Microchip Technology - ATSAM4SD16CB-CN Datasheet

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"

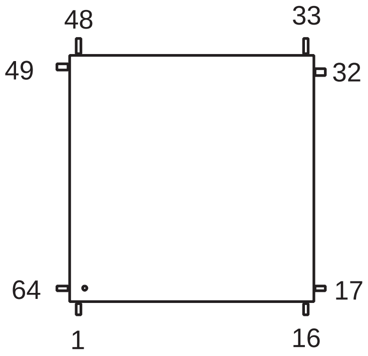
Details	
Product Status	Active
Core Processor	ARM® Cortex®-M4
Core Size	32-Bit Single-Core
Speed	120MHz
Connectivity	EBI/EMI, I²C, IrDA, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	79
Program Memory Size	1MB (1M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	160K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 16x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TFBGA
Supplier Device Package	100-TFBGA (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4sd16cb-cn

4.2 64-lead Packages and Pinouts

Refer to Table 1-1 and Table 1-2 for the overview of devices available in 64-lead packages.

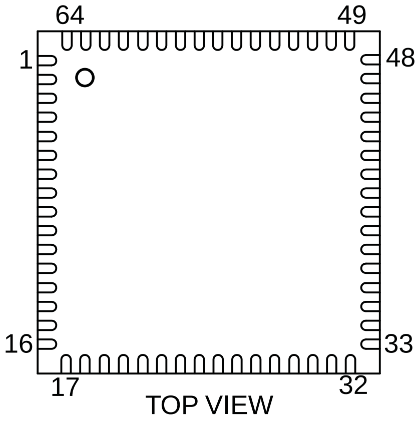
4.2.1 64-lead LQFP Package Outline

Figure 4-4. Orientation of the 64-lead LQFP Package



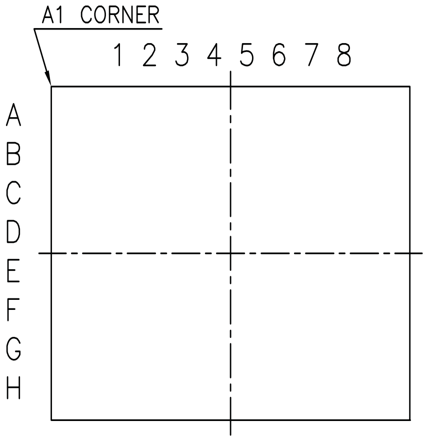
4.2.2 64-lead QFN Package Outline

Figure 5-5. Orientation of the 64-lead QFN Package



4.2.3 64-ball WLCSP Package Outline

Figure 4-6. Orientation of the 64-ball WLCSP Package



4. PIODCENx/PIODCx has priority over WKUPx. Refer to Section 31.5.13 “Parallel Capture Mode”.
5. To select this extra function, refer to Section 42.5.3 “Analog Inputs”.
6. To select this extra function, refer to “Section 31.5.13 “Parallel Capture Mode”.

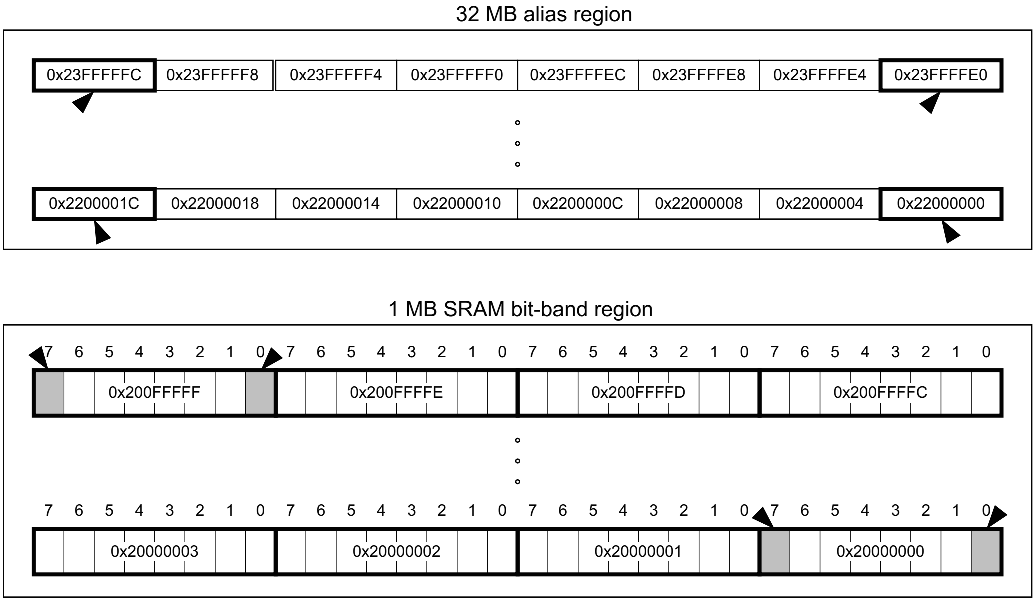
11.2.2 PIO Controller B Multiplexing

Table 11-3. Multiplexing on PIO Controller B (PIOB)

I/O Line	Peripheral A	Peripheral B	Peripheral C	Extra Function	System Function	Comments
PB0	PWMH0			AD4/RTCOUT0[1]		
PB1	PWMH1			AD5/RTCOUT1[1]		
PB2	URXD1	NPCS2		AD6/WKUP12[2]		
PB3	UTXD1	PCK2		AD7[3]		
PB4	TWD1	PWMH2			TDI[4]	
PB5	TWCK1	PWML0		WKUP13[2]	TDO/TRACESWO[4]	
PB6					TMS/SWDIO[4]	
PB7					TCK/SWCLK[4]	
PB8					XOUT[4]	
PB9					XIN[4]	
PB10					DDM	
PB11					DDP	
PB12	PWML1				ERASE[4]	
PB13	PWML2	PCK0		DAC0[5]		64-/100-pin versions
PB14	NPCS1	PWMH3		DAC1[5]		64-/100-pin versions

Notes:
1. Analog input has priority over RTCOUTx pin. See Section 16.5.8 “Waveform Generation”.
2. WKUPx can be used if PIO controller defines the I/O line as "input".
3. To select this extra function, refer to Section 42.5.3 “Analog Inputs”.
4. Refer to Section 6.2 “System I/O Lines”.
5. DAC0 is selected when DACC_CHER.CH0 is set. DAC1 is selected when DACC_CHER.CH1 is set. See Section 43.7.3 “DACC Channel Enable Register”.

Figure 12-4. Bit-band Mapping



Directly Accessing an Alias Region

Writing to a word in the alias region updates a single bit in the bit-band region.

Bit[0] of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit[0] set to 1 writes a 1 to the bit-band bit, and writing a value with bit[0] set to 0 writes a 0 to the bit-band bit.

Bits[31:1] of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0xFF. Writing 0x00 has the same effect as writing 0x0E.

Reading a word in the alias region:

- 0x00000000 indicates that the targeted bit in the bit-band region is set to 0
- 0x00000001 indicates that the targeted bit in the bit-band region is set to 1

Directly Accessing a Bit-band Region

"Behavior of Memory Accesses" describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

12.4.2.6 Memory Endianness

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0–3 hold the first stored word, and bytes 4–7 hold the second stored word. "Little-endian Format" describes how words of data are stored in memory.

Little-endian Format

In little-endian format, the processor stores the least significant byte of a word at the lowest-numbered byte, and the most significant byte at the highest-numbered byte. For example:

- If subtracting a negative value from a positive value generates a negative value.

The Compare operations are identical to subtracting, for CMP, or adding, for CMN, except that the result is discarded. See the instruction descriptions for more information.

Note: Most instructions update the status flags only if the S suffix is specified. See the instruction descriptions for more information.

Condition Code Suffixes

The instructions that can be conditional have an optional condition code, shown in syntax descriptions as {cond}. Conditional execution requires a preceding IT instruction. An instruction with a condition code is only executed if the condition code flags in the APSR meet the specified condition. Table 12-16 shows the condition codes to use.

A conditional execution can be used with the IT instruction to reduce the number of branch instructions in code.

Table 12-16 also shows the relationship between condition code suffixes and the N, Z, C, and V flags.

Table 12-16. Condition Code Suffixes

Suffix	Flags	Meaning
EQ	Z = 1	Equal
NE	Z = 0	Not equal
CS or HS	C = 1	Higher or same, unsigned ≥
CC or LO	C = 0	Lower, unsigned <
MI	N = 1	Negative
PL	N = 0	Positive or zero
VS	V = 1	Overflow
VC	V = 0	No overflow
HI	C = 1 and Z = 0	Higher, unsigned >
LS	C = 0 or Z = 1	Lower or same, unsigned ≤
GE	N = V	Greater than or equal, signed ≥
LT	N != V	Less than, signed <
GT	Z = 0 and N = V	Greater than, signed >
LE	Z = 1 and N != V	Less than or equal, signed ≤
AL	Can have any value	Always. This is the default when no suffix is specified.

Absolute Value

The example below shows the use of a conditional instruction to find the absolute value of a number. R0 = ABS(R1).

```
MOVS    R0, R1          ; R0 = R1, setting flags
IT      MI              ; IT instruction for the negative condition
RSBMI   R0, R1, #0      ; If negative, R0 = -R1
```

Compare and Update Value

The example below shows the use of conditional instructions to update the value of R4 if the signed values R0 is greater than R1 and R2 is greater than R3.

```
CMP     R0, R1   ; Compare R0 and R1, setting flags
ITT     GT       ; IT instruction for the two GT conditions
CMPGT   R2, R3   ; If 'greater than', compare R2 and R3, setting flags
MOVGT   R4, R5   ; If still 'greater than', do R4 = R5
```

Examples

```
LDR     R8, [R10]               ; Loads R8 from the address in R10.
LDRNE   R2, [R5, #960]!         ; Loads (conditionally) R2 from a word
                                ; 960 bytes above the address in R5, and
                                ; increments R5 by 960.
STR     R2, [R9,#const-struc]   ; const-struc is an expression evaluating
                                ; to a constant in the range 0-4095.
STRH    R3, [R4], #4            ; Store R3 as halfword data into address in
                                ; R4, then increment R4 by 4
LDRD    R8, R9, [R3, #0x20]     ; Load R8 from a word 32 bytes above the
                                ; address in R3, and load R9 from a word 36
                                ; bytes above the address in R3
STRD    R0, R1, [R8], #-16      ; Store R0 to address in R8, and store R1 to
                                ; a word 4 bytes above the address in R8,
                                ; and then decrement R8 by 16.
```

12.6.4.3 LDR and STR, Register Offset

Load and Store with register offset.

Syntax

```
      op{type}{cond} Rt, [Rn, Rm {, LSL #n}]
```

where:

op is one of:

- LDR Load Register.
- STR Store Register.

type is one of:

- B unsigned byte, zero extend to 32 bits on loads.
- SB signed byte, sign extend to 32 bits (LDR only).
- H unsigned halfword, zero extend to 32 bits on loads.
- SH signed halfword, sign extend to 32 bits (LDR only).
- - omit, for word.

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

Rt is the register to load or store.

Rn is the register on which the memory address is based.

Rm is a register containing a value to be used as the offset.

LSL #*n* is an optional shift, with *n* in the range 0 to 3.

Operation

LDR instructions load a register with a value from memory.

STR instructions store a register value into memory.

The memory address to load from or store to is at an offset from the register *Rn*. The offset is specified by the register *Rm* and can be shifted left by up to 3 bits using LSL.

The value to load or store can be a byte, halfword, or word. For load instructions, bytes and halfwords can either be signed or unsigned. See “Address Alignment” .

Restrictions

In these instructions:

- *Rn* must not be PC

12.6.5.9 SADD16 and SADD8

Signed Add 16 and Signed Add 8

Syntax

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

where:

op is any of:

SADD16 Performs two 16-bit signed integer additions.

SADD8 Performs four 8-bit signed integer additions.

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

Rd is the destination register.

Rn is the first register holding the operand.

Rm is the second register holding the operand.

Operation

Use these instructions to perform a halfword or byte add in parallel:

The SADD16 instruction:

1. Adds each halfword from the first operand to the corresponding halfword of the second operand.
2. Writes the result in the corresponding halfwords of the destination register.

The SADD8 instruction:

1. Adds each byte of the first operand to the corresponding byte of the second operand.

Writes the result in the corresponding bytes of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not change the flags.

Examples

```
SADD16 R1, R0     ; Adds the halfwords in R0 to the corresponding
                  ; halfwords of R1 and writes to corresponding halfword
                  ; of R1.
SADD8  R4, R0, R5 ; Adds bytes of R0 to the corresponding byte in R5 and
                  ; writes to the corresponding byte in R4.
```

Examples

```
      QADD16   R7, R4, R2  ; Adds halfwords of R4 with corresponding halfword of
                           ; R2, saturates to 16 bits and writes to
                           ; corresponding halfword of R7
      QADD8    R3, R1, R6  ; Adds bytes of R1 to the corresponding bytes of R6,
                           ; saturates to 8 bits and writes to corresponding
                           ; byte of R3
      QSUB16   R4, R2, R3  ; Subtracts halfwords of R3 from corresponding
                           ; halfword of R2, saturates to 16 bits, writes to
                           ; corresponding halfword of R4
      QSUB8    R4, R2, R5  ; Subtracts bytes of R5 from the corresponding byte
                           ; in R2, saturates to 8 bits, writes to corresponding
                           ; byte of R4.
```

12.6.7.4 QASX and QSAX

Saturating Add and Subtract with Exchange and Saturating Subtract and Add with Exchange, signed.

Syntax

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

where:

op is one of:

QASX Add and Subtract with Exchange and Saturate.

QSAX Subtract and Add with Exchange and Saturate.

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

Rd is the destination register.

Rn, Rm are registers holding the first and second operands.

Operation

The QASX instruction:

1. Adds the top halfword of the source operand with the bottom halfword of the second operand.
2. Subtracts the top halfword of the second operand from the bottom highword of the first operand.
3. Saturates the result of the subtraction and writes a 16-bit signed integer in the range $-2^{15} \leq x \leq 2^{15} - 1$, where *x* equals 16, to the bottom halfword of the destination register.
4. Saturates the results of the sum and writes a 16-bit signed integer in the range $-2^{15} \leq x \leq 2^{15} - 1$, where *x* equals 16, to the top halfword of the destination register.

The QSAX instruction:

1. Subtracts the bottom halfword of the second operand from the top highword of the first operand.
2. Adds the bottom halfword of the source operand with the top halfword of the second operand.
3. Saturates the results of the sum and writes a 16-bit signed integer in the range $-2^{15} \leq x \leq 2^{15} - 1$, where *x* equals 16, to the bottom halfword of the destination register.
4. Saturates the result of the subtraction and writes a 16-bit signed integer in the range $-2^{15} \leq x \leq 2^{15} - 1$, where *x* equals 16, to the top halfword of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the condition code flags.

24.5.1 UART0 Serial Port

Communication is performed through the UART0 initialized to 115200 Baud, 8, n, 1.

The Send and Receive File commands use the Xmodem protocol to communicate. Any terminal performing this protocol can be used to send the application file to the target. The size of the binary file to send depends on the SRAM size embedded in the product. In all cases, the size of the binary file must be lower than the SRAM size because the Xmodem protocol requires some SRAM memory to work. See Section 24.2 ”Hardware and Software Constraints”.

24.5.2 Xmodem Protocol

The Xmodem protocol supported is the 128-byte length block. This protocol uses a two-character CRC-16 to guarantee detection of a maximum bit error.

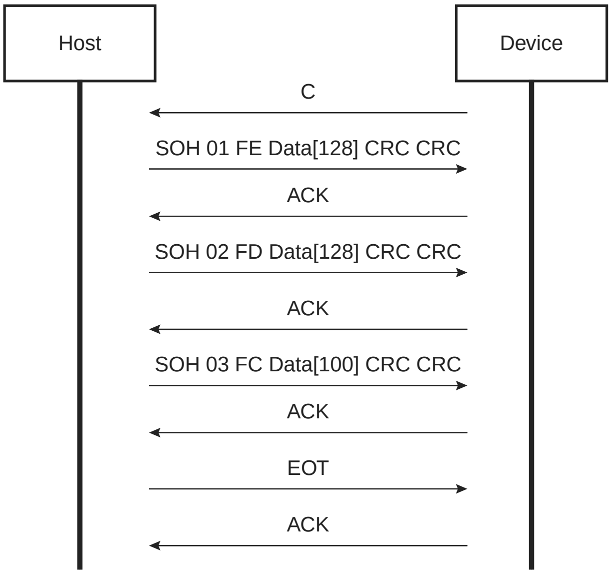
Xmodem protocol with CRC is accurate provided both sender and receiver report successful transmission. Each block of the transfer looks like:

<SOH><blk #><255-blk #><--128 data bytes--><checksum> in which:

- <SOH> = 01 hex
- <blk #> = binary number, starts at 01, increments by 1, and wraps 0FFH to 00H (not to 01)
- <255-blk #> = 1’s complement of the blk#.
- <checksum> = 2 bytes CRC16

Figure 24-2 shows a transmission using this protocol.

Figure 24-2. Xmodem Transfer Example


Host
Device
C
SOH 01 FE Data[128] CRC CRC
ACK
SOH 02 FD Data[128] CRC CRC
ACK
SOH 03 FC Data[100] CRC CRC
ACK
EOT
ACK

24.5.3 USB Device Port

The device uses the USB communication device class (CDC) drivers to take advantage of the installed PC RS-232 software to talk over the USB. The CDC class is implemented in all releases of Windows®, beginning with Windows 98 SE. The CDC document, available at www.usb.org, describes a way to implement devices such as ISDN modems and virtual COM ports.

The Vendor ID (VID) is Atmel’s vendor ID 0x03EB. The product ID (PID) is 0x6124. These references are used by the host operating system to mount the correct driver. On Windows systems, the INF files contain the correspondence between vendor ID and product ID.

the slow clock until it reaches 0. At this time, the LOCK bit is set in PMC_SR and can trigger an interrupt to the processor. The user has to load the number of slow clock cycles required to cover the PLL transient time into the PLLCOUNT field.

The PLL clock can be divided by 2 by writing the PLLDIV2 (PLLADIV2, PLLBDIV2) bit in PMC_MCKR.

It is prohibited to change the 4/8/12 MHz fast RC oscillator or the main oscillator selection in CKGR_MOR while the master clock source is the PLL and the PLL reference clock is the fast RC oscillator.

The user must:

1. Switch on the main RC oscillator by writing a 1 to the CSS field of PMC_MCKR.
2. Change the frequency (MOSCRCF) or oscillator selection (MOSCSEL) in CKGR_MOR.
3. Wait for MOSCRCS (if frequency changes) or MOSCSELS (if oscillator selection changes) in PMC_SR.
4. Disable and then enable the PLL.
5. Wait for the LOCK flag in PMC_SR.
6. Switch back to the PLL by writing the appropriate value to the CSS field of PMC_MCKR.

- Receive data

36.6.5.1 IrDA Modulation

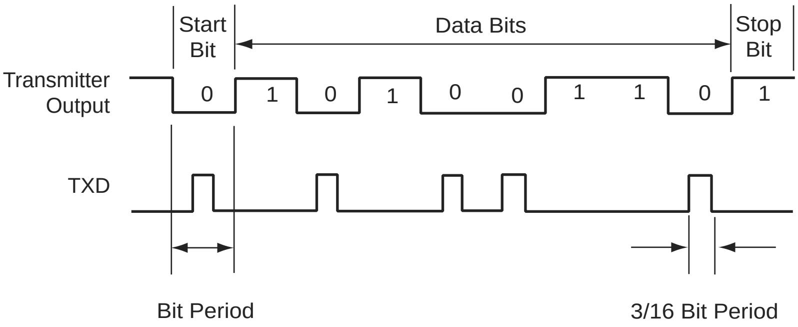
For baud rates up to and including 115.2 kbit/s, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 36-11.

Table 36-11. IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
2.4 kbit/s	78.13 µs
9.6 kbit/s	19.53 µs
19.2 kbit/s	9.77 µs
38.4 kbit/s	4.88 µs
57.6 kbit/s	3.26 µs
115.2 kbit/s	1.63 µs

Figure 36-33 shows an example of character transmission.

Figure 36-33. IrDA Modulation



36.6.5.2 IrDA Baud Rate

Table 36-12 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

Table 36-12. IrDA Baud Rate Error

Peripheral Clock	Baud Rate (bit/s)	CD	Baud Rate Error	Pulse Time (µs)
3,686,400	115,200	2	0.00%	1.63
20,000,000	115,200	11	1.38%	1.63
32,768,000	115,200	18	1.25%	1.63
40,000,000	115,200	22	1.38%	1.63
3,686,400	57,600	4	0.00%	3.26
20,000,000	57,600	22	1.38%	3.26
32,768,000	57,600	36	1.25%	3.26
40,000,000	57,600	43	0.93%	3.26
3,686,400	38,400	6	0.00%	4.88
20,000,000	38,400	33	1.38%	4.88
32,768,000	38,400	53	0.63%	4.88
40,000,000	38,400	65	0.16%	4.88

- **CHRL: Character Length**

Value	Name	Description
0	5_BIT	Character length is 5 bits
1	6_BIT	Character length is 6 bits
2	7_BIT	Character length is 7 bits
3	8_BIT	Character length is 8 bits

- **SYNC: Synchronous Mode Select**

0: USART operates in Asynchronous mode.

1: USART operates in Synchronous mode.

- **PAR: Parity Type**

Value	Name	Description
0	EVEN	Even parity
1	ODD	Odd parity
2	SPACE	Parity forced to 0 (Space)
3	MARK	Parity forced to 1 (Mark)
4	NO	No parity
6	MULTIDROP	Multidrop mode

- **NBSTOP: Number of Stop Bits**

Value	Name	Description
0	1_BIT	1 stop bit
1	1_5_BIT	1.5 stop bit (SYNC = 0) or reserved (SYNC = 1)
2	2_BIT	2 stop bits

- **CHMODE: Channel Mode**

Value	Name	Description
0	NORMAL	Normal mode
1	AUTOMATIC	Automatic Echo. Receiver input is connected to the TXD pin.
2	LOCAL_LOOPBACK	Local Loopback. Transmitter output is connected to the Receiver Input.
3	REMOTE_LOOPBACK	Remote Loopback. RXD pin is internally connected to the TXD pin.

- **MSBF: Bit Order**

0: Least significant bit is sent/received first.

1: Most significant bit is sent/received first.

- **MODE9: 9-bit Character Length**

0: CHRL defines character length

1: 9-bit character length

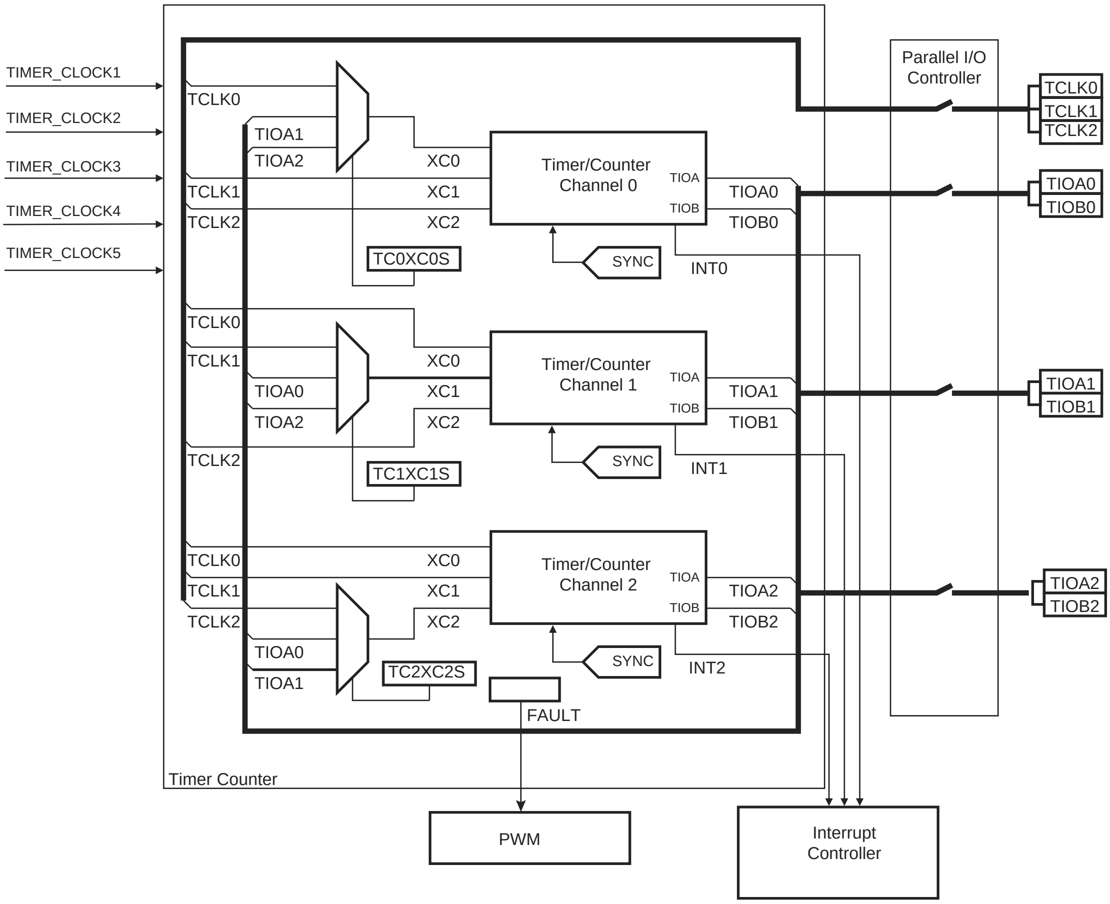
37.3 Block Diagram

Table 37-1. Timer Counter Clock Assignment

Name	Definition
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5	SLCK

Note: 1. When SLCK is selected for Peripheral Clock (CSS = 0 in PMC Master Clock Register), SLCK input is equivalent to Peripheral Clock.

Figure 37-1. Timer Counter Block Diagram



Note: The QDEC connections are detailed in Figure 37-15.

- **ETRGS: External Trigger Status (cleared on read)**

0: External trigger has not occurred since the last read of the Status Register.

1: External trigger has occurred since the last read of the Status Register.

- **CLKSTA: Clock Enabling Status**

0: Clock is disabled.

1: Clock is enabled.

- **MTIOA: TIOA Mirror**

0: TIOA is low. If TC_CMRx.WAVE = 0, this means that TIOA pin is low. If TC_CMRx.WAVE = 1, this means that TIOA is driven low.

1: TIOA is high. If TC_CMRx.WAVE = 0, this means that TIOA pin is high. If TC_CMRx.WAVE = 1, this means that TIOA is driven high.

- **MTIOB: TIOB Mirror**

0: TIOB is low. If TC_CMRx.WAVE = 0, this means that TIOB pin is low. If TC_CMRx.WAVE = 1, this means that TIOB is driven low.

1: TIOB is high. If TC_CMRx.WAVE = 0, this means that TIOB pin is high. If TC_CMRx.WAVE = 1, this means that TIOB is driven high.

39.7.30 PWM Write Protection Control Register

Name: PWM_WPCR

Address: 0x400200E4

Access: Write-only

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**
WPRG5	WPRG4	WPRG3	WPRG2	WPRG1	WPRG0	WPCMD	

See Section 39.6.6 “Register Write Protection” for the list of registers that can be write-protected.

- **WPCMD: Write Protection Command**

This command is performed only if the WPKEY corresponds to 0x50574D (“PWM” in ASCII).

Value	Name	Description
0	DISABLE_SW_PROT	Disables the software write protection of the register groups of which the bit WPRGx is at ‘1’.
1	ENABLE_SW_PROT	Enables the software write protection of the register groups of which the bit WPRGx is at ‘1’.
2	ENABLE_HW_PROT	Enables the hardware write protection of the register groups of which the bit WPRGx is at ‘1’. Only a hardware reset of the PWM controller can disable the hardware write protection. Moreover, to meet security requirements, the PIO lines associated with the PWM can not be configured through the PIO interface.

- **WPRGx: Write Protection Register Group x**

0: The WPCMD command has no effect on the register group x.

1: The WPCMD command is applied to the register group x.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x50574D	PASSWD	Writing any other value in this field aborts the write operation of the WPCMD field. Always reads as 0

41.7.2 ACC Mode Register

Name: ACC_MR

Address: 0x40040004

Access: Read/Write

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td>–</td><td>FE</td><td>SELFS</td><td>INV</td><td>–</td><td colspan="2">EDGETYP</td><td>ACEN</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>–</td><td colspan="3">SELPLUS</td><td>–</td><td colspan="3">SELMINUS</td></tr>
</table>

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

- **SELMINUS: Selection for Minus Comparator Input**

0..7: Selects the input to apply on analog comparator SELMINUS comparison input.

Value	Name	Description
0	TS	Select TS
1	ADVREF	Select ADVREF
2	DAC0	Select DAC0
3	DAC1	Select DAC1
4	AD0	Select AD0
5	AD1	Select AD1
6	AD2	Select AD2
7	AD3	Select AD3

- **SELPLUS: Selection For Plus Comparator Input**

0..7: Selects the input to apply on analog comparator SELPLUS comparison input.

Value	Name	Description
0	AD0	Select AD0
1	AD1	Select AD1
2	AD2	Select AD2
3	AD3	Select AD3
4	AD4	Select AD4
5	AD5	Select AD5
6	AD6	Select AD6
7	AD7	Select AD7

- **ACEN: Analog Comparator Enable**

0 (DIS): Analog comparator disabled.

1 (EN): Analog comparator enabled.

Figure 42-2. Sequence of ADC Conversions When Tracking Time > Conversion Time

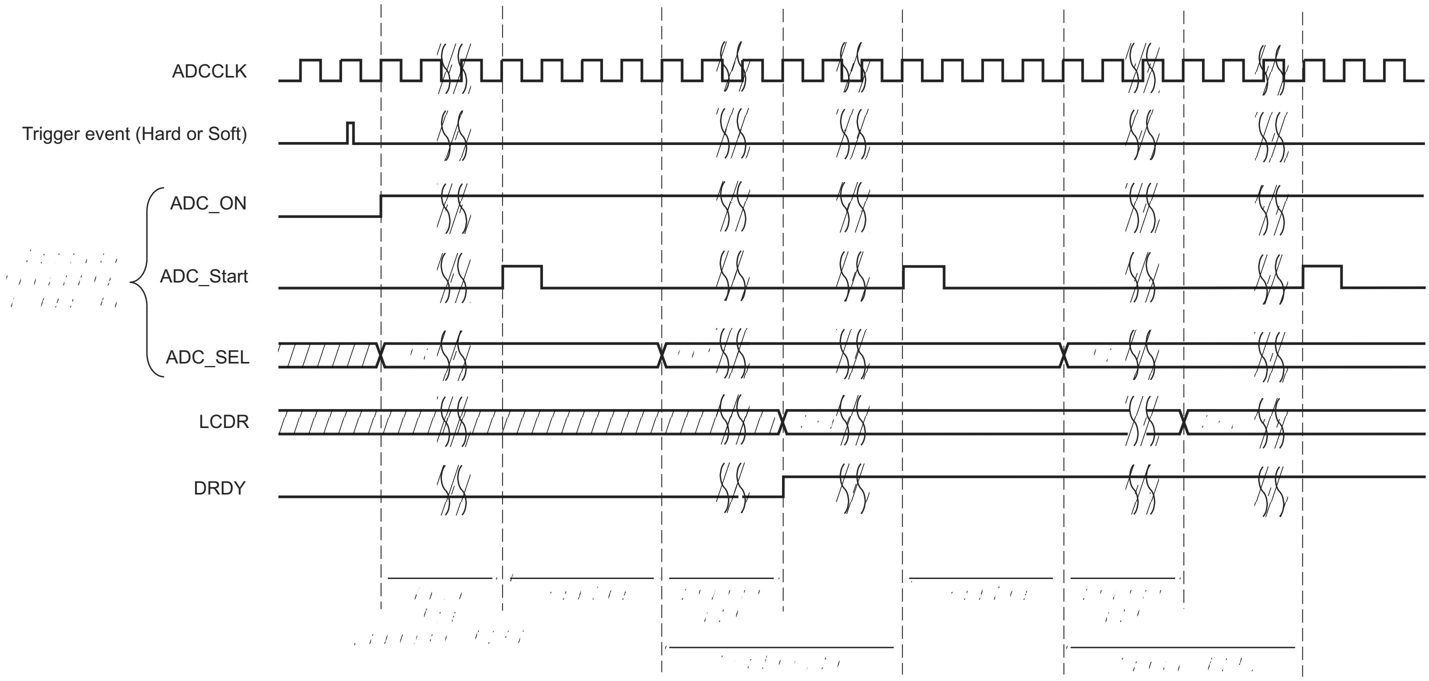


Figure 42-3. Sequence of ADC Conversions When Tracking Time < Conversion Time

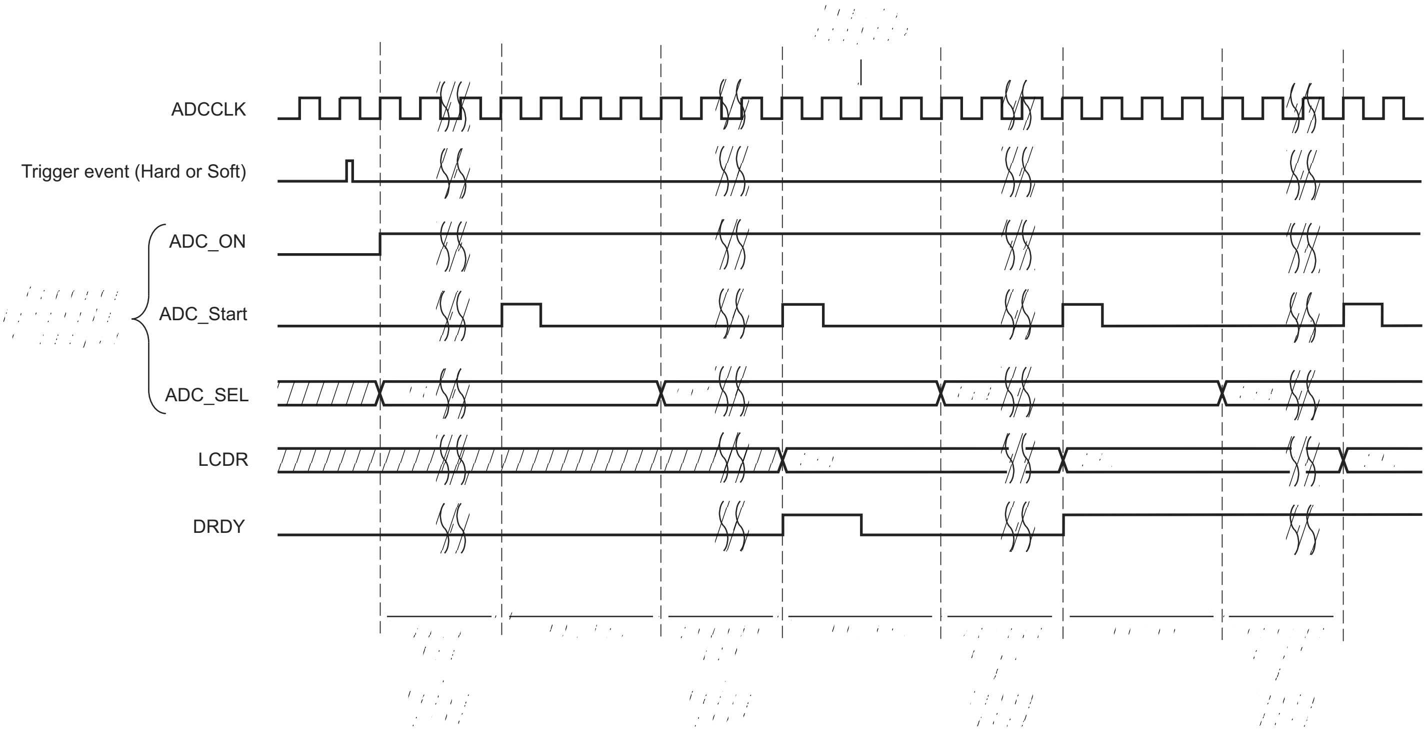


Table 45-20. 64-ball WLCSP Package Dimensions (in mm) (Continued)

	Symbol	Common Dimensions		
		MIN.	NOM.	MAX.
Package Edge Tolerance	aaa	0.03		
Coplanarity (Whole Wafer)	ccc	0.075		
Ball/Bump Offset (Package)	ddd	0.05		
Ball/Bump Offset (Ball)	eee	0.015		

Notes: 1. For SAM4SD32/SAM4SD16

2. For SAM4S16/SAM4S8

3. For SAM4S4/SAM4S2

4. SAM4SD32/SD16/S8/S16 are pin-to-pin compatible.

Figure 45-8. UBM Pad Installation

Table 45-21. WLCSP Package Reference - Soldering Information (Substrate Level)

UBM Pad (Under Bump Metallurgy) (E)	200 µm
PBO2 Opening (j)	240 µm

Table 45-22. Device and 64-ball WLCSP Package Maximum Weight

SAM4S	TBD	mg

Table 45-23. 64-ball WLCSP Package Characteristics

Moisture Sensitivity Level	1

Table 45-24. 64-ball WLCSP Package Reference

JEDEC Drawing Reference	Not JEDEC
JESD97 Classification	e1

48.2.2 Flash

Issue: Read Error after a GPNVM or Lock Bit Writing

The sequence below leads to a bad read value.

Fail sequence is:

Read Flash @ address XXX

Programming Flash: Write GPNVM or Lock Bit instructions

Read Flash @ address XXX

Workaround: A dummy read at another address needs to be included in the sequence.

Sequence is:

Read Flash @ address XXX

Programming Flash: Write GPNVM or Lock Bit instructions

Read Flash @ address YYY (dummy read)

Read Flash @ address XXX

48.2.3 Watchdog

Issue: Watchdog Not Stopped in Wait Mode

When the Watchdog is enabled and the bit WAITMODE = 1 is used to enter wait mode, the watchdog is not halted. If the time spent in Wait Mode is longer than the Watchdog time-out, the device will be reset if Watchdog reset is enabled.

Workaround: When entering wait mode, the Wait For Event (WFE) instruction of the processor Cortex-M4 must be used with the SLEEPDEEP of the System Control Register (SCB_SCR) of the Cortex-M = 0.

48.2.4 Brownout Detector

Issue: Unpredictable Behavior if BOD is Disabled, VDDCORE is Lost and VDDIO is Connected

In active mode or in wait mode, if the Brownout Detector is disabled (SUPC_MR.BODDIS = 1) and power is lost on VDDCORE while VDDIO is powered, the device might not be properly reset and may behave unpredictably.

Workaround: When the Brownout Detector is disabled in active or in wait mode, VDDCORE always needs to be powered.

Table 49-3. SAM4S Datasheet Rev. 11100I Revision History (Continued)

Doc. Date	Changes
03-Apr-15	**Section 44., “Electrical Characteristics”**: Updated Table 44-27, “4/8/12 MHz RC Oscillators Characteristics” Updated parameter definition for $R_{para(standby)}$ and $C_{para(standby)}$ in Table 44-33, “XIN Clock Electrical Characteristics (In Bypass Mode)” Updated SSC4 and SSC7 values and removed Note 1 in Table 44-66, “SSC Timings” Removed footnotes from Table 44-31, “3 to 20 MHz Crystal Oscillator Characteristics” Updated $t_{TRACKTIM}$ value in Table 44-41, “ADC Timing Characteristics” Replaced t_{of} symbol with t_{fo} in Table 44-72, “Two-wire Serial Bus Requirements”
	Section 45., “Mechanical Characteristics” Updated Section 45.6, “64-ball WLCSP Mechanical Characteristics”
	Section 48., “Errata” Added limitation on PB4 Input Voltage Range on Rev. A and Rev. B Parts (Section 48.1.6, “PIO”, Section 48.2.5, “PIO”, Section 48.3.5, “PIO” and Section 48.4.3, “PIO”)