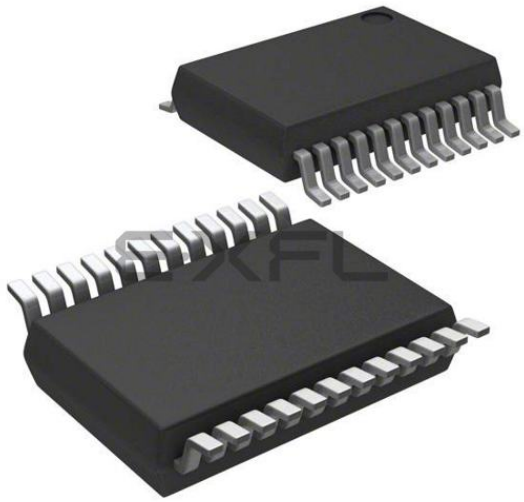


Welcome to E-XFL.COM

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®-M23
Core Size	32-Bit Single-Core
Speed	32MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	17
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	16K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.63V
Data Converters	A/D 5x12b; D/A 1x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	24-SSOP (0.209", 5.30mm Width)
Supplier Device Package	24-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsaml10d16a-yft

Bit Pos.	Name	Usage	Factory Setting	Related Peripheral Register
95:40	Reserved	Reserved	Reserved	Reserved
127:96	ROMVERSION	ROM Code Version	Device Dependent	Boot ROM
511:128	Reserved	Reserved	Reserved	Reserved
639:512	CRCKEY	CRC Key	All '1's	Boot ROM
2047:640	Reserved	Reserved	Reserved	Reserved

Table 10-17. SAM L10 BOCOR Mapping

Offset	Bit Pos.	Name
0x00-0x03	31:0	Reserved
0x04	39:32	BOOTPROT
0x05-0x0B	95:40	Reserved
0x0C-0x0F	127:96	ROMVERSION
0x10-0x3F	511:128	Reserved
0x40-0x4F	639:512	CRCKEY
0x50-0xFF	2047:640	Reserved

10.2.4.2 SAM L11 Boot Configuration Row

Table 10-18. SAM L11 BOCOR Bitfields Definition

Bit Pos.	Name	Usage	Factory Setting	Related Peripheral Register
7:0	Reserved	Reserved	Reserved	Reserved
15:8	BS	Boot Flash Secure Size = BS*0x100	0x00	IDAU.SCFGB
21:16	BNSC	Boot Flash Non-Secure Callable Size = BNSC*0x20	0x00	IDAU.SCFGB
23:22	Reserved	Reserved	Reserved	Reserved
31:24	BOOTOPT	Boot Option	0xA0	Boot ROM
39:32	BOOTPROT	Boot Protection size = BOOTPROT*0x100	0x00	IDAU.SCFGB
47:40	Reserved	Reserved	Reserved	Reserved
48	BCWEN	Boot Configuration Write Enable	0x1	NVMCTRL.SCFGB
49	BCREN	Boot Configuration Read Enable	0x1	NVMCTRL.SCFGB
63:50	Reserved	Reserved	Reserved	Reserved
95:64	BOCORCRC	Boot Configuration CRC for bit 63:0	0xC1D7ECC3	Boot ROM
127:96	ROMVERSION	ROM Code Version	0x0000003A	Boot ROM
255:128	CEKEY0	Chip Erase Key 0	All 1s	Boot ROM
383:256	CEKEY1	Chip Erase Key 1	All 1s	Boot ROM
511:384	CEKEY2	Chip Erase Key 2	All 1s	Boot ROM
639:512	CRCKEY	CRC Key	All 1s	Boot ROM
895:640	BOOTKEY	Secure Boot Key	All 1s	Boot ROM
1791:896	Reserved	Reserved	Reserved	Reserved
2047:1792	BOCORHASH	Boot Configuration Row Hash	All 1s	Boot ROM

15.7.12 Peripheral Non-Secure Status - Bridge A

Name: NONSECA
Offset: 0x54
Reset: x initially determined from NVM User Row after reset
Property: Write-Secure

Important: This register is only available for **SAM L11** and has no effect for **SAM L10**.

Reading NONSEC register returns peripheral security attribution status:

Value	Description
0	Peripheral is secured.
1	Peripheral is non-secured.

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
			AC	PORT	FREQM	EIC	RTC	WDT
Access			R/R/R	R/R/R	R/R/R	R/R/R	R/R/R	R/R/R
Reset			x	x	x	x	x	x

Bit	7	6	5	4	3	2	1	0
	GCLK	SUPC	OSC32KCTRL	OSCCTRL	RSTC	MCLK	PM	PAC
Access	R/R/R	R/R/R	R/R/R	R/R/R	R/R/R	R/R/R	R/R/R	R/R/R
Reset	x	x	x	x	x	x	x	0

Bit 13 – AC Peripheral AC Non-Secure

Bit 12 – PORT Peripheral PORT Non-Secure

Bit 11 – FREQM Peripheral FREQM Non-Secure

Bit 10 – EIC Peripheral EIC Non-Secure

Bit 9 – RTC Peripheral RTC Non-Secure

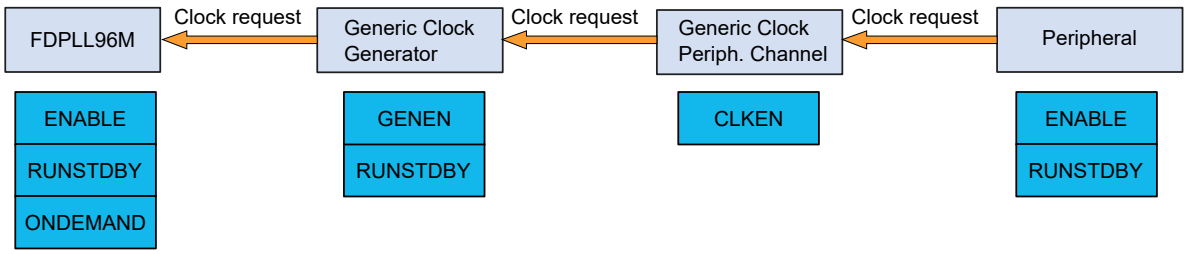
17.4 Enabling a Peripheral

In order to enable a peripheral that is clocked by a Generic Clock, the following parts of the system needs to be configured:

- A running Clock Source
- A clock from the Generic Clock Generator must be configured to use one of the running Clock Sources, and the Generator must be enabled.
- The Peripheral Channel that provides the Generic Clock signal to the peripheral must be configured to use a running Generic Clock Generator, and the Generic Clock must be enabled.
- The user interface of the peripheral needs to be unmasked in the Main Clock Controller (MCLK). If this is not done the peripheral registers will read all '0's and any writing attempts to the peripheral will be discarded.

17.5 On Demand Clock Requests

Figure 17-4. Clock Request Routing



All clock sources in the system can be run in an on-demand mode: the clock source is in a stopped state unless a peripheral is requesting the clock source. Clock requests propagate from the peripheral, via the GCLK, to the clock source. If one or more peripheral is using a clock source, the clock source will be started/kept running. As soon as the clock source is no longer needed and no peripheral has an active request, the clock source will be stopped until requested again.

The clock request can reach the clock source only if the peripheral, the generic clock and the clock from the Generic Clock Generator in-between are enabled. The time taken from a clock request being asserted to the clock source being ready is dependent on the clock source startup time, clock source frequency as well as the divider used in the Generic Clock Generator. The total startup time T_{start} from a clock request until the clock is available for the peripheral is between:

T_{start_max} = Clock source startup time + 2 × clock source periods + 2 × divided clock source periods

T_{start_min} = Clock source startup time + 1 × clock source period + 1 × divided clock source period

The time between the last active clock request stopped and the clock is shut down, T_{stop}, is between:

T_{stop_min} = 1 × divided clock source period + 1 × clock source period

T_{stop_max} = 2 × divided clock source periods + 2 × clock source periods

The On-Demand function can be disabled individually for each clock source by clearing the ONDEMAND bit located in each clock source controller. Consequently, the clock will always run whatever the clock request status is. This has the effect of removing the clock source startup time at the cost of power consumption.

The clock request mechanism can be configured to work in standby mode by setting the RUNSDTBY bits of the modules (see Figure 17-4).

index(m)	Name	Description
9	GCLK_EVSYS_CHANNEL_3	EVSYS_CHANNEL_3
10	GCLK_SERCOM[0,1,2]_SLOW	SERCOM[0,1,2]_SLOW
11	GCLK_SERCOM0_CORE	SERCOM0_CORE
12	GCLK_SERCOM1_CORE	SERCOM1_CORE
13	GCLK_SERCOM2_CORE	SERCOM2_CORE
14	GCLK_TC0, GCLK_TC1	TC0,TC1
15	GCLK_TC2	TC2
16	GCLK_ADC	ADC
17	GCLK_AC	AC
18	GCLK_DAC	DAC
19	GCLK_PTC	PTC
20	GCLK_CCL	CCL

Related Links

33. EVSYS – Event System

27.6.6 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 will continue 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* for more information.

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 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.COUNTSYNC)

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.

27.8.10 Counter Value in COUNT32 mode (CTRLA.MODE=0)

Name: COUNT
Offset: 0x18
Reset: 0x00000000
Property: PAC Write-Protection, 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.

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.

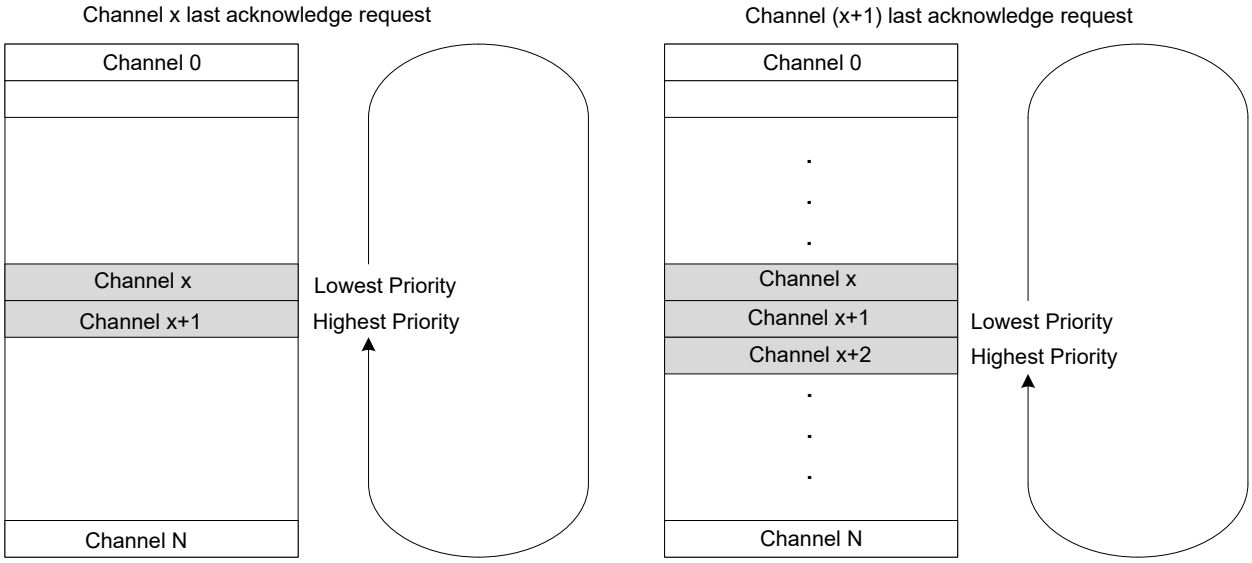
Bit 1 – ENABLE Enable Synchronization Busy Status

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

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

Figure 28-6. Dynamic (Round-Robin) Priority Scheduling



28.6.2.5 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 (refer to DMA Block Diagram section) 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, refer to the section on Addressing.

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.

system frequencies, a programmable number of wait states can be used to optimize performance. When changing the AHB bus frequency, the user must ensure that the NVM Controller is configured with the proper number of wait states. Refer to the Electrical Characteristics for the exact number of wait states to be used for a particular frequency range.

30.5.3 Interrupts

The NVM Controller interrupt request line is connected to the interrupt controller. Using the NVMCTRL interrupt requires the interrupt controller to be programmed first.

30.5.4 Events

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

- Write zeroes in one Data FLASH row: Refer to 30.6.7 Tamper Erase for details.
- Write a page in the FLASH or in the Data FLASH: Refer to 30.6.6 Event Automatic Write for details.

The NVMCTRL uses only asynchronous events, so the asynchronous Event System channel path must be configured. By default, the NVMCTRL will detect a rising edge on the incoming event. If the NVMCTRL action must be performed on the falling edge of the incoming event, the event line must be inverted first. This is done by setting the corresponding Event Invert Enable bit in Event Control register (NVMCTRL.AUTOWINV=1).

30.5.5 Debug Operation

When an external debugger forces the CPU into debug mode, the peripheral continues normal operation except that FLASH reads are not cached so that the cache state is not altered by debug tools.

30.5.6 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 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.

When TrustZone is supported (**SAM L11** only), all register reads are allowed. Non-secure writes to APB registers are limited as follows. Illegal writes will be ignored.

- Some commands written to CTRLA such as Write/Erase and Lock/Unlock are only permitted to non-secure application and data space.
- Writes to all other registers except CTRLC and INTFLAG are not allowed.

Related Links

15. PAC - Peripheral Access Controller

30.5.7 SAM L11 TrustZone Specific Register Access Protection

The NVMCTRL is a split-secure APB module, all registers are available in the secure alias and only a subset of registers is available in the non-secure alias with limited access.

When NONSEC.WRITE is read zero, all APB write accesses to the non-secure APB alias and all non-secure AHB write accesses to the Page Buffer are discarded. The latter returns a hardfault. Any attempt to change the configuration via the non-secure alias is silently ignored.

30.8.8 Status

Name: STATUS
Offset: 0x18
Reset: 0x0X00
Property: Write-Secure

Important: For **SAM L11 Non-Secure** accesses, write accesses (W*) are allowed only if Non-Secure Write is set in NONSEC register.

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				DALFUSE[1:0]		READY	LOAD	PRM
Access				R/R/R	R/R/R	R/R/R	R/R/R	R/R/R
Reset				x	x	0	0	0

Bits 4:3 – DALFUSE[1:0] DAL Fuse Value
This field is the current debugger access level fuse value.

Value	Description
0	DAL = 0 : Access to very limited features.
1	DAL = 1 (**SAM L11 only**): Access to all non-secure memory. Can debug non-secure CPU code.
2	DAL = 2 : Access to all memory. Can debug Secure and non-secure CPU code.
3	Reserved

Bit 2 – READY NVM Ready

Value	Description
0	The NVM controller is busy programming or erasing.
1	The NVM controller is ready to accept a new command.

Bit 1 – LOAD NVM Page Buffer Active Loading
This bit indicates that the NVM page buffer has been loaded with one or more words. Immediately after an NVM load has been performed, this flag is set. It remains set until a page write or a page buffer clear (PBC) command is given.

Bit 0 – PRM Power Reduction Mode
This bit indicates the current NVM power reduction state. The NVM block can be set in power reduction mode in two ways: through the command interface or automatically when entering sleep with SLEEPPRM set accordingly.

Value	Description
	to change the pin configuration through the non-secure alias will be silently ignored and reads will return 0.
1	The corresponding I/O pin in the PORT group is configured as non-secured. The I/O line configuration for this pin is available through the non-secure alias.

33.7.3 Priority Control

Name: PRICTRL
Offset: 0x08
Reset: 0x00
Property: PAC Write-Protection, Secure

Bit	7	6	5	4	3	2	1	0
	RREN						PRI[1:0]	
Access	RW/RW/RW						RW/-/RW	RW/-/RW
Reset	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 1:0 – PRI[1:0] 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).

33.7.5 Interrupt Status

Name: INTSTATUS
Offset: 0x14
Reset: 0x00000000
Property: Mix-Secure

Important: For **SAM L11 Non-Secure** accesses, read accesses (R*) are allowed only if the security attribution for the corresponding channel (CHANNELx) is set as Non-Secured in the NONSECCHAN register.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

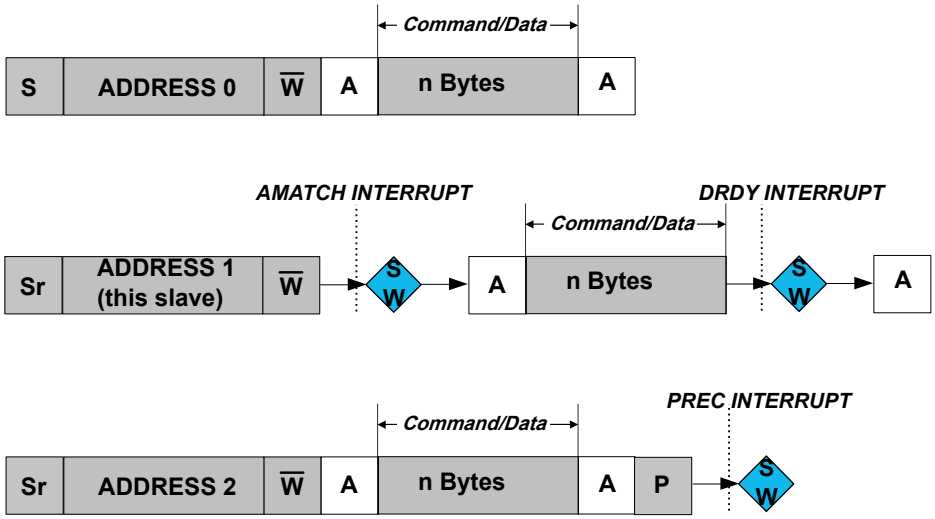
Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					CHINT3	CHINT2	CHINT1	CHINT0
Access					R/R*/R	R/R*/R	R/R*/R	R/R*/R
Reset					0	0	0	0

Bits 0, 1, 2, 3 – CHINT 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.

Figure 37-13. PMBus Group Command Example



37.6.3 Additional Features

37.6.3.1 SMBus

The I^2C includes three hardware SCL low time-outs which allow a time-out to occur for SMBus SCL low time-out, master extend time-out, and slave 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 32KHz oscillator. The I^2C interface also allows for a SMBus compatible SDA hold time.

- $T_{TIMEOUT}$: SCL low time of 25..35ms – Measured for a single SCL low period. It is enabled by CTRLA.LOWTOUTEN.
- $T_{LOW:SEXT}$: Cumulative clock low extend time of 25 ms – Measured as the cumulative SCL low extend time by a slave device in a single message from the initial START to the STOP. It is enabled by CTRLA.SEXTTOEN.
- $T_{LOW:MEXT}$: Cumulative clock low extend time of 10 ms – Measured as the cumulative SCL low extend time by the master device within a single byte from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is enabled by CTRLA.MEXTTOEN.

37.6.3.2 Smart Mode

The I^2C interface has a smart mode that simplifies application code and minimizes the user interaction needed to adhere to the I^2C 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.

37.6.3.3 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^2C tri-state drivers are bypassed, and an external I^2C compliant tri-state driver is needed when connecting to an I^2C bus.

Table 37-2. Module Request for SERCOM I²C Master

Condition	Request		
	DMA	Interrupt	Event
Data needed for transmit (TX) (Master transmit mode)	Yes (request cleared when data is written)		NA
Data needed for transmit (RX) (Master transmit mode)	Yes (request cleared when data is read)		
Master on Bus (MB)		Yes	
Stop received (SB)		Yes	
Error (ERROR)		Yes	

37.6.4.1 DMA Operation

Smart mode must be enabled for DMA operation in the Control B register by writing CTRLB.SMEN=1.

37.6.4.1.1 Slave DMA

When using the I²C slave with DMA, an address match will cause the address interrupt flag (INTFLAG.ADDRMATCH) to be raised. After the interrupt has been serviced, data transfer will be performed through DMA.

The I²C slave generates the following requests:

- Write data received (RX): The request is set when master 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 master read operation. The request is cleared when DATA is written.

37.6.4.1.2 Master DMA

When using the I²C master 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 master reads) and a STOP.

If a NACK is received by the slave for a master write transaction before ADDR.LEN bytes, a STOP will be automatically generated and the length error (STATUS.LENERR) will be raised along with the INTFLAG.ERROR interrupt.

The I²C master generates the following requests:

- Read data received (RX): The request is set when master 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 master write operation. The request is cleared when DATA is written.

37.8.6 Status

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

Bit	15	14	13	12	11	10	9	8
					LENERR	HS	SEXTTOUT	
Access					R/W	R/W	R/W	
Reset					0	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 when responding to a new start condition with ACK or NACK (CTRLB.CMD=0x3) 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.

Bit 10 – HS High-speed
This bit is set if the slave detects a START followed by a Master Code transmission.

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

Writing a '1' to this bit will clear the status. However, this flag is automatically cleared when a STOP is received.

Bit 9 – SEXTTOUT Slave SCL Low Extend Time-Out
This bit is set if a slave 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.

38.7.2.7 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
				MCx			ERR	OVF
Access				R/W			R/W	R/W
Reset				0			0	0

Bit 4 – 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.

42.7 Register Summary

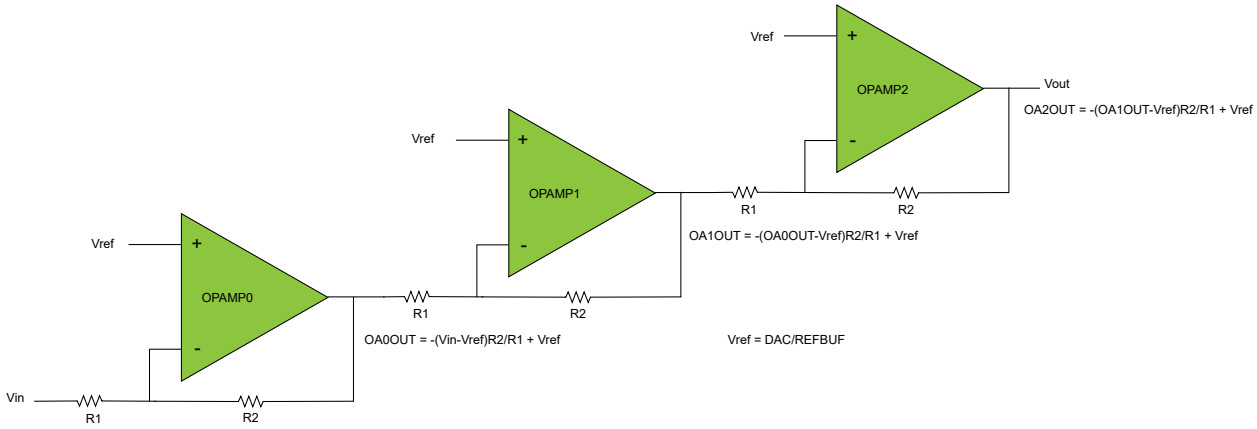
Offset	Name	Bit Pos.								
0x00	CTRLA	7:0							ENABLE	SWRST
0x01	CTRLB	7:0							STARTx	STARTx
0x02	EVCTRL	7:0				WINEO0			COMPEOx	COMPEOx
		15:8			INVEIx	INVEIx			COMPEIx	COMPEIx
0x04	INTENCLR	7:0				WIN0			COMPx	COMPx
0x05	INTENSET	7:0				WIN0			COMPx	COMPx
0x06	INTFLAG	7:0				WIN0			COMPx	COMPx
0x07	STATUSA	7:0			WSTATE0[1:0]				STATEx	STATEx
0x08	STATUSB	7:0							READYx	READYx
0x09	DBGCTRL	7:0								DBGRUN
0x0A	WINCTRL	7:0						WINTSEL0[1:0]		WEN0
0x0B	Reserved									
0x0C	SCALER0	7:0			VALUE[5:0]					
0x0D	SCALER1	7:0			VALUE[5:0]					
0x0E ... 0x0F	Reserved									
0x10	COMPCTRL0	7:0		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE	
		15:8	SWAP	MUXPOS[2:0]				MUXNEG[2:0]		
		23:16			HYST[1:0]		HYSTEN		SPEED[1:0]	
		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			HYST[1:0]		HYSTEN		SPEED[1:0]	
		31:24			OUT[1:0]			FLEN[2:0]		
0x18 ... 0x1F	Reserved									
0x20	SYNCBUSY	7:0				COMPCTRLx	COMPCTRLx	WINCTRL	ENABLE	SWRST
		15:8								
		23:16								
		31:24								

42.8 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 Register Access Protection.

Figure 44-5. Cascaded Inverting PGA



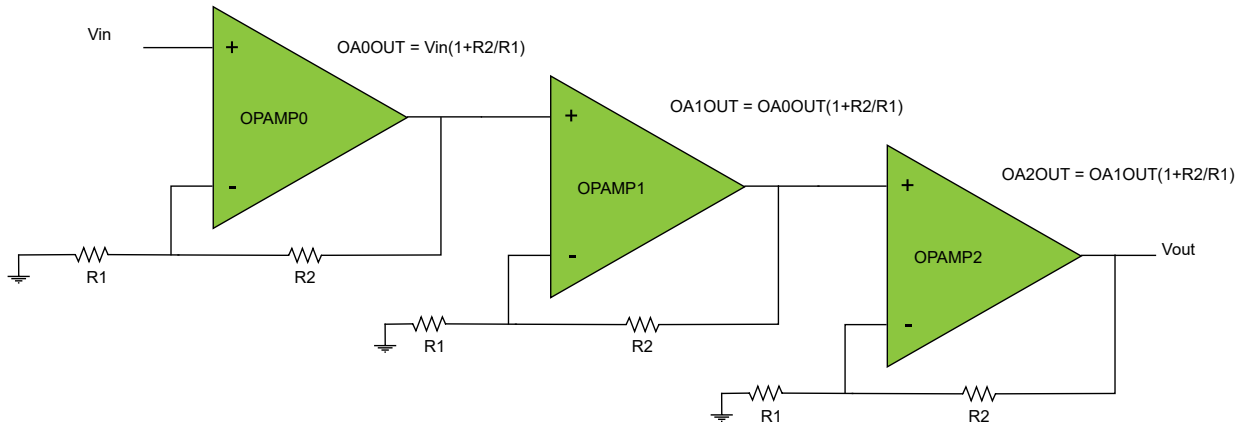
44.6.10.5 Cascaded Non-Inverting PGA

The OPAMPs can be configured as three cascaded, non-inverting PGAs using these settings in OPAMPCTRLx:

Table 44-6. Cascaded Non-Inverting PGA (Exemple: R1=4R, R2=12R)

	MUXPOS	MUXNEG	RES1MUX	POTMUX	RES2VCC	RES2OUT	RES1EN	ANAOUT
OPAMP0	0000	001	011	100	0	1	1	0
OPAMP1	0100	001	011	100	0	1	1	0
OPAMP2	0100	001	011	100	0	1	1	0

Figure 44-6. Cascaded Non-Inverting PGA



44.6.10.6 Two OPAMPs Differential Amplifier

In this mode, OPAMP0 can be coupled with OPAMP1 or OPAMP1 with OPAMP2 in order to amplify a differential signal.

To configure OPAMP0 and OPAMP1 as differential amplifier, the OPAMPCTRLx register can be configured as follows: