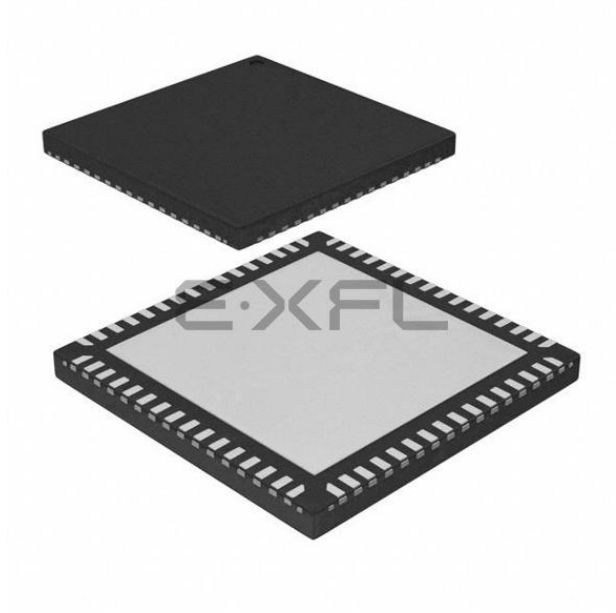


Welcome to E-XFL.COM

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 ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	64-VFQFN Exposed Pad
Supplier Device Package	64-QFN (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4sa16bb-mnr

Figure 2-5. SAM4S4/S2 100-pin Version Block Diagram

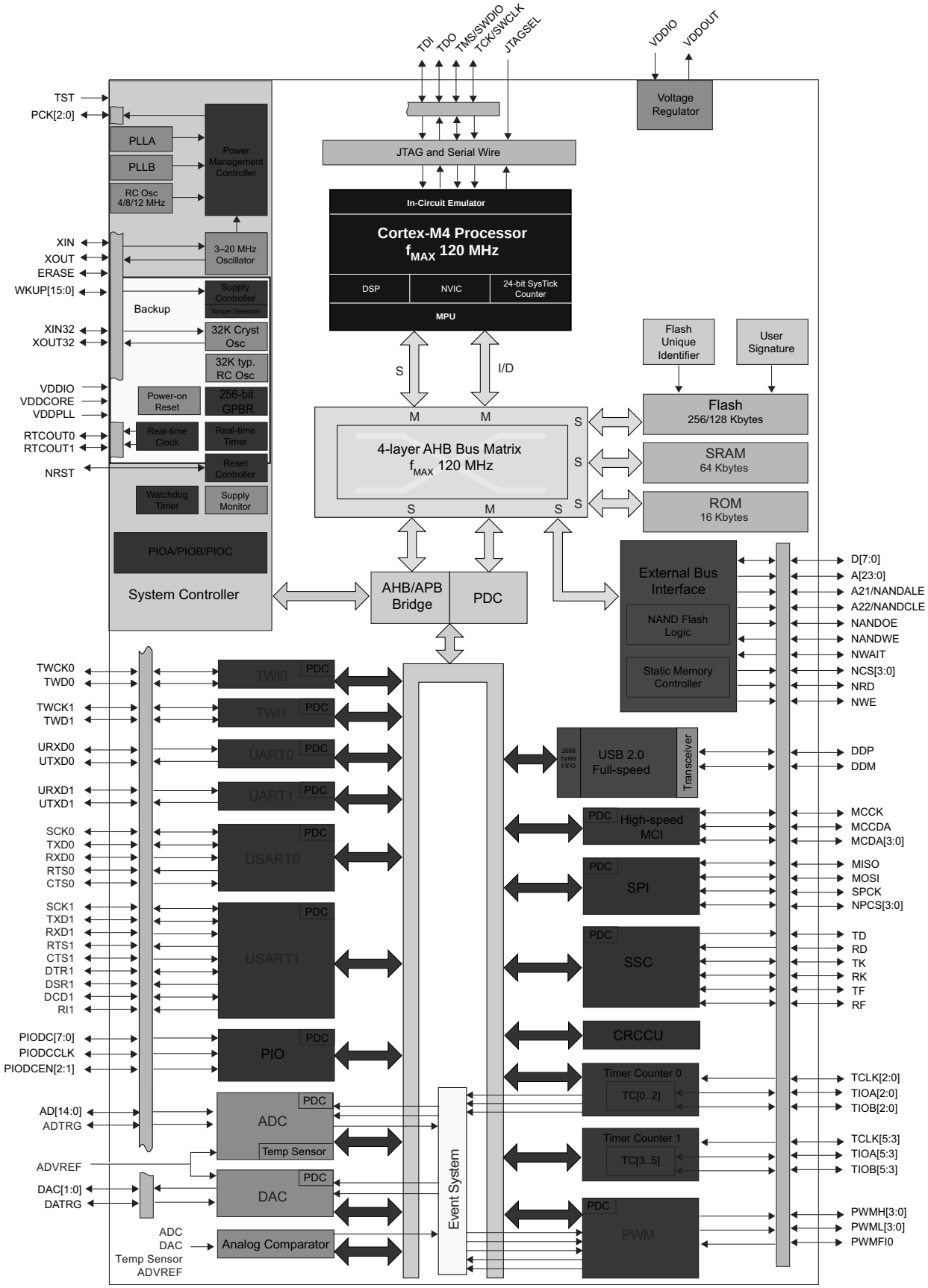


Table 12-13. Cortex-M4 Instructions (Continued)

Mnemonic	Operands	Description	Flags
QSUB16	{Rd,} Rn, Rm	Saturating Subtract 16	–
QSUB8	{Rd,} Rn, Rm	Saturating Subtract 8	–
RBIT	Rd, Rn	Reverse Bits	–
REV	Rd, Rn	Reverse byte order in a word	–
REV16	Rd, Rn	Reverse byte order in each halfword	–
REVSH	Rd, Rn	Reverse byte order in bottom halfword and sign extend	–
ROR, RORS	Rd, Rm, <Rs|#n>	Rotate Right	N,Z,C
RRX, RRXS	Rd, Rm	Rotate Right with Extend	N,Z,C
RSB, RSBS	{Rd,} Rn, Op2	Reverse Subtract	N,Z,C,V
SADD16	{Rd,} Rn, Rm	Signed Add 16	GE
SADD8	{Rd,} Rn, Rm	Signed Add 8 and Subtract with Exchange	GE
SASX	{Rd,} Rn, Rm	Signed Add	GE
SBC, SBCS	{Rd,} Rn, Op2	Subtract with Carry	N,Z,C,V
SBFX	Rd, Rn, #lsb, #width	Signed Bit Field Extract	–
SDIV	{Rd,} Rn, Rm	Signed Divide	–
SEL	{Rd,} Rn, Rm	Select bytes	–
SEV	–	Send Event	–
SHADD16	{Rd,} Rn, Rm	Signed Halving Add 16	–
SHADD8	{Rd,} Rn, Rm	Signed Halving Add 8	–
SHASX	{Rd,} Rn, Rm	Signed Halving Add and Subtract with Exchange	–
SHSAX	{Rd,} Rn, Rm	Signed Halving Subtract and Add with Exchange	–
SHSUB16	{Rd,} Rn, Rm	Signed Halving Subtract 16	–
SHSUB8	{Rd,} Rn, Rm	Signed Halving Subtract 8	–
SMLABB, SMLABT, SMLATB, SMLATT	Rd, Rn, Rm, Ra	Signed Multiply Accumulate Long (halfwords)	Q
SMLAD, SMLADX	Rd, Rn, Rm, Ra	Signed Multiply Accumulate Dual	Q
SMLAL	RdLo, RdHi, Rn, Rm	Signed Multiply with Accumulate ($32 \times 32 + 64$), 64-bit result	–
SMLALBB, SMLALBT, SMLALTB, SMLALTT	RdLo, RdHi, Rn, Rm	Signed Multiply Accumulate Long, halfwords	–
SMLALD, SMLALDX	RdLo, RdHi, Rn, Rm	Signed Multiply Accumulate Long Dual	–
SMLAWB, SMLAWT	Rd, Rn, Rm, Ra	Signed Multiply Accumulate, word by halfword	Q
SMLSD	Rd, Rn, Rm, Ra	Signed Multiply Subtract Dual	Q
SMLSLD	RdLo, RdHi, Rn, Rm	Signed Multiply Subtract Long Dual	
SMMLA	Rd, Rn, Rm, Ra	Signed Most significant word Multiply Accumulate	–
SMMLS, SMMLR	Rd, Rn, Rm, Ra	Signed Most significant word Multiply Subtract	–
SMMUL, SMMULR	{Rd,} Rn, Rm	Signed Most significant word Multiply	–
SMUAD	{Rd,} Rn, Rm	Signed dual Multiply Add	Q

12.6.10 Branch and Control Instructions

The table below shows the branch and control instructions.

Table 12-25. Branch and Control Instructions

Mnemonic	Description
B	Branch
BL	Branch with Link
BLX	Branch indirect with Link
BX	Branch indirect
CBNZ	Compare and Branch if Non Zero
CBZ	Compare and Branch if Zero
IT	If-Then
TBB	Table Branch Byte
TBH	Table Branch Halfword

- All conditional instructions except B*cond* must be inside an IT block. B*cond* can be either outside or inside an IT block but has a larger branch range if it is inside one
- Each instruction inside the IT block must specify a condition code suffix that is either the same or logical inverse as for the other instructions in the block.

Your assembler might place extra restrictions on the use of IT blocks, such as prohibiting the use of assembler directives within them.

Condition Flags

This instruction does not change the flags.

Example

```
ITTE   NE           ; Next 3 instructions are conditional
ANDNE  R0, R0, R1   ; ANDNE does not update condition flags
ADDSNE R2, R2, #1   ; ADDSNE updates condition flags
MOVEQ  R2, R3       ; Conditional move


CMP    R0, #9       ; Convert R0 hex value (0 to 15) into ASCII
                    ; ('0'-'9', 'A'-'F')
ITE    GT           ; Next 2 instructions are conditional
ADDGT  R1, R0, #55  ; Convert 0xA -> 'A'
ADDLE  R1, R0, #48  ; Convert 0x0 -> '0'


IT     GT           ; IT block with only one conditional instruction
ADDGT  R1, R1, #1   ; Increment R1 conditionally


ITTEE  EQ           ; Next 4 instructions are conditional
MOVEQ  R0, R1       ; Conditional move
ADDEQ  R2, R2, #10  ; Conditional add
ANDNE  R3, R3, #1   ; Conditional AND
BNE.W  dloop        ; Branch instruction can only be used in the last
                    ; instruction of an IT block


IT     NE           ; Next instruction is conditional
ADD    R0, R0, R1   ; Syntax error: no condition code used in IT block
```

12.6.10.4 TBB and TBH

Table Branch Byte and Table Branch Halfword.

Syntax

```
TBB [Rn, Rm]
TBH [Rn, Rm, LSL #1]
```

where:

Rn is the register containing the address of the table of branch lengths.

If *Rn* is PC, then the address of the table is the address of the byte immediately following the TBB or TBH instruction.

Rm is the index register. This contains an index into the table. For halfword tables, LSL #1 doubles the value in *Rm* to form the right offset into the table.

Depending on the number of bits to be programmed within the page, the EEFC adapts the write operations required to program the Flash.

When a ‘Write Page’ (WP) command is issued, the EEFC starts the programming sequence and all the bits written at 0 in the latch buffer are cleared in the Flash memory array.

During programming, i.e., until EEFC_FSR.FDRY rises, access to the Flash is not allowed.

Full Page Programming

To program a full page, all the bits of the page must be erased before writing the latch buffer and issuing the WP command. The latch buffer must be written in ascending order, starting from the first address of the page. See Figure 20-8 "Full Page Programming".

Partial Page Programming

To program only part of a page using the WP command, the following constraints must be respected:

- Data to be programmed must be contained in integer multiples of 64-bit address-aligned words.
- 64-bit words can be programmed only if all the corresponding bits in the Flash array are erased (at logical value 1).

See Figure 20-9 "Partial Page Programming".

Programming Bytes

Individual bytes can be programmed using the Partial page programming mode.
In this case, an area of 64 bits must be reserved for each byte.

Refer to Figure 20-10 "Programming Bytes in the Flash".

23.6 Transfer Control Registers Memory Mapping

Table 23-2. Transfer Control Register Memory Mapping

Offset	Register	Name	Access	Reset
CRCCU_DSCR + 0x0	CRCCU Transfer Address Register	TR_ADDR	Read/Write	0x00000000
CRCCU_DSCR + 0x4	CRCCU Transfer Control Register	TR_CTRL	Read/Write	0x00000000
CRCCU_DSCR + 0xC–0x10	Reserved	–	–	–
CRCCU_DSCR+0x10	CRCCU Transfer Reference Register	TR_CRC	Read/Write	0x00000000

Note: These registers are memory mapped.

Three round-robin algorithms are implemented:

- Round-Robin arbitration without default master
- Round-Robin arbitration with last access master
- Round-Robin arbitration with fixed default master

25.5.2.1 Round-Robin arbitration without default master

Round-robin arbitration without default master is the main algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to dispatch requests from different masters to the same slave in a pure round-robin manner. At the end of the current access, if no other request is pending, the slave is disconnected from all masters. This configuration incurs one latency cycle for the first access of a burst. Arbitration without default master can be used for masters that perform significant bursts.

25.5.2.2 Round-Robin arbitration with last access master

Round-robin arbitration with last access master is a biased round-robin algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to remove the one latency cycle for the last master that accessed the slave. At the end of the current transfer, if no other master request is pending, the slave remains connected to the last master that performs the access. Other non-privileged masters incur one latency cycle if they want to access the same slave. This technique can be used for masters that mainly perform single accesses.

25.5.2.3 Round-Robin arbitration with fixed default master

Round-robin arbitration with fixed default master is an algorithm used by the Bus Matrix arbiters to remove the one latency cycle for the fixed default master per slave. At the end of the current access, the slave remains connected to its fixed default master. Every request attempted by the fixed default master does not incur latency, whereas other non-privileged masters still incur one latency cycle. This technique can be used for masters that mainly perform single accesses.

25.5.3 Fixed Priority Arbitration

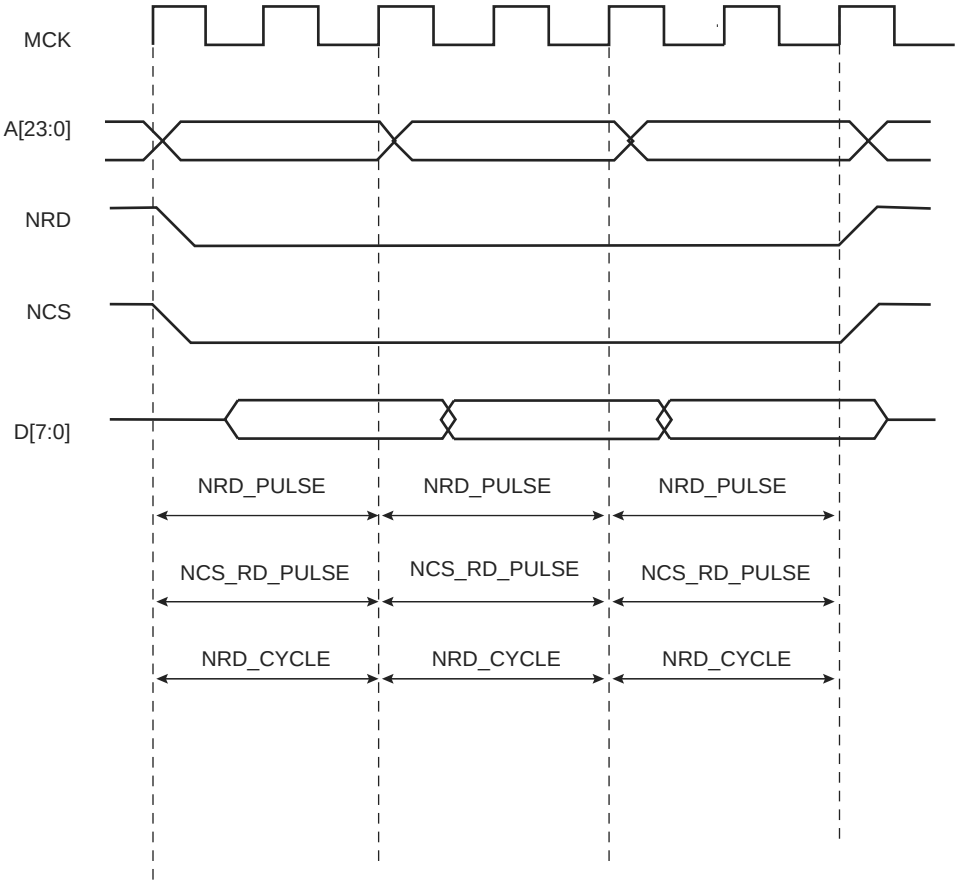
The fixed priority arbitration algorithm is used by the Bus Matrix arbiters to dispatch the requests from different masters to the same slave by using the fixed priority defined by the user. If requests from two or more masters are active at the same time, the master with the highest priority is serviced first. If requests from two or more masters with the same priority are active at the same time, the master with the highest number is serviced first.

For each slave, the priority of each master may be defined through the Priority registers for slaves (MATRIX_PRAS and MATRIX_PRBS).

25.6 System I/O Configuration

The System I/O Configuration register (CCFG_SYSIO) configures I/O lines in system I/O mode (such as JTAG, ERASE, USB, etc.) or as general-purpose I/O lines. Enabling or disabling the corresponding I/O lines in peripheral mode or in PIO mode (PIO_PER or PIO_PDR registers) in the PIO controller has no effect. However, the direction (input or output), pull-up, pull-down and other mode control is still managed by the PIO controller.

Figure 26-6. No Setup, No Hold on NRD and NCS Read Signals



26.9.1.5 Null Pulse

Programming null pulse is not permitted. Pulse must be at least set to 1. A null value leads to unpredictable behavior.

26.9.2 Read Mode

As NCS and NRD waveforms are defined independently of one other, the SMC needs to know when the read data is available on the data bus. The SMC does not compare NCS and NRD timings to know which signal rises first. The READ_MODE parameter in the SMC_MODE register of the corresponding chip select indicates which signal of NRD and NCS controls the read operation.

26.9.2.1 Read is Controlled by NRD (READ_MODE = 1):

Figure 26-7 shows the waveforms of a read operation of a typical asynchronous RAM. The read data is available t_{PACC} after the falling edge of NRD, and turns to ‘Z’ after the rising edge of NRD. In this case, the READ_MODE must be set to 1 (read is controlled by NRD), to indicate that data is available with the rising edge of NRD. The SMC samples the read data internally on the rising edge of Master Clock that generates the rising edge of NRD, whatever the programmed waveform of NCS may be.

26.14 Slow Clock Mode

The SMC is able to automatically apply a set of “Slow clock mode” read/write waveforms when an internal signal driven by the Power Management Controller is asserted because MCK has been turned to a very slow clock rate (typically 32kHz clock rate). In this mode, the user-programmed waveforms are ignored and the Slow clock mode waveforms are applied. This mode is provided so as to avoid reprogramming the User Interface with appropriate waveforms at very slow clock rate. When activated, the Slow mode is active on all chip selects.

26.14.1 Slow Clock Mode Waveforms

Figure 26-28 illustrates the read and write operations in Slow clock mode. They are valid on all chip selects. Table 26-6 indicates the value of read and write parameters in Slow clock mode.

Figure 26-28. Read/Write Cycles in Slow Clock Mode

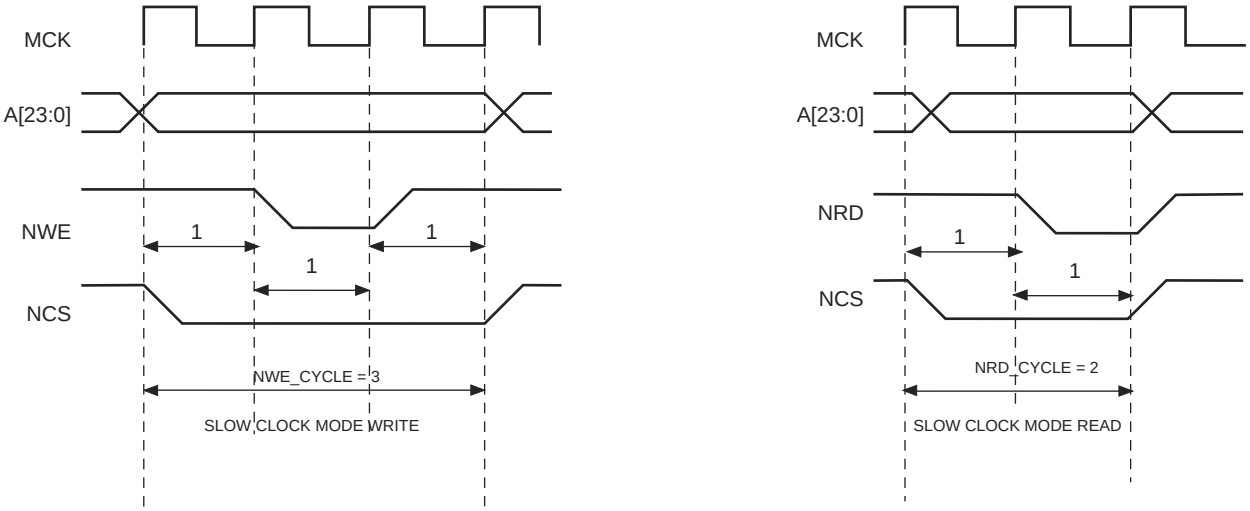


Table 26-6. Read and Write Timing Parameters in Slow Clock Mode

Read Parameters	Duration (cycles)	Write Parameters	Duration (cycles)
NRD_SETUP	1	NWE_SETUP	1
NRD_PULSE	1	NWE_PULSE	1
NCS_RD_SETUP	0	NCS_WR_SETUP	0
NCS_RD_PULSE	2	NCS_WR_PULSE	3
NRD_CYCLE	2	NWE_CYCLE	3

26.16.3 SMC Cycle Register

Name: SMC_CYCLE[0..3]

Address: 0x400E0008 [0], 0x400E0018 [1], 0x400E0028 [2], 0x400E0038 [3]

Access: Read/Write

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

23	22	21	20	19	18	17	16
NRD_CYCLE							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NWE_CYCLE

7	6	5	4	3	2	1	0
NWE_CYCLE							

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

- **NWE_CYCLE: Total Write Cycle Length**

The total write cycle length is the total duration in clock cycles of the write cycle. It is equal to the sum of the setup, pulse and hold steps of the NWE and NCS signals. It is defined as:

Write cycle length = (NWE_CYCLE[8:7]*256 + NWE_CYCLE[6:0]) clock cycles

- **NRD_CYCLE: Total Read Cycle Length**

The total read cycle length is the total duration in clock cycles of the read cycle. It is equal to the sum of the setup, pulse and hold steps of the NRD and NCS signals. It is defined as:

Read cycle length = (NRD_CYCLE[8:7]*256 + NRD_CYCLE[6:0]) clock cycles

Table 29-3. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0110	Oscillator Calibration Register	PMC_OCR	Read/Write	0x0040_4040
0x114–0x120	Reserved	–	–	–
0134–0x144	Reserved	–	–	–

Note: If an offset is not listed in the table it must be considered as “reserved”.

29.17.14PMC Interrupt Enable Register

Name: PMC_IER

Address: 0x400E0460

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCXTS

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **MOSCXTS: Main Crystal Oscillator Status Interrupt Enable**
- **LOCKA: PLLA Lock Interrupt Enable**
- **LOCKB: PLLB Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**
- **MOSCSELS: Main Oscillator Selection Status Interrupt Enable**
- **MOSCRCS: Main On-Chip RC Status Interrupt Enable**
- **CFDEV: Clock Failure Detector Event Interrupt Enable**

- **FSOS: Receive Frame Sync Output Selection**

Value	Name	Description
0	NONE	None, RF pin is an input
1	NEGATIVE	Negative Pulse, RF pin is an output
2	POSITIVE	Positive Pulse, RF pin is an output
3	LOW	Driven Low during data transfer, RF pin is an output
4	HIGH	Driven High during data transfer, RF pin is an output
5	TOGGLING	Toggling at each start of data transfer, RF pin is an output

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on Frame Sync will generate the interrupt RXSYN in the SSC Status Register.

Value	Name	Description
0	POSITIVE	Positive Edge Detection
1	NEGATIVE	Negative Edge Detection

- **FSLEN_EXT: FSLEN Field Extension**

Extends FSLEN field. For details, refer to FSLEN bit description on page 664.

35.6.6 UART Status Register

Name: UART_SR

Address: 0x400E0614 (0), 0x400E0814 (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
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: Receiver Ready**

0: No character has been received since the last read of the UART_RHR, or the receiver is disabled.

1: At least one complete character has been received, transferred to UART_RHR and not yet read.

- **TXRDY: Transmitter Ready**

0: A character has been written to UART_THR and not yet transferred to the internal shift register, or the transmitter is disabled.

1: There is no character written to UART_THR not yet transferred to the internal shift register.

- **ENDRX: End of Receiver Transfer**

0: The end of transfer signal from the receiver PDC channel is inactive.

1: The end of transfer signal from the receiver PDC channel is active.

- **ENDTX: End of Transmitter Transfer**

0: The end of transfer signal from the transmitter PDC channel is inactive.

1: The end of transfer signal from the transmitter PDC channel is active.

- **OVRE: Overrun Error**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0: No framing error has occurred since the last RSTSTA.

1: At least one framing error has occurred since the last RSTSTA.

- **PARE: Parity Error**

0: No parity error has occurred since the last RSTSTA.

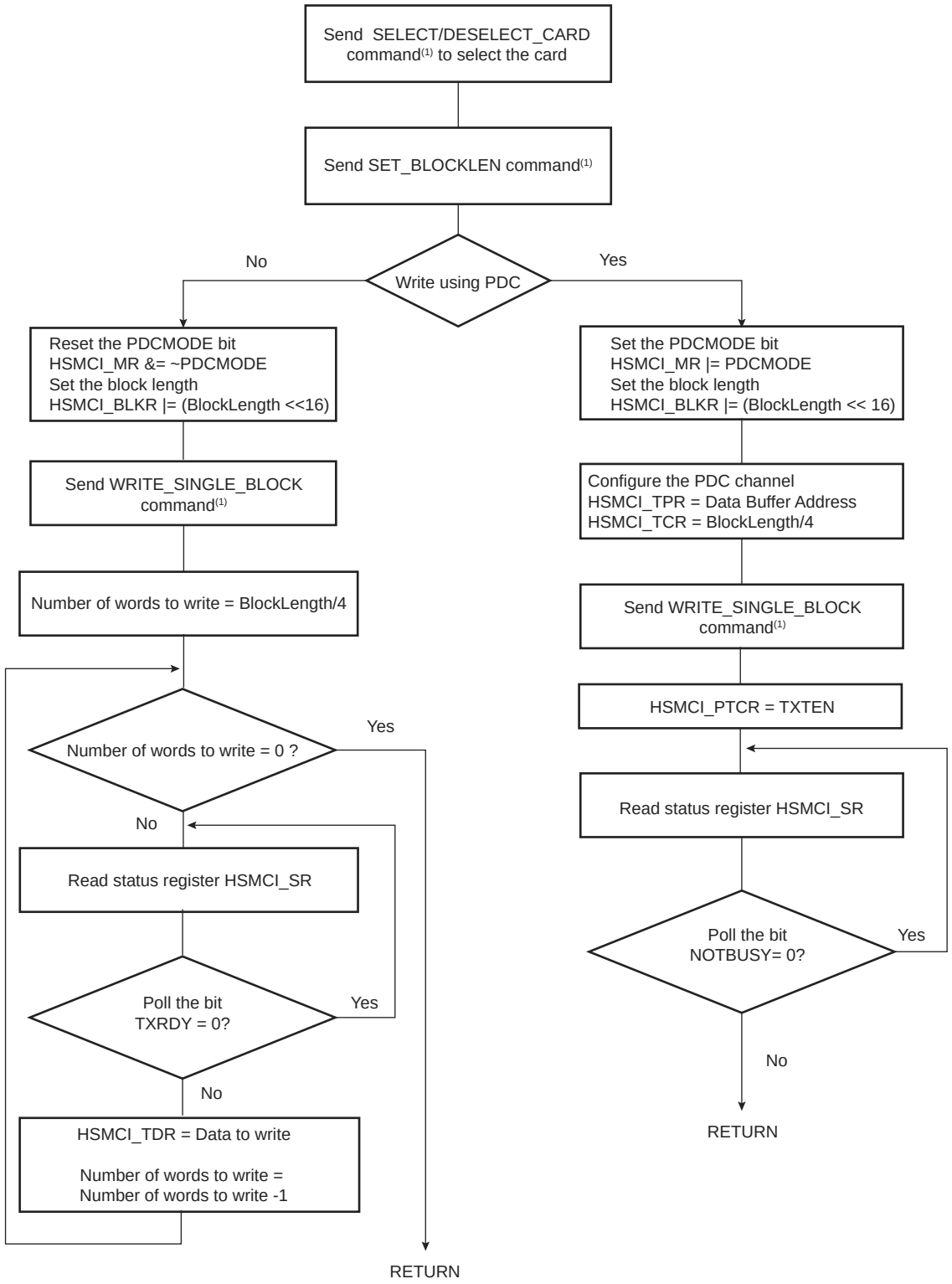
1: At least one parity error has occurred since the last RSTSTA.

- **TXEMPTY: Transmitter Empty**

0: There are characters in UART_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1: There are no characters in UART_THR and there are no characters being processed by the transmitter.

Figure 38-9. Write Functional Flow Diagram



Note: 1. It is assumed that this command has been correctly sent (see Figure 38-7).

The flowchart in Figure 38-10 shows how to manage a multiple write block transfer with the PDC. Polling or interrupt method can be used to wait for the end of write according to the contents of the HSMCI_IMR.

- **PADV: Padding Value**

0: 0x00 value is used when padding data in write transfer.

1: 0xFF value is used when padding data in write transfer.

PADV may be only in manual transfer.

- **PDCMODE: PDC-oriented Mode**

0: Disables PDC transfer

1: Enables PDC transfer. In this case, UNRE and OVRE flags in the HSMCI Status Register (HSMCI_SR) are deactivated after the PDC transfer has been completed.

43. Digital-to-Analog Converter Controller (DACC)

43.1 Description

The Digital-to-Analog Converter Controller (DACC) provides up to 2 analog outputs, making it possible for the digital-to-analog conversion to drive up to 2 independent analog lines.

The DACC supports 12-bit resolution. Data to be converted are sent in a common register for all channels. External triggers or free-running mode are configurable.

43.2 Embedded Characteristics

- Up to Two Independent Analog Outputs
- 12-bit Resolution
- Individual Enable and Disable of Each Analog Channel
- Hardware Trigger
  - External Trigger Pins
- PDC Support
- Internal FIFO
- Register Write Protection

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

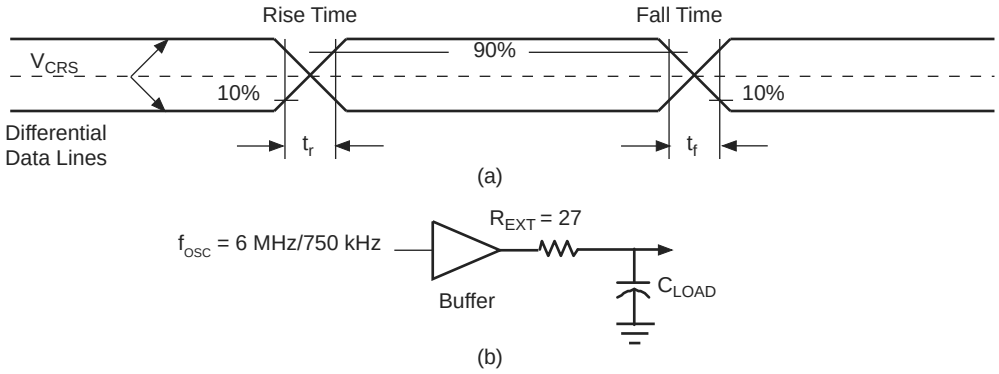


Table 44-66. SSC Timings

Symbol	Parameter	Conditions		Min	Max	Unit
Transmitter						
SSC_0	TK Edge to TF/TD (TK Output, TF Output)	1.8V domain 3.3V domain		-3 -2.6	5.4 5.0	ns
SSC_1	TK Edge to TF/TD (TK Input, TF Output)	1.8V domain 3.3V domain		4.5 3.8	19.6 13.3	ns
SSC_2	TF Setup Time before TK Edge (TK Output)	1.8V domain 3.3V domain		18.9 12.0	–	ns
SSC_3	TF Hold Time after TK Edge (TK Output)	1.8V domain 3.3V domain		0	–	ns
SSC_4	TK Edge to TF/TD (TK Output, TF Input)	1.8V domain	–	2.6	5.4	ns
			STTDLY = 0 START = 4, 5 or 7	$2.6 + (2 \times t_{CPMCK})$ (1)	$5.4 + (2 \times t_{CPMCK})$ (1)	
		3.3V domain	–	2.3	5.0	
			STTDLY = 0 START = 4, 5 or 7	$2.3 + (2 \times t_{CPMCK})$ (1)	$5.0 + (2 \times t_{CPMCK})$ (1)	
SSC_5	TF Setup Time before TK Edge (TK Input)	1.8V domain 3.3V domain		0	–	ns
SSC_6	TF Hold Time after TK edge (TK Input)	1.8V domain 3.3V domain		t_{CPMCK}	–	ns
SSC_7	TK Edge to TF/TD (TK Input, TF Input)	1.8V domain	–	4.5	16.3	ns
			STTDLY = 0 START = 4, 5 or 7	$4.5 + (3 \times t_{CPMCK})$ (1)	$16.3 + (3 \times t_{CPMCK})$ (1)	
		3.3V domain	–	3.8	13.3	
			STTDLY = 0 START = 4, 5 or 7	$3.8 + (3 \times t_{CPMCK})$ (1)	$13.3 + (3 \times t_{CPMCK})$ (1)	
Receiver						
SSC_8	RF/RD Setup Time before RK Edge (RK Input)	1.8V domain 3.3V domain		0	–	ns
SSC_9	RF/RD Hold Time after RK Edge (RK Input)	1.8V domain 3.3V domain		t_{CPMCK}	–	ns
SSC_{10}	RK Edge to RF (RK Input)	1.8V domain 3.3V domain		4.7 4	16.1 12.8	ns
SSC_{11}	RF/RD Setup Time before RK Edge (RK Output)	1.8V domain 3.3V domain		$15.8 - t_{CPMCK}$ $12.5 - t_{CPMCK}$	–	ns
SSC_{12}	RF/RD Hold Time after RK Edge (RK Output)	1.8V domain 3.3V domain		$t_{CPMCK} - 4.3$ $t_{CPMCK} - 3.6$	–	ns
SSC_{13}	RK Edge to RF (RK Output)	1.8V domain 3.3V domain		-3 -2.6	4.3 3.8	ns

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.