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®-M4
Core Size	32-Bit Single-Core
Speed	120MHz
Connectivity	I²C, IrDA, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	47
Program Memory Size	1MB (1M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	160K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 11x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	64-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4sd16ba-au

Table 11-1. Peripheral Identifiers (Continued)

Instance ID	Instance Name	NVIC Interrupt	PMC Clock Control	Instance Description
29	ADC	X	X	Analog-to-Digital Converter
30	DACC	X	X	Digital-to-Analog Converter Controller
31	PWM	X	X	Pulse Width Modulation
32	CRCCU	X	X	CRC Calculation Unit
33	ACC	X	X	Analog Comparator Controller
34	UDP	X	X	USB Device Port

11.2 Peripheral Signal Multiplexing on I/O Lines

The SAM4S features two PIO controllers on 64-pin versions (PIOA and PIOB) or three PIO controllers on the 100-pin version (PIOA, PIOB and PIOC), that multiplex the I/O lines of the peripheral set.

The SAM4S 64-pin and 100-pin PIO controllers control up to 32 lines. Each line can be assigned to one of three peripheral functions: A, B or C. The multiplexing tables in the following tables define how the I/O lines of the peripherals A, B and C are multiplexed on the PIO Controllers. The column “Comments” has been inserted in this table for the user’s own comments; it may be used to track how pins are defined in an application.

Note that some peripheral functions which are output only, might be duplicated within the tables.

12.6.7.2 SSAT16 and USAT16

Signed Saturate and Unsigned Saturate to any bit position for two halfwords.

Syntax

```
op{cond} Rd, #n, Rm
```

where:

op	is one of:
	SSAT16 Saturates a signed halfword value to a signed range.
	USAT16 Saturates a signed halfword value to an unsigned range.
cond	is an optional condition code, see “Conditional Execution” .
Rd	is the destination register.
n	specifies the bit position to saturate to:
n ranges from 1	n ranges from 0 to 15 for USAT.
to 16 for SSAT	
Rm	is the register containing the value to saturate.

Operation

The SSAT16 instruction:

Saturates two signed 16-bit halfword values of the register with the value to saturate from selected by the bit position in *n*.

Writes the results as two signed 16-bit halfwords to the destination register.

The USAT16 instruction:

Saturates two unsigned 16-bit halfword values of the register with the value to saturate from selected by the bit position in *n*.

Writes the results as two unsigned halfwords in the destination register.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the condition code flags.

If saturation occurs, these instructions set the Q flag to 1.

Examples

```
SSAT16    R7, #9, R2   ; Saturates the top and bottom highwords of R2
                       ; as 9-bit values, writes to corresponding halfword
                       ; of R7
USAT16NE  R0, #13, R5  ; Conditionally saturates the top and bottom
                       ; halfwords of R5 as 13-bit values, writes to
                       ; corresponding halfword of R0.
```

12.6.7.3 QADD and QSUB

Saturating Add and Saturating Subtract, signed.

Syntax

```
op{cond} {Rd}, Rn, Rm
op{cond} {Rd}, Rn, Rm
```

where:

- **UNSTKERR: Bus Fault on Unstacking for a Return From Exception**

This is part of “BFSR: Bus Fault Status Subregister” .

0: No unstacking fault.

1: Unstack for an exception return has caused one or more bus faults.

This fault is chained to the handler. This means that when the processor sets this bit to 1, the original return stack is still present. The processor does not adjust the SP from the failing return, does not performed a new save, and does not write a fault address to the BFAR.

- **STKERR: Bus Fault on Stacking for Exception Entry**

This is part of “BFSR: Bus Fault Status Subregister” .

0: No stacking fault.

1: Stacking for an exception entry has caused one or more bus faults.

When the processor sets this bit to 1, the SP is still adjusted but the values in the context area on the stack might be incorrect. The processor does not write a fault address to the SCB_BFAR.

- **BFARVALID: Bus Fault Address Register (BFAR) Valid flag**

This is part of “BFSR: Bus Fault Status Subregister” .

0: The value in SCB_BFAR is not a valid fault address.

1: SCB_BFAR holds a valid fault address.

The processor sets this bit to 1 after a bus fault where the address is known. Other faults can set this bit to 0, such as a memory management fault occurring later.

If a bus fault occurs and is escalated to a hard fault because of priority, the hard fault handler must set this bit to 0. This prevents problems if returning to a stacked active bus fault handler whose SCB_BFAR value has been overwritten.

- **UNDEFINSTR: Undefined Instruction Usage Fault**

This is part of “UFSR: Usage Fault Status Subregister” .

0: No undefined instruction usage fault.

1: The processor has attempted to execute an undefined instruction.

When this bit is set to 1, the PC value stacked for the exception return points to the undefined instruction.

An undefined instruction is an instruction that the processor cannot decode.

- **INVSTATE: Invalid State Usage Fault**

This is part of “UFSR: Usage Fault Status Subregister” .

0: No invalid state usage fault.

1: The processor has attempted to execute an instruction that makes illegal use of the EPSR.

When this bit is set to 1, the PC value stacked for the exception return points to the instruction that attempted the illegal use of the EPSR.

This bit is not set to 1 if an undefined instruction uses the EPSR.

- **INVPC: Invalid PC Load Usage Fault**

This is part of “UFSR: Usage Fault Status Subregister” . It is caused by an invalid PC load by EXC_RETURN:

0: No invalid PC load usage fault.

1: The processor has attempted an illegal load of EXC_RETURN to the PC, as a result of an invalid context, or an invalid EXC_RETURN value.

- **FARG: Flash Command Argument**

GETD, GLB, GGPB, STUI, SPUI, GCALB, WUS, EUS, STUS, SPUS, EA	Commands requiring no argument, including Erase all command	FARG is meaningless, must be written with 0
ES	Erase sector command	FARG must be written with any page number within the sector to be erased
EPA	Erase pages command	FARG[1:0] defines the number of pages to be erased The start page must be written in FARG[15:2]. FARG[1:0] = 0: Four pages to be erased. FARG[15:2] = Page_Number / 4 FARG[1:0] = 1: Eight pages to be erased. FARG[15:3] = Page_Number / 8, FARG[2]=0 FARG[1:0] = 2: Sixteen pages to be erased. FARG[15:4] = Page_Number / 16, FARG[3:2]=0 FARG[1:0] = 3: Thirty-two pages to be erased. FARG[15:5] = Page_Number / 32, FARG[4:2]=0 Refer to Table 20-4 “EEFC_FCR.FARG Field for EPA Command”.
WP, WPL, EWP, EWPL	Programming commands	FARG must be written with the page number to be programmed
SLB, CLB	Lock bit commands	FARG defines the page number to be locked or unlocked
SGPB, CGPB	GPNVM commands	FARG defines the GPNVM number to be programmed

- **FKEY: Flash Writing Protection Key**

Value	Name	Description
0x5A	PASSWD	The 0x5A value enables the command defined by the bits of the register. If the field is written with a different value, the write is not performed and no action is started.

Table 21-7. Write Command (Continued)

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
n+1	Write handshaking	ADDR1	Memory Address
n+2	Write handshaking	DATA	*Memory Address++
n+3	Write handshaking	DATA	*Memory Address++
...	...	...	...

The Flash command **Write Page and Lock (WPL)** is equivalent to the Flash Write Command. However, the lock bit is automatically set at the end of the Flash write operation. As a lock region is composed of several pages, the programmer writes to the first pages of the lock region using Flash write commands and writes to the last page of the lock region using a Flash write and lock command.

The Flash command **Erase Page and Write (EWP)** is equivalent to the Flash Write Command. However, before programming the load buffer, the page is erased.

The Flash command **Erase Page and Write the Lock (EWPL)** combines EWP and WPL commands.

21.3.5.3 Flash Full Erase Command

This command is used to erase the Flash memory planes.

All lock regions must be unlocked before the Full Erase command by using the CLB command. Otherwise, the erase command is aborted and no page is erased.

Table 21-8. Full Erase Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	EA
2	Write handshaking	DATA	0

21.3.5.4 Flash Lock Commands

Lock bits can be set using WPL or EWPL commands. They can also be set by using the **Set Lock** command **(SLB)**. With this command, several lock bits can be activated. A Bit Mask is provided as argument to the command. When bit 0 of the bit mask is set, then the first lock bit is activated.

In the same way, the **Clear Lock** command **(CLB)** is used to clear lock bits.

Table 21-9. Set and Clear Lock Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	SLB or CLB
2	Write handshaking	DATA	Bit Mask

Lock bits can be read using **Get Lock Bit** command **(GLB)**. The n^{th} lock bit is active when the bit n of the bit mask is set.

Table 21-10. Get Lock Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	GLB
2	Read handshaking	DATA	Lock Bit Mask Status 0 = Lock bit is cleared 1 = Lock bit is set

21.3.5.5 Flash General-purpose NVM Commands

General-purpose NVM bits (GP NVM bits) can be set using the **Set GPNVM** command **(SGPB)**. This command also activates GP NVM bits. A bit mask is provided as argument to the command. When bit 0 of the bit mask is set, then the first GP NVM bit is activated.

In the same way, the **Clear GPNVM** command **(CGPB)** is used to clear general-purpose NVM bits. The general-purpose NVM bit is deactivated when the corresponding bit in the pattern value is set to 1.

Table 21-11. Set/Clear GP NVM Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	SGPB or CGPB
2	Write handshaking	DATA	GP NVM bit pattern value

General-purpose NVM bits can be read using the **Get GPNVM Bit** command **(GGPB)**. The n^{th} GP NVM bit is active when bit n of the bit mask is set.

Table 21-12. Get GP NVM Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	GGPB
2	Read handshaking	DATA	GP NVM Bit Mask Status 0 = GP NVM bit is cleared 1 = GP NVM bit is set

21.3.5.6 Flash Security Bit Command

A security bit can be set using the **Set Security Bit** command (SSE). Once the security bit is active, the Fast Flash programming is disabled. No other command can be run. An event on the Erase pin can erase the security bit once the contents of the Flash have been erased.

Table 21-13. Set Security Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	SSE
2	Write handshaking	DATA	0

Once the security bit is set, it is not possible to access FFPI. The only way to erase the security bit is to erase the Flash.

In order to erase the Flash, the user must perform the following:

1. Power-off the chip.
2. Power-on the chip with TST = 0.
3. Assert Erase during a period of more than 220 ms.
4. Power-off the chip.

Then it is possible to return to FFPI mode and check that Flash is erased.

21.3.5.7 Memory Write Command

This command is used to perform a write access to any memory location.

4. Wait for the MOSCXTS flag to be 1 in PMC_SR to get the end of a start-up period of the fast crystal oscillator.
5. Then, MOSCSEL must be programmed to 1 in CKGR_MOR to select fast main crystal oscillator for the main clock.
6. MOSCSEL must be read until its value equals 1.
7. Then the MOSCSELS status flag must be checked in PMC_SR.

At this point, two cases may occur (either MOSCSELS = 0 or MOSCSELS = 1).

- If MOSCSELS = 1: There is a valid crystal connected and its frequency can be determined by initiating a frequency measure by programming RCMEAS in CKGR_MCFR.
- If MOSCSELS = 0:
  - There is no fast crystal clock (either no crystal connected or a crystal clock out of specification). A frequency measure can reinforce this status by initiating a frequency measure by programming RCMEAS in CKGR_MCFR.
  - If MOSCSELS = 0, the selection of the main clock must be programmed back to the main RC oscillator by writing MOSCSEL to 0 prior to disabling the fast crystal oscillator.
  - If MOSCSELS = 0, the crystal oscillator can be disabled (MOSCXTEN = 0 in CKGR_MOR).

28.5.8 Main Clock Frequency Counter

The device features a main clock frequency counter that provides the frequency of the main clock.

The main clock frequency counter is reset and starts incrementing at the main clock speed after the next rising edge of the slow clock in the following cases:

- When the 4/8/12 MHz fast RC oscillator clock is selected as the source of main clock and when this oscillator becomes stable (i.e., when the MOSCRCS bit is set)
- When the 3 to 20 MHz crystal or ceramic resonator-based oscillator is selected as the source of main clock and when this oscillator becomes stable (i.e., when the MOSCXTS bit is set)
- When the main clock oscillator selection is modified
- When the RCMEAS bit of CKGR_MFCR is written to 1.

Then, at the 16th falling edge of slow clock, the MAINFRDY bit in CKGR_MCFR) is set and the counter stops counting. Its value can be read in the MAINF field of CKGR_MCFR and gives the number of main clock cycles during 16 periods of slow clock, so that the frequency of the 4/8/12 MHz fast RC oscillator or 3 to 20 MHz crystal or ceramic resonator-based oscillator can be determined.

31.6.29 PIO Slow Clock Divider Debouncing Register

Name: PIO_SCDR

Address: 0x400E0E8C (PIOA), 0x400E108C (PIOB), 0x400E128C (PIOC)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	DIV					

7	6	5	4	3	2	1	0
DIV							

- **DIV: Slow Clock Divider Selection for Debouncing**

$t_{div\_slck} = ((DIV + 1) \times 2) \times t_{slck}$

31.6.39 PIO Edge Select Register

Name: PIO_ESR

Address: 0x400E0EC0 (PIOA), 0x400E10C0 (PIOB), 0x400E12C0 (PIOC)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Edge Interrupt Selection**

0: No effect.

1: The interrupt source is an edge-detection event.

34.7.4 Multi-master Mode

34.7.4.1 Definition

In Multi-master mode, more than one master may handle the bus at the same time without data corruption by using arbitration.

Arbitration starts as soon as two or more masters place information on the bus at the same time, and stops (arbitration is lost) for the master that intends to send a logical one while the other master sends a logical zero.

As soon as a master lose arbitration, it stops sending data and listens to the bus in order to detect a stop. When the stop is detected, the master may put its data on the bus by performing arbitration.

Arbitration is illustrated in Figure 34-21.

34.7.4.2 Two Multi-master Modes

Two Multi-master modes may be distinguished:

1. TWI is considered as a master only and will never be addressed.
2. TWI may be either a master or a slave and may be addressed.

Note: Arbitration is supported in both Multi-master modes.

TWI as Master Only

In this mode, TWI is considered as a Master only (MSEN is always one) and must be driven like a Master with the ARBLST (Arbitration Lost) flag in addition.

If arbitration is lost (ARBLST = 1), the user must reinitiate the data transfer.

If the user starts a transfer (ex.: DADR + START + W + Write in THR) and if the bus is busy, the TWI automatically waits for a STOP condition on the bus to initiate the transfer (see Figure 34-20).

Note: The state of the bus (busy or free) is not shown in the user interface.

TWI as Master or Slave

The automatic reversal from Master to Slave is not supported in case of a lost arbitration.

Then, in the case where TWI may be either a Master or a Slave, the user must manage the pseudo Multi-master mode described in the steps below.

1. Program TWI in Slave mode (SADR + MSDIS + SVEN) and perform a slave access (if TWI is addressed).
2. If the TWI has to be set in Master mode, wait until the TXCOMP flag is at 1.
3. Program the Master mode (DADR + SVDIS + MSEN) and start the transfer (ex: START + Write in THR).
4. As soon as the Master mode is enabled, the TWI scans the bus in order to detect if it is busy or free. When the bus is considered free, TWI initiates the transfer.
5. As soon as the transfer is initiated and until a STOP condition is sent, the arbitration becomes relevant and the user must monitor the ARBLST flag.
6. If the arbitration is lost (ARBLST is set to 1), the user must program the TWI in Slave mode in case the Master that won the arbitration is required to access the TWI.
7. If the TWI has to be set in Slave mode, wait until TXCOMP flag is at 1 and then program the Slave mode.

Note: If the arbitration is lost and the TWI is addressed, the TWI will not acknowledge even if it is programmed in Slave mode as soon as ARBLST is set to 1. Then the Master must repeat SADR.

36.7.3 USART Mode Register

Name: US_MR

Address: 0x40024004 (0), 0x40028004 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
ONEBIT	MODSYNC	MAN	FILTER	–	MAX_ITERATION		

23	22	21	20	19	18	17	16
INVDATA	VAR_SYNC	DSNACK	INACK	OVER	CLKO	MODE9	MSBF

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC

7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

For SPI configuration, see Section 36.7.4 ”USART Mode Register (SPI_MODE)”.

- **USART_MODE: USART Mode of Operation**

Value	Name	Description
0x0	NORMAL	Normal mode
0x1	RS485	RS485
0x2	HW_HANDSHAKING	Hardware Handshaking
0x3	MODEM	Modem
0x4	IS07816_T_0	IS07816 Protocol: T = 0
0x6	IS07816_T_1	IS07816 Protocol: T = 1
0x8	IRDA	IrDA
0xE	SPI_MASTER	SPI master
0xF	SPI_SLAVE	SPI Slave

The PDC transfers are supported in all USART modes of operation.

- **USCLKS: Clock Selection**

Value	Name	Description
0	MCK	Peripheral clock is selected
1	DIV	Peripheral clock divided (DIV=8) is selected
2	—	Reserved
3	SCK	Serial clock (SCK) is selected

36.7.21 USART Manchester Configuration Register

Name: US_MAN

Address: 0x40024050 (0), 0x40028050 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	DRIFT	ONE	RX_MPOL	–	–	RX_PP	

23	22	21	20	19	18	17	16
–	–	–	–	RX_PL			

15	14	13	12	11	10	9	8
–	–	–	TX_MPOL	–	–	TX_PP	

7	6	5	4	3	2	1	0
–	–	–	–	TX_PL			

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **TX_PL: Transmitter Preamble Length**

0: The transmitter preamble pattern generation is disabled

1–15: The preamble length is TX_PL × Bit Period

- **TX_PP: Transmitter Preamble Pattern**

The following values assume that TX_MPOL field is not set:

Value	Name	Description
0	ALL_ONE	The preamble is composed of ‘1’s
1	ALL_ZERO	The preamble is composed of ‘0’s
2	ZERO_ONE	The preamble is composed of ‘01’s
3	ONE_ZERO	The preamble is composed of ‘10’s

- **TX_MPOL: Transmitter Manchester Polarity**

0: Logic zero is coded as a zero-to-one transition, Logic one is coded as a one-to-zero transition.

1: Logic zero is coded as a one-to-zero transition, Logic one is coded as a zero-to-one transition.

- **RX_PL: Receiver Preamble Length**

0: The receiver preamble pattern detection is disabled

1–15: The detected preamble length is RX_PL × Bit Period

37.7.6 TC Register A

Name: TC_RAx [x=0..2]

Address: 0x40010014 (0)[0], 0x40010054 (0)[1], 0x40010094 (0)[2], 0x40014014 (1)[0], 0x40014054 (1)[1], 0x40014094 (1)[2]

Access: Read-only if TC_CMRx.WAVE = 0, Read/Write if TC_CMRx.WAVE = 1

31	30	29	28	27	26	25	24
RA							

23	22	21	20	19	18	17	16
RA							

15	14	13	12	11	10	9	8
RA							

7	6	5	4	3	2	1	0
RA							

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **RA: Register A**

RA contains the Register A value in real time.

IMPORTANT: For 16-bit channels, RA field size is limited to register bits 15:0.

37.7.20 TC Write Protection Mode Register

Name: TC_WPMR

Address: 0x400100E4 (0), 0x400140E4 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x54494D (“TIM” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x54494D (“TIM” in ASCII).

The Timer Counter clock of the first channel must be enabled to access this register.

See Section 37.6.17 “Register Write Protection” for a list of registers that can be write-protected and Timer Counter clock conditions.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x54494D	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

39.7.42 PWM Channel Dead Time Register

Name: PWM_DTx [x=0..3]

Address: 0x40020218 [0], 0x40020238 [1], 0x40020258 [2], 0x40020278 [3]

Access: Read/Write

31	30	29	28	27	26	25	24
DTL							

23	22	21	20	19	18	17	16
DTL							

15	14	13	12	11	10	9	8
DTH							

7	6	5	4	3	2	1	0
DTH							

This register can only be written if bits WPSWS4 and WPHWS4 are cleared in the PWM Write Protection Status Register.

Only the first 12 bits (dead-time counter size) of fields DTH and DTL are significant.

- **DTH: Dead-Time Value for PWMHx Output**

Defines the dead-time value for PWMHx output. This value must be defined between 0 and the value (CPRD – CDTY) (PWM_CPRDx and PWM_CDTYx).

- **DTL: Dead-Time Value for PWMLx Output**

Defines the dead-time value for PWMLx output. This value must be defined between 0 and CDTY (PWM_CDTYx).

40.7.7 UDP Interrupt Status Register

Name: UDP_ISR

Address: 0x4003401C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBUSRES	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
EP7INT	EP6INT	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Endpoint 0 Interrupt Status**
- **EP1INT: Endpoint 1 Interrupt Status**
- **EP2INT: Endpoint 2 Interrupt Status**
- **EP3INT: Endpoint 3 Interrupt Status**
- **EP4INT: Endpoint 4 Interrupt Status**
- **EP5INT: Endpoint 5 Interrupt Status**
- **EP6INT: Endpoint 6 Interrupt Status**
- **EP7INT: Endpoint 7Interrupt Status**

0: No Endpoint0 Interrupt pending

1: Endpoint0 Interrupt has been raised

Several signals can generate this interrupt. The reason can be found by reading UDP_CSR0:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP0INT is a sticky bit. Interrupt remains valid until EP0INT is cleared by writing in the corresponding UDP_CSR0 bit.

- **RXSUSP: UDP Suspend Interrupt Status**

0: No UDP Suspend Interrupt pending

1: UDP Suspend Interrupt has been raised

The USB device sets this bit when it detects no activity for 3 ms. The USB device enters Suspend mode.

40.7.9 UDP Reset Endpoint Register

Name: UDP_RST_EP

Address: 0x40034028

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0

- **EP0: Reset Endpoint 0**
- **EP1: Reset Endpoint 1**
- **EP2: Reset Endpoint 2**
- **EP3: Reset Endpoint 3**
- **EP4: Reset Endpoint 4**
- **EP5: Reset Endpoint 5**
- **EP6: Reset Endpoint 6**
- **EP7: Reset Endpoint 7**

This flag is used to reset the FIFO associated with the endpoint and the bit RXBYTECOUNT in the UDP_CSRx. It also resets the data toggle to DATA0. It is useful after removing a HALT condition on a BULK endpoint. Refer to Chapter 5.8.5 in the *USB Serial Bus Specification, Rev.2.0*.

Warning: This flag must be cleared at the end of the reset. It does not clear UDP_CSRx flags.

0: No reset

1: Forces the corresponding endpoint FIF0 pointers to 0, therefore RXBYTECNT field is read at 0 in UDP_CSRx

Resetting the endpoint is a two-step operation:

1. Set the corresponding EPx field.
2. Clear the corresponding EPx field.

Table 44-24. SAM4SD32/SA16/SD16 Typical Active Power Consumption with VDDCORE@ 1.2V running from Flash Memory (AMP2) or SRAM

	CoreMark					
	Cache Enable (CE)		Cache Disable (CD)			
Core Clock (MHz)	128-bit Flash access(1)	64-bit Flash access(1)	128-bit Flash access(1)	64-bit Flash access(1)	SRAM	Unit
120	23.2	23.2	27.8	20.9	22.1	mA
100	19.6	19.6	25.3	19.0	18.5	
84	16.6	16.5	21.6	16.2	15.7	
64	12.8	12.8	18.0	13.7	12.1	
48	9.7	9.7	14.9	11.9	9.2	
32	6.7	6.7	11.2	9.5	6.3	
24	5.2	5.2	9.5	8.4	4.9	
12	2.5	2.5	5.4	4.6	2.4	
8	1.8	1.8	4.5	3.9	1.7	
4	1.1	1.1	2.8	2.8	1.0	
2	0.7	0.7	2.0	2.0	0.7	
1	0.5	0.5	1.2	1.2	0.5	
0.5	0.4	0.4	0.8	0.8	0.4	

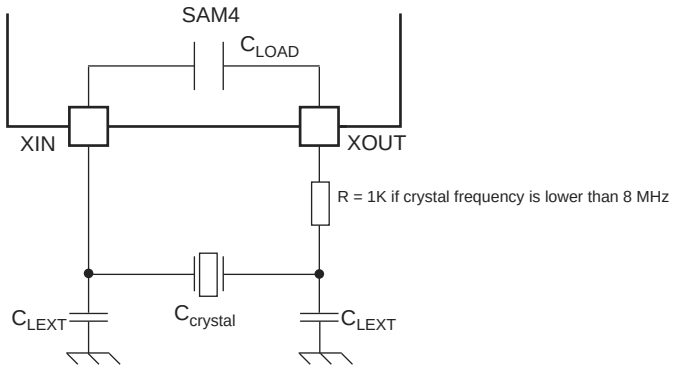
Note: 1. Flash Wait State (FWS) in EEFC_FMR adjusted versus core frequency

44.5.6 3 to 20 MHz Crystal Oscillator Characteristics

Table 44-31. 3 to 20 MHz Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OSC}	Operating Frequency	Normal mode with crystal	3	16	20	MHz
	Duty Cycle		40	50	60	%
t_{START}	Startup Time	3 MHz, C_{SHUNT} = 3 pF 8 MHz, C_{SHUNT} = 7 pF 16 MHz, C_{SHUNT} = 7 pF with C_m = 8 fF 16 MHz, C_{SHUNT} = 7 pF with C_m = 1.6 fF 20 MHz, C_{SHUNT} = 7 pF	–	–	14.5 4 1.4 2.5 1	ms
I_{DDON}	Current Consumption (on VDDIO)	3 MHz 8 MHz 16 MHz 20 MHz	–	230 300 390 450	350 400 470 560	µA
P_{ON}	Drive Level	3 MHz 8 MHz 16 MHz, 20 MHz	–	–	15 30 50	µW
R_f	Internal Resistor	Between XIN and XOUT	–	0.5	–	MΩ
C_{LEXT}	Maximum External Capacitor on XIN and XOUT		–	–	17	pF
$C_{crystal}$	Allowed Crystal Capacitance Load	From crystal specification	12.5	–	17.5	pF
C_{LOAD}	Internal Equivalent Load Capacitance	Integrated load capacitance (XIN and XOUT in series)	7.5	9.5	10.5	pF

Figure 44-15. 3 to 20 MHz Crystal Oscillator Schematic



$C_{LEXT} = 2 \times (C_{crystal} - C_{LOAD} - C_{PCB})$.

Where C_{PCB} is the capacitance of the printed circuit board (PCB) track layout from the crystal to the SAM4 pin.

Table 47-1. Ordering Codes for SAM4S Devices (Continued)

Ordering Code	MRL	Flash (Kbytes)	SRAM (Kbytes)	Package	Carrier Type	Operating Temperature Range
ATSAM4S4AA-AN	A	256	64	LQFP48	Tray	Industrial (-40°C to +105°C)
ATSAM4S4AB-AN	B					
ATSAM4S2CA-CU	A	128	64	TFBGA100	Tray	Industrial (-40°C to +85°C)
ATSAM4S2CB-CU	B					
ATSAM4S2CA-CFU	A	128	64	VFBGA100	Tray	Industrial (-40°C to +85°C)
ATSAM4S2CB-CFU	B					
ATSAM4S2CA-AU	A	128	64	LQFP100	Tray	Industrial (-40°C to +85°C)
ATSAM4S2CB-AU	B					
ATSAM4S2CA-AN	A	128	64	LQFP100	Tray	Industrial (-40°C to +105°C)
ATSAM4S2CB-AN	B					
ATSAM4S2BA-MU	A	128	64	QFN64	Tray	Industrial (-40°C to +85°C)
ATSAM4S2BB-MU	B					
ATSAM4S2BA-AU	A	128	64	LQFP64	Tray	Industrial (-40°C to +85°C)
ATSAM4S2BB-AU	B					
ATSAM4S2BA-UUR	A	128	64	WLCSP64	Reel	Industrial (-40°C to +85°C)
ATSAM4S2BB-UUR	B					
ATSAM4S2BA-AN	A	128	64	LQFP64	Tray	Industrial (-40°C to +105°C)
ATSAM4S2BB-AN	B					
ATSAM4S2AA-MU	A	128	64	QFN48	Tray	Industrial (-40°C to +85°C)
ATSAM4S2AB-MU	B					
ATSAM4S2AA-AU	A	128	64	LQFP48	Tray	Industrial (-40°C to +85°C)
ATSAM4S2AB-AU	B					
ATSAM4S2AA-AN	A	128	64	LQFP48	Tray	Industrial (-40°C to +105°C)
ATSAM4S2AB-AN	B					