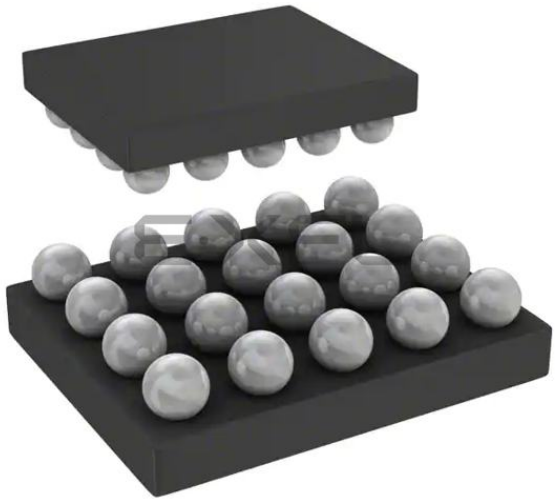


Welcome to E-XFL.COM

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	I²C, LINbus, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, WDT
Number of I/O	18
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.63V
Data Converters	A/D 8x12b; D/A 1x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-XFBGA, WLCSP
Supplier Device Package	20-WLCSP (2.43x1.93)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsamd11d14a-uut

12.13.1 Control

Name: CTRL

Offset: 0x0000

Reset: 0x00

Access: Write-Only

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	SMSA	ARR		CE	MBIST	CRC		SWRST
Access	W	W	R	W	W	W	R	W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – SMSA: Start Memory Stream Access**

- **Bit 6 – ARR: Auxiliary Row Read**

- **Bit 5 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 4 – CE: Chip-Erase**
  Writing a zero to this bit has no effect.
  Writing a one to this bit starts the Chip-Erase operation.

- **Bit 3 – MBIST: Memory built-in self-test**
  Writing a zero to this bit has no effect.
  Writing a one to this bit starts the memory BIST algorithm.

- **Bit 2 – CRC: 32-bit Cyclic Redundancy Code**
  Writing a zero to this bit has no effect.
  Writing a one to this bit starts the cyclic redundancy check algorithm.

- **Bit 1 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bit 0 – SWRST: Software Reset**
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets the module.

Writing a zero to this bit has no effect.

Writing a one to this bit will set the DFLL Lock Coarse Interrupt Enable bit, which enables the DFLL Lock Coarse interrupt.

- **Bit 6 – DFLLLCKF: DFLL Lock Fine Interrupt Enable**
  0: The DFLL Lock Fine interrupt is disabled.

  1: The DFLL Lock Fine interrupt is enabled, and an interrupt request will be generated when the DFLL Lock Fine Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the DFLL Lock Fine Interrupt Disable/Enable bit, disable the DFLL Lock Fine interrupt and set the corresponding interrupt request.

- **Bit 5 – DFLLOOB: DFLL Out Of Bounds Interrupt Enable**
  0: The DFLL Out Of Bounds interrupt is disabled.

  1: The DFLL Out Of Bounds interrupt is enabled, and an interrupt request will be generated when the DFLL Out Of Bounds Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the DFLL Out Of Bounds Interrupt Enable bit, which enables the DFLL Out Of Bounds interrupt.

- **Bit 4 – DFLLRDY: DFLL Ready Interrupt Enable**
  0: The DFLL Ready interrupt is disabled.

  1: The DFLL Ready interrupt is enabled, and an interrupt request will be generated when the DFLL Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the DFLL Ready Interrupt Enable bit, which enables the DFLL Ready interrupt and set the corresponding interrupt request.

- **Bit 3 – OSC8MRDY: OSC8M Ready Interrupt Enable**
  0: The OSC8M Ready interrupt is disabled.

  1: The OSC8M Ready interrupt is enabled, and an interrupt request will be generated when the OSC8M Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the OSC8M Ready Interrupt Enable bit, which enables the OSC8M Ready interrupt.

- **Bit 2 – OSC32KRDY: OSC32K Ready Interrupt Enable**
  0: The OSC32K Ready interrupt is disabled.

  1: The OSC32K Ready interrupt is enabled, and an interrupt request will be generated when the OSC32K Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the OSC32K Ready Interrupt Enable bit, which enables the OSC32K Ready interrupt.

- **Bit 1 – XOSC32KRDY: XOSC32K Ready Interrupt Enable**
  0: The XOSC32K Ready interrupt is disabled.

  1: The XOSC32K Ready interrupt is enabled, and an interrupt request will be generated when the XOSC32K Ready Interrupt flag is set.

  Writing a zero to this bit has no effect.

  Writing a one to this bit will set the XOSC32K Ready Interrupt Enable bit, which enables the XOSC32K Ready interrupt.

17.4 Signal Description

Not applicable.

17.5 Product Dependencies

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

17.5.1 I/O Lines

Not applicable.

17.5.2 Power Management

The WDT can continue to operate in any sleep mode where the selected source clock is running. The WDT 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. Refer to "PM – Power Manager" on page 110 for details on the different sleep modes.

17.5.3 Clocks

The WDT bus clock (CLK_WDT_APB) is enabled by default, and can be enabled and disabled in the Power Manager. Refer to "PM – Power Manager" on page 110 for details.

A generic clock (GCLK_WDT) is required to clock the WDT. This clock must be configured and enabled in the Generic Clock Controller before using the WDT. Refer to "GCLK – Generic Clock Controller" on page 88 for details.

This generic clock is asynchronous to the user interface clock (CLK_WDT_APB). Due to this asynchronicity, accessing certain registers will require synchronization between the clock domains. Refer to "Synchronization" on page 214 for further details.

GCLK_WDT is intended to be sourced from the clock of the internal ultra-low-power (ULP) oscillator. Due to the ultra-low-power design, the oscillator is not very accurate, and so the exact time-out period may vary from device to device. This variation must be kept in mind when designing software that uses the WDT to ensure that the time-out periods used are valid for all devices. For more information on ULP oscillator accuracy, consult the "Ultra Low Power Internal 32kHz RC Oscillator (OSCULP32K) Characteristics" on page 890.

GCLK_WDT can also be clocked from other sources if a more accurate clock is needed, but at the cost of higher power consumption.

17.5.4 DMA

Not applicable.

17.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. Using the WDT interrupts requires the interrupt controller to be configured first. Refer to "Nested Vector Interrupt Controller" on page 26 for details.

17.5.6 Events

Not applicable.

17.5.7 Debug Operation

When the CPU is halted in debug mode, the WDT will halt normal operation. This peripheral can be forced to continue operation during debugging.

17.5.8 Register Access Protection

All registers with write-access are optionally write-protected by the peripheral access controller (PAC), except the following registers:

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 "EVSYS – Event System" on page 405 for more information.

18.6.8.1 Shutdown Mode

Any of the RTC interrupt sources can wake up the device from the Shutdown mode if the RTC clock is configured to use a clock that is available in Shutdown mode. The wake-up sources are enabled if the corresponding bit in the Interrupt Enable registers (INTENCLR/SET) is one.

When waking up from Shutdown mode, all RTC registers will have the same value as before the Shutdown mode was entered, except the following registers: Interrupt Enable (INTENCLR/SET), Read Request (READREQ), Interrupt Flag Status and Clear (INTFLAG) and Debug (DBGCTRL). Note that INTENCLR/SET will be reset, with all interrupts turned off. The software must first reconfigure the Interrupt Controller, and then enable the interrupts again in the RTC. The Status register (STATUS) will show the status of the RTC, including the status bits set during shutdown operation.

When waking up the system from shutdown, the CPU will start executing code from the reset start address.

18.6.9 Synchronization

Due to the asynchronicity between CLK_RTC_APB and GCLK_RTC some registers must be synchronized when accessed. A register can require:

- Synchronization when written
- Synchronization when read
- Synchronization when written and read
- No synchronization

When executing an operation that requires synchronization, the Synchronization Busy bit in the Status register (STATUS.SYNCBUSY) will be set immediately, and cleared when synchronization is complete. The synchronization Ready interrupt can be used to signal when sync is complete. This can be accessed via the Synchronization Ready Interrupt Flag in the Interrupt Flag Status and Clear register (INTFLAG.SYNCRDY).

If an operation that requires synchronization is executed while STATUS.SYNCBUSY is one, the bus will be stalled. All operations will complete successfully, but the CPU will be stalled and interrupts will be pending as long as the bus is stalled.

The following bits need synchronization when written:

- Software Reset bit in the Control register (CTRL.SWRST)
- Enable bit in the Control register (CTRL.ENABLE)

The following registers need synchronization when written:

- The Counter Value register (COUNT)
- The Clock Value register (CLOCK)
- The Counter Period register (PER)
- The Compare n Value registers (COMPn)
- The Alarm n Value registers (ALARMn)
- The Frequency Correction register (FREQCORR)
- The Alarm n Mask register (MASKn)

Write-synchronization is denoted by the Write-Synchronization property in the register description.

The following registers need synchronization when read:

- The Counter Value register (COUNT)
- The Clock Value register (CLOCK)

Read-synchronization is denoted by the Read-Synchronization property in the register description.

18.8.27 Alarm n Mask - MODE2

Name: MASK

Offset: 0x1C

Reset: 0x00

Access: Read-Write

Property: -

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

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 2:0 – SEL[2:0]: Alarm Mask Selection**
  These bits define which bit groups of Alarm n are valid.

Table 18-11. Alarm Mask Selection

SEL[2:0]	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

19.8.1.11 Interrupt Status

Name: INTSTATUS

Offset: 0x24

Reset: 0x00000000

Access: Read-Only

Property: -

Bit	31	30	29	28	27	26	25	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
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	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
			CHINT5	CHINT4	CHINT3	CHINT2	CHINT1	CHINT0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

- **Bits 31:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 5:0 – CHINTx [x=5..0]: Channel x Pending Interrupt**
  This bit is set when Channel x has pending interrupt.
  This bit is cleared when the corresponding Channel x interrupts are disabled or the interrupts sources are cleared.

By default, the input synchronizer is clocked only when an input read is requested, which will delay the read operation by two CLK_PORT cycles. To remove that delay, the input synchronizers for each group of eight pins can be configured to be always active, but this comes at the expense of higher power consumption. This is controlled by writing a one to the corresponding SAMPLINGn bit group of the CTRL register, where n = (y%32) / 8.

To use pin y as one of the available peripheral functions for that pin, configure it by writing a one to the corresponding PMUXEN bit of the PINCFGy register. The PINCFGy register for pin y is at byte offset (PINCFG0 + (y%32)).

The peripheral function can be selected by writing to the PMUXO or PMUXE bit group in the PMUXn register. The PMUXO/PMUXE bit group is at byte offset (PMUX0 + (y%32) / 2), in bits 3:0 if y is even and in bits 7:4 if y is odd.

The chosen peripheral must also be configured and enabled.

22.6.4 I/O Pin Configuration

The Pin Configuration register (PINCFGy) is used for additional I/O pin configuration. A pin can be set in a totem-pole, open-drain or pull configuration.

Because pull configuration is done through the Pin Configuration register, all intermediate PORT states during switching of pin direction and pin values are avoided.

The I/O pin configurations are described further in this chapter, and summarized in Table 22-1.

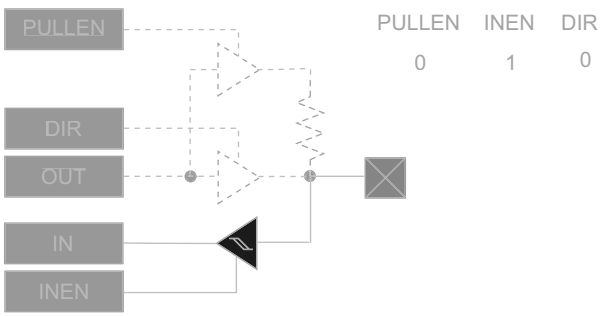
22.6.4.1 Pin Configurations Summary

Table 22-1. Pin Configurations Summary

DIR	INEN	PULLEN	OUT	Configuration
0	0	0	X	Reset or analog I/O; all digital disabled
0	0	1	0	Pull-down; input disabled
0	0	1	1	Pull-up; input disabled
0	1	0	X	Input
0	1	1	0	Input with pull-down
0	1	1	1	Input with pull-up
1	0	X	X	Output; input disabled
1	1	X	X	Output; input enabled

22.6.4.2 Input Configuration

Figure 22-4. I/O Configuration - Standard Input



Events propagated in the asynchronous path cannot generate any interrupts, and no channel status bits will indicate the state of the channel. No edge detection is available; this must be handled in the event user.

When the event generator and the event user share the same generic clock, using the asynchronous path will propagate the event with the least amount of latency.

Synchronous Path

The synchronous path should be used when the event generator and the event channel share the same 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 capable of generating interrupts. The channel status bits in the Channel Status register (CHSTATUS) are also updated and available for use.

If the Generic Clocks Request bit in the Control register (CTRL.GCLKREQ) is zero, the channel operates in SleepWalking mode and request the configured generic clock only when an event is to be propagated through the channel. If CTRL.GCLKREQ is one, the generic clock will always be on for the configured channel.

Resynchronized Path

The resynchronized path should be used when the event generator and the event channel do not share the same 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 capable of generating interrupts. The channel status bits in the Channel Status register (CHSTATUS) are also updated and available for use.

If the Generic Clocks Request bit in the Control register (CTRL.GCLKREQ) is zero, the channel operates in SleepWalking mode and request the configured generic clock only when an event is to be propagated through the channel. If CTRL.GCLKREQ is one, the generic clock will always be on for the configured channel.

23.6.2.6 Edge Detection

When synchronous or resynchronized paths are used, edge detection must be used. The event system can perform 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 in the Channel register (CHANNEL.EDGSEL).

If the generator event is a pulse, both edges cannot be selected. Use the rising edge or falling edge detection methods, depending on the generator event default level.

23.6.2.7 Channel Status

The Channel Status register (CHSTATUS) contains the status of the channels when a synchronous or resynchronized path is in use. There are two different status bits in CHSTATUS for each of the available channels: The Channel Busy bit (CHSTATUS.CHBUSYx) is set to one if an event on the corresponding channel x has not been handled by all event users connected to that channel.

The CHSTATUS.USRRDYx bit is set to one if all event users connected to the corresponding channel x are ready to handle incoming events on that channel.

23.6.2.8 Software Event

A software event can be initiated on a channel by writing a one to the Software Event bit in the Channel register (CHANNEL.SWEVT) at the same time as writing the Channel bits (CHANNEL.CHANNEL). This will generate a software event on the selected channel.

The software event can be used for application debugging, and functions like any event generator. To use the software event, the event path must be configured to either a synchronous path or resynchronized path

- USART mode with external or internal clock must be selected first by writing 0x0 or 0x1 to the Operating Mode bit group in the Control A register (CTRLA.MODE)
- Communication mode (asynchronous or synchronous) must be selected by writing to the Communication Mode bit in the Control A register (CTRLA.CMODE)
- SERCOM pad to use for the receiver must be selected by writing to the Receive Data Pinout bit group in the Control A register (CTRLA.RXPO)
- SERCOM pads to use for the transmitter and external clock must be selected by writing to the Transmit Data Pinout bit in the Control A register (CTRLA.TXPO)
- Character size must be selected by writing to the Character Size bit group in the Control B register (CTRLB.CHSIZE)
- MSB- or LSB-first data transmission must be selected by writing to the Data Order bit in the Control A register (CTRLA.DORD)
- When parity mode is to be used, even or odd parity must be selected by writing to the Parity Mode bit in the Control B register (CTRLB.PMODE) and enabled by writing 0x1 to the Frame Format bit group in the Control A register (CTRLA.FORM)
- Number of stop bits must be selected by writing to the Stop Bit Mode bit in the Control B register (CTRLB.SBMODE)
- When using an internal clock, the Baud register (BAUD) must be written to generate the desired baud rate
- The transmitter and receiver can be enabled by writing ones to the Receiver Enable and Transmitter Enable bits in the Control B register (CTRLB.RXEN and CTRLB.TXEN)

25.6.2.2 Enabling, Disabling and Resetting

The USART is enabled by writing a one to the Enable bit in the Control A register (CTRLA.ENABLE). The USART is disabled by writing a zero to CTRLA.ENABLE.

The USART is reset by writing a one to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the USART, except DBGCTRL, will be reset to their initial state, and the USART will be disabled. Refer to the CTRLA register for details.

25.6.2.3 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. Synchronous mode is selected by writing a one to the Communication Mode bit in the Control A register (CTRLA.CMODE) and asynchronous mode is selected by writing a zero to CTRLA.CMODE. The internal clock source is selected by writing 0x1 to the Operation Mode bit group in the Control A register (CTRLA.MODE) and the external clock source is selected by writing 0x0 to CTRLA.MODE.

The SERCOM baud-rate generator is configured as shown in Figure 25-3. When CTRLA.CMODE is zero, the baud-rate generator is automatically set to asynchronous mode and the 16-bit Baud register value is used. When CTRLA.CMODE is one, the baud-rate generator is automatically set to synchronous mode and the eight LSBs of the Baud register are used. Refer to “Clock Generation – Baud-Rate Generator” on page 433 for details on configuring the baud rate.

- **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.
  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.

Table 25-11. Run In Standby

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 can wake up on Receive Start or Transfer Complete interrupt.
0x1	Wake on Receive Start or Receive Complete interrupt.	Generic clock is enabled in all sleep modes. Any interrupt can wake up the device.

- **Bits 6:5 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 4:2 – MODE: Operating Mode**
  These bits must be written to 0x0 or 0x1 to select the USART serial communication interface of the SERCOM.
  0x0: USART with external clock.
  0x1: USART with internal clock.
  These bits are not synchronized.

- **Bit 1 – ENABLE: Enable**
  0: The peripheral is disabled or being disabled.
  1: The peripheral is enabled or being enabled.
  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.

- **Bit 0 – SWRST: Software Reset**
  0: There is no reset operation ongoing.
  1: The reset operation is ongoing.
  Writing a zero to this bit has no effect.
  Writing a one to this bit resets all registers in the SERCOM, except DBGCTRL, to their initial state, and the SERCOM will be disabled.
  Writing a one 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.

29.8 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). Write-protection is denoted by the Write-Protected property in each individual register description. Please refer to the “Register Access Protection” on page 616 and the PAC chapter for details.

Some registers require synchronization when read and/or written. Synchronization is denoted by the Write-Synchronized or Read-Synchronized property in each individual register description. Please refer to the “Synchronization” on page 646 for details.

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

29.8.3 Control B Set

Name: CTRLBSET

Offset: 0x05

Reset: 0x00

Access: Read-Write

Property: Write-Protected, Write-Synchronized

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

This register allows the user to change the below mentioned functionalities without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Clear (CTRLBCLR) register.

- **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 field will be read back as zero. The commands are executed on the next prescaled GCLK_TCC clock cycle.
  Writing a zero to this bit group has no effect
  Writing a valid value into this bit group will set the associated command, as shown in the table below.

Table 29-12. TCC Command

CMD[2:0]	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Clear start, restart or retrigger
0x2	STOP	Force stop
0x3	UPDATE	Force update or double buffered registers
0x4	READSYNC	Force COUNT read synchronization
0x5-0x7		Reserved

- **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, according to the table below. 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 this field has no effect.
  Writing a valid value into this field will set a command.

Table 29-13. Ramp Index Command

IDXCMD[1:0]	Name	Description
0x0	DISABLE	Command disabled: Index toggles between cycles A and B

- **Bit 17 – WAVEB: Wave Buffer Busy**
  This bit is cleared when the synchronization of WAVEB register between the clock domains is complete.
  This bit is set when the synchronization of WAVEB register between clock domains is started.
- **Bit 16 – PATTB: Pattern Buffer Busy**
  This bit is cleared when the synchronization of PATTB register between the clock domains is complete.
  This bit is set when the synchronization of PATTB register between clock domains is started.
- **Bits 15:12 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 11:8 – CCx [x=3..0]: Compare Channel x Busy**
  This bit is cleared when the synchronization of Compare/Capture Channel x register between the clock domains is complete.
  This bit is set when the synchronization of Compare/Capture Channel x register between clock domains is started.
  CCx bit is available only for existing Compare/Capture Channels. For details on CC channels number, refer to each TCC feature list.
  This bit is set when the synchronization of CCx register between clock domains is started.
- **Bit 7 – PER: Period busy**
  This bit is cleared when the synchronization of PER register between the clock domains is complete.
  This bit is set when the synchronization of PER register between clock domains is started.
- **Bit 6 – WAVE: Wave Busy**
  This bit is cleared when the synchronization of WAVE register between the clock domains is complete.
  This bit is set when the synchronization of WAVE register between clock domains is started.
- **Bit 5 – PATT: Pattern Busy**
  This bit is cleared when the synchronization of PATT register between the clock domains is complete.
  This bit is set when the synchronization of PATT register between clock domains is started.
- **Bit 4 – COUNT: Count Busy**
  This bit is cleared when the synchronization of COUNT register between the clock domains is complete.
  This bit is set when the synchronization of COUNT register between clock domains is started.
- **Bit 3 – STATUS: Status Busy**
  This bit is cleared when the synchronization of STATUS register between the clock domains is complete.
  This bit is set when the synchronization of STATUS register between clock domains is started.
- **Bit 2 – CTRLB: Ctrlb Busy**
  This bit is cleared when the synchronization of CTRLB register between the clock domains is complete.
  This bit is set when the synchronization of CTRLB register between clock domains is started.
- **Bit 1 – ENABLE: Enable Busy**
  This bit is cleared when the synchronization of ENABLE register bit between the clock domains is complete.
  This bit is set when the synchronization of ENABLE register bit between clock domains is started.
- **Bit 0 – SWRST: Swrst Busy**
  This bit is cleared when the synchronization of SWRST register bit between the clock domains is complete.
  This bit is set when the synchronization of SWRST register bit between clock domains is started.

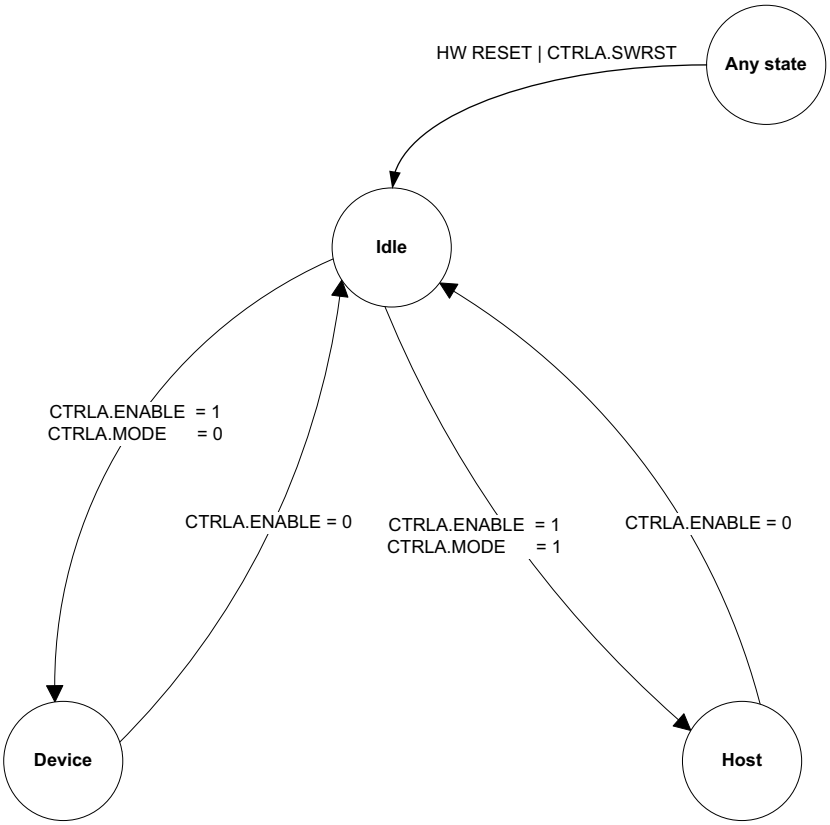
30.6 Functional Description

30.6.1 USB General Operation

30.6.1.1 Initialization

After a hardware reset, the USB is disabled. The user should first enable the USB (CTRLA.ENABLE) in device mode.

Figure 30-2. General States



After a hardware reset, the USB is in the idle state. In this state:

- The module is disabled. The USB Enable bit in the Control A register (CTRLA.ENABLE) is reset.
- The module clock is stopped in order to minimize power consumption.
- The USB pad is in suspend mode.
- The internal states and registers of the device are reset.

Before using the USB, the Pad Calibration register (PADCAL) must be loaded with production calibration values from the NVM Software Calibration Area. Refer to "NVM Software Calibration Row Mapping" on page 24 for further details.

The USB is enabled by writing a one to CTRLA.ENABLE. The USB is disabled by writing a zero to CTRLA.ENABLE.

The USB is reset by writing a one to the Software Reset bit in CTRLA (CTRLA.SWRST). All registers in the USB will be reset to their initial state, and the USB will be disabled. Refer to the CTRLA register for details.

The user can configure pads and speed before enabling the USB by writing to the Speed Configuration field in the Control B register (CTRLB.SPDCONF). These values are taken into account once the USB has been enabled by writing a one to CTRLA.ENABLE.

After writing a one to CTRLA.ENABLE, the USB enters device mode (according to CTRLA.MODE). Please refer Figure 30-2.

30.8.2.2 Device Address

Name: DADD

Offset: 0x0A

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	ADDEN	**DADD[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 – ADDEN: Device Address Enable**
  0: Writing a zero will deactivate the DADD field (USB device address) and return the device to default address 0.
  1: Writing a one will activate the DADD field (USB device address).
  This bit is cleared when a USB reset is received.

- **Bits 6:0 – DADD: Device Address**
  These bits define the device address. The DADD register is reset when a USB reset is received.

Writing a one to the bit EPSTATUSSET.CURBK will set this bit.

- **Bit 1 – DTGLIN: Data Toggle IN Sequence**
  0: The PID of the next expected IN transaction will be zero: data 0.
  1: The PID of the next expected IN transaction will be one: data 1.
  Writing a zero to the bit EPSTATUSCLR.DTGLINCLR will clear this bit.
  Writing a one to the bit EPSTATUSSET.DTGLINSET will set this bit.

- **Bit 0 – DTGLOUT: Data Toggle OUT Sequence**
  0: The PID of the next expected OUT transaction will be zero: data 0.
  1: The PID of the next expected OUR transaction will be one: data 1.
  Writing a zero to the bit EPSTATUSCLR.DTGLOUTCLR will clear this bit.
  Writing a one to the bit EPSTATUSSET.DTGLOUTSET will set this bit.

30.8.4 Device Registers - Endpoint RAM

30.8.4.1 Endpoint Descriptor structure

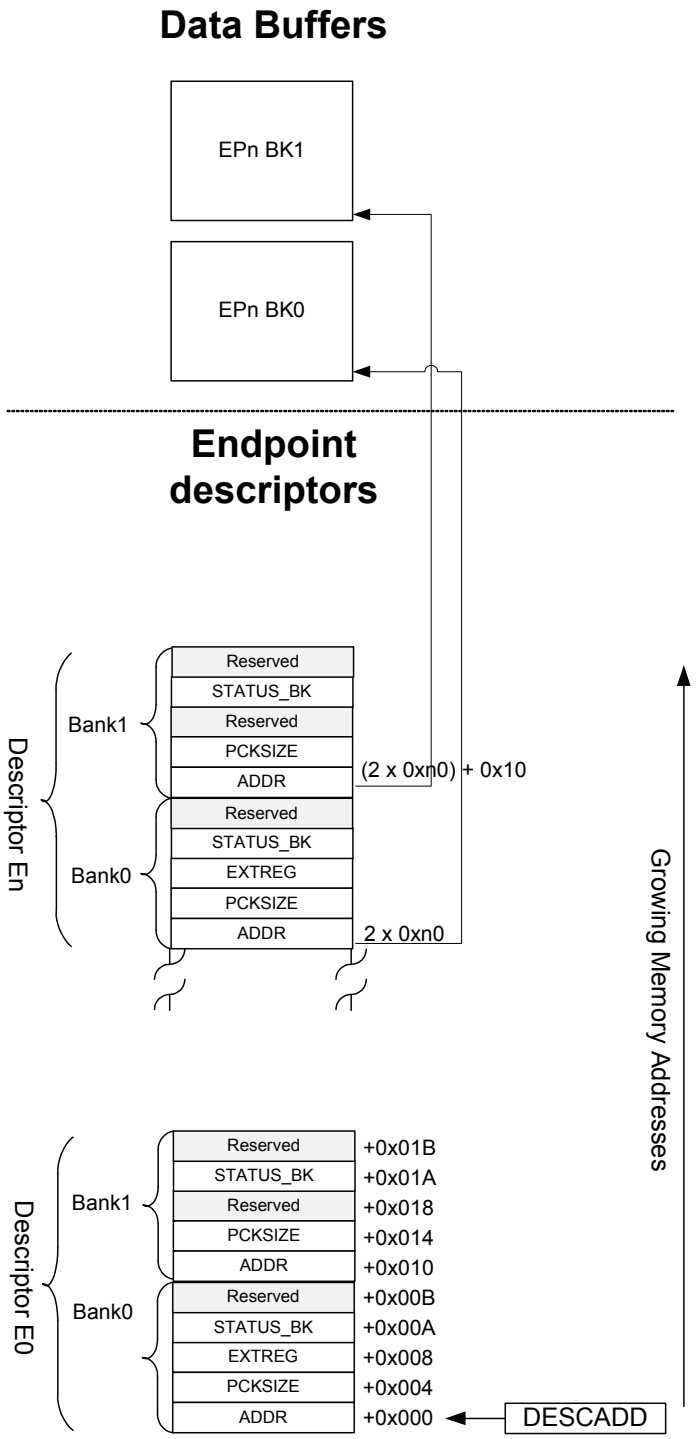


Table 30-12. Endpoint Size

SIZE[2:0]	Description
0x0	8 Byte
0x1	16 Byte
0x2	32 Byte
0x3	64 Byte
0x4	128 Byte (1)
0x5	256 Byte (1)
0x6	512 Byte (1)
0x7	1023 Byte (1)

1. for Isochronous endpoints only.

- **Bits 27:14 – MULTI_PACKET_SIZE: Multiple Packet Size**
  These bits define the 14-bit value that is used for multi-packet transfers.
  For IN endpoints, MULTI_PACKET_SIZE holds the total number of bytes sent. MULTI_PACKET_SIZE should be written to zero when setting up a new transfer.
  For OUT endpoints, MULTI_PACKET_SIZE holds the total data size for the complete transfer. This value must be a multiple of the maximum packet size.
- **Bits 13:0 – BYTE_COUNT: Byte Count**
  These bits define the 14-bit value that is used for the byte count.
  For IN endpoints, BYTE_COUNT holds the number of bytes to be sent in the next IN transaction.
  For OUT endpoint or SETUP endpoints, BYTE_COUNT holds the number of bytes received upon the last OUT or SETUP transaction.

32.9.10 Window Control

Name: WINCTRL

Offset: 0x0C

Reset: 0x00

Access: Read-Write

Property: Write-Protected, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
						WINTSEL0[1:0]		WEN0
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

- **Bits 2:1 – WINTSEL0[1:0]: Window 0 Interrupt Selection**
  These bits configure the interrupt mode for the comparator window 0 mode.

Table 32-5. Window 0 Interrupt Selection

WINTSEL0[1:0]	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 – WEN0: Window 0 Mode Enable**
  0: Window mode is disabled for comparators 0 and 1.
  1: Window mode is enabled for comparators 0 and 1.

33.6.3.3 Data Buffer

The Data Buffer register (DATABUF) and the Data register (DATA) are linked together to form a two-stage FIFO. The DAC uses the Start Conversion event to load data from DATABUF into DATA and start a new conversion. The Start Conversion event is enabled by writing a one to the Start Event Input bit in the Event Control register (EVCTRL.STARTEI). If a Start Conversion event occurs when DATABUF is empty, an Underrun interrupt request is generated if the Underrun interrupt is enabled.

The DAC can generate a Data Buffer Empty event when DATABUF becomes empty and new data can be loaded to the buffer. The Data Buffer Empty event is enabled by writing a one to the Empty Event Output bit in the Event Control register (EVCTRL.EMPTYEO). A Data Buffer Empty interrupt request is generated if the Data Buffer Empty interrupt is enabled.

33.6.3.4 Voltage Pump

When the DAC is used at operating voltages lower than 2.5V, the voltage pump must be enabled. This enabling is done automatically, depending on operating voltage.

The voltage pump can be disabled by writing a one to the Voltage Pump Disable bit in the Control B register (CTRLB.VPD). This can be used to reduce power consumption when the operating voltage is above 2.5V.

The voltage pump uses the asynchronous GCLK_DAC clock, and requires that the clock frequency be at least four times higher than the sampling period.

33.6.3.5 Sampling Period

As there is no automatic indication that a conversion is done, the sampling period must be greater than or equal to the specified conversion time.

33.6.4 DMA, Interrupts and Events

Table 33-1. Moduel Request for ADC

Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
Data Buffer Empty	x	x		x	When DATABUF is written
Underrun	x				
Synchronization Ready	x				
Start Conversion			x		

33.6.4.1 DMA Operation

The DAC generates the following DMA request:

- Data Buffer Empty (EMPTY): the request is set when the Data Buffer register is empty (data transferred from DATABUF to DATA). The request is cleared when DATABUF is written.

For each Start Conversion event, DATABUF is transferred into DATA and the conversion starts. When DATABUF is empty, the DAC generates the DMA request for new data. As DATABUF is initially not empty, it must be written by the CPU before the first event occurs.

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

When DAC registers are write-protected by Peripheral Access Controller, DATABUF cannot be written. To bypass DATABUF write protection, Bypass DATABUF Write Protection bit (CTRLB.BDWP) must be written to one.