**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

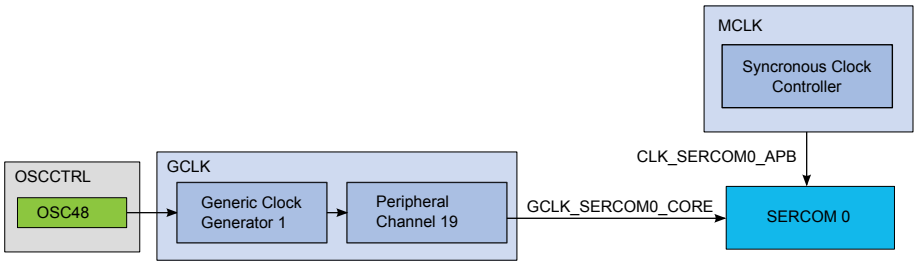
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | ARM® Cortex®-M0+ |
| Core Size | 32-Bit Single-Core |
| Speed | 48MHz |
| Connectivity | CANbus, I²C, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, DMA, POR, WDT |
| Number of I/O | 52 |
| Program Memory Size | 128KB (128K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 16K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.5V |
| Data Converters | A/D 20x12b, 3x16b; D/A 1x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 64-TQFP |
| Supplier Device Package | 64-TQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atsamc21j17a-aut |

The figure below shows an example where SERCOM0 is clocked by the OSC48M. The OSC48M is enabled, the Generic Clock Generator 1 uses the OSCLL48M as its clock source, and the generic clock 19, also called GCLK_SERCOM0_CORE, that is connected to SERCOM0 uses generator 1 as its source. The SERCOM0 interface, clocked by CLK_SERCOM0_APB, has been unmasked in the APBC Mask register in the MCLK.

**Figure 15-2. Example of SERCOM clock**




## 15.2 Synchronous and Asynchronous Clocks

As the CPU and the peripherals can be clocked from different clock sources, possibly with widely different clock speeds, some peripheral accesses by the CPU needs to be synchronized between the different clock domains. In these cases the peripheral includes a SYNCBUSY status register that can be used to check if a sync operation is in progress. As the nature of the synchronization might vary between different peripherals, detailed description for each peripheral can be found in the sub-chapter “synchronization” for each peripheral where this is necessary.

In the datasheet references to synchronous clocks are referring to the CPU and bus clocks, while asynchronous clocks are clock generated by generic clocks.

## 15.3 Register Synchronization

### 15.3.1 Overview

All peripherals are composed of one digital bus interface, which is connected to the APB or AHB bus and clocked using a corresponding synchronous clock, and one core clock, which is clocked using a generic clock. Access between these clock domains must be synchronized. As this mechanism is implemented in hardware the synchronization process takes place even if the different clocks domains are clocked from the same source and on the same frequency. All registers in the bus interface are accessible without synchronization. All core registers in the generic clock domain must be synchronized when written. Some core registers must be synchronized when read. Registers that need synchronization has this denoted in each individual register description.

### 15.3.2 General Write-Synchronization

Inside the same module, each core register, denoted by the Write-Synchronized property, use its own synchronization mechanism so that writing to different core registers can be done without waiting for the end of synchronization of previous core register access.

However a second write access to the same core register, while synchronization is on going, is discarded and an error is reported through the PAC. To write again to the same core register in the same module, user must wait for the end of synchronization.

For each core register, that can be written, a synchronization status bit is associated

**Example:**

**Bit 12 – TC0: TC0 APBC Mask Clock Enable**

| Value | Description |
|---|---|
| 0 | The APBC clock for the TC0 is stopped. |
| 1 | The APBC clock for the TC0 is enabled. |

**Bit 11 – TCC2: TCC2 APBC Mask Clock Enable**

| Value | Description |
|---|---|
| 0 | The APBC clock for the TCC2 is stopped. |
| 1 | The APBC clock for the TCC2 is enabled. |

**Bit 10 – TCC1: TCC1 APBC Mask Clock Enable**

| Value | Description |
|---|---|
| 0 | The APBC clock for the TCC1 is stopped. |
| 1 | The APBC clock for the TCC1 is enabled. |

**Bit 9 – TCC0: TCC0 APBC Mask Clock Enable**

| Value | Description |
|---|---|
| 0 | The APBC clock for the TCC0 is stopped. |
| 1 | The APBC clock for the TCC0 is enabled. |

**Bit 6 – SERCOM5: SERCOM5 APBC Mask Clock Enable**

| Value | Description |
|---|---|
| 0 | The APBC clock for the SERCOM5 is stopped. |
| 1 | The APBC clock for the SERCOM5 is enabled. |

**Bit 5 – SERCOM4: SERCOM4 APBC Mask Clock Enable**

| Value | Description |
|---|---|
| 0 | The APBC clock for the SERCOM4 is stopped. |
| 1 | The APBC clock for the SERCOM4 is enabled. |

**Bit 4 – SERCOM3: SERCOM3 APBC Mask Clock Enable**

| Value | Description |
|---|---|
| 0 | The APBC clock for the SERCOM3 is stopped. |
| 1 | The APBC clock for the SERCOM3 is enabled. |

**Bit 3 – SERCOM2: SERCOM2 APBC Mask Clock Enable**

| Value | Description |
|---|---|
| 0 | The APBC clock for the SERCOM2 is stopped. |
| 1 | The APBC clock for the SERCOM2 is enabled. |

**Bit 2 – SERCOM1: SERCOM1 APBC Mask Clock Enable**

| Value | Description |
|---|---|
| 0 | The APBC clock for the SERCOM1 is stopped. |
| 1 | The APBC clock for the SERCOM1 is enabled. |

**Related Links**

GCLK - Generic Clock Controller

### 20.6.3 Clock Failure Detection Operation

The Clock Failure Detector (CFD) allows the user to monitor the external clock or crystal oscillator signal provided by the external oscillator (XOSC). The CFD detects failing operation of the XOSC clock with reduced latency, and allows to switch to a safe clock source in case of clock failure. The user can also switch from the safe clock back to XOSC in case of recovery. The safe clock is derived from the OSC48M oscillator with a configurable prescaler. This allows to configure the safe clock in order to fulfill the operative conditions of the microcontroller.

In sleep modes, CFD operation is automatically disabled when the external oscillator is not requested to run by a peripheral. See the Sleep Behavior table above when this is the case.

The user interface registers allow to enable, disable, and configure the CFD. The Status register provides status flags on failure and clock switch conditions. The CFD can optionally trigger an interrupt or an event when a failure is detected.

#### Clock Failure Detection

The CFD is reset only at power-on (POR). The CFD does not monitor the XOSC clock when the oscillator is disabled (XOSCCTRL.ENABLE=0).

Before starting CFD operation, the user must start and enable the safe clock source (OSC48M oscillator).

CFD operation is started by writing a '1' to the CFD Enable bit in the External Oscillator Control register (XOCCTRL.CFDEN). After starting or restarting the XOSC, the CFD does not detect failure until the start-up time has elapsed. The start-up time is configured by the Oscillator Start-Up Time in the External Multipurpose Crystal Oscillator Control register (XOSCCTRL.STARTUP). Once the XOSC Start-Up Time is elapsed, the XOSC clock is constantly monitored.

During a period of 4 safe clocks (monitor period), the CFD watches for a clock activity from the XOSC. There must be at least one rising and one falling XOSC clock edge during 4 safe clock periods to meet non-failure conditions. If no or insufficient activity is detected, the failure status is asserted: The Clock Failure Detector status bit in the Status register (STATUS.CLKFAIL) and the Clock Failure Detector interrupt flag bit in the Interrupt Flag register (INTFLAG.CLKFAIL) are set. If the CLKFAIL bit in the Interrupt Enable Set register (INTENSET.CLKFAIL) is set, an interrupt is generated as well. If the Event Output enable bit in the Event Control register (EVCTRL.CFDEO) is set, an output event is generated, too.

After a clock failure was issued the monitoring of the XOSC clock is continued, and the Clock Failure Detector status bit in the Status register (STATUS.CLKFAIL) reflects the current XOSC activity.

#### Clock Switch

When a clock failure is detected, the XOSC clock is replaced by the safe clock in order to maintain an active clock during the XOSC clock failure. The safe clock source is the OSC48M oscillator clock. The safe clock source can be scaled down by a configurable prescaler to ensure that the safe clock frequency does not exceed the operating conditions selected by the application. When the XOSC clock is switched to the safe clock, the Clock Switch bit in the Status register (STATUS.CLKSW) is set.

When the CFD has switched to the safe clock, the XOSC is not disabled. If desired, the application must take the necessary actions to disable the oscillator. The application must also take the necessary actions to configure the system clocks to continue normal operations.

**Clock Failure Detection**

The CFD is reset only at power-on (POR). The CFD does not monitor the XOSC32K clock when the oscillator is disabled (XOSC32K.ENABLE=0).

Before starting CFD operation, the user must start and enable the safe clock source (OSCULP32K oscillator).

CFD operation is started by writing a '1' to the CFD Enable bit in the External Oscillator Control register (CFDCTRL.CFDEN). After starting or restarting the XOSC32K, the CFD does not detect failure until the start-up time has elapsed. The start-up time is configured by the Oscillator Start-Up Time in the External Multipurpose Crystal Oscillator Control register (XOSC32K.STARTUP). Once the XOSC32K Start-Up Time is elapsed, the XOSC32K clock is constantly monitored.

During a period of 4 safe clocks (monitor period), the CFD watches for a clock activity from the XOSC32K. There must be at least one rising and one falling XOSC32K clock edge during 4 safe clock periods to meet non-failure conditions. If no or insufficient activity is detected, the failure status is asserted: The Clock Failure Detector status bit in the Status register (STATUS.CLKFAIL) and the Clock Failure Detector interrupt flag bit in the Interrupt Flag register (INTFLAG.CLKFAIL) are set. If the CLKFAIL bit in the Interrupt Enable Set register (INTENSET.CLKFAIL) is set, an interrupt is generated as well. If the Event Output enable bit in the Event Control register (EVCTRL.CFDEO) is set, an output event is generated, too.

After a clock failure was issued the monitoring of the XOSC32K clock is continued, and the Clock Failure Detector status bit in the Status register (STATUS.CLKFAIL) reflects the current XOSC32K activity.

**Clock Switch**

When a clock failure is detected, the XOSC32K clock is replaced by the safe clock in order to maintain an active clock during the XOSC32K clock failure. The safe clock source is the OSCULP32K oscillator clock. Both 32KHz and 1KHz outputs of the XOSC32K are replaced by the respective OSCULP32K 32KHz and 1KHz outputs. The safe clock source can be scaled down by a configurable prescaler to ensure that the safe clock frequency does not exceed the operating conditions selected by the application. When the XOSC32K clock is switched to the safe clock, the Clock Switch bit in the Status register (STATUS.CLKSW) is set.

When the CFD has switched to the safe clock, the XOSC32K is not disabled. If desired, the application must take the necessary actions to disable the oscillator. The application must also take the necessary actions to configure the system clocks to continue normal operations. In the case the application can recover the XOSC32K, the application can switch back to the XOSC32K clock by writing a '1' to Switch Back Enable bit in the Clock Failure Control register (CFDCTRL.SWBACK). Once the XOSC32K clock is switched back, the Switch Back bit (CFDCTRL.SWBACK) is cleared by hardware.

**Prescaler**

The CFD has an internal configurable prescaler to generate the safe clock from the OSCULP32K oscillator. The prescaler size allows to scale down the OSCULP32K oscillator so the safe clock frequency is not higher than the XOSC32K clock frequency monitored by the CFD. The maximum division factor is 2.

The prescaler is applied on both outputs (32KHz and 1KHz) of the safe clock.

**Example**

For an external crystal oscillator at 32KHz and the OSCULP32K frequency is 32KHz, the XOSC32K.CFDPRESC should be set to 0 for a safe clock of equal frequency.

# 24. RTC – Real-Time Counter

## 24.1 Overview

The Real-Time Counter (RTC) is a 32-bit counter with a 10-bit programmable prescaler that typically runs continuously to keep track of time. The RTC can wake up the device from sleep modes using the alarm/compare wake up, periodic wake up, or overflow wake up mechanisms.

The RTC 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.

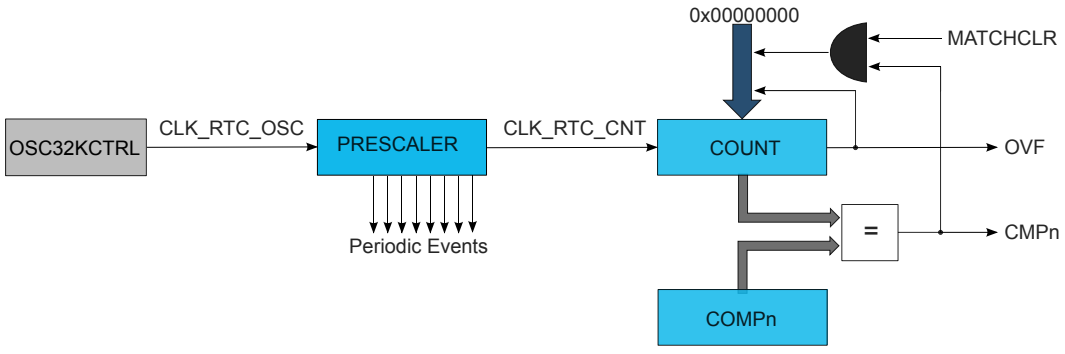
The 10-bit programmable prescaler can scale down the clock source. By this, a wide range of resolutions and time-out periods can be configured. With a 32.768kHz clock source, the minimum counter tick interval is 30.5µs, and time-out periods can range up to 36 hours. For a counter tick interval of 1s, the maximum time-out period is more than 136 years.

## 24.2 Features

- 32-bit counter with 10-bit prescaler
- Multiple clock sources
- 32-bit or 16-bit counter mode
- One 32-bit or two 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

## 24.3 Block Diagram

**Figure 24-1. RTC Block Diagram (Mode 0 — 32-Bit Counter)**



**Name:** INTENSET
**Offset:** 0x0A
**Reset:** 0x0000
**Property:** PAC Write-Protection

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | OVF | | | | | | | ALARM0 |
| Access | R/W | | | | | | | R/W |
| Reset | 0 | | | | | | | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PERn | PERn | PERn | PERn | PERn | PERn | PERn | PERn |
| Access | R/W | 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 8 – ALARM0: Alarm 0 Interrupt Enable**
Writing a '0' to this bit has no effect. Writing a '1' to this bit will set the Alarm 0 Interrupt Enable bit, which enables the Alarm 0 interrupt.

| Value | Description |
|---|---|
| 0 | The Alarm 0 interrupt is disabled. |
| 1 | The Alarm 0 interrupt is enabled. |

**Bits 7:0 – 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. |

### 24.12.5 Interrupt Flag Status and Clear in Clock/Calendar mode (CTRLA.MODE=2)

**Name:** INTFLAG
**Offset:** 0x0C
**Reset:** 0x0000
**Property:** -

# 25. DMAC – Direct Memory Access Controller

## 25.1 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 also the Block Diagram. 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 master interfaces but the AHB/APB Bridge bus, which is an APB slave interface.

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.

## 25.2 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 12 channels

    - Optionally enable the Resume software command
  - If the DMA is executing the same descriptor as the one which requires changes:
    - Set the Channel Suspend software command and wait for the Suspend interrupt
    - Update the next descriptor address (DESCRADDR) 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.

**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:
   2.1. Set the descriptor A VALID bit to '0'.
   2.2. Set the DESCADDR value of descriptor A to point to descriptor C instead of descriptor B.
   2.3. Set the DESCADDR value of descriptor C to point to descriptor B.
   2.4. Set the descriptor A VALID bit to '1'.
3. If DMA is executing descriptor A:
   3.1. Apply the software suspend command to the channel and
   3.2. Perform steps 2.1 through 2.4.
   3.3. Apply the software resume command to the channel.

### 25.6.3.2 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, refer to section Transfer Descriptors.

**Related Links**

CHCTRLB

CHINTFLAG

BTCTRL

**Name:** CHID
**Offset:** 0x3F [ID-00001ece]
**Reset:** 0x00
**Property:** -

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

**Bits 3:0 – ID[3:0]: Channel ID**
These bits define the channel number that will be affected by the channel registers (CH*). Before reading or writing a channel register, the channel ID bit group must be written first.

### 25.8.18 Channel Control A

This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

**Name:** CHCTRLA
**Offset:** 0x40 [ID-00001ece]
**Reset:** 0x00
**Property:** PAC Write-Protection, Enable-Protected

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | RUNSTDBY | | | | | ENABLE | SWRST |
| Access | R | R/W | R | R | R | R | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**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.

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 common interrupt request line for all the interrupt sources, and one interrupt request line for the NMI. The user must read the INTFLAG (or NMIFLAG) register to determine which interrupt condition is present.

**Note:** Interrupts must be globally enabled for interrupt requests to be generated.

**Note:** If an external interrupts (EXTINT) is common on two or more I/O pins, only one will be active (the first one programmed).

**Related Links**

Processor and Architecture

### 26.6.7 Events

The EIC can generate the following output events:

- External event from pin (EXTINTx).

Setting an Event Output Control register (EVCTRL.EXTINTEO) enables the corresponding output event. Clearing this bit disables the corresponding output event. Refer to *Event System* for details on configuring the Event System.

When the condition on pin EXTINTx matches the configuration in the CONFIGn register, the corresponding event is generated, if enabled.

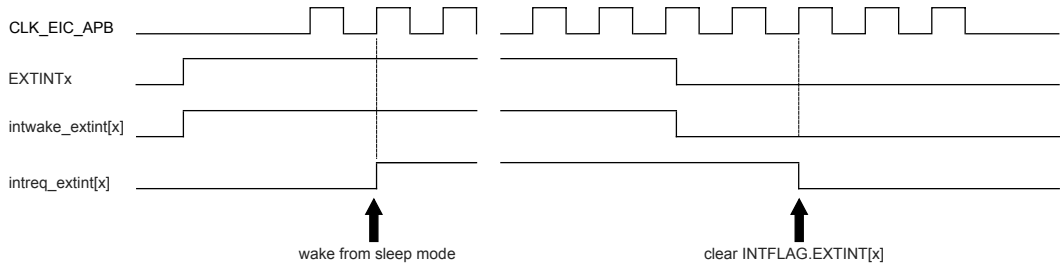
**Related Links**

EVSYS – Event System

### 26.6.8 Sleep Mode Operation

In sleep modes, an EXTINTx pin can wake up the device if the corresponding condition matches the configuration in CONFIG0, CONFIG1, CONFIG2, CONFIG3 register, and the corresponding bit in the Interrupt Enable Set register (INTENSET) is written to '1'.

**Figure 26-5. Wake-up Operation Example (High-Level Detection, No Filter, Interrupt Enable Set)**




### 26.6.9 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.

**Related Links**

PM – Power Manager

### 30.5.3 Clocks

The SERCOM bus clock (CLK_SERCOMx_APB) can be enabled and disabled in the Main Clock Controller. Refer to *Peripheral Clock Masking* for details and default status of this clock.

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 master. 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 Generic Clock Controller (GCLK) before using the SERCOM.

The generic clocks are asynchronous to the user interface clock (CLK_SERCOMx_APB). Due to this asynchronicity, writing to certain registers will require synchronization between the clock domains. Refer to Synchronization for details.

**Related Links**

GCLK - Generic Clock Controller

MCLK – Main Clock

### 30.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). The DMAC must be configured before the SERCOM DMA requests are used.

**Related Links**

DMAC – Direct Memory Access Controller

### 30.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller (NVIC). The NVIC must be configured before the SERCOM interrupts are used.

**Related Links**

Nested Vector Interrupt Controller

### 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.

### 30.5.8 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 Clear and Status register (INTFLAG)
- Status register (STATUS)
- Data register (DATA)

| Value | Description |
|---|---|
| 0 | The Stop Received interrupt is disabled. |
| 1 | The Stop Received interrupt is enabled. |

### 33.8.5 Interrupt Flag Status and Clear

**Name:** INTFLAG
**Offset:** 0x18 [ID-00001bb3]
**Reset:** 0x00
**Property:** -

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ERROR | | | | | DRDY | AMATCH | PREC |
| Access | R/W | | | | | R/W | R/W | R/W |
| Reset | 0 | | | | | 0 | 0 | 0 |

**Bit 7 – ERROR: Error**
This bit is set when any error is detected. Errors that will set this flag have corresponding status flags in the STATUS register. The corresponding bits in STATUS are SEXTTOUT, LOWTOUT, COLL, and BUSERR.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the flag.

**Bit 2 – DRDY: Data Ready**
This flag is set when a $I^2C$ slave byte transmission is successfully completed.

The flag is cleared by hardware when either:

- Writing to the DATA register.
- Reading the DATA register with smart mode enabled.
- Writing a valid command to the CMD register.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear the Data Ready interrupt flag.

**Bit 1 – AMATCH: Address Match**
This flag is set when the $I^2C$ slave 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 master and another slave 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.

| 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 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | RBDS[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | F1DS[2:0] | | | | F0DS[2:0] | | |
| Access | | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bits 10:8 – RBDS[2:0]: Rx Buffer Data Field Size**
In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer, only the number of bytes as configured by RXESC are stored to the Rx Buffer element. The rest of the frame's data field is ignored.

| Value | Name | Description |
|---|---|---|
| 0x0 | DATA8 | 8 byte data field. |
| 0x1 | DATA12 | 12 byte data field. |
| 0x2 | DATA16 | 16 byte data field. |
| 0x3 | DATA20 | 20 byte data field. |
| 0x4 | DATA24 | 24 byte data field. |
| 0x5 | DATA32 | 32 byte data field. |
| 0x6 | DATA48 | 48 byte data field. |
| 0x7 | DATA64 | 64 byte data field. |

**Bits 6:4 – F1DS[2:0]: Rx FIFO 1 Data Field Size**
In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx FIFO 1, only the number of bytes as configured by RXESC are stored to the Rx FIFO 1 element. The rest of the frame's data field is ignored.

| Value | Name | Description |
|---|---|---|
| 0x0 | DATA8 | 8 byte data field. |
| 0x1 | DATA12 | 12 byte data field. |
| 0x2 | DATA16 | 16 byte data field. |
| 0x3 | DATA20 | 20 byte data field. |
| 0x4 | DATA24 | 24 byte data field. |
| 0x5 | DATA32 | 32 byte data field. |
| 0x6 | DATA48 | 48 byte data field. |
| 0x7 | DATA64 | 64 byte data field. |

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). |

#### 35.7.2.3 Control B Set

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).

**Name:** CTRLBSET
**Offset:** 0x05
**Reset:** 0x00
**Property:** PAC Write-Protection, Read-synchronized, Write-Synchronized

| 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 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

| 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 |
|---|---|---|---|---|---|---|---|---|
| | | CCx | PER | COUNT | STATUS | CTRLB | ENABLE | SWRST |
| Access | | R | R | R | R | R | R | R |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 6 – CCx: Compare/Capture Channel x Synchronization Busy**
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.

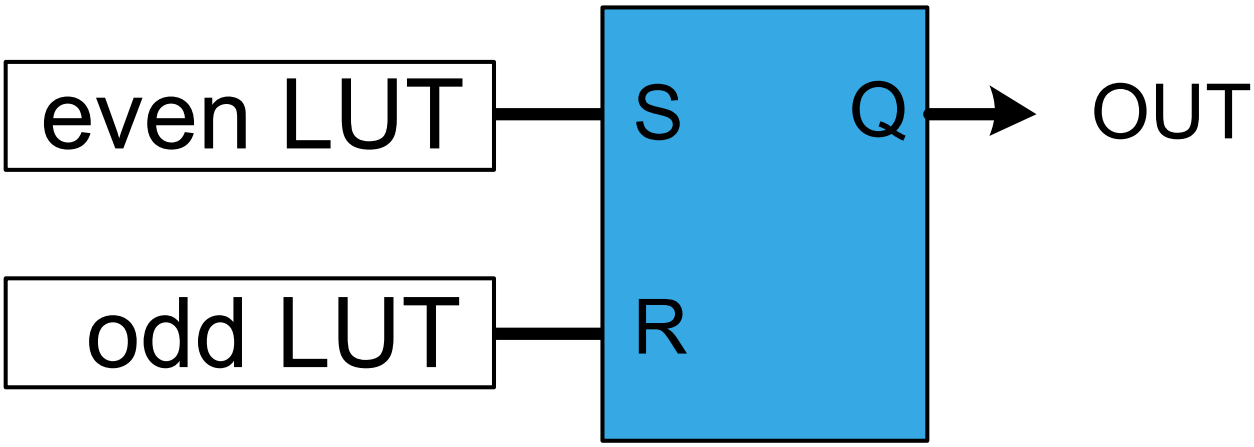
**Table 37-4. D-Latch Characteristics**

| G | D | OUT |
|---|---|---|
| 0 | X | Hold state (no change) |
| 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 LUT2), and the R-input is driven by the odd LUT output (LUT1 and LUT3), as shown in Figure 37-17.

**Figure 37-17. RS-Latch**




When the even LUT is disabled LUTCTRL0.ENABLE=0 / LUTCTRL2.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 Table 37-5.

**Table 37-5. RS-Latch Characteristics**

| S | R | OUT |
|---|---|---|
| 0 | 0 | Hold state (no change) |
| 0 | 1 | Clear |
| 1 | 0 | Set |
| 1 | 1 | Forbidden state |

### 37.6.3 Events

The CCL can generate the following output events:

- OUTx: Lookup Table Output Value

Writing a '1' to the LUT Control Event Output Enable bit (LUTCTRL.LUTEO) enables the corresponding output event. Writing a '0' to this bit disables the corresponding output event.

The CCL can take the following actions on an input event:

- INSELx: The event is used as input for the TRUTH table. For further details refer to 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.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | DUALSEL[1:0] | | | WINMODE[2:0] | | |
| 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 |
|---|---|---|---|---|---|---|---|---|
| | R2R | | RESSEL[1:0] | | CORREN | FREERUN | LEFTADJ | DIFFMODE |
| Access | R/W | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 13:12 – DUALSEL[1:0]: Dual Mode Trigger Selection**
These bits define the trigger mode. These bits are available in the master ADC and have no effect if the master-slave operation is disabled (ADC1.CTRLA.SLAVEEN=0).

| Value | Name | Description |
|---|---|---|
| 0x0 | BOTH | Start event or software trigger will start a conversion on both ADCs. |
| 0x1 | INTERLEAVE | Start event or software trigger will alternatingly start a conversion on ADC0 and ADC1. |
| 0x2 - 0x3 | - | Reserved |

**Bits 10:8 – WINMODE[2:0]: Window Monitor Mode**
These bits enable and define the window monitor mode.

| Value | Name | Description |
|---|---|---|
| 0x0 | DISABLE | No window mode (default) |
| 0x1 | MODE1 | RESULT > WINLT |
| 0x2 | MODE2 | RESULT < WINUT |
| 0x3 | MODE3 | WINLT < RESULT < WINUT |
| 0x4 | MODE4 | WINUT < RESULT < WINLT |
| 0x5 - 0x7 | | Reserved |

**Bit 7 – R2R: Rail-to-Rail Operation**

| Value | Description |
|---|---|
| 0 | Disable rail-to-rail operation. |
| 1 | Enable rail-to-rail operation to increase the allowable range of the input common mode voltage ($V_{CMIN}$). When R2R is one, a sampling period of four cycles is required. Offset compensation (SAMPCTRL.OFFCOMP) must be written to one when using this period. |

**Bits 5:4 – RESSEL[1:0]: Conversion Result Resolution**
These bits define whether the ADC completes the conversion 12-, 10- or 8-bit result resolution.

| Value | Name | Description |
|---|---|---|
| 0x0 | 12BIT | 12-bit result |
| 0x1 | 16BIT | For averaging mode output |
| 0x2 | 10BIT | 10-bit result |
| 0x3 | 8BIT | 8-bit result |

**Bit 3 – CORREN: Digital Correction Logic Enabled**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | REFSEL[1:0] | | DITHER | | VPD | LEFTADJ | IOEN | EOEN |
| Access | R/W | R/W | R/W | | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | | 0 | 0 | 0 | 0 |

**Bits 7:6 – REFSEL[1:0]: Reference Selection**
This bit field selects the Reference Voltage for the DAC.

| Value | Name | Description |
|---|---|---|
| 0x0 | INTREF | Internal voltage reference |
| 0x1 | VDDANA | Analog voltage supply |
| 0x2 | VREFA | External reference |
| 0x3 | | Reserved |

**Bit 5 – DITHER: Dithering Mode**
This bit controls dithering operation according to Dithering mode.

| Value | Description |
|---|---|
| 0 | Dithering mode is disabled. |
| 1 | Dithering mode is enabled. |

**Bit 3 – VPD: Voltage Pump Disabled**
This bit controls the behavior of the voltage pump.

| Value | Description |
|---|---|
| 0 | Voltage pump is turned on/off automatically |
| 1 | Voltage pump is disabled. |

**Bit 2 – LEFTADJ: Left-Adjusted Data**
This bit controls how the 10-bit conversion data is adjusted in the Data and Data Buffer registers.

| Value | Description |
|---|---|
| 0 | DATA and DATABUF registers are right-adjusted. |
| 1 | DATA and DATABUF registers are left-adjusted. |

**Bit 1 – IOEN: Internal Output Enable**

| Value | Description |
|---|---|
| 0 | Internal DAC output not enabled. |
| 1 | Internal DAC output enabled to be used by the AC or ADC. |

**Bit 0 – EOEN: External Output Enable**

| Value | Description |
|---|---|
| 0 | The DAC output is turned off. |
| 1 | The high-drive output buffer drives the DAC output to the $V_{OUT}$ pin. |

### 41.8.3 Event Control

#### 44.6.2.3 Measurement

In the Configuration A register, the Number of Reference Clock Cycles field (CFGA.REFNUM) selects the duration of the measurement. The measurement is given in number of GCLK_FREQM_REF periods.
**Note:** The REFNUM field must be written before the FREQM is enabled.

After the FREQM is enabled, writing a '1' to the START bit in the Control B register (CTRLB.START) starts the measurement. The BUSY bit in Status register (STATUS.BUSY) is set when the measurement starts, and cleared when the measurement is complete.

There is also an interrupt request for Measurement Done: When the Measurement Done bit in Interrupt Enable Set register (INTENSET.DONE) is '1' and a measurement is finished, the Measurement Done bit in the Interrupt Flag Status and Clear register (INTFLAG.DONE) will be set and an interrupt request is generated.

The result of the measurement can be read from the Value register (VALUE.VALUE). The frequency of the measured clock GCLK_FREQM_MSR is then:

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

**Note:** In order to make sure the measurement result (VALUE.VALUE[23:0]) is valid, the overflow status (STATUS.OVF) should be checked.

In case 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.

### 44.6.3 DMA Operation

Not applicable.

### 44.6.4 Interrupts

The FREQM has one interrupt source:

- DONE: A frequency measurement is done.

The interrupt flag in the Interrupt Flag Status and Clear (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 (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 FREQM is reset. See 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 INTFLAG register to determine which interrupt condition is present.

This interrupt is a synchroneous wake-up source.

Note that interrupts must be globally enabled for interrupt requests to be generated.

### 44.6.5 Events

Not applicable.

### 44.6.6 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.

**Table 48-23. Device and Package Maximum Weight**

| | |
|---|---|
| 90 | mg |

**Table 48-24. Package Characteristics**

| | |
|---|---|
| Moisture Sensitivity Level | MSL3 |

**Table 48-25. Package Reference**

| | |
|---|---|
| JEDEC Drawing Reference | MO-220 |
| JESD97 Classification | E3 |

## 48.3 Soldering Profile

The following table gives the recommended soldering profile from J-STD-20.

**Table 48-26.**

| Profile Feature | Green Package |
|---|---|
| Average Ramp-up Rate (217°C to peak) | 3°C/s max. |
| Preheat Temperature 175°C ±25°C | 150-200°C |
| Time Maintained Above 217°C | 60-150s |
| Time within 5°C of Actual Peak Temperature | 30s |
| Peak Temperature Range | 260°C |
| Ramp-down Rate | 6°C/s max. |
| Time 25°C to Peak Temperature | 8 minutes max. |

A maximum of three reflow passes is allowed per component.