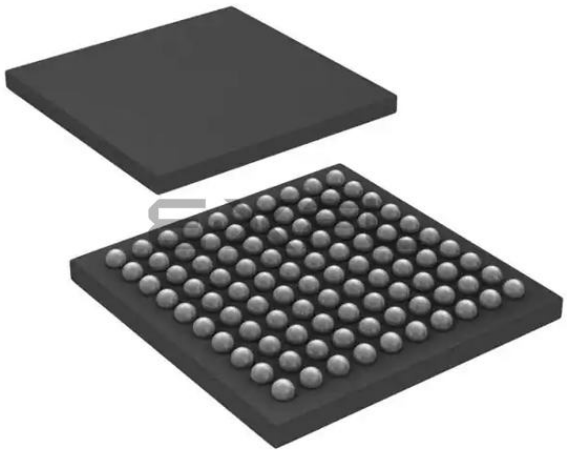


Welcome to E-XFL.COM

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	EBI/EMI, I²C, IrDA, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	79
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	64K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 16x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	100-VFBGA
Supplier Device Package	100-VFBGA (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4s2cb-cfnr

4.2.5 64-ball WLCSP Pinout

Table 4-5. SAM4SD32/S32/SD16/S16/S8 64-ball WLCSP Pinout

A1	PA31	C1	GND	E1	PA29	G1	PA5
A2	PB7	C2	PA1	E2	TST	G2	PA6
A3	VDDCORE	C3	PA0	E3	NRST	G3	PA9
A4	PB10	C4	PB12	E4	PA28	G4	PA11
A5	VDDIO	C5	ADVREF	E5	PA25	G5	VDDCORE
A6	GND	C6	PB3	E6	PA23	G6	PA14
A7	PB9	C7	PB1	E7	PA18	G7	PA20
A8	PB14	C8	PB0	E8	VDDIN	G8	PA19
B1	PB5	D1	VDDIO	F1	PA27	H1	PA7
B2	JTAGSEL	D2	PA3	F2	VDDCORE	H2	PA8
B3	PB6	D3	PA30	F3	PA4	H3	PA10
B4	PB11	D4	PA2	F4	PB4	H4	PA12
B5	PB13	D5	PA13	F5	PA26	H5	PA24
B6	VDDPLL	D6	PA21	F6	PA16	H6	PA15
B7	PB8	D7	PA17	F7	PA22	H7	VDDIO
B8	GND	D8	PB2	F8	VDDOUT	H8	GND

Table 4-6. SAM4S4/S2 64-ball WLCSP Pinout

A1	PB5	C1	GND	E1	PA3	G1	VDDCORE
A2	PA31	C2	PA0	E2	PA30	G2	PA4
A3	VDDCORE	C3	PB7	E3	PA29	G3	PA9
A4	VDDIO	C4	PB12	E4	PA27	G4	PA11
A5	GND	C5	PA10	E5	PA24	G5	PA25
A6	PB8	C6	PB0	E6	PA18	G6	PA14
A7	PB9	C7	PB2	E7	PA17	G7	VDDIO
A8	ADVREF	C8	PB1	E8	VDDIN	G8	PA19
B1	PA1	D1	VDDIO	F1	TST	H1	PB4
B2	JTAGSEL	D2	PA2	F2	NRST	H2	PA7
B3	PB10	D3	PA28	F3	PA5	H3	PA8
B4	PB11	D4	PB6	F4	PA6	H4	PA12
B5	PB13	D5	PA26	F5	PA13	H5	VDDCORE
B6	VDDPLL	D6	PA23	F6	PA22	H6	PA15
B7	PB14	D7	PA16	F7	PA21	H7	GND
B8	GNDANA	D8	PB3	F8	VDDOUT	H8	PA20

12.6.5.10 SHADD16 and SHADD8

Signed Halving Add 16 and Signed Halving Add 8

Syntax

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

where:

op is any of:

SHADD16 Signed Halving Add 16.

SHADD8 Signed Halving Add 8.

cond is an optional condition code, see “Conditional Execution” .

Rd is the destination register.

Rn is the first operand register.

Rm is the second operand register.

Operation

Use these instructions to add 16-bit and 8-bit data and then to halve the result before writing the result to the destination register:

The SHADD16 instruction:

1. Adds each halfword from the first operand to the corresponding halfword of the second operand.
2. Shuffles the result by one bit to the right, halving the data.
3. Writes the halfword results in the destination register.

The SHADDB8 instruction:

1. Adds each byte of the first operand to the corresponding byte of the second operand.
2. Shuffles the result by one bit to the right, halving the data.
3. Writes the byte results in the destination register.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not change the flags.

Examples

```
SHADD16 R1, R0     ; Adds halfwords in R0 to corresponding halfword of R1
                   ; and writes halved result to corresponding halfword in
                   ; R1
SHADD8  R4, R0, R5 ; Adds bytes of R0 to corresponding byte in R5 and
                   ; writes halved result to corresponding byte in R4.
```

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:

- **MSTKERR: Memory Manager Fault on Stacking for Exception Entry**

This is part of “MMFSR: Memory Management Fault Status Subregister” .

0: No stacking fault.

1: Stacking for an exception entry has caused one or more access violations.

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

- **MMARVALID: Memory Management Fault Address Register (SCB_MMFAR) Valid Flag**

This is part of “MMFSR: Memory Management Fault Status Subregister” .

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

1: SCB_MMFAR holds a valid fault address.

If a memory management 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 on return to a stacked active memory management fault handler whose SCB_MMFAR value has been overwritten.

- **IBUSERR: Instruction Bus Error**

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

0: No instruction bus error.

1: Instruction bus error.

The processor detects the instruction bus error on prefetching an instruction, but it sets the IBUSERR flag to 1 only if it attempts to issue the faulting instruction.

When the processor sets this bit to 1, it does not write a fault address to the BFAR.

- **PRECISERR: Precise Data Bus Error**

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

0: No precise data bus error.

1: A data bus error has occurred, and the PC value stacked for the exception return points to the instruction that caused the fault.

When the processor sets this bit to 1, it writes the faulting address to the SCB_BFAR.

- **IMPRECISERR: Imprecise Data Bus Error**

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

0: No imprecise data bus error.

1: A data bus error has occurred, but the return address in the stack frame is not related to the instruction that caused the error.

When the processor sets this bit to 1, it does not write a fault address to the SCB_BFAR.

This is an asynchronous fault. Therefore, if it is detected when the priority of the current process is higher than the bus fault priority, the bus fault becomes pending and becomes active only when the processor returns from all higher priority processes. If a precise fault occurs before the processor enters the handler for the imprecise bus fault, the handler detects that both this bit and one of the precise fault status bits are set to 1.

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**

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

32.8.8 Loop Mode

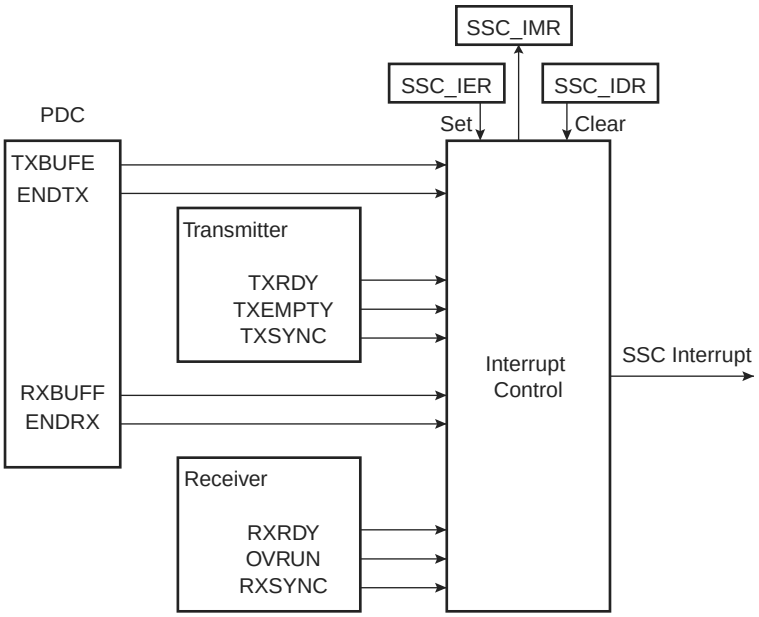
The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in the SSC_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

32.8.9 Interrupt

Most bits in the SSC_SR have a corresponding bit in interrupt management registers.

The SSC can be programmed to generate an interrupt when it detects an event. The interrupt is controlled by writing the Interrupt Enable Register (SSC_IER) and Interrupt Disable Register (SSC_IDR). These registers enable and disable, respectively, the corresponding interrupt by setting and clearing the corresponding bit in the Interrupt Mask Register (SSC_IMR), which controls the generation of interrupts by asserting the SSC interrupt line connected to the interrupt controller.

Figure 32-19. Interrupt Block Diagram



33.8.1 SPI Control Register

Name: SPI_CR

Address: 0x40008000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	–	–	–	–	SPIDIS	SPIEN

- **SPIEN: SPI Enable**

0: No effect.

1: Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

0: No effect.

1: Disables the SPI.

All pins are set in Input mode after completion of the transmission in progress, if any.

If a transfer is in progress when SPIDIS is set, the SPI completes the transmission of the shifter register and does not start any new transfer, even if the SPI_THR is loaded.

Note: If both SPIEN and SPIDIS are equal to one when the SPI_CR is written, the SPI is disabled.

- **SWRST: SPI Software Reset**

0: No effect.

1: Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

The SPI is in Slave mode after software reset.

PDC channels are not affected by software reset.

- **LASTXFER: Last Transfer**

0: No effect.

1: The current NPCS is de-asserted after the character written in TD has been transferred. When SPI_CSRx.CSAAT is set, the communication with the current serial peripheral can be closed by raising the corresponding NPCS line as soon as TD transfer is completed.

Refer to Section 33.7.3.5 “Peripheral Selection” for more details.

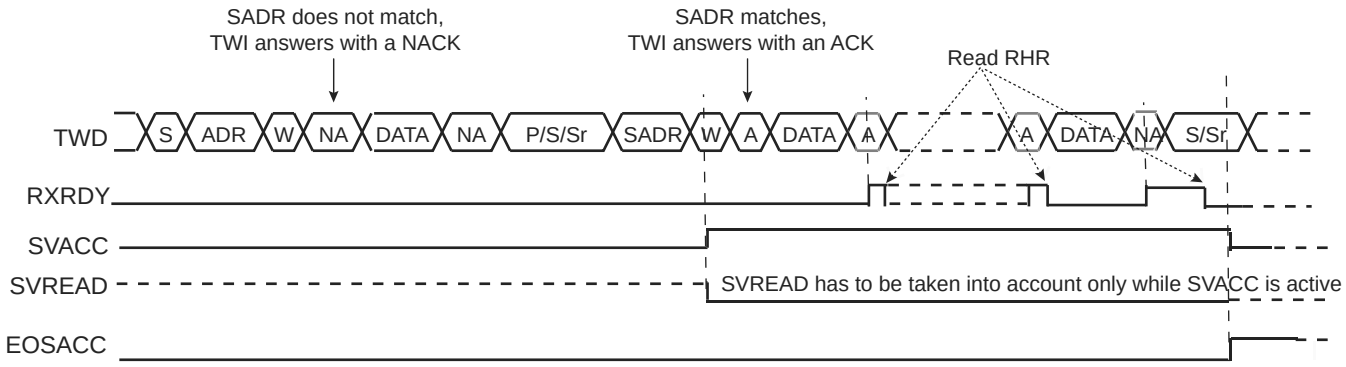
After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

Until a STOP or REPEATED START condition is detected, TWI stores the received data in the TWI_RHR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 34-25 describes the write operation.

Figure 34-25. Write Access Ordered by a Master



Notes: 1. When SVACC is low, the state of SVREAD becomes irrelevant.

2. RXRDY is set when data has been transmitted from the internal shifter to the TWI_RHR and reset when this data is read.

General Call

The general call is performed in order to change the address of the slave.

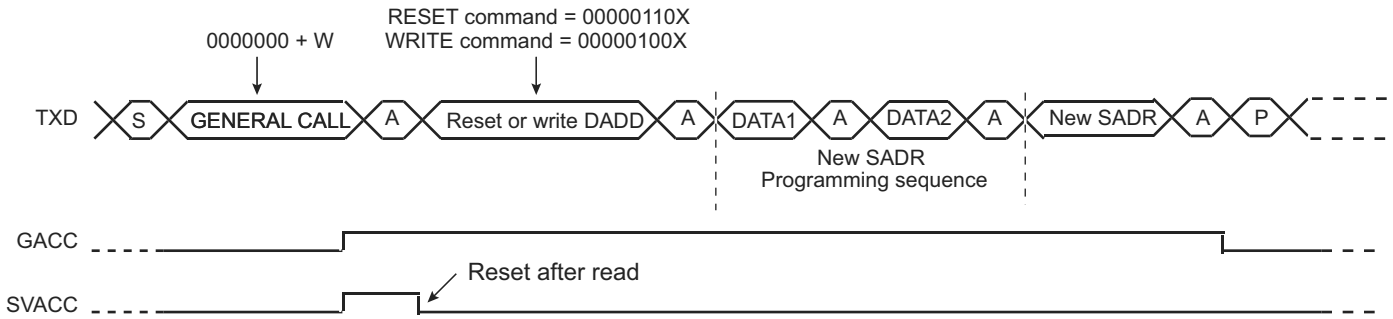
If a GENERAL CALL is detected, GACC is set.

After the detection of GENERAL CALL, it is up to the programmer to decode the commands which come afterwards.

In case of a WRITE command, the programmer has to decode the programming sequence and program a new SADR if the programming sequence matches.

Figure 34-26 describes the GENERAL CALL access.

Figure 34-26. Master Performs a General Call



Note: This method allows the user to create a personal programming sequence by choosing the programming bytes and the number of them. The programming sequence has to be provided to the master.

34.8.10 TWI Receive Holding Register

Name: TWI_RHR

Address: 0x40018030 (0), 0x4001C030 (1)

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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
RXDATA							

- **RXDATA: Master or Slave Receive Holding Data**

35.4 Product Dependencies

35.4.1 I/O Lines

The UART pins are multiplexed with PIO lines. The user must first configure the corresponding PIO Controller to enable I/O line operations of the UART.

Table 35-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
UART0	URXD0	PA9	A
UART0	UTXD0	PA10	A
UART1	URXD1	PB2	A
UART1	UTXD1	PB3	A

35.4.2 Power Management

The UART clock can be controlled through the Power Management Controller (PMC). In this case, the user must first configure the PMC to enable the UART clock. Usually, the peripheral identifier used for this purpose is 1.

35.4.3 Interrupt Sources

The UART interrupt line is connected to one of the interrupt sources of the Interrupt Controller. Interrupt handling requires programming of the Interrupt Controller before configuring the UART.

Table 35-3. Peripheral IDs

Instance	ID
UART0	8
UART1	9

35.5 Functional Description

The UART operates in Asynchronous mode only and supports only 8-bit character handling (with parity). It has no clock pin.

The UART is made up of a receiver and a transmitter that operate independently, and a common baud rate generator. Receiver timeout and transmitter time guard are not implemented. However, all the implemented features are compatible with those of a standard USART.

35.5.1 Baud Rate Generator

The baud rate generator provides the bit period clock named baud rate clock to both the receiver and the transmitter. The baud rate clock is the peripheral clock divided by 16 times the clock divisor (CD) value written in the Baud Rate Generator register (UART_BRGR). If UART_BRGR is set to 0, the baud rate clock is disabled and the UART remains inactive. The maximum allowable baud rate is peripheral clock divided by 16. The minimum allowable baud rate is peripheral clock divided by (16 x 65536).

If STTTO is performed, the counter clock is stopped until a first character is received. The idle state on RXD before the start of the frame does not provide a time-out. This prevents having to obtain a periodic interrupt and enables a wait of the end of frame when the idle state on RXD is detected.

If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

Figure 36-23 shows the block diagram of the Receiver Time-out feature.

Figure 36-23. Receiver Time-out Block Diagram

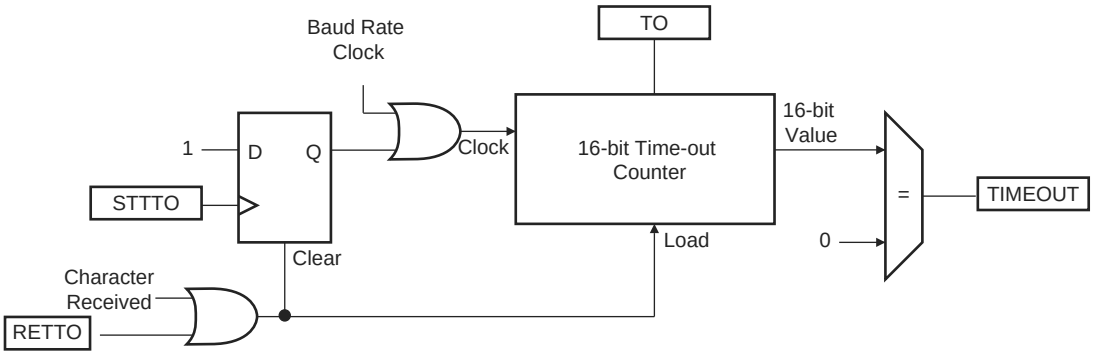


Table 36-10 gives the maximum time-out period for some standard baud rates.

Table 36-10. Maximum Time-out Period

Baud Rate (bit/s)	Bit Time (µs)	Time-out (ms)
600	1,667	109,225
1,200	833	54,613
2,400	417	27,306
4,800	208	13,653
9,600	104	6,827
14,400	69	4,551
19,200	52	3,413
28,800	35	2,276
38,400	26	1,704
56,000	18	1,170
57,600	17	1,138
200,000	5	328

36.6.3.12 Framing Error

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of US_CSR. The FRAME bit is asserted in the middle of the stop bit as soon as the framing error is detected. It is cleared by writing a 1 to the RSTSTA bit in the US_CR.

36.7.1 USART Control Register

Name: US_CR

Address: 0x40024000 (0), 0x40028000 (1)

Access: Write-only

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

23	22	21	20	19	18	17	16
–	–	–	–	RTSDIS	RTSEN	DTRDIS	DTREN

15	14	13	12	11	10	9	8
RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

For SPI control, see Section 36.7.2 ”USART Control Register (SPI_MODE)”.

- **RSTRX: Reset Receiver**

0: No effect.

1: Resets the receiver.

- **RSTTX: Reset Transmitter**

0: No effect.

1: Resets the transmitter.

- **RXEN: Receiver Enable**

0: No effect.

1: Enables the receiver, if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: Disables the receiver.

- **TXEN: Transmitter Enable**

0: No effect.

1: Enables the transmitter if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: Disables the transmitter.

- **RSTSTA: Reset Status Bits**

0: No effect.

1: Resets the status bits PARE, FRAME, OVRE, MANERR and RXBRK in US_CSR.

- **ONEBIT: Start Frame Delimiter Selector**

0: Start frame delimiter is COMMAND or DATA SYNC.

1: Start frame delimiter is one bit.

- **ETRGS: External Trigger**

0: No effect.

1: Enables the External Trigger Interrupt.

38.14.3 HSMCI Data Timeout Register

Name: HSMCI_DTOR

Address: 0x40000008

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
–	DTOMUL			DTOCYC			

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

- **DTOCYC: Data Timeout Cycle Number**

This field determines the maximum number of Master Clock cycles that the HSMCI waits between two data block transfers. It equals (DTOCYC x Multiplier).

- **DTOMUL: Data Timeout Multiplier**

Value	Name	Description
0	1	DTOCYC
1	16	DTOCYC x 16
2	128	DTOCYC x 128
3	256	DTOCYC x 256
4	1024	DTOCYC x 1024
5	4096	DTOCYC x 4096
6	65536	DTOCYC x 65536
7	1048576	DTOCYC x 1048576

If the data time-out set by DTOCYC and DTOMUL has been exceeded, the Data Time-out Error flag (DTOE) in the HSMCI Status Register (HSMCI_SR) rises.

40.7.5 UDP Interrupt Disable Register

Name: UDP_IDR

Address: 0x40034014

Access: Write-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	–	SOFINT	EXTRSM	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
EP7INT	EP6INT	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

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

0: No effect

1: Disables corresponding Endpoint Interrupt

- **RXSUSP: Disable UDP Suspend Interrupt**

0: No effect

1: Disables UDP Suspend Interrupt

- **RXRSM: Disable UDP Resume Interrupt**

0: No effect

1: Disables UDP Resume Interrupt

- **SOFINT: Disable Start Of Frame Interrupt**

0: No effect

1: Disables Start Of Frame Interrupt

The sequence can be customized by programming the Sequence Channel Registers ADC_SEQR1 and ADC_SEQR2 and setting the USEQ bit of the Mode Register (ADC_MR). The user can choose a specific order of channels and can program up to 16 conversions by sequence. The user is free to create a personal sequence by writing channel numbers in ADC_SEQR1 and ADC_SEQR2. Not only can channel numbers be written in any sequence, channel numbers can be repeated several times. When the bit USEQ in ADC_MR is set, the fields USCHx in ADC_SEQR1 and ADC_SEQR2 are used to define the sequence. Only enabled USCHx fields will be part of the sequence. Each USCHx field has a corresponding enable, CHx, in ADC_CHER (USCHx field with the lowest x index is associated with bit CHx of the lowest index).

If all ADC channels (i.e., 16) are used on an application board, there is no restriction of usage of the user sequence. However, if some ADC channels are not enabled for conversion but rather used as pure digital inputs, the respective indexes of these channels cannot be used in the user sequence fields (see ADC_SEQRx). For example, if channel 4 is disabled (ADC_CSR[4] = 0), ADC_SEQRx fields USCH1 up to USCH16 must not contain the value 4. Thus the length of the user sequence may be limited by this behavior.

As an example, if only four channels over 16 (CH0 up to CH3) are selected for ADC conversions, the user sequence length cannot exceed four channels. Each trigger event may launch up to four successive conversions of any combination of channels 0 up to 3 but no more (i.e., in this case the sequence CH0, CH0, CH1, CH1, CH1 is impossible).

A sequence that repeats the same channel several times requires more enabled channels than channels actually used for conversion. For example, the sequence CH0, CH0, CH1, CH1 requires four enabled channels (four free channels on application boards) whereas only CH0, CH1 are really converted.

Note: The reference voltage pins always remain connected in Normal mode as in Sleep mode.

42.6.8 Comparison Window

The ADC Controller features automatic comparison functions. It compares converted values to a low threshold, a high threshold or both, depending on the value of the CMPMODE bit in ADC_EMR. The comparison can be done on all channels or only on the channel specified in the CMPSEL field of ADC_EMR. To compare all channels, the CMPALL bit of ADC_EMR must be set.

The flag can be read on the COMPE bit of the Interrupt Status register (ADC_ISR) and can trigger an interrupt.

The high threshold and the low threshold can be read/write in the Compare Window register (ADC_CWR).

42.6.9 Differential Inputs

The ADC can be used either as a single-ended ADC (DIFF bit = 0 in ADC_COR) or as a fully differential ADC (DIFF bit = 1 in ADC_COR) as shown in Figure 42-7. By default, after a reset, the ADC is in Single-ended mode.

If ANACH is set in ADC_MR, the ADC can apply a different mode on each channel. Otherwise the parameters of CH0 are applied to all channels.

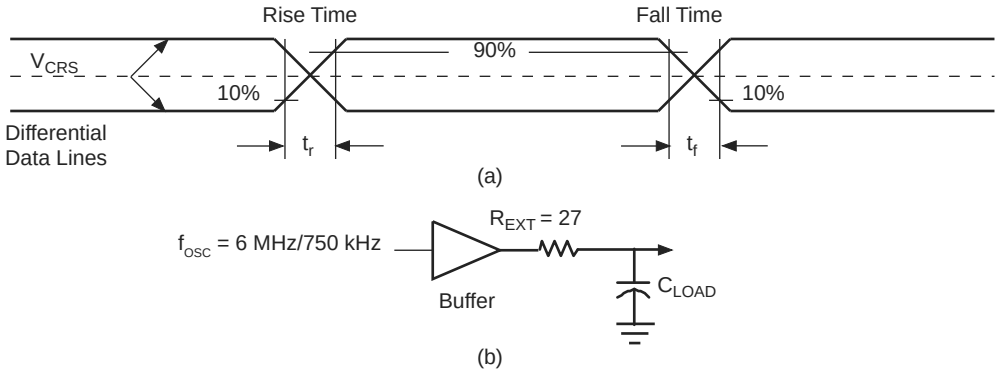
The same inputs are used in Single-ended or Differential mode.

In Single-ended mode, inputs are managed by a 16:1-channel analog multiplexer. In Fully Differential mode, inputs are managed by an 8:1-channel analog multiplexer. See Table 42-4.

Table 42-4. Input Pins and Channel Numbers

Input Pin	Channel Number	
	Single-ended Mode	Differential Mode
AD0	CH0	CH0
AD1	CH1	
AD2	CH2	CH1
AD3	CH3	

Figure 44-17. USB Data Signal Rise and Fall Times



48. Errata

48.1 Errata SAM4SD32/SD16/SA16/S16/S8 Rev. A Parts

The errata are applicable to the devices in Table 48-1.

Table 48-1. Device List for Errata Described in Section 48.1

Device Name	Revision	Chip ID
SAM4SD32C	A	0x29A7_0EE0
SAM4SD32B	A	0x2997_0EE0
SAM4SD16C	A	0x29A7_0CE0
SAM4SD16B	A	0x2997_0CE0
SAM4SA16C	A	0x28A7_0CE0
SAM4SA16B	A	0x2897_0CE0
SAM4S16C	A	0x28AC_0CE0
SAM4S16B	A	0x289C_0CE0
SAM4S8C	A	0x28AC_0AE0
SAM4S8B	A	0x289C_0AE0

48.1.1 Flash Controller (EEFC)

Issue: Flash Buffer Not Cleared

The Write Buffer in the embedded Flash is not cleared after trying to write to a locked region. Therefore, the data that was previously loaded into the Write Buffer would remain in the buffer while the next page write command (e.g., WP) is being executed.

Workaround: Do not do partial programming (Fill completely the Write Buffer). Note that this problem occurs only if the software tries to write into a locked region.

Issue: Code Loop Optimization Cannot Be Disabled

The EFC does not work after the buffer for loop optimization is disabled; in Flash Mode Register (EEFC_FMR), CLOE = 0.

Workaround: The CLOE bit must be kept at 1.

Issue: Erase Sector Command Cannot Be Performed If a Subsector Is Locked (ONLY in Flash Sector0)

If one of subsector (Small Sector 0, Small Sector1 and Larger Sector) is locked, the Erase Sector Command (ES) is not possible on non-locked subsectors.

Workaround: All the lock bits of the sector0 must be cleared prior to issuing the ES command. After the ES command has been issued, the first sector lock bits must be reverted to the state before clearing them.

Table 49-9. SAM4S Datasheet Rev. 11100C 09-Jan-13 Revision History (Continued)

Doc. Rev. 11100C	Comments	Change Request Ref.
	TWI	
	NVIC and AIC changed to Interrupt Controller. Section 33.10.4.5 “PDC” removed. “This bit is only used in Master mode” removed from bitfields ENDRX, ENDTX, RXBUFF, and TXBUFE in Section 34.11.6 “TWI Status Register”.	7844
	Figure 34-23 updated: SVREAD = 1 and first occurrence of RXRDY = 1.	7884
	Removed “20” at the end of the 1st paragraph in Section 34.1 “Description”.	7921
	Table 34-7 ”Register Mapping”, replaced “0x100 - 0x124” with “0x100 - 0x128” and “Reserved for the PDC” with “Reserved for PDC registers” in the PDC line.	7973
	Section 34.10.6 “Using the Peripheral DMA Controller (PDC) in Slave Mode” reworked.	rfo
	UART	
	Table 35-3 ”Register Mapping”, PDC registers info for register mapping updated.	7967
	USART	
	Section 36.7.1 “Baud Rate Generator”, replaced “or 6” with “or 6 times lower” in the last phrase.	rfo
	HSMCI	
	Phrase “not only for Write operations now” removed from NOTBUSY bitfield descriptionI in Section 38.14.12 “HSMCI Status Register”.	8394
	replaced BCNT bitfield table with the corresponding description and updated Warning note in BCNT bitfield description in Section 38.14.7 “HSMCI Block Register”.	8431
	In Section 38.6.3 “Interrupt”, replaced references to NVIC/AIC with “interrupt controller”.	rfo
	PWM	
	Typo corrected in line Timer0 in Table 39-4 ”Fault Inputs”.	8438
	Replaced ‘Main OSC’ with ‘Main OSC (PMC)’ in Table 39-4 ”Fault Inputs”.	rfo
	UDP	
	Pull-up’ and ‘pull-down’ spelling harmonized in the whole chapter.	7867
	Added UDP_CSRx (ISOENDPT) alternate register in Section 40.7.11 “UDP Endpoint Control and Status Register (ISOCHRONOUS)”.	8414
	ADC	
	Removed “...and EOC bit corresponding to the last converted channel” from the last phrase of the third paragraph in Section 42.6.4 “Conversion Results”.	8357
	TRANSFER value set to 2 in TRANSFER bitfield description in Section 42.7.2 “ADC Mode Register”.	8462
	Text amended in Section 42.1 “Description”.	rfo
	SLEEP and FWUP bitfield description texts in tables updated in Section 42.7.2 “ADC Mode Register”.	
	Electrical Characteristics	
	Whole chapter reworked to add SAM4SD32/SD16/SA16 data, various values added or updated.	rfo, 8435
	Clext values changed in Table 44-30.	8391
	Configurations A and B updated in Section 44.4.1 “Backup Mode Current Consumption”.	8422
	Mechanical Characteristics	
	QFN64 package drawing and table updated in Figure 45-5.	8529