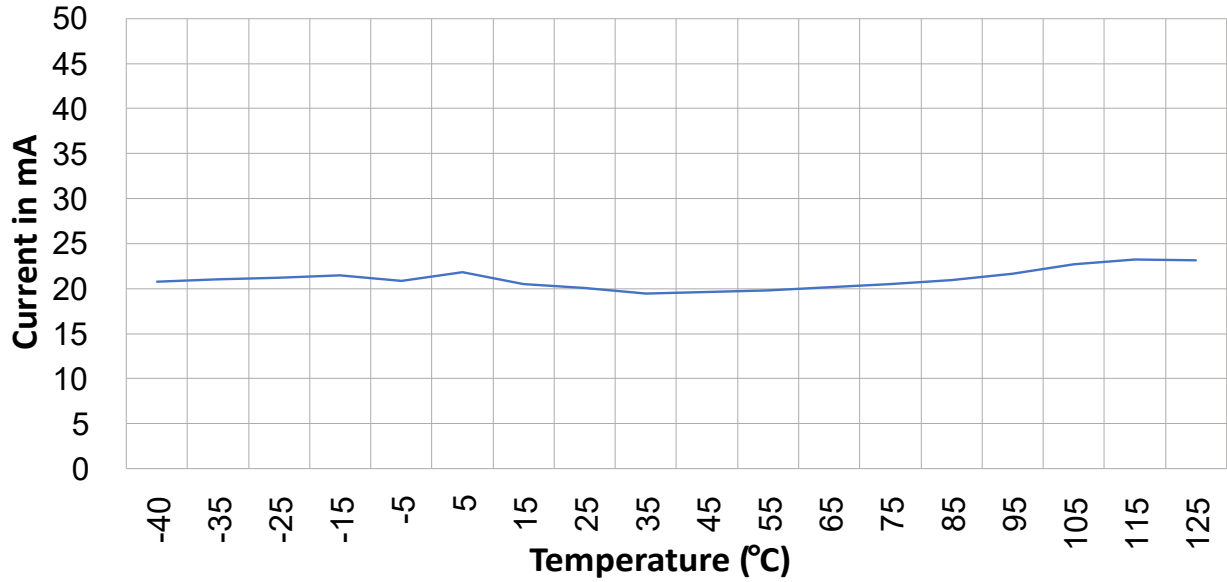
Figure 43-49. Zigbee Receive Sensitivity vs. VDD Supply Voltage



**Notes:**

- RX sensitivity is measured at 2440 MHz at 25°C across voltage, 250 kbps.
- Sensitivity is measured according to the 802.15.4 specifications.
- Sensitivity is measured after RX calibration.

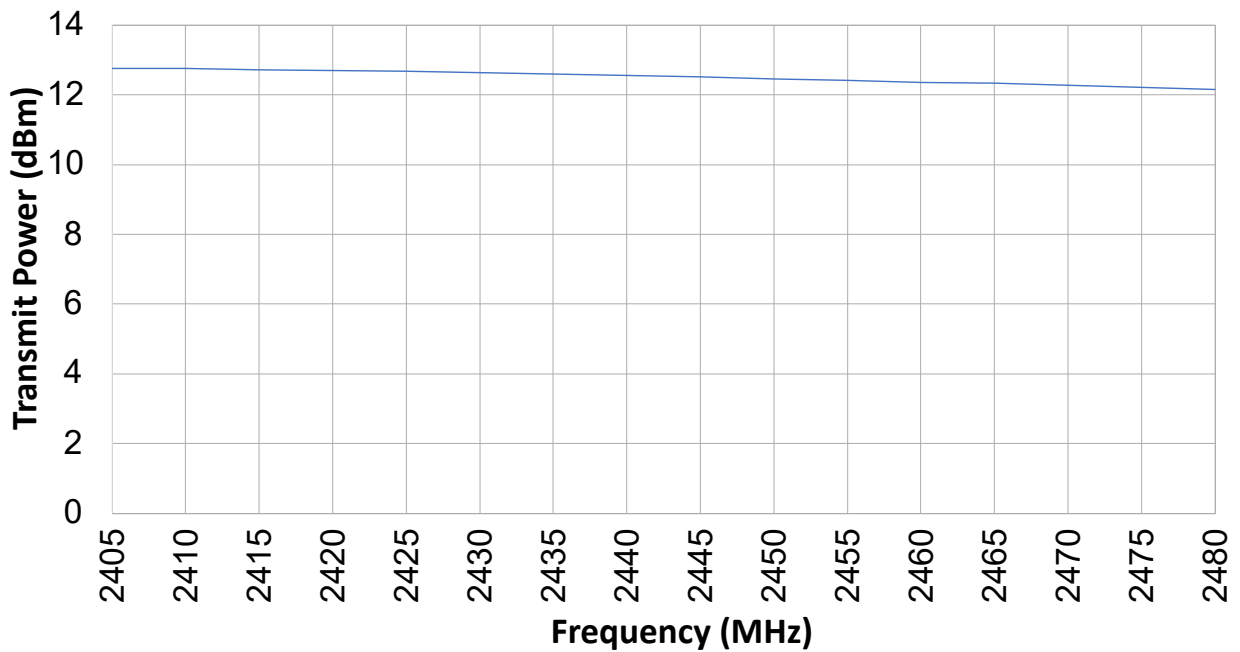
**Figure 43-50.** Zigbee Receive Current vs. Temperature



**Notes:**

- Receiver operating at 2440 MHz, 3.3V, 25°C at maximum LNA gain.
- Measured after receiver calibration.

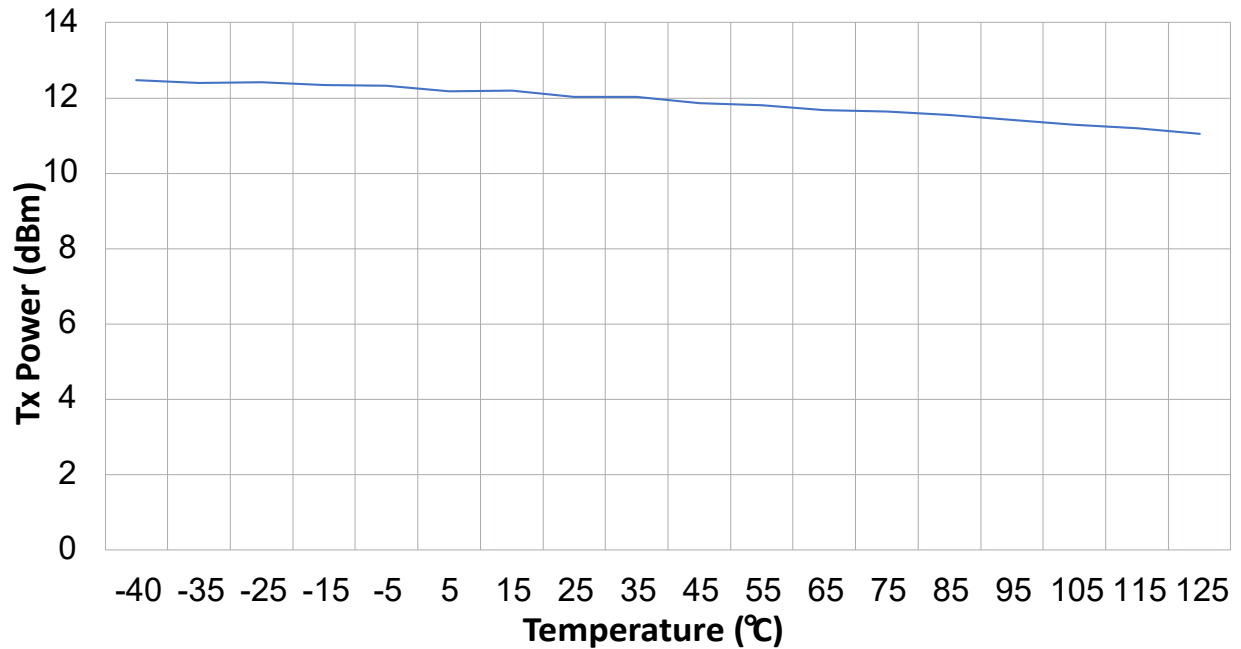
**Figure 43-51.** Zigbee Transmit Power vs Frequency



**Notes:**

- Transmit power is measured after calibration.
- Transmit power is measured across the channels at 3.6V at 25°C.
- Transmit power is measured after the PA matching and LPF.

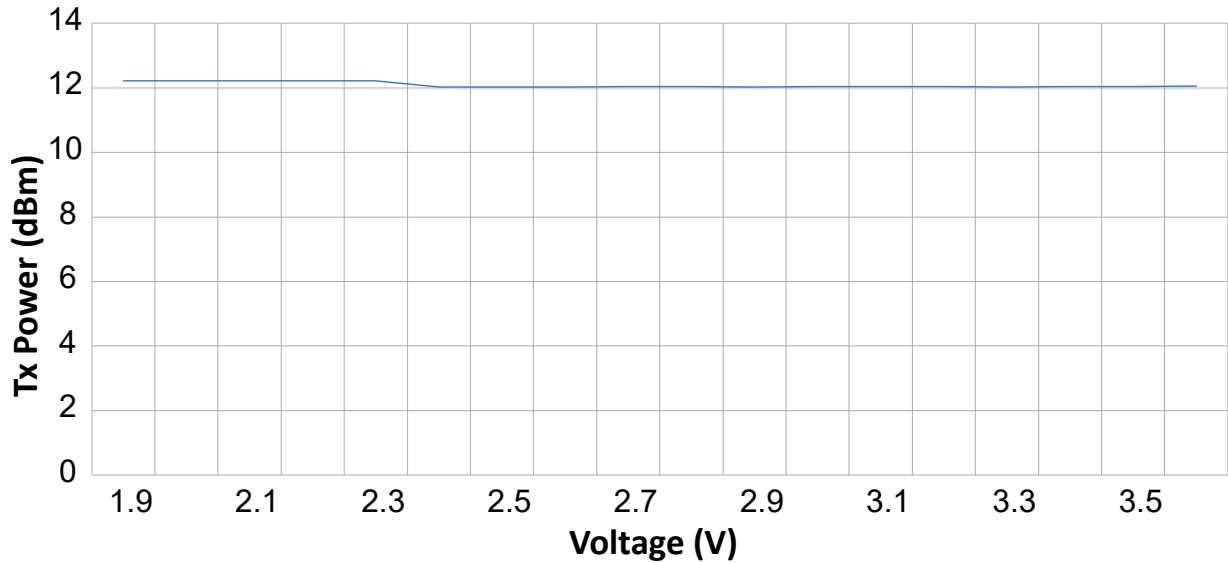
**Figure 43-52.** Zigbee Transmit Power vs Temperature



**Notes:**

- Transmit power is measured after calibration.
- Transmit power is measured at 2440 MHz at 3.6V across temperature.
- Transmit power is measured after the PA matching and LPF.
- Transmit power compensation is triggered before measurement across temperature.

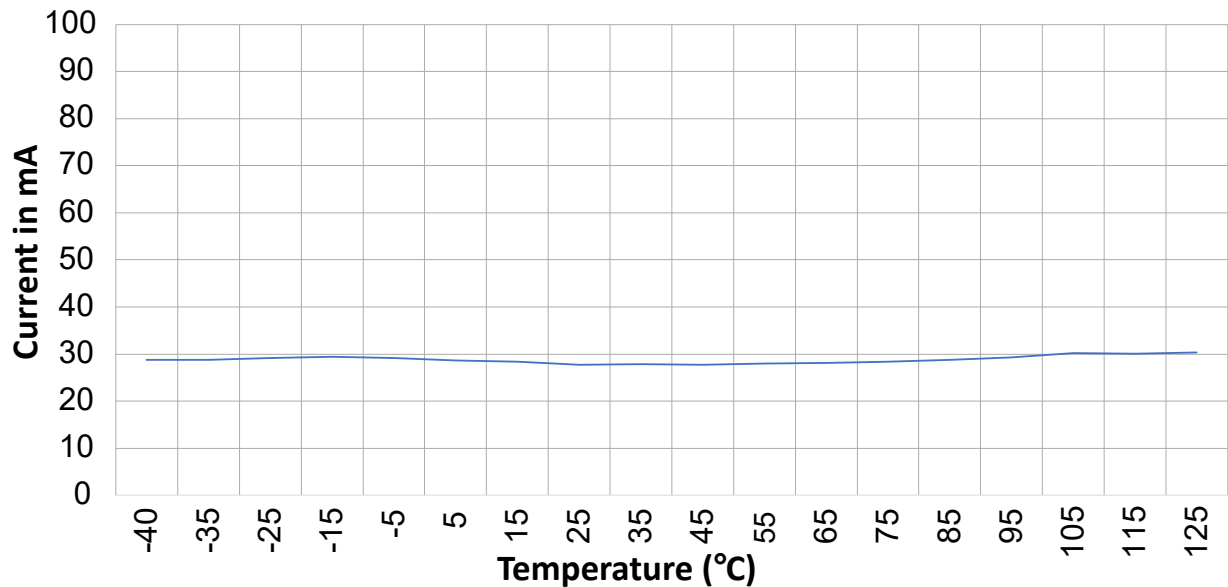
Figure 43-53. Zigbee Transmit Power vs. VDD Supply Voltage



**Notes:**

- Transmit power is measured after calibration.
- Transmit power is measured across voltage at 2440 MHz and 25°C.
- Transmit power is measured on reference board after the PA matching and LPF.

Figure 43-54. Zigbee Transmit Current vs. Temperature



**Notes:**

- Transmit current is measured at input to SoC (includes SoC power consumption).
- Transmit current is measured at 2440 MHz at 3.3V (Buck mode).
- Transmit power is calibrated to +12 dBm ( $\pm 0.5$  dBm on MPA mode).



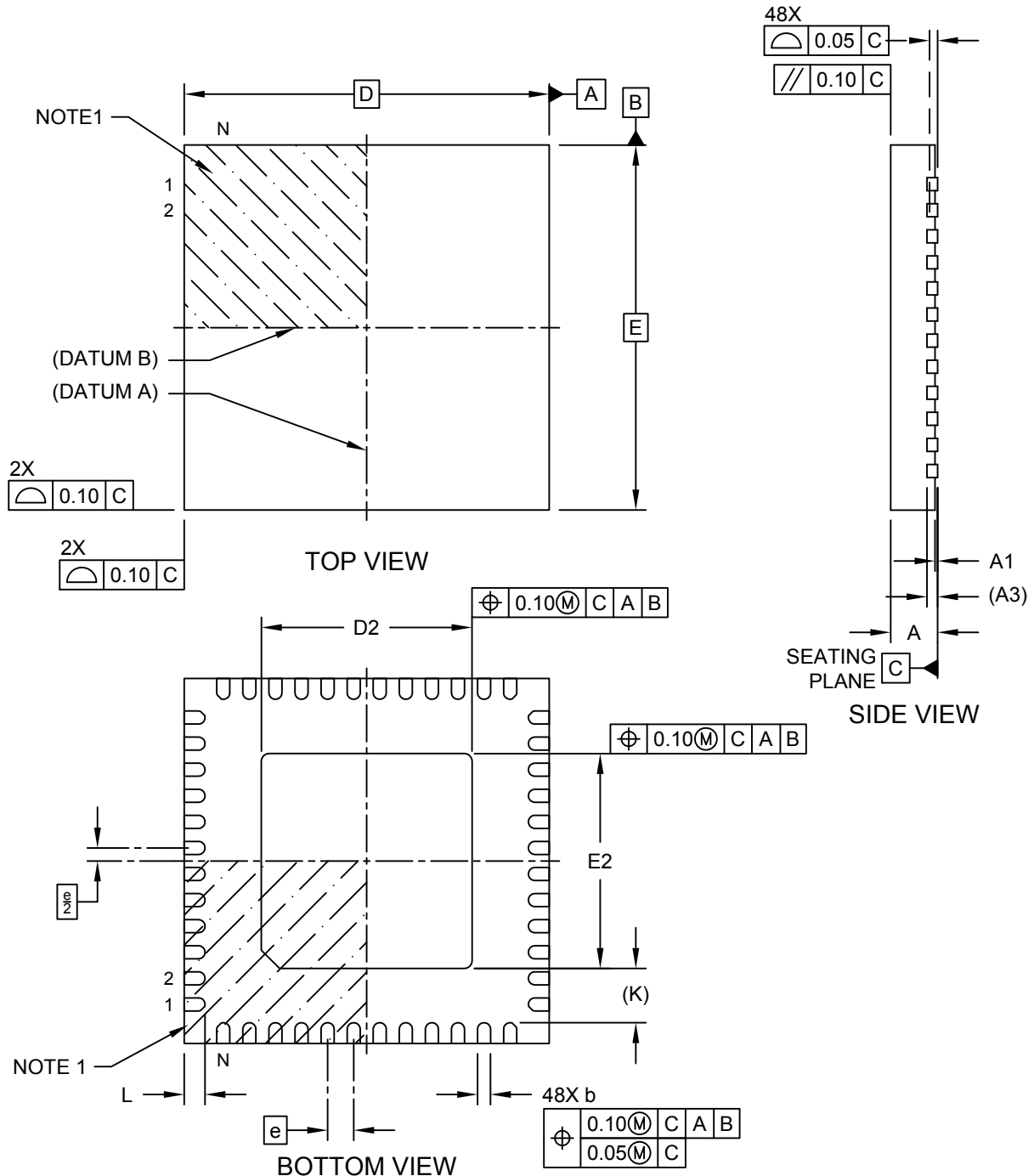
## 44. Packaging Information

This chapter provides the information on package markings, dimension and footprint of the PIC32CX-BZ2 and WBZ45 family.

## 44.1 PIC32CX1012BZ25048 SoC Packaging Information

### 48-Lead Very Thin Quad Flat, No Lead Package (MYX) 7x7x0.9 mm Body [VQFN] With 4.04x4.12 mm Exposed Pad

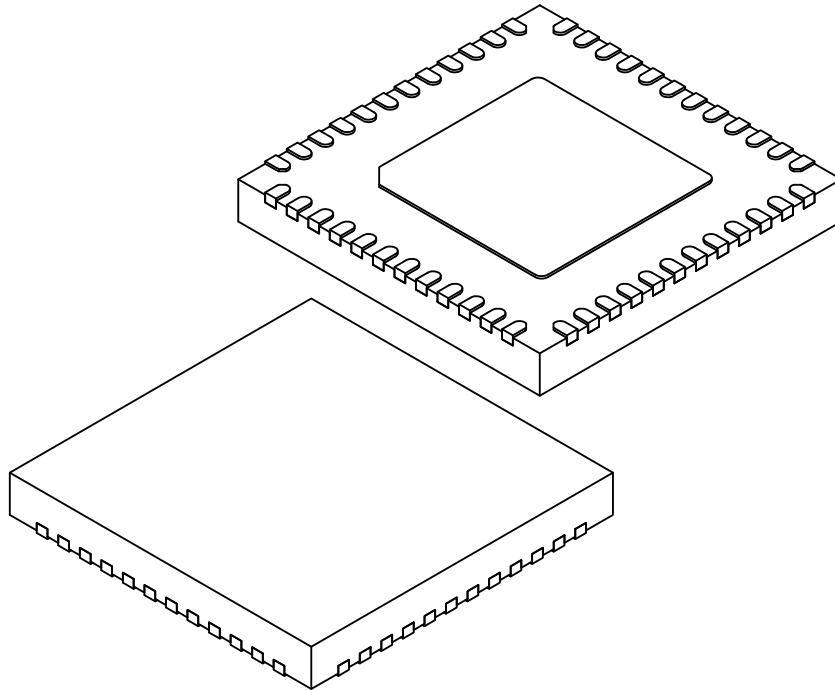
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Microchip Technology Drawing C04-507 Rev A Sheet 1 of 2

**48-Lead Very Thin Quad Flat, No Lead Package (MYX) 7x7x0.9 mm Body [VQFN]  
With 4.04x4.12 mm Exposed Pad**

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Number of Terminals	N	48		
Pitch	e	0.50 BSC		
Overall Height	A	0.80	0.85	0.90
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3	0.20 REF		
Overall Length	D	7.00 BSC		
Exposed Pad Length	D2	3.94	4.04	4.14
Overall Width	E	7.00 BSC		
Exposed Pad Width	E2	4.02	4.12	4.22
Terminal Width	b	0.18	0.25	0.30
Terminal Length	L	0.35	0.40	0.45
Terminal-to-Exposed-Pad	K	1.04 REF		

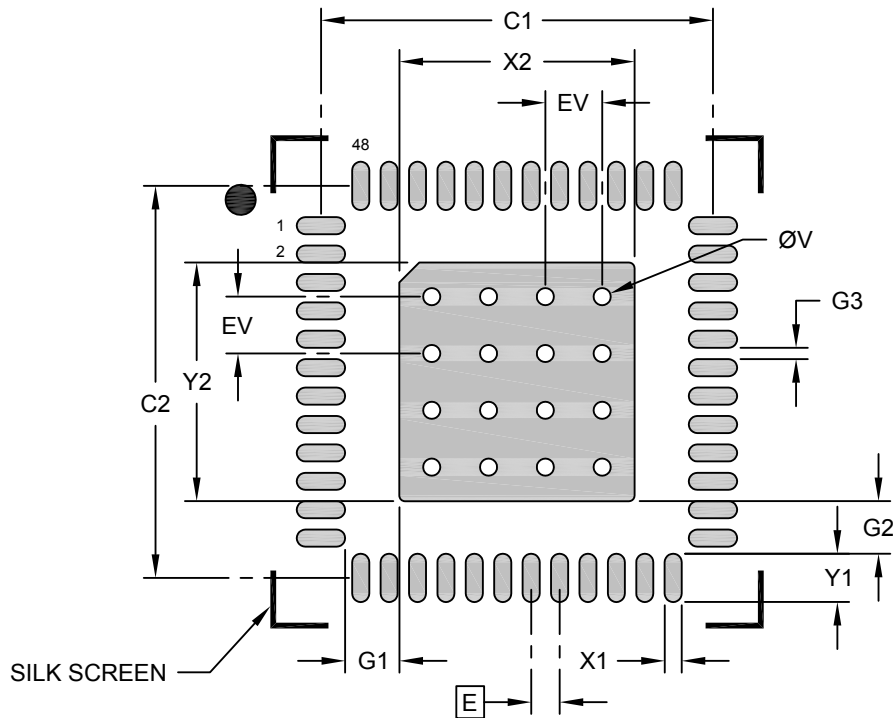
**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M  
 BSC: Basic Dimension. Theoretically exact value shown without tolerances.  
 REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-507 Rev A Sheet 2 of 2

**48-Lead Very Thin Quad Flat, No Lead Package (MYX) 7x7x0.9 mm Body [VQFN]  
With 4.04x4.12 mm Exposed Pad**

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



**RECOMMENDED LAND PATTERN**

Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Contact Pitch	E	0.50 BSC		
Optional Center Pad Width	X2			4.14
Optional Center Pad Length	Y2			4.20
Contact Pad Spacing	C1		6.90	
Contact Pad Spacing	C2		6.90	
Contact Pad Width (X48)	X1			0.30
Contact Pad Length (X48)	Y1			0.85
Contact Pad to Center Pad (X24)	G1	0.96		
Contact Pad to Center Pad (X24)	G2	0.93		
Contact Pad to Contact Pad (X44)	G3	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

**Notes:**

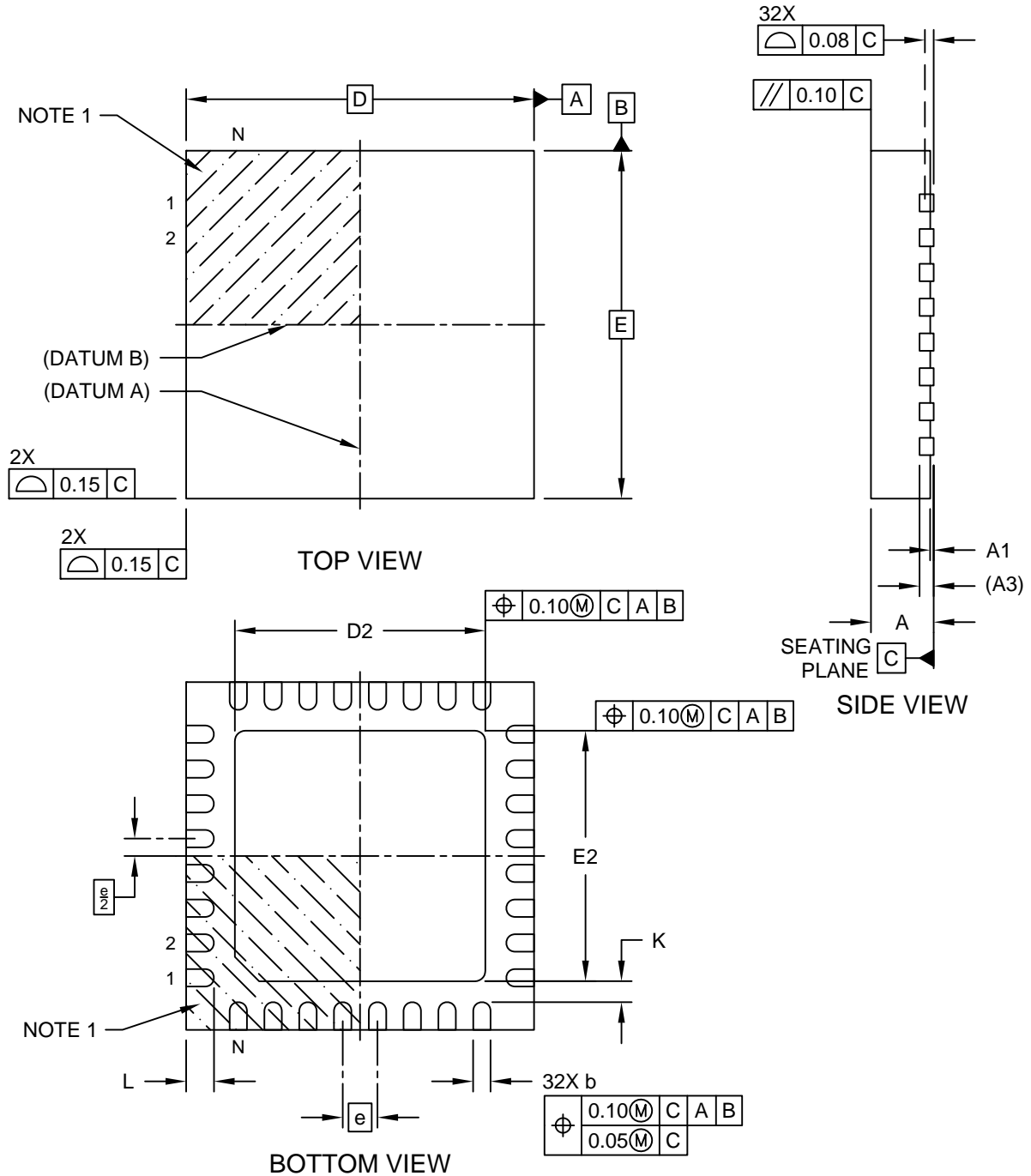
- Dimensioning and tolerancing per ASME Y14.5M  
BSC: Basic Dimension. Theoretically exact value shown without tolerances.
- For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2507 Rev A

## 44.2 PIC32CX1012BZ24032 SoC Packaging Information

### 32-Lead Very Thin Plastic Quad Flat, No Lead Package (S8B) - 5x5 mm Body [VQFN] With 3.60 mm Exposed Pad; Atmel Legacy Global Package Code ZKV

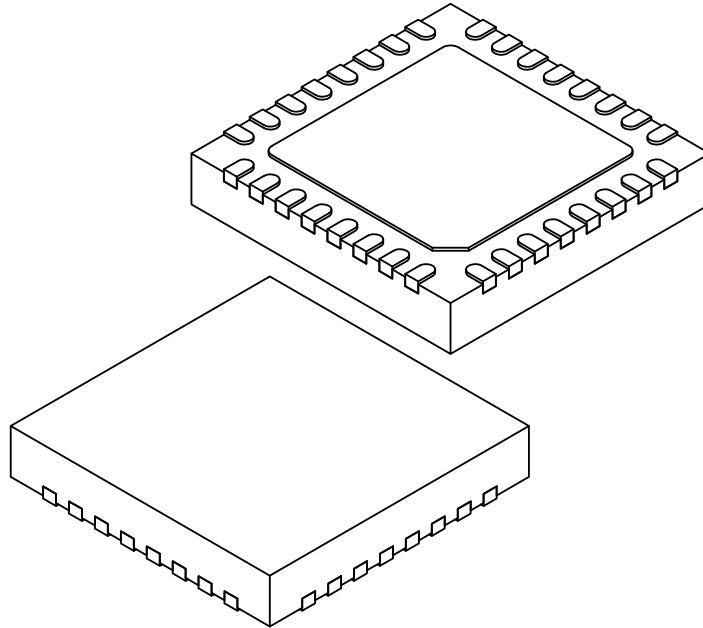
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Microchip Technology Drawing C04-21402 Rev A Sheet 1 of 2

**32-Lead Very Thin Plastic Quad Flat, No Lead Package (S8B) - 5x5 mm Body [VQFN]  
With 3.60 mm Exposed Pad; Atmel Legacy Global Package Code ZKV**

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Number of Terminals	N	32		
Pitch	e	0.50 BSC		
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3	0.20 REF		
Overall Length	D	5.00 BSC		
Exposed Pad Length	D2	3.50	3.60	3.70
Overall Width	E	5.00 BSC		
Exposed Pad Width	E2	3.50	3.60	3.70
Terminal Width	b	0.18	0.25	0.30
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K	0.20	-	-

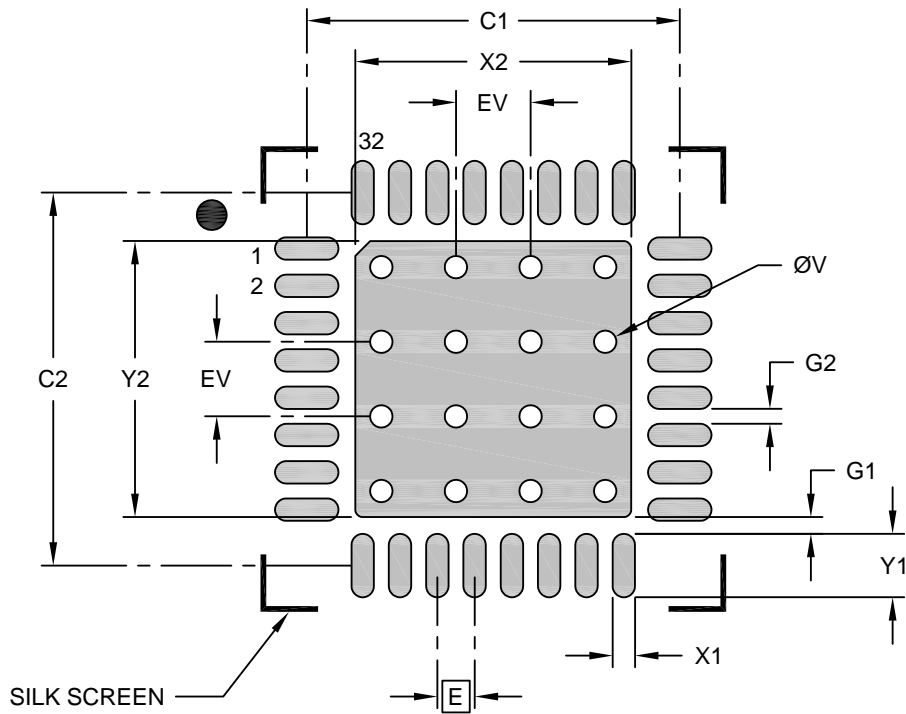
Notes:

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Package is saw singulated
- Dimensioning and tolerancing per ASME Y14.5M  
BSC: Basic Dimension. Theoretically exact value shown without tolerances.  
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-21402 Rev A Sheet 1 of 2

**32-Lead Very Thin Plastic Quad Flat, No Lead Package (S8B) - 5x5 mm Body [VQFN]  
With 3.60 mm Exposed Pad; Atmel Legacy Global Package Code ZKV**

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



**RECOMMENDED LAND PATTERN**

Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Contact Pitch	E	0.50 BSC		
Optional Center Pad Width	X2			3.70
Optional Center Pad Length	Y2			3.70
Contact Pad Spacing	C1		5.00	
Contact Pad Spacing	C2		5.00	
Contact Pad Width (X32)	X1			0.30
Contact Pad Length (X32)	Y1			0.85
Contact Pad to Center Pad (X32)	G1	0.23		
Contact Pad to Contact Pad (X28)	G2	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

**Notes:**

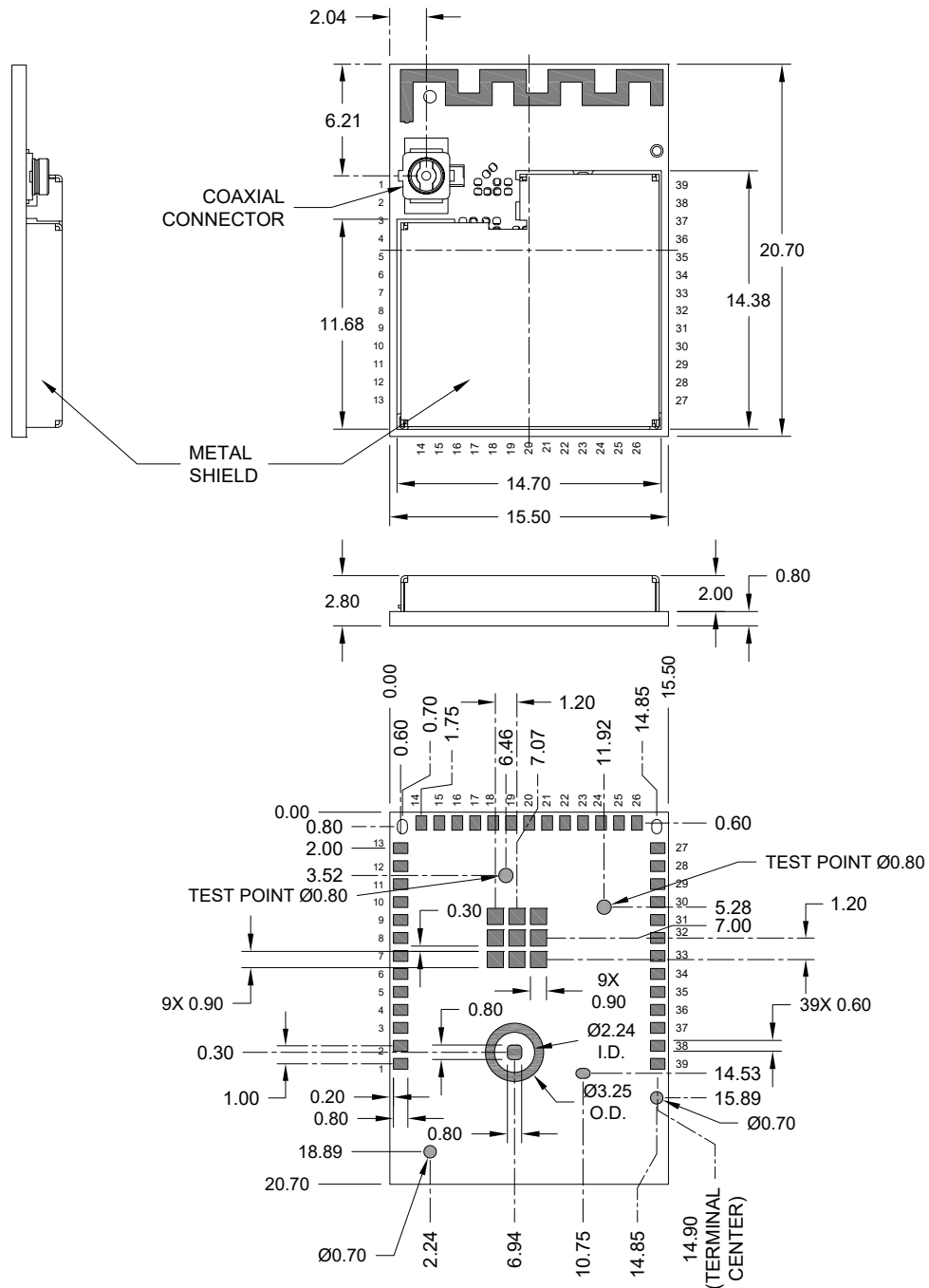
1. Dimensioning and tolerancing per ASME Y14.5M  
BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-23402 Rev A

### 44.3 WBZ451 Module Packaging Information

#### 39-Lead PCB Module (ZSX) - 15.5x20.7x2.8 mm Body [MODULE] With Metal Shield and Coaxial Connector

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>

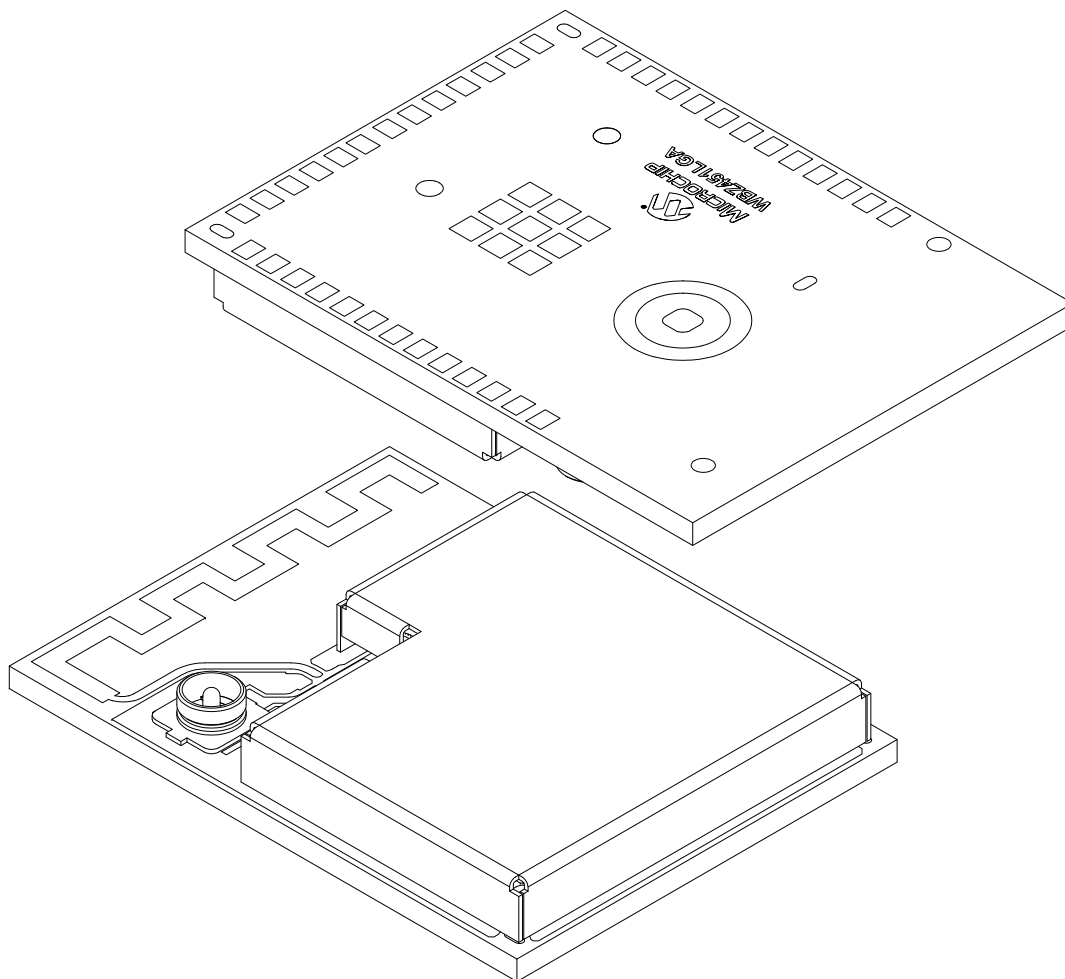


Microchip Technology Drawing C04-10052 Rev B Sheet 1 of 2



**39-Lead PCB Module (ZSX) - 15.5x20.7x2.8 mm Body [MODULE]  
With Metal Shield and Coaxial Connector**

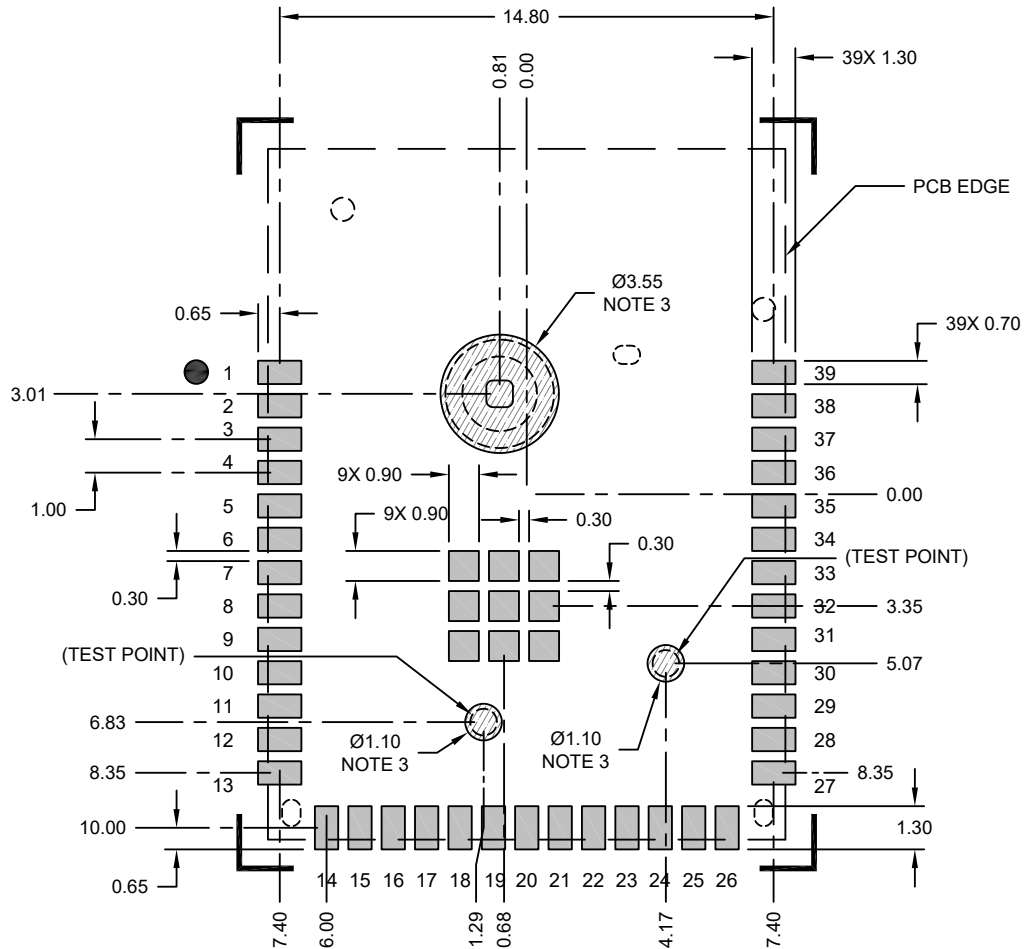
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>




Microchip Technology Drawing C04-10052 Rev B Sheet 2 of 2

### 39-Lead PCB Module (ZSX) - 15.5x20.7x2.8 mm Body [MODULE] With Metal Shield and Coaxial Connector

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



 COPPER KEEPOUT ZONE

#### RECOMMENDED LAND PATTERN

**Notes:**

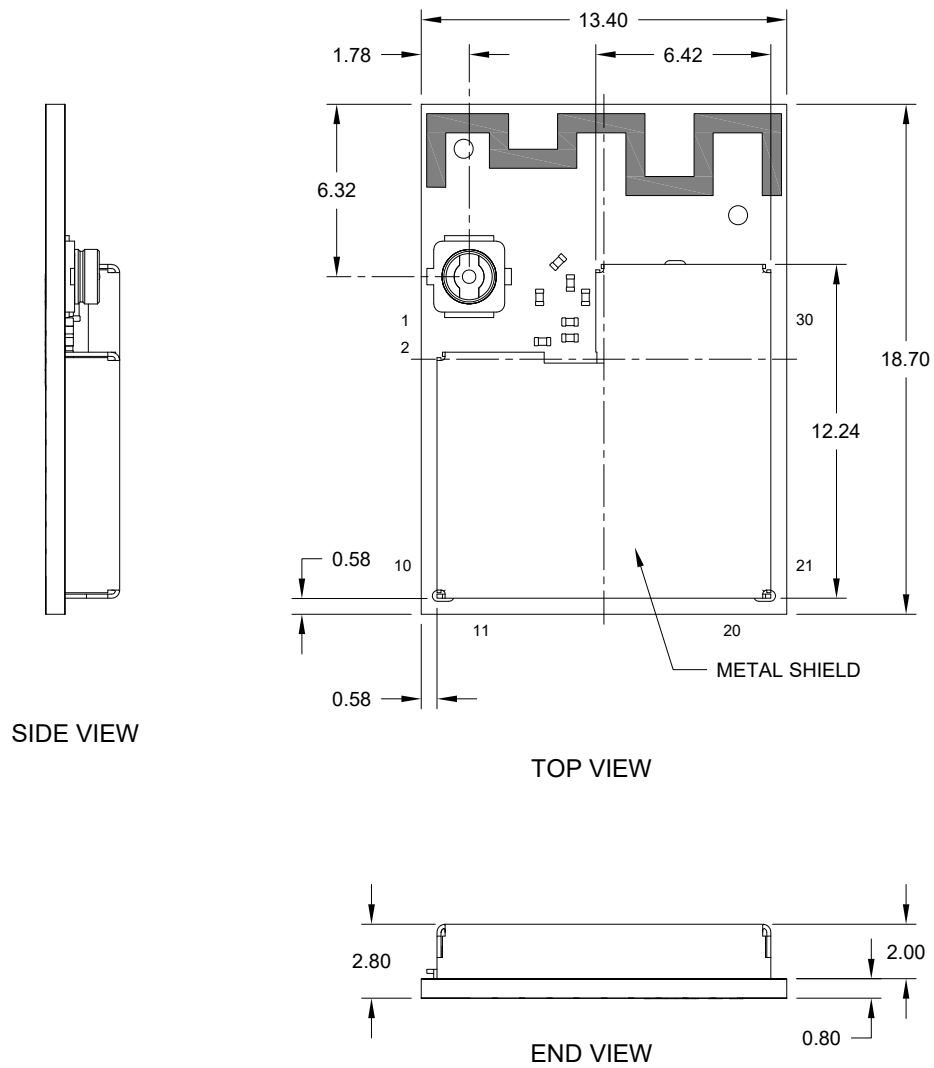
1. All dimensions are in millimeters.
2. Keep this area free from all metal, including ground fill.
3. Keep these areas free from routes and exposed copper. Ground fill with solder mask may be placed here.

Microchip Technology Drawing C04-12052 Rev B

## 44.4 WBZ450 Module Packaging Information

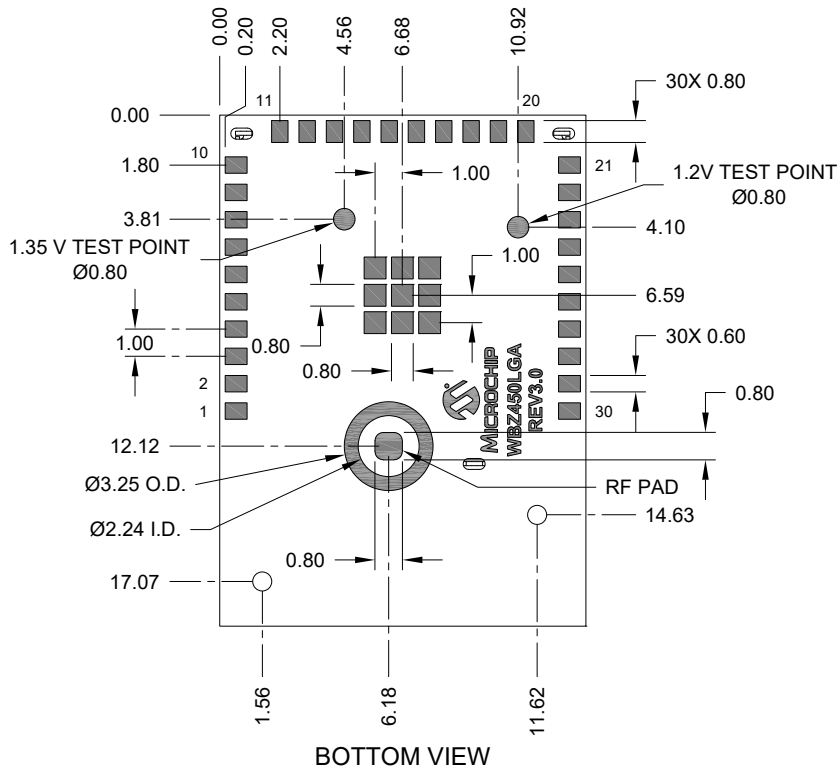
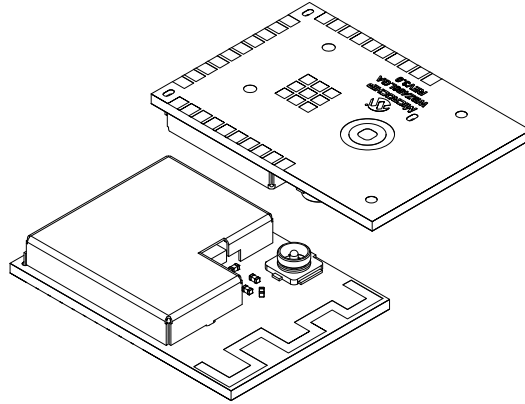
### 30-Lead PCB Module (ZRX) -13.4x18.7x2.8 mm Body [MODULE] For WBZ450 With Metal Shield and Coaxial Connector

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



**30-Lead PCB Module (ZRX) -13.4x18.7x2.8 mm Body [MODULE] For WBZ450  
With Metal Shield and Coaxial Connector**

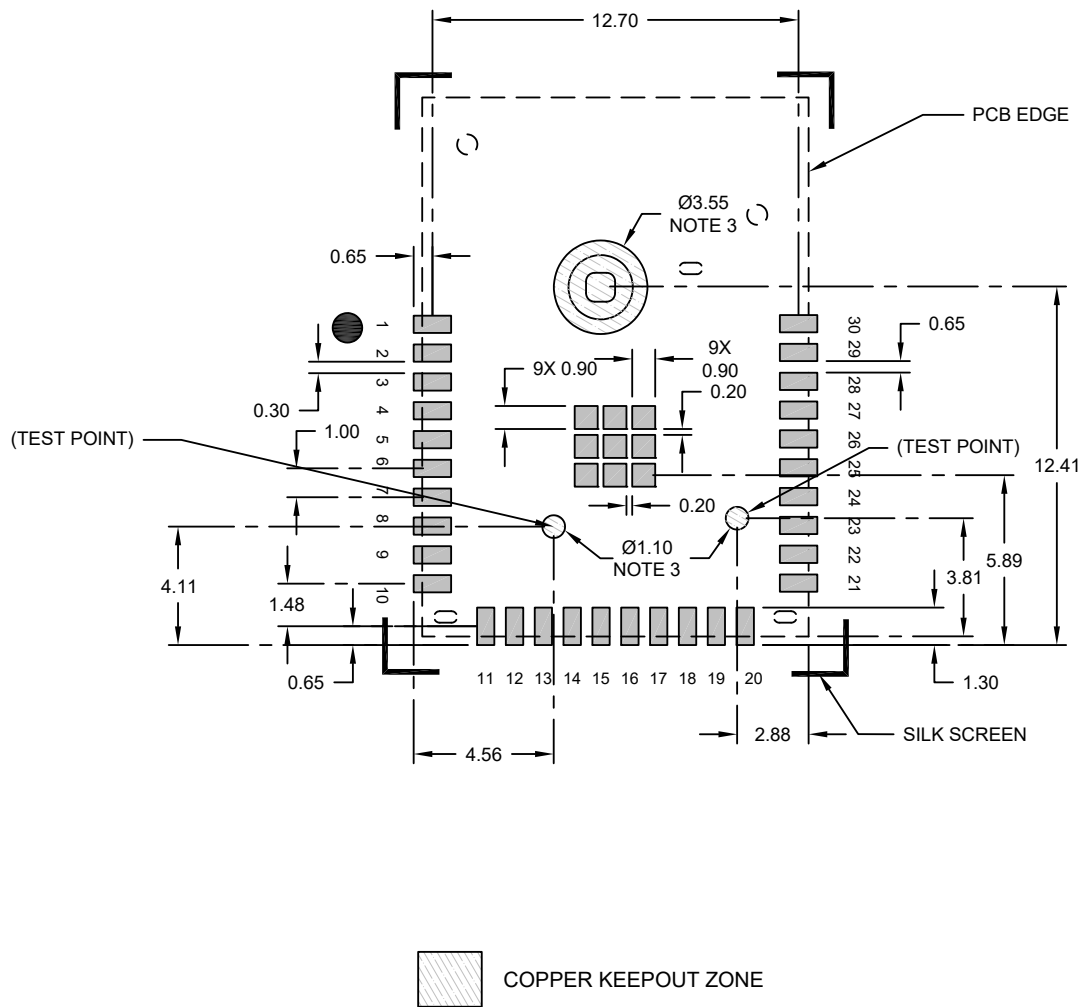
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Notes:

**30-Lead PCB Module (ZRX) - 13.4x18.7X2.8 mm Body [Module] For WBZ450  
With Metal Shield and Coaxial Connector**

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



**RECOMMENDED LAND PATTERN**

**Notes:**

1. All dimensions are in millimeters.
2. Keep this area free from all metal, including ground fill.
3. Keep these areas free from routes and exposed copper. Ground fill with solder mask may be placed here.

Microchip Technology Drawing C04-12051 Rev A

## 45. Appendix A: Regulatory Approval

The WBZ450PC module has received regulatory approval for the following countries:

- Bluetooth Special Interest Group (SIG) QDID: D056857
- United States/FCC ID: 2ADHKWBZ450
- Canada/ISED:
  - IC: 20266-WBZ450
  - HVIN: WBZ450PC
- Europe/CE
- Japan/MIC
- Korea/KCC
- Taiwan/NCC

The WBZ450PE module has received regulatory approval for the following countries:

- Bluetooth Special Interest Group (SIG) QDID: D056857
- United States/FCC ID: 2ADHKWBZ450
- Canada/ISED:
  - IC: 20266-WBZ450
  - HVIN: WBZ450PE
- Europe/CE
- Japan/MIC
- Korea/KCC
- Taiwan/NCC
- China: 2023DJ2426

The WBZ450UC module has received regulatory approval for the following countries:

- Bluetooth Special Interest Group (SIG) QDID: D056857
- United States/FCC ID: 2ADHKWBZ450
- Canada/ISED:
  - IC: 20266-WBZ450
  - HVIN: WBZ450UC
- Europe/CE
- Japan/MIC
- Korea/KCC
- Taiwan/NCC

The WBZ450UE module has received regulatory approval for the following countries:

- Bluetooth Special Interest Group (SIG) QDID: D056857
- United States/FCC ID: 2ADHKWBZ450
- Canada/ISED:
  - IC: 20266-WBZ450
  - HVIN: WBZ450PE
- Europe/CE
- Japan/MIC

- Korea/KCC
- Taiwan/NCC
- China: 2023DJ2427(M)

The WBZ451PC module has received regulatory approval for the following countries:

- Bluetooth Special Interest Group (SIG) QDID: D056857
- United States/FCC ID: 2ADHKWBZ451
- Canada/ISED:
  - IC: 20266-WBZ451
  - HVIN: WBZ451PC
- Europe/CE
- Japan/MIC
- Korea/KCC
- Taiwan/NCC

The WBZ451PE module has received regulatory approval for the following countries:

- Bluetooth Special Interest Group (SIG) QDID: D056857
- United States/FCC ID: 2ADHKWBZ451
- Canada/ISED:
  - IC: 20266-WBZ451
  - HVIN: WBZ451PE
- Europe/CE
- Japan/MIC
- Korea/KCC
- Taiwan/NCC
- China: 2023DJ2471

The WBZ451UC module has received regulatory approval for the following countries:

- Bluetooth Special Interest Group (SIG) QDID: D056857
- United States/FCC ID: 2ADHKWBZ451
- Canada/ISED:
  - IC: 20266-WBZ451
  - HVIN: WBZ451UC
- Europe/CE
- Japan/MIC
- Korea/KCC
- Taiwan/NCC

The WBZ451PE module has received regulatory approval for the following countries:

- Bluetooth Special Interest Group (SIG) QDID: D056857
- United States/FCC ID: 2ADHKWBZ451
- Canada/ISED:
  - IC: 20266-WBZ451
  - HVIN: WBZ451PE
- Europe/CE

- Japan/MIC
- Korea/KCC
- Taiwan/NCC
- China: 2023DJ2471

## 45.1 United States

The WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules have received Federal Communications Commission (FCC) CFR47 Telecommunications, Part 15 Subpart C “Intentional Radiators” single-modular approval in accordance with Part 15.212 Modular Transmitter approval. Single-modular transmitter approval is defined as a complete RF transmission sub-assembly, designed to be incorporated into another device, that must demonstrate compliance with FCC rules and policies independent of any host. A transmitter with a modular grant can be installed in different end-use products (referred to as a host, host product or host device) by the grantee or other equipment manufacturer, then the host product may not require additional testing or equipment authorization for the transmitter function provided by that specific module or limited module device.

The user must comply with all of the instructions provided by the Grantee, which indicate installation and/or operating conditions necessary for compliance.

A host product itself is required to comply with all other applicable FCC equipment authorization regulations, requirements, and equipment functions that are not associated with the transmitter module portion. For example, compliance must be demonstrated: to regulations for other transmitter components within a host product; to requirements for unintentional radiators (Part 15 Subpart B), such as digital devices, computer peripherals, radio receivers, etc.; and to additional authorization requirements for the non-transmitter functions on the transmitter module (i.e., Suppliers Declaration of Conformity (SDoC) or certification) as appropriate (e.g., Bluetooth and Wi-Fi transmitter modules may also contain digital logic functions).

### 45.1.1 Labeling and User Information Requirements

The WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules have been labeled with its own FCC ID number, and if the FCC ID is not visible when the module is installed inside another device, then the outside of the finished product into which the module is installed must display a label referring to the enclosed module. This exterior label must use the following wording:

For the WBZ450PC/WBZ450PE/ WBZ450UC/WBZ450UE module	Contains Transmitter Module FCC ID: 2ADHKWBZ450 or Contains FCC ID: 2ADHKWBZ450
For the WBZ451PC/WBZ451PE/ WBZ451UC/WBZ451UE module	Contains Transmitter Module FCC ID: 2ADHKWBZ451 or Contains FCC ID: 2ADHKWBZ451

**This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.**

The user's manual for the finished product must include the following statement:



This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy, and if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna
- Increase the separation between the equipment and receiver
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected
- Consult the dealer or an experienced radio/TV technician for help

Additional information on labeling and user information requirements for Part 15 devices can be found in KDB Publication 784748, which is available at the FCC Office of Engineering and Technology (OET) Laboratory Division Knowledge Database (KDB) [apps.fcc.gov/oetcf/kdb/index.cfm](https://apps.fcc.gov/oetcf/kdb/index.cfm).

## 45.1.2 RF Exposure

All transmitters regulated by FCC must comply with RF exposure requirements. KDB 447498 General RF Exposure Guidance provides guidance in determining whether proposed or existing transmitting facilities, operations or devices comply with limits for human exposure to Radio Frequency (RF) fields adopted by the Federal Communications Commission (FCC).

From the FCC Grant: Output power listed is conducted. This grant is valid only when the module is sold to OEM integrators and must be installed by the OEM or OEM integrators. This transmitter is restricted for use with the specific antenna(s) tested in this application for Certification and must not be co-located or operating in conjunction with any other antenna or transmitters within a host device, except in accordance with FCC multi-transmitter product procedures.

WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE: These modules are approved for installation into mobile or/and portable host platforms.

## 45.1.3 Helpful Web Sites

- Federal Communications Commission (FCC): [www.fcc.gov](http://www.fcc.gov).
- FCC Office of Engineering and Technology (OET) Laboratory Division Knowledge Database (KDB) [apps.fcc.gov/oetcf/kdb/index.cfm](https://apps.fcc.gov/oetcf/kdb/index.cfm).

## 45.2 Canada

The WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules have been certified for use in Canada under Innovation, Science and Economic Development Canada (ISED, formerly Industry Canada) Radio Standards Procedure (RSP) RSP-100, Radio Standards Specification (RSS) RSS-Gen and RSS-247. Modular approval permits the installation of a module in a host device without the need to recertify the device.

### 45.2.1 Labeling and User Information Requirements

Labeling Requirements (from RSP-100 - Issue 12, Section 5): The host product shall be properly labeled to identify the module within the host device.

The Innovation, Science and Economic Development Canada certification label of a module shall be clearly visible at all times when installed in the host device; otherwise, the host product must be labeled to display the Innovation, Science and Economic Development Canada certification number of the module, preceded by the word "Contains" or similar wording expressing the same meaning, as follows:

For the WBZ450PC/WBZ450PE/  
WBZ450UC/WBZ450UE module

**Contains IC: 20266-WBZ450**

For the WBZ451PC/WBZ451PE/  
WBZ451UC/WBZ451UE module

**Contains IC: 20266-WBZ451**

User Manual Notice for License-Exempt Radio Apparatus (from Section 8.4 RSS-Gen, Issue 5, February 2021): User manuals for license-exempt radio apparatus shall contain the following or equivalent notice in a conspicuous location in the user manual or alternatively on the device or both:

**This device contains license-exempt transmitter(s)/receiver(s) that comply with Innovation, Science and Economic Development Canada's license-exempt RSS(s). Operation is subject to the following two conditions:**

**(1) This device may not cause interference;**

**(2) This device must accept any interference, including interference that may cause undesired operation of the device.**

**L'émetteur/récepteur exempt de licence contenu dans le présent appareil est conforme aux CNR d'Innovation, Sciences et Développement économique Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes:**

**1. L'appareil ne doit pas produire de brouillage;**

**2. L'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.**

Transmitter Antenna (From Section 6.8 RSS-GEN, Issue 5, February 2021): User manuals, for transmitters shall display the following notice in a conspicuous location:

**This radio transmitter IC: 20266-WBZ450 and IC: 20266-WBZ451 have been approved by Innovation, Science and Economic Development Canada to operate with the antenna types listed below, with the maximum permissible gain indicated. Antenna types not included in this list that have a gain greater than the maximum gain indicated for any type listed are strictly prohibited for use with this device.**

**Le présent émetteur radio IC: 20266-WBZ450 and IC: 20266-WBZ451 a été approuvé par Innovation, Sciences et Développement économique Canada pour fonctionner avec les types d'antenne énumérés cidessous et ayant un gain admissible maximal. Les types d'antenne non inclus dans cette liste, et dont le gain est supérieur au gain maximal indiqué pour tout type figurant sur la liste, sont strictement interdits pour l'exploitation de l'émetteur.**

Immediately following the above notice, the manufacturer shall provide a list of all antenna types approved for use with the transmitter, indicating the maximum permissible antenna gain (in dBi) and required impedance for each.

## 45.2.2 RF Exposure

All transmitters regulated by Innovation, Science and Economic Development Canada (ISED) must comply with RF exposure requirements listed in RSS-102 - Radio Frequency (RF) Exposure Compliance of Radio communication Apparatus (All Frequency Bands).

This transmitter is restricted for use with a specific antenna tested in this application for certification, and must not be co-located or operating in conjunction with any other antenna or transmitters within a host device, except in accordance with Canada multi-transmitter product procedures.

IC: 20266-WBZ450 and IC: 20266-WBZ451 : The devices operate at an output power level which is within the ISED SAR test exemption limits at any user distance.

## 45.2.3 Helpful Web Sites

Innovation, Science and Economic Development Canada (ISED): [www.ic.gc.ca/](http://www.ic.gc.ca/).

## 45.3 Europe

The WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules is/are a Radio Equipment Directive (RED) assessed radio module that is CE marked and has been manufactured and tested with the intention of being integrated into a final product.

The WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules has/have been tested to RED 2014/53/EU Essential Requirements mentioned in the following European Compliance table.

**Table 45-1.** European Compliance Information

Certification	Standard	Article
Safety	EN 62368	3.1a
Health	EN 62311	
EMC	EN 301 489-1	3.1b
	EN 301 489-17	
Radio	EN 300 328	3.2

The ETSI provides guidance on modular devices in the “*Guide to the application of harmonised standards covering articles 3.1b and 3.2 of the RED 2014/53/EU (RED) to multi-radio and combined radio and non-radio equipment*” document available at [http://www.etsi.org/deliver/etsi\\_eg/203300\\_203399/20\\_3367/01.01.01\\_60/eg\\_203367v010101p.pdf](http://www.etsi.org/deliver/etsi_eg/203300_203399/20_3367/01.01.01_60/eg_203367v010101p.pdf).

**Note:** To maintain conformance to the standards listed in the preceding European Compliance table, the module shall be installed in accordance with the installation instructions in this data sheet and shall not be modified. When integrating a radio module into a completed product, the integrator becomes the manufacturer of the final product and is therefore responsible for demonstrating compliance of the final product with the essential requirements against the RED.

#### 45.3.1 Labeling and User Information Requirements

The label on the final product that contains the WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules must follow CE marking requirements.

#### 45.3.2 Conformity Assessment

From ETSI Guidance Note EG 203367, section 6.1, when non-radio products are combined with a radio product:

If the manufacturer of the combined equipment installs the radio product in a host non-radio product in equivalent assessment conditions (i.e. host equivalent to the one used for the assessment of the radio product) and according to the installation instructions for the radio product, then no additional assessment of the combined equipment against article 3.2 of the RED is required.

##### 45.3.2.1 Simplified EU Declaration of Conformity

Hereby, Microchip Technology Inc. declares that the radio equipment type WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules is/are in compliance with Directive 2014/53/EU.

The full text of the EU declaration of conformity, for this product, is available at [www.microchip.com/design-centers/wireless-connectivity/](http://www.microchip.com/design-centers/wireless-connectivity/).

#### 45.3.3 Helpful Websites

A document that can be used as a starting point in understanding the use of Short Range Devices (SRD) in Europe is the European Radio Communications Committee (ERC) Recommendation 70-03 E, which can be downloaded from the European Communications Committee (ECC) at: <http://www.ecodocdb.dk/>.

Additional helpful web sites are:

- Radio Equipment Directive (2014/53/EU): [https://ec.europa.eu/growth/single-market/european-standards/harmonised-standards/red\\_en](https://ec.europa.eu/growth/single-market/european-standards/harmonised-standards/red_en)
- European Conference of Postal and Telecommunications Administrations (CEPT): <http://www.cept.org>
- European Telecommunications Standards Institute (ETSI): <http://www.etsi.org>
- The Radio Equipment Directive Compliance Association (REDCA):

<http://www.redca.eu/>

## 45.4 Japan

The WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules have received type certification and is required to be labeled with its own technical conformity mark and certification number as required to conform to the technical standards regulated by the Ministry of Internal Affairs and Communications (MIC) of Japan pursuant to the Radio Act of Japan.

Integration of this module into a final product does not require additional radio certification provided installation instructions are followed and no modifications of the module are allowed. Additional testing may be required:

- If the host product is subject to electrical appliance safety (for example, powered from an AC mains), the host product may require Product Safety Electrical Appliance and Material (PSE) testing. The integrator should contact their conformance laboratory to determine if this testing is required
- There is an voluntary Electromagnetic Compatibility (EMC) test for the host product administered by VCCI: [www.vcci.jp/vcci\\_e/index.html](http://www.vcci.jp/vcci_e/index.html)

### 45.4.1 Labeling and User Information Requirements

The label on the final product which contains the WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE module(s) must follow Japan marking requirements. The integrator of the module should refer to the labeling requirements for Japan available at the Ministry of Internal Affairs and Communications (MIC) website.

For the WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules, due to a limited module size, the technical conformity logo and ID is displayed in the data sheet and/or packaging and cannot be displayed on the module label. The final product in which this module is being used must have a label referring to the type certified module inside:

WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE module



WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE module



### 45.4.2 Helpful Web Sites

- Ministry of Internal Affairs and Communications (MIC): [www.tele.soumu.go.jp/e/index.htm](http://www.tele.soumu.go.jp/e/index.htm).
- Association of Radio Industries and Businesses (ARIB): [www.arib.or.jp/english/](http://www.arib.or.jp/english/).

## 45.5 Korea

The WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules have received certification of conformity in accordance with the Radio Waves Act. Integration of this module into a final product does not require additional radio certification provided installation instructions are followed and no modifications of the module are allowed.

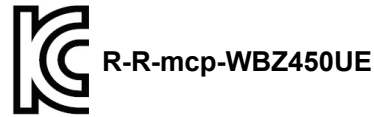
### 45.5.1 Labeling and User Information Requirements

The label on the final product which contains the WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE module(s) must follow KC marking requirements. The

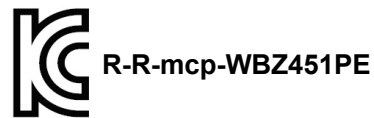
integrator of the module should refer to the labeling requirements for Korea available on the Korea Communications Commission (KCC) website.

The WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules are labeled with its own KC mark. The final product requires the KC mark and certificate number of the module:

WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE module



WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE module



## 45.5.2 Helpful Websites

- Korea Communications Commission (KCC): [www.kcc.go.kr](http://www.kcc.go.kr).
- National Radio Research Agency (RRA): [rra.go.kr](http://rra.go.kr).

## 45.6 Taiwan

The WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules have received compliance approval in accordance with the Telecommunications Act. Customers seeking to use the compliance approval in their product should contact Microchip Technology sales or distribution partners to obtain a Letter of Authority.

Integration of this module into a final product does not require additional radio certification provided installation instructions are followed and no modifications of the module are allowed.

### 45.6.1 Labeling and User Information Requirements

For the WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules, due to the limited module size, the NCC mark and ID are displayed in the data sheet only and cannot be displayed on the module label:

WBZ450PC module



WBZ450PE module



WBZ450UC module



WBZ450UE module



WBZ451PC module



WBZ451PE module



WBZ451UC module



WBZ451UE module



The user's manual should contain following warning (for RF device) in traditional Chinese:

根據 NCC LP0002 低功率射頻器材技術規範\_章節 3.8.2:

取得審驗證明之低功率射頻器材，非經核准，公司、商號或使用者均不得擅自變更頻率、加大功率或變更原設計之特性及功能。

低功率射頻器材之使用不得影響飛航安全及干擾合法通信；經發現有干擾現象時，應立即停用，並改善至無干擾時方得繼續使用。

前述合法通信，指依電信管理法規定作業之無線電通信。

低功率射頻器材須忍受合法通信或工業、科學及醫療用電波輻射性電機設備之干擾。

#### 45.6.2 Helpful Web Sites

National Communications Commission (NCC): [www.ncc.gov.tw](http://www.ncc.gov.tw)

#### 45.7 China

The WBZ450PE/WBZ451PE modules has/have received certification of conformity in accordance with the China MIIT Notice 2014-01 of State Radio Regulation Committee (SRRC) certification scheme. Integration of this module into a final product does not require additional radio certification,

provided installation instructions are followed and no modifications of the module are allowed. Refer to SRRC certificate available in WBZ450PE/WBZ451PE product page for expiry date.

The WBZ450UE/WBZ451UE modules have received certification of conformity in accordance with the China MIIT Notice 2014-01 of State Radio Regulation Committee (SRRC) certification scheme under Limited Modular Approval (LMA). Integration of this module into a final product does require additional radio certification (radiated spurious emission test and EMC test), provided installation instructions are followed and no modifications of the module are allowed. Refer to SRRC certificate available in WBZ450UE/WBZ451UE product page for expiry date.

#### 45.7.1 Labeling and User Information Requirements

The WBZ450PE/WBZ451PE and WBZ450UE/WBZ451UE modules are labeled with its own CMIIT ID as follows:

WBZ450PE module

CMIIT ID: 2023DJ2426

WBZ451PE module

CMIIT ID: 2023DJ2471

WBZ450UE module

CMIIT ID: 2023DJ2427(M)

WBZ451UE module

CMIIT ID: 2023DJ2475(M)

When Host system is using an approved Full Modular Approval (FMA) radio: The host must bear a label containing the statement "This device contains SRRC approved Radio module CMIIT ID: 2023DJ2426, CMIIT ID: 2023DJ2427(M), CMIIT ID: 2023DJ2471, CMIIT ID: 2023DJ2475(M).

#### 45.8 UKCA (UK Conformity Assessed)

The WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE module is a UK conformity assessed radio module that meets all the essential requirements according to CE RED requirements.

##### 45.8.1 Labeling Requirements for Module and User's Requirements

The label on the final product that contains the WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE module must follow UKCA marking requirements.

**UK  
CA**

The UKCA mark above is printed on the module itself or on the packing label.

Additional details for the label requirement are available at:

<https://www.gov.uk/guidance/using-the-ukca-marking#check-whether-you-need-to-use-the-new-ukca-marking>.

#### 45.8.2 UKCA Declaration of Conformity

Hereby, Microchip Technology Inc. declares that the radio equipment type the WBZ450PC/WBZ450PE/WBZ450UC/WBZ450UE and WBZ451PC/WBZ451PE/WBZ451UC/WBZ451UE modules are in compliance with the Radio Equipment Regulations 2017. The full text of the UKCA declaration of conformity for this product is available (under *Documents > Certifications*) at: [www.microchip.com/en-us/product/WBZ451PE](http://www.microchip.com/en-us/product/WBZ451PE).

#### 45.8.3 Helpful Websites

For more information on the UKCA regulatory approvals, refer to the [www.gov.uk/guidance/placing-manufactured-goods-on-the-market-in-great-britain](http://www.gov.uk/guidance/placing-manufactured-goods-on-the-market-in-great-britain).

### 45.9 Other Regulatory Information

- For information about other countries' jurisdictions not covered here, refer to the [www.microchip.com/design-centers/wireless-connectivity/certifications](http://www.microchip.com/design-centers/wireless-connectivity/certifications).
- Should other regulatory jurisdiction certification be required by the customer, or the customer needs to recertify the module for other reasons, contact Microchip for the required utilities and documentation.



## 46. Document Revision History

Revision	Date	Section	Description
B	12/2023	Bluetooth	<ul style="list-style-type: none"> <li>Typical receiver power sensitivity values updated</li> <li>Updated digital RSSI indicator -30 dBm to -50 dBm</li> </ul>
		802.15.4/Zigbee Modulation Scheme	Changed RX sensitivity 103 dBm to 100 dBm
		System Peripherals	Minor change
		Package	Updated with WBZ450 information
		WBZ45 Module Features	Updated the sentence
		WBZ45 Module Variants	Updated with WBZ450 information
		Package and Operating Conditions	
		Certifications	Added one of the notes
		1.1. PIC32CX-BZ2 SoC Ordering Information	Added PIC32CX1012BZ24032 ordering information
		1.2. WBZ45 Module Ordering Information	Added WBZ450 ordering information
		2. Configuration Summary	Updated with WBZ450 information
		3.2. Pinout Diagram	Added PIC32CX1012BZ24032 pinout diagram
		4.1. Pinout Diagram	Added WBZ450 pinout diagram
		4.2. Basic Connection Requirement	Added WBZ450 diagram
		4.2.4. Unused I/O Pins	Removed the note
		4.6.1. PCB Antenna	Added antenna radiation pattern images
		4.6.2. External Antenna Placement Recommendations	Updated the figure
		5. Pinout and Signal Descriptions List	Updated with WBZ450 information
		9.12.1. CHECON	Minor change
		13.17.9. PBxDIV, 13.17.8. REFOxTRIM, 13.17.7. REFOxCON, 13.17.10. SLEWCON, 13.17.12. CLK_DIAG, 13.17.3. SPLLCON	Updated the Reset values
		15.2. Power Modes	Minor change
		15.4. DSCON	Added new topic
		18.4.1. CFGCON0(L)	Minor change
		18.4.2. CFGCON1(L)	Added new bits
		18.4.6. CFGPCLKGEN1	Minor change
		39.2. Features	Added COMP and COMP1 related points
		42.2. Features	Typical receiver power sensitivity values updated
		42.7. RF Physical Layer	Updated transceiver architecture block diagram
		42.8.1. Transmit Mixer and Power Amplifier	Minor change
		43. Electrical Characteristics	Updated all the topics for this chapter
		44.4. WBZ450 Module Packaging Information	Added WBZ450 packaging information
45. Appendix A: Regulatory Approval	<ul style="list-style-type: none"> <li>Updated with WBZ450 information</li> <li>Added 45.4. Japan, 45.5. Korea, 45.6. Taiwan, China</li> </ul>		
A	10/2022	Document	Initial revision

## Microchip Information

### The Microchip Website

Microchip provides online support via our website at [www.microchip.com/](http://www.microchip.com/). This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

### Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to [www.microchip.com/pcn](http://www.microchip.com/pcn) and follow the registration instructions.

### Customer Support

Users of Microchip products can receive assistance through several channels:

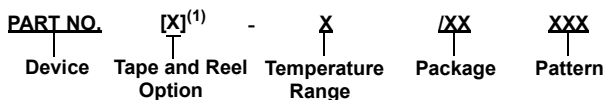
- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: [www.microchip.com/support](http://www.microchip.com/support)

## Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



Device:	PIC16F18313, PIC16LF18313, PIC16F18323, PIC16LF18323	
Tape and Reel Option:	Blank	= Standard packaging (tube or tray)
	T	= Tape and Reel <sup>(1)</sup>
Temperature Range:	I	= -40°C to +85°C (Industrial)
	E	= -40°C to +125°C (Extended)
Package: <sup>(2)</sup>	JQ	= UQFN
	P	= PDIP
	ST	= TSSOP
	SL	= SOIC-14
	SN	= SOIC-8
	RF	= UDFN
Pattern:	QTP, SQTP, Code or Special Requirements (blank otherwise)	

Examples:

- PIC16LF18313- I/P Industrial temperature, PDIP package
- PIC16F18313- E/SS Extended temperature, SSOP package

### Notes:

1. Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
2. Small form-factor packaging options may be available. Please check [www.microchip.com/packaging](http://www.microchip.com/packaging) for small-form factor package availability, or contact your local Sales Office.

## Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

## Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for

additional support or, obtain additional support at [www.microchip.com/en-us/support/design-help/client-support-services](http://www.microchip.com/en-us/support/design-help/client-support-services).

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

## Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Klear, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, TimeCesium, TimeHub, TimePictra, TimeProvider, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, Clockstudio, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, EyeOpen, GridTime, IdealBridge, IGaT, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, IntelliMOS, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, MarginLink, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, mSiC, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, Power MOS IV, Power MOS 7, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, Trusted Time, TSHARC, Turing, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2022-2023, Microchip Technology Incorporated and its subsidiaries. All Rights Reserved.

ISBN: 978-1-6683-3582-6

## Quality Management System

For information regarding Microchip's Quality Management Systems, please visit [www.microchip.com/quality](http://www.microchip.com/quality).

# Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
<p><b>Corporate Office</b> 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: <a href="http://www.microchip.com/support">www.microchip.com/support</a> Web Address: <a href="http://www.microchip.com">www.microchip.com</a></p> <p><b>Atlanta</b> Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455</p> <p><b>Austin, TX</b> Tel: 512-257-3370</p> <p><b>Boston</b> Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088</p> <p><b>Chicago</b> Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075</p> <p><b>Dallas</b> Addison, TX Tel: 972-818-7423 Fax: 972-818-2924</p> <p><b>Detroit</b> Novi, MI Tel: 248-848-4000</p> <p><b>Houston, TX</b> Tel: 281-894-5983</p> <p><b>Indianapolis</b> Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380</p> <p><b>Los Angeles</b> Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800</p> <p><b>Raleigh, NC</b> Tel: 919-844-7510</p> <p><b>New York, NY</b> Tel: 631-435-6000</p> <p><b>San Jose, CA</b> Tel: 408-735-9110 Tel: 408-436-4270</p> <p><b>Canada - Toronto</b> Tel: 905-695-1980 Fax: 905-695-2078</p>	<p><b>Australia - Sydney</b> Tel: 61-2-9868-6733</p> <p><b>China - Beijing</b> Tel: 86-10-8569-7000</p> <p><b>China - Chengdu</b> Tel: 86-28-8665-5511</p> <p><b>China - Chongqing</b> Tel: 86-23-8980-9588</p> <p><b>China - Dongguan</b> Tel: 86-769-8702-9880</p> <p><b>China - Guangzhou</b> Tel: 86-20-8755-8029</p> <p><b>China - Hangzhou</b> Tel: 86-571-8792-8115</p> <p><b>China - Hong Kong SAR</b> Tel: 852-2943-5100</p> <p><b>China - Nanjing</b> Tel: 86-25-8473-2460</p> <p><b>China - Qingdao</b> Tel: 86-532-8502-7355</p> <p><b>China - Shanghai</b> Tel: 86-21-3326-8000</p> <p><b>China - Shenyang</b> Tel: 86-24-2334-2829</p> <p><b>China - Shenzhen</b> Tel: 86-755-8864-2200</p> <p><b>China - Suzhou</b> Tel: 86-186-6233-1526</p> <p><b>China - Wuhan</b> Tel: 86-27-5980-5300</p> <p><b>China - Xian</b> Tel: 86-29-8833-7252</p> <p><b>China - Xiamen</b> Tel: 86-592-2388138</p> <p><b>China - Zhuhai</b> Tel: 86-756-3210040</p>	<p><b>India - Bangalore</b> Tel: 91-80-3090-4444</p> <p><b>India - New Delhi</b> Tel: 91-11-4160-8631</p> <p><b>India - Pune</b> Tel: 91-20-4121-0141</p> <p><b>Japan - Osaka</b> Tel: 81-6-6152-7160</p> <p><b>Japan - Tokyo</b> Tel: 81-3-6880-3770</p> <p><b>Korea - Daegu</b> Tel: 82-53-744-4301</p> <p><b>Korea - Seoul</b> Tel: 82-2-554-7200</p> <p><b>Malaysia - Kuala Lumpur</b> Tel: 60-3-7651-7906</p> <p><b>Malaysia - Penang</b> Tel: 60-4-227-8870</p> <p><b>Philippines - Manila</b> Tel: 63-2-634-9065</p> <p><b>Singapore</b> Tel: 65-6334-8870</p> <p><b>Taiwan - Hsin Chu</b> Tel: 886-3-577-8366</p> <p><b>Taiwan - Kaohsiung</b> Tel: 886-7-213-7830</p> <p><b>Taiwan - Taipei</b> Tel: 886-2-2508-8600</p> <p><b>Thailand - Bangkok</b> Tel: 66-2-694-1351</p> <p><b>Vietnam - Ho Chi Minh</b> Tel: 84-28-5448-2100</p>	<p><b>Austria - Wels</b> Tel: 43-7242-2244-39 Fax: 43-7242-2244-393</p> <p><b>Denmark - Copenhagen</b> Tel: 45-4485-5910 Fax: 45-4485-2829</p> <p><b>Finland - Espoo</b> Tel: 358-9-4520-820</p> <p><b>France - Paris</b> Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79</p> <p><b>Germany - Garching</b> Tel: 49-8931-9700</p> <p><b>Germany - Haan</b> Tel: 49-2129-3766400</p> <p><b>Germany - Heilbronn</b> Tel: 49-7131-72400</p> <p><b>Germany - Karlsruhe</b> Tel: 49-721-625370</p> <p><b>Germany - Munich</b> Tel: 49-89-627-144-0 Fax: 49-89-627-144-44</p> <p><b>Germany - Rosenheim</b> Tel: 49-8031-354-560</p> <p><b>Israel - Ra'anana</b> Tel: 972-9-744-7705</p> <p><b>Italy - Milan</b> Tel: 39-0331-742611 Fax: 39-0331-466781</p> <p><b>Italy - Padova</b> Tel: 39-049-7625286</p> <p><b>Netherlands - Drunen</b> Tel: 31-416-690399 Fax: 31-416-690340</p> <p><b>Norway - Trondheim</b> Tel: 47-72884388</p> <p><b>Poland - Warsaw</b> Tel: 48-22-3325737</p> <p><b>Romania - Bucharest</b> Tel: 40-21-407-87-50</p> <p><b>Spain - Madrid</b> Tel: 34-91-708-08-90 Fax: 34-91-708-08-91</p> <p><b>Sweden - Gothenberg</b> Tel: 46-31-704-60-40</p> <p><b>Sweden - Stockholm</b> Tel: 46-8-5090-4654</p> <p><b>UK - Wokingham</b> Tel: 44-118-921-5800 Fax: 44-118-921-5820</p>