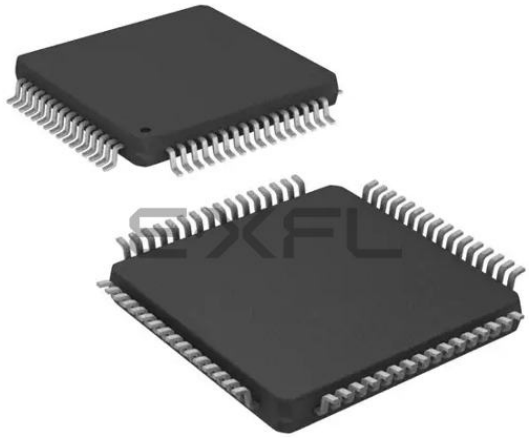


Welcome to E-XFL.COM

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®-M4F
Core Size	32-Bit Single-Core
Speed	120MHz
Connectivity	CANbus, EBI/EMI, I²C, IrDA, LINbus, MMC/SD, QSPI, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, I²S, POR, PWM
Number of I/O	51
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128K x 8
Voltage - Supply (Vcc/Vdd)	1.71V ~ 3.63V
Data Converters	A/D 24x12b; D/A 2x12b
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/atsame51j18a-au

21.8.1 Control A in COUNT32 mode (CTRLA.MODE=0)

Name: CTRLA
Offset: 0x00
Reset: 0x0000
Property: PAC Write-Protection, Enable-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	COUNTSYNC	GPTRST			PRESCALER[3:0]			
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

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

Bit 15 – COUNTSYNC COUNT Read Synchronization Enable
The COUNT register requires synchronization when reading. Disabling the synchronization will prevent reading valid values from the COUNT register.
This bit is not enable-protected.

Value	Description
0	COUNT read synchronization is disabled
1	COUNT read synchronization is enabled

Bit 14 – GPTRST GP Registers Reset On Tamper Enable
Only GP registers enabled by the CTRLB.GPnEN bits are affected. This bit can be written only when the peripheral is disabled.
This bit is not synchronized.

Bits 11:8 – PRESCALER[3:0] Prescaler
These bits define the prescaling factor for the RTC clock source (GCLK_RTC) to generate the counter clock (CLK_RTC_CNT). Periodic events and interrupts are not available when the prescaler is off. These bits are not synchronized.

Value	Name	Description
0x0	OFF	CLK_RTC_CNT = GCLK_RTC/1
0x1	DIV1	CLK_RTC_CNT = GCLK_RTC/1
0x2	DIV2	CLK_RTC_CNT = GCLK_RTC/2
0x3	DIV4	CLK_RTC_CNT = GCLK_RTC/4
0x4	DIV8	CLK_RTC_CNT = GCLK_RTC/8
0x5	DIV16	CLK_RTC_CNT = GCLK_RTC/16
0x6	DIV32	CLK_RTC_CNT = GCLK_RTC/32
0x7	DIV64	CLK_RTC_CNT = GCLK_RTC/64
0x8	DIV128	CLK_RTC_CNT = GCLK_RTC/128
0x9	DIV256	CLK_RTC_CNT = GCLK_RTC/256
0xA	DIV512	CLK_RTC_CNT = GCLK_RTC/512

21.8.3 Event Control in COUNT32 mode (CTRLA.MODE=0)

Name: EVCTRL
Offset: 0x04
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

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

Bit	15	14	13	12	11	10	9	8
	OVFEO	TAMPEREO					CMPEOn[1:0]	
Access	R/W	R/W					R/W	R/W
Reset	0	0					0	0

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

Bit 16 – TAMPEVEI Tamper Event Input Enable

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

Bit 15 – OVFEO Overflow Event Output Enable

Value	Description
0	Overflow event is disabled and will not be generated.
1	Overflow event is enabled and will be generated for every overflow.

Bit 14 – TAMPEREO Tamper Event Output Enable

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

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

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

Bits 7:0 – PEREOn[7:0] Periodic Interval n Event Output Enable [n = 7..0]

Value	Description
0	Periodic Interval n event is disabled and will not be generated.
1	Periodic Interval n event is enabled and will be generated.

Value	Name	Description
0x2A	CH10	DMA channel 10
0x2B	CH11	DMA channel 11
0x2C	CH12	DMA channel 12
0x2D	CH13	DMA channel 13
0x2E	CH14	DMA channel 14
0x2F	CH15	DMA channel 15
0x30	CH16	DMA channel 16
0x31	CH17	DMA channel 17
0x32	CH18	DMA channel 18
0x33	CH19	DMA channel 19
0x34	CH20	DMA channel 20
0x35	CH21	DMA channel 21
0x36	CH22	DMA channel 22
0x37	CH23	DMA channel 23
0x38	CH24	DMA channel 24
0x39	CH25	DMA channel 25
0x3A	CH26	DMA channel 26
0x3B	CH27	DMA channel 27
0x3C	CH28	DMA channel 28
0x3D	CH29	DMA channel 29
0x3E	CH30	DMA channel 30
0x3F	CH31	DMA channel 31

Bits 3:2 – CRCPOLY[1:0] CRC Polynomial Type
These bits select the CRC polynomial type.

Value	Name	Description
0x0	CRC16	CRC-16 (CRC-CCITT)
0x1	CRC32	CRC32 (IEEE 802.3)
0x2-0x3		Reserved

Bits 1:0 – CRCBEATSIZE[1:0] CRC Beat Size
These bits define the size of the data transfer for each bus access when the CRC is used with I/O interface.

Value	Name	Description
0x0	BYTE	8-bit bus transfer
0x1	HWORD	16-bit bus transfer
0x2	WORD	32-bit bus transfer
0x3		Reserved

minimum packet buffer memory size should be chosen to satisfy the maximum frame to be transmitted in the application.

In full store and forward mode, once the complete transmit frame is written into the packet buffer memory, a trigger is sent across to the MAC transmitter, which will then begin reading the frame from the packet buffer memory. Since the whole frame is present and stable in the packet buffer memory an underflow of the transmitter is not possible. The frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be retransmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from AHB system memory.

In Partial Store and Forward mode, a trigger is sent across to the MAC transmitter as soon as sufficient packet data is available, which will then begin fetching the frame from the packet buffer memory. If, after this point, the MAC transmitter is able to fetch data from the packet buffer faster than the AHB DMA can fill it, an underflow of the transmitter is possible. In this case, the transmission is terminated early, and the packet buffer is completely flushed. Transmission can only be restarted by writing a '1' to the Transmit Start bit in the Network Control register (NCR.TSTART).

In half duplex mode, the frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be retransmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from AHB system memory.

In full duplex mode, the frame is removed from the packet buffer on the fly.

Other than underflow, the only MAC related errors that can occur are due to collisions during half duplex transmissions. When a collision occurs the frame still exists in the packet buffer memory so can be retried directly from there. After sixteen failed transmit attempts, the frame will be flushed from the packet buffer.

24.6.3.8 Receive Packet Buffer

The receive packet buffer stores frames from the MAC receiver along with their status and statistics. Frames with errors are flushed from the packet buffer memory, while good frames are pushed onto the DMA AHB interface.

The receiver packet buffer monitors the FIFO write interface from the MAC receiver and translates the FIFO pushes into packet buffer writes. At the end of the received frame the status and statistics are buffered so that the information can be used when the frame is read out. When programmed in full store and forward mode and the frame has an error, the frame data is immediately flushed from the packet buffer memory allowing subsequent frames to utilize the freed up space. The status and statistics for bad frames are still used to update the GMAC registers.

To accommodate the status and statistics associated with each frame, three words per packet (or two if configured in 64-bit datapath mode) are reserved at the end of the packet data. If the packet is bad and requires to be dropped, the status and statistics are the only information held on that packet.

The receiver packet buffer will also detect a full condition so that an overflow condition can be detected. If this occurs, subsequent packets are dropped and an RX overflow interrupt is raised.

For full store and forward, the DMA only begins packet fetches once the status and statistics for a frame are available. If the frame has a bad status due to a frame error, the status and statistics are passed on to the GMAC registers. If the frame has a good status, the information is used to read the frame from the packet buffer memory and burst onto the AHB using the DMA buffer management protocol. Once the last frame data has been transferred to the packet buffer, the status and statistics are updated to the GMAC registers.

24.9.53 GMAC Multiple Collision Frames Register

Name: MCF
Offset: 0x13C
Reset: 0x00000000
Property: Read-Only

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

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

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

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

Bits 17:0 – MCOL[17:0] Multiple Collision
This register counts the number of frames experiencing between two and fifteen collisions prior to being successfully transmitted, i.e., no underrun and not too many retries.

The APB ADDR register is updated upon:

- APB writes to the ADDR register address
- AHB writes to the page buffer

ADDR APB writes are discarded and report an INTFLAG.ADDRE error in the following cases:

- When written from APB while a command is reading it.
- ADDR APB write access while writing the page buffer (AHB write): ADDR is written upon AHB writes and must stay valid until the page buffer has been written and also until automatic write command has been issued to the command interface when in automatic write mode (WMODE configured as ADW or AQW or AP).
- ADDR APB write access while the command interface reads it.
- A command is executed at an illegal address

All commands that require an address are discarded when INTFLAG.ADDRE is set. INTFLAG.PROGE is set in this case. INTFLAG.ADDRE must be cleared before issuing such commands.

25.6.6.1 NVM Read

Reading from the NVM main address space is performed via the AHB bus by addressing the NVM main address space or auxiliary address space directly. Read data is available after the number of read wait states has passed as configured in NVMCTRL.CTRLA.RWS.

The number of cycles data are delayed to the AHB bus is determined by the read wait states.

It is not possible to read two banks at the same time. In case of simultaneous read operations, transactions are arbitrated by the internal matrix. Arbitration scheme is fixed priority, AHB0 has the highest priority, AHB1 has priority over AHB2. In case of conflict, AHB interfaces with lower priority are stalled.

Reading in a bank stalls the bus when it is being programmed or erased except when the suspend feature is used.

Reading in a bank does not stall the bus when the other bank is being programmed or erased.

Related Links

25.6.6.4 Suspend/Resume

25.6.6.2 NVM Write

The entire NVM main address space except the BOOTPROT section can be erased by a debugger Chip Erase command. Alternatively, blocks or pages can be individually erased using the Erase Page (EP) or Erase Block (EB) depending on the targeted address space. The NVM can be programmed using the Write Page (WP) or Write Quad Word (WQW) commands depending on the targeted address space. AHB writes automatically update the ADDR register. ADDR is write locked by the NVMCTRL until the pagebuffer write completes or until the appropriate write command has been passed to the command interface when in automatic write mode. Write commands are not supported in all address spaces, see the table below. These commands are detailed further in this section.

26.8.11 User Initial Hash Value Register

Name: UIHVALx
Offset: 0x38 + x*0x04 [x=0..7]
Reset: 0
Property: -

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

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

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

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

Bits 31:0 – VAL[31:0] Initial Hash Value
When UIHASH bit of CFG register is set, the Initial Hash Value is user-programmable.

To meet the desired standard, use the following example values.

For UIHVAL0 field:

Example	Comment
0x67452301	SHA1 algorithm
0xC1059ED8	SHA224 algorithm
0x6A09E667	SHA256 algorithm

For UIHVAL1 field:

Example	Comment
0xEFCDAB89	SHA1 algorithm
0x367CD507	SHA224 algorithm
0xBB67AE85	SHA256 algorithm

For UIHVAL2 field:

27.7.9 Peripheral Interrupt Flag Status - Bridge D

Name: INTFLAGD
Offset: 0x20
Reset: 0x00000000
Property: –

These flags are set when a Peripheral Access Error occurs while accessing the peripheral associated with the respective INTFLAGx bit, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Writing a '0' to these bits has no effect.

Writing a '1' to these bits will clear the corresponding INTFLAGx interrupt flag.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					PCC	I2S	DAC	ADC1
Access					RW	RW	RW	RW
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADC0	TC7	TC6	TCC4	SERCOM7	SERCOM6	SERCOM5	SERCOM4
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit 11 – PCC Interrupt Flag for PCC
This flag is set when a Peripheral Access Error occurs while accessing the PCC, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Writing a '0' to these bits has no effect.

Writing a '1' to these bits will clear the PCC interrupt flag.

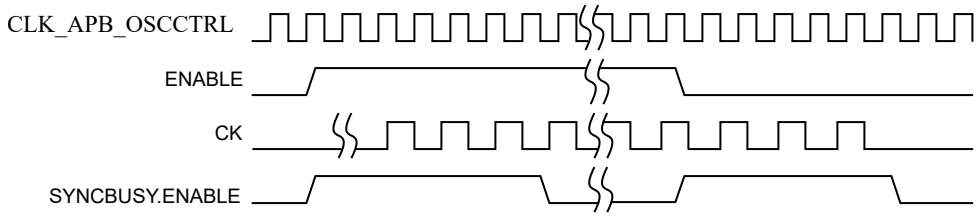
Bit 10 – I2S Interrupt Flag for I2S
This flag is set when a Peripheral Access Error occurs while accessing the I2S, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Writing a '0' to these bits has no effect.

Writing a '1' to these bits will clear the I2S interrupt flag.

Bit 9 – DAC Interrupt Flag for DAC
This flag is set when a Peripheral Access Error occurs while accessing the DAC, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Figure 28-3. Enable synchronization busy operation



The frequency of the DPLLCn output clock CLK_DPLLn is stable when the module is enabled and when the LOCK bit is set. When DPLLnCTRLB.LTIME is different from 0, a user defined lock time is used to validate the lock operation. In that case the lock time is constant. If DPLLnCTRLB.LTIME is zero, the lock signal is linked with the status bit of the DPLLCn (DPLLnSTATUS.LOCK), the lock time vary depending on the filter selection and final target frequency. When DPLLnCTRLB.WUF is set the wake up fast mode is activated. In that mode the clock gating cell is enabled at the end of the startup time. At that time the final frequency is not stable as it is still the acquisition period, but it allows to save hundreds of microseconds. After First acquisition, DPLLnCTRLB.LBYPASS indicates if the Lock signal is discarded from the control of the clock gater (CG) generating the output clock CLK_DPLLn.

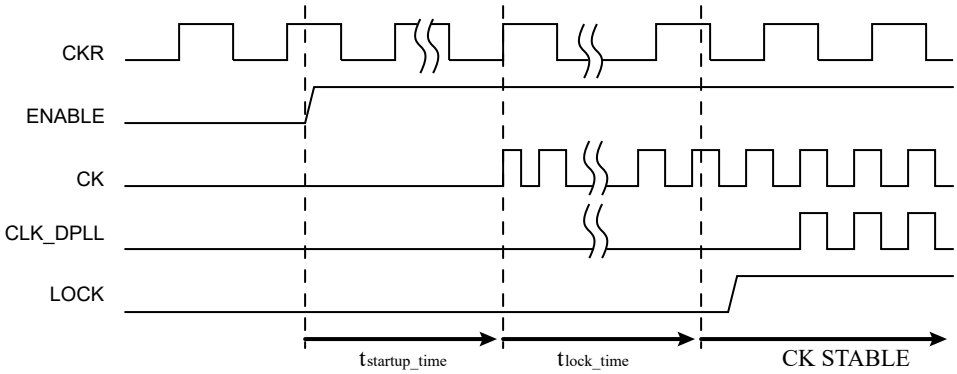
Table 28-3. CLK_DPLLn behavior from startup to first edge detection.

WUF	LTIME	CLK_DPLLn Behavior
0	0	Normal Mode: First Edge when lock is asserted
0	Not Equal To Zero	Lock Timer Timeout mode: First Edge when the timer down-counts to 0.
1	X	Wake Up Fast Mode: First Edge when CK is active (startup time)

Table 28-4. CLK_DPLLn behavior after First Edge detection.

LBYPASS	CLK_DPLLn Behavior
0	Normal Mode: the CLK_DPLLn is turned off when lock signal is low.
1	Lock Bypass Mode: the CLK_DPLLn is always running, lock is irrelevant.

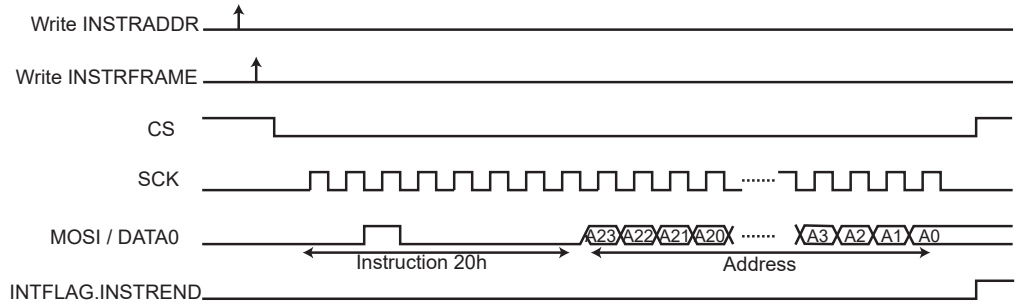
Figure 28-4. CK and CLK_DPLL output from DPLL off mode to running mode



34.7 Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY			MODE[2:0]			ENABLE	SWRST
		15:8	SAMPR[2:0]					RXINV	TXINV	IBON
		23:16	SAMPA[1:0]		RXPO[1:0]				TXPO[1:0]	
		31:24		DORD	CPOL	CMODE	FORM[3:0]			
0x04	CTRLB	7:0		SBMODE				CHSIZE[2:0]		
		15:8			PMODE			ENC	SFDE	COLDEN
		23:16							RXEN	TXEN
		31:24							LINCMD[1:0]	
0x08	CTRLC	7:0						GTIME[2:0]		
		15:8					HDRDLY[1:0]		BRKLEN[1:0]	
		23:16		MAXITER[2:0]					DSNACK	INACK
		31:24							DATA32B[1:0]	
0x0C	BAUD	7:0	BAUD[7:0]							
		15:8	BAUD[15:8]							
0x0E	RXPL	7:0	RXPL[7:0]							
0x0F ... 0x13	Reserved									
0x14	INTENCLR	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
0x15	Reserved									
0x16	INTENSET	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
0x17	Reserved									
0x18	INTFLAG	7:0	ERROR		RXBRK	CTSIC	RXS	RXC	TXC	DRE
0x19	Reserved									
0x1A	STATUS	7:0	ITER	TXE	COLL	ISF	CTS	BUFOVF	FERR	PERR
		15:8								
0x1C	SYNCBUSY	7:0				LENGTH	RXERRCNT	CTRLB	ENABLE	SWRST
		15:8								
		23:16								
		31:24								
0x20	RXERRCNT	7:0	RXERRCNT[7:0]							
0x21	Reserved									
0x22	LENGTH	7:0	LEN[7:0]							
		15:8							LENEN[1:0]	
0x24 ... 0x27	Reserved									
0x28	DATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							
0x2C ...	Reserved									

Figure 37-14. Instruction Transmission Waveform 3



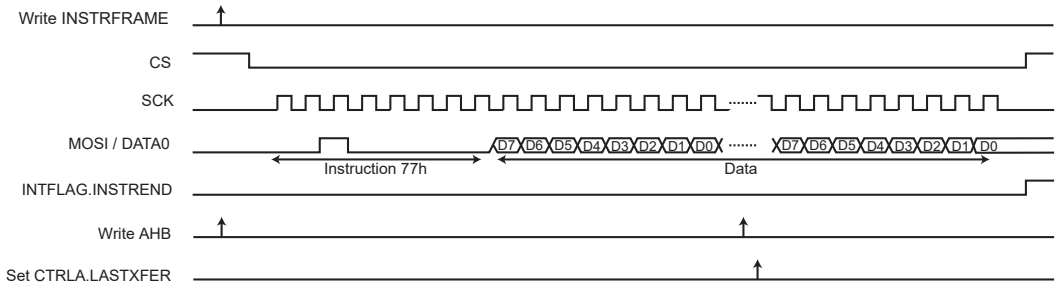
Example 37-4. Example 4

Instruction in Single-bit SPI, without address, without option, with data write in Single-bit SPI.

Command: SET BURST (77h)

- Write 0x0000_0077 to INSTRCTRL register.
- Write 0x0000_2090 to INSTRFRAME register.
- Read INSTRFRAME register (dummy read) to synchronize system bus accesses.
- Write data to the system bus memory space (0x0400_0000–0x0500_0000). The address of the system bus write accesses is not used.
- Write the LASTXFR bit in CTRLA register to '1'.
- Wait for INTFLAG.INSTREND to rise.

Figure 37-15. Instruction Transmission Waveform 4



Example 37-5. Example 5

Instruction in Single-bit SPI, with address in Dual SPI, without option, with data write in Dual SPI.

Command: BYTE/PAGE PROGRAM (02h)

- Write 0x0000_0002 to INSTRCTRL register.
- Write 0x0000_30B3 to INSTRFRAME register.
- Read INSTRFRAME register (dummy read) to synchronize system bus accesses.
- Write data to the QSPI system bus memory space (0x040 00000–0x0500_0000).

  The address of the first system bus write access is sent in the instruction frame.

  The address of the next system bus write accesses is not used.
- Write LASTXFR bit in CTRLA register to '1'.

- When all valid data blocks in the peripheral have been transferred to the system and no current block transfers are being sent as a result of the Stop At Block Gap Request (STPBGR) of BGCR being set to 1.

A change from 1 to 0 rises the Transfer Complete (TRFC) status flag in NISTR if NISTER.TRFC is set to 1. An interrupt is generated if NISIER.TRFC is set to 1.

Bit 8 – WTACT Write Transfer Active
This bit indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the peripheral. Refer to section “Write Transaction Wait / Continue Timing” in the “SD Host Controller Simplified Specification V3.00” for more details on the sequence of events.

This bit is set to 1 in either of the following conditions:

- After the end bit of the write command.
- When a write operation is restarted by writing a 1 to BGCR.CONTR (Continue Request).

This bit is cleared to 0 in either of the following conditions:

- After getting the CRC status of the last data block as specified by the transfer count (single and multiple). In case of ADMA2, transfer count is designated by the descriptor table.
- After getting the CRC status of any block where a data transmission is about to be stopped by a Stop At Block Gap Request (STPBGR) of BGCR.

During a write transaction and as the result of the Stop At Block Gap Request (STPBGR) being set, a change from 1 to 0 rises the Block Gap Event (BLKGE) status flag in NISTR if NISTER.BLKGE is set to 1. An interrupt is generated if BLKGE is set to 1 in NISIER. This status is useful to determine whether non-DAT line commands can be issued during Write Busy.

Bit 2 – DLACT DAT Line Active
This bit indicates whether one of the DAT lines on the bus is in use.

In the case of read transactions:

This status indicates whether a read transfer is executing on the bus. A change from 1 to 0 resulting from setting the Stop At Block Gap Request (STPBGR) rises the Block Gap Event (BLKGE) status flag in NISTR if NISTER.BLKGE is set to 1. An interrupt is generated if NISIER.BLKGE is set to 1. Refer to section “Read Transaction Wait / Continue Timing” in the “SD Host Controller Simplified Specification V3.00” for details on timing.

This bit is set in either of the following cases:

- After the end bit of the read command.
- When writing 1 to BGCR.CONTR (Continue Request) to restart a read transfer.

This bit is peripheral cleared in either of the following cases:

- When the end bit of the last data block is sent from the bus to the peripheral. In case of ADMA2, the last block is designated by the last transfer of the Descriptor Table.
- When a read transfer is stopped at the block gap initiated by a Stop At Block Gap Request (STPBGR).

The peripheral stops a read operation at the start of the interrupt cycle by driving the Read Wait (DAT[2] line) or by stopping the SD Clock. If the Read Wait signal is already driven (due to the fact that the data buffer cannot receive data), the peripheral can continue to stop the read operation by driving the Read Wait signal. It is necessary to support the Read Wait in order to use the Suspend/Resume operation.

In the case of write transactions:

43.3.4.10.10 Status Returned Values

Table 43-34. Square Service Return Codes

Returned status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly

43.3.4.11 Integral (Euclidean) Division

43.3.4.11.1 Purpose

The purpose of this service is to compute the Euclidean Division of two multiple precision numbers in GF(p) or polynomial in GF(2^n). The Numerator is divided by the Denominator giving the Quotient “Quo” and the Remainder “R”.

The following options are available:

- Work in the GF(2^n) field or in the standard integer arithmetic field GF(p)

43.3.4.11.2 How to Use the Service

43.3.4.11.3 Description

This service processes the calculus corresponding to:

$$Num = Mod \times Quo + R \quad with \quad 0 \leq R < Mod \quad and \quad Quo = \left[\frac{Num}{Mod}\right]$$

The Numerator is Num.

The Divisior (Modulus) is Mod.

The Quotient is Quo.

The Remainder is R.

The Inputs are, the Numerator Num, and the Denominator Mod. The service calculates the Quotient and the Remainder. The Remainder overwrites the Numerator and is copied to the R area.

If the parameter nu1QuoBase equals zero, the Quotient is not stored in memory.

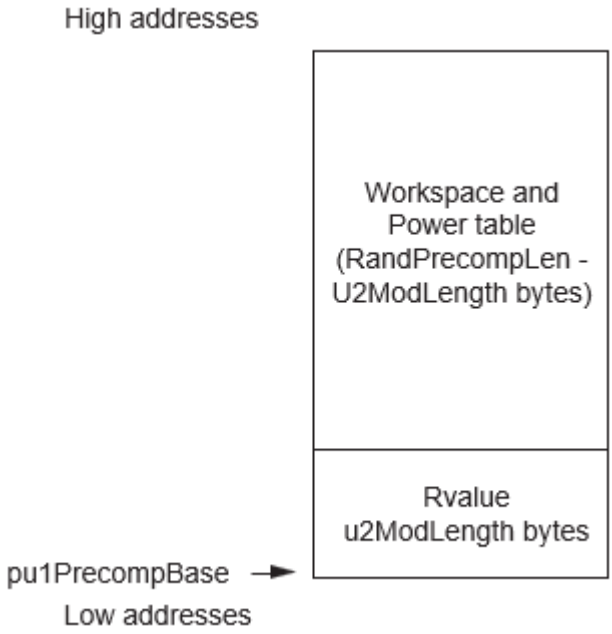
If nu1QuoBase is different from zero, the Quotient length is (<Numerator Length> - <Denominator Length>) + 4 bytes.

In this computation, the following areas need to be provided:

- Num (pointed by {nu1NumBase,u2NumLength}) filled with the Numerator (with MSB word to zero).
- Mod (pointed by {nu1ModBase,u2ModLength}) filled with the Denominator.
- Workspace (pointed by {nu1CnsBase,64 or68}).
- Quo (pointed by {nu1QuoBase,u2NumLength - u2ModLength + 4}) to contain calculated Quotient.
  - When the quotient is not needed, the nu1QuoBase pointer can be provided as NULL. In that case, only the remainder will be provided as a result.
- R (pointed by {nu1RBase,u2ModLength}) to contain the calculated Remainder.

The service name for this operation is `Div`.

Figure 43-5. Value Rval and Precomp in {nu1PrecompBase, RandPrecompLen}



43.3.5.4.9 CRT Service Modular Exponentiation Maximum Size

The following table details the maximum size in bits of P or Q, of N and of EP or EQ.

- The maximum size in bits of P or Q equals:
  <Max Size Bits P> = <Max Size Bits Q> = 8 * <Max u2ModLength bytes>
- The maximum size in bits of N=P*Q equals:
  <Max Size Bits N> = 2 * <Max Size Bits P>
- The maximum size in bits of EP or EQ equals:
  <Max Size Bits EP> = <Max Size Bits EQ> = 8 * <Max u2ExpLength bytes>
- In case of the PUKCL_EXPMOD_EXPINPUKCCRAM option is specified, for the computation of the maximum acceptable size, it is assumed the Exponent is entirely in the Crypto RAM and its length equal the Modulus one.
- Otherwise, the Exponent is entirely out of the Crypto RAM and so the computation do not depend on its length.

Table 43-66. CRT Service Maximum Sizes

Characteristics of the Operation	P or Q Max Bit Sizes	N Max Bit Sizes	EP or EQ Max Bit Sizes
Exponent in Crypto RAM, 1 bit window	2912	5824	2912
Exponent in Crypto RAM, 2 bits window	2688	5376	2688
Exponent in Crypto RAM, 3 bits window	2464	4928	2464
Exponent in Crypto RAM, 4 bits window	2304	4608	2304
Exponent not in Crypto RAM, 1 bit window	3584	7168	<application dependent>
Exponent not in Crypto RAM, 2 bits window	3232	6464	<application dependent>

- Control A (CTRLA), except ENABLE and SWRST bits
- Event Control register (EVCTRL)
- Calibration register (CALIB)

Enable-protection is denoted by the "Enable-Protected" property in the register description.

45.6.2.2 Enabling, Disabling, and Resetting

The ADC is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The ADC is disabled by writing CTRLA.ENABLE=0.

The ADC is reset by writing a '1' to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the ADC, except DBGCTRL, will be reset to their initial state, and the ADC will be disabled. Refer to 45.8.1 CTRLA for details.

45.6.2.3 Operation

In the most basic configuration, the ADC samples values from the configured internal or external sources (INPUTCTRL register). The rate of the conversion depends on the combination of the GCLK_ADCx frequency and the clock prescaler.

To convert analog values to digital values, the ADC needs to be initialized first, as described in the Initialization section. Data conversion can be started either manually by setting the Start bit in the Software Trigger register (SWTRIG.START=1), or automatically by configuring an automatic trigger to initiate the conversions. The ADC starts sampling the input only after the start of conversion is triggered. This means that even after the MUX selection is made, sample and hold (S&H) operation starts only on the conversion trigger. A free-running mode can be used to continuously convert an input channel. When using free-running mode the first conversion must be started, while subsequent conversions will start automatically at the end of previous conversions.

The ADC starts sampling the input only after the start of a conversion is triggered. This means that even after the MUX selection is made, sample and hold operation starts only on the conversion trigger.

The result of the conversion is stored in the Result register (RESULT) overwriting the result from the previous conversion.

To avoid data loss, if more than one channel is enabled, the conversion result must be read as soon as it is available (INTFLAG.RESRDY). Failing to do so will result in an overrun error condition, indicated by the OVERRUN bit in the Interrupt Flag Status and Clear register (INTFLAG.OVERRUN).

To enable one of the available interrupts sources, the corresponding bit in the Interrupt Enable Set register (INTENSET) must be written to '1'.

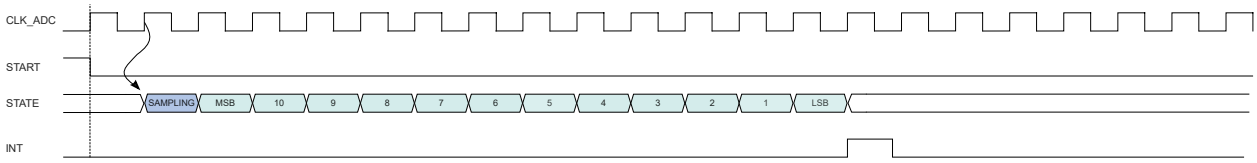
Related Links

45.6.2.1 Initialization

45.6.2.4 Prescaler Selection

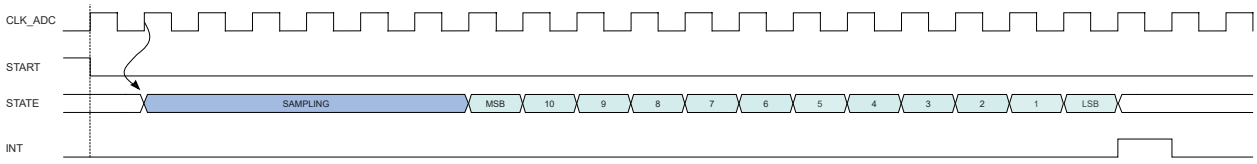
The ADC is clocked by GCLK_ADCx. There is also a prescaler in the ADC to enable conversion at lower clock rates. Refer to CTRLA for details on prescaler settings. Refer to 45.6.2.8 Conversion Timing and Sampling Rate for details on timing and sampling rate.

Figure 45-3. ADC Timing for One Conversion in 12-bit Resolution



The sampling time can be increased by using the Sampling Time Length bit group in the Sampling Time Control register (SAMPCTRL.SAMPLEN). As example, the next figure is showing the timing conversion with sampling time increased to six CLK_ADC cycles.

Figure 45-4. ADC Timing for One Conversion with Increased Sampling Time, 12-bit



The ADC provides also offset compensation, see the following figure. The offset compensation is enabled by the Offset Compensation bit in the Sampling Control register (SAMPCTRL.OFFCOMP).

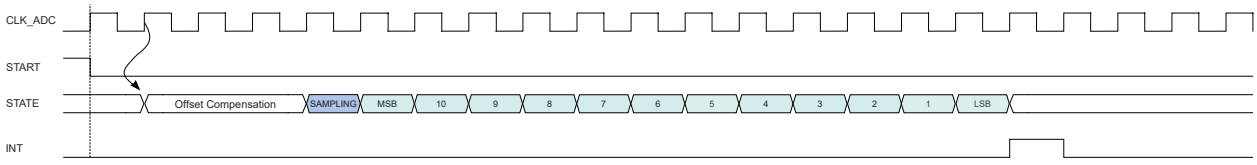
Note: If offset compensation is used, the sampling time must be set to one cycle of CLK_ADCx.

In free running mode, the sampling rate R_S is calculated by

$R_S = f_{CLK\_ADC} / (n_{SAMPLING} + n_{OFFCOMP} + n_{DATA})$

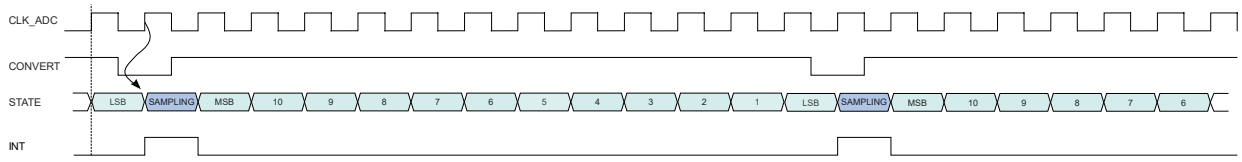
Here, $n_{SAMPLING}$ is the sampling duration in CLK_ADC cycles, $n_{OFFCOMP}$ is the offset compensation duration in clock cycles, and n_{DATA} is the bit resolution. f_{CLK_ADC} is the ADC clock frequency from the internal prescaler: $f_{CLK\_ADC} = f_{GCLK\_ADC} / 2^{(1 + CTRLA.PRESCALER)}$

Figure 45-5. ADC Timing for One Conversion with Offset Compensation, 12-bit



The impact of resolution on the sampling rate is seen in the next two figures, where free-running sampling in 12-bit and 8-bit resolution are compared.

Figure 45-6. ADC Timing for Free Running in 12-bit Resolution



Value	Description
0	The ADC is disabled.
1	The ADC is enabled.

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

Writing a '1' to this bit resets all registers in the ADC, except DBGCTRL, to their initial state, and the ADC will be disabled.

Writing a '1' to CTRL.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded.

Due to synchronization there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the reset is complete.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

52.7 Register Summary

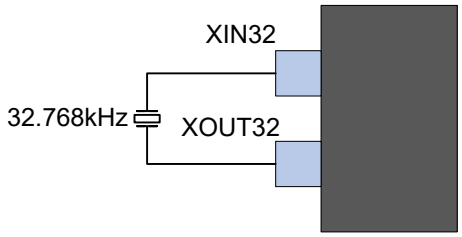
Offset	Name	Bit Pos.								
0x00	MR	7:0			DSIZE[1:0]					PCEN
		15:8					FRSTS	HALFS	ALWYS	SCALE
		23:16						ISIZE[2:0]		
		31:24	CID[1:0]							
0x04	IER	7:0					RXBUF	ENDRX	OVRE	DRDY
		15:8								
		23:16								
		31:24								
0x08	IDR	7:0					RXBUFF	ENDRX	OVRE	DRDY
		15:8								
		23:16								
		31:24								
0x0C	IMR	7:0					RXBUFF	ENDRX	OVRE	DRDY
		15:8								
		23:16								
		31:24								
0x10	ISR	7:0					RXBUFF	ENDRX	OVRE	DRDY
		15:8								
		23:16								
		31:24								
0x14	RHR	7:0	RDATA[7:0]							
		15:8	RDATA[15:8]							
		23:16	RDATA[23:16]							
		31:24	RDATA[31:24]							
0x18 ... 0xDF	Reserved									
0xE0	WPMR	7:0								WPEN
		15:8	WPKEY[7:0]							
		23:16	WPKEY[15:8]							
		31:24	WPKEY[23:16]							
0xE4	WPSR	7:0								WPVS
		15:8	WPVSRC[7:0]							
		23:16	WPVSRC[15:8]							
		31:24								

52.8 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.

The typical parasitic load capacitance values are available in the Electrical Characteristics section. This capacitance and PCB capacitance can allow using a crystal inferior to 12.5pF load capacitance without external capacitors as shown in the next figure.

Figure 56-10. External Real Time Oscillator without Load Capacitor



To improve accuracy and Safety Factor, the crystal datasheet can recommend adding external capacitors as shown the figure below.

To find suitable load capacitance for a 32.768kHz crystal, consult the crystal datasheet.

Figure 56-11. External Real Time Oscillator with Load Capacitor

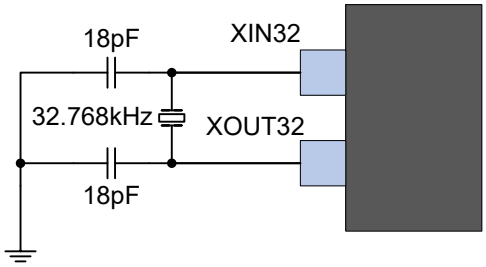


Table 56-6. External Real Time Oscillator Checklist

Signal Name	Recommended Pin Connection	Description
XIN32	Load capacitor 18pF(1)(2)	Timer oscillator input
XOUT32	Load capacitor 18pF(1)(2)	Timer oscillator output

1. These values are only given as typical examples.

2. The capacitors should be placed close to the device for each supply pin pair in the signal group.

Note: In order to minimize the cycle-to-cycle jitter of the external oscillator, keep the neighboring pins as steady as possible. For neighboring pin details, refer to the Oscillator Pinout section.

56.6.4 Calculating the Correct Crystal Decoupling Capacitor

The model shown in Figure 56-12 can be used to calculate correct load capacitor for a given crystal. This model includes internal capacitors C_{Ln}, external parasitic capacitance C_{ELn} and external load capacitance C_{Pn}.