**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 | Obsolete |
| Core Processor | ARM® Cortex®-M4/M4F |
| Core Size | 32-Bit Dual-Core |
| Speed | 120MHz |
| Connectivity | EBI/EMI, I²C, IrDA, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, DMA, LCD, POR, PWM, WDT |
| Number of I/O | 57 |
| Program Memory Size | 1MB (1M x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 128K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.62V ~ 3.6V |
| Data Converters | A/D 6x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 100-LQFP |
| Supplier Device Package | 100-LQFP (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atsam4cms16cb-au |

# 5. Power Supply and Power Control

## 5.1 Power Supplies

The SAM4CM has several types of power supply pins. In most cases, a single supply scheme for all power supplies (except VDDBU) is possible. Figure 5-1 shows power domains according to the different power supply pins.

**Figure 5-1. Power Domains**

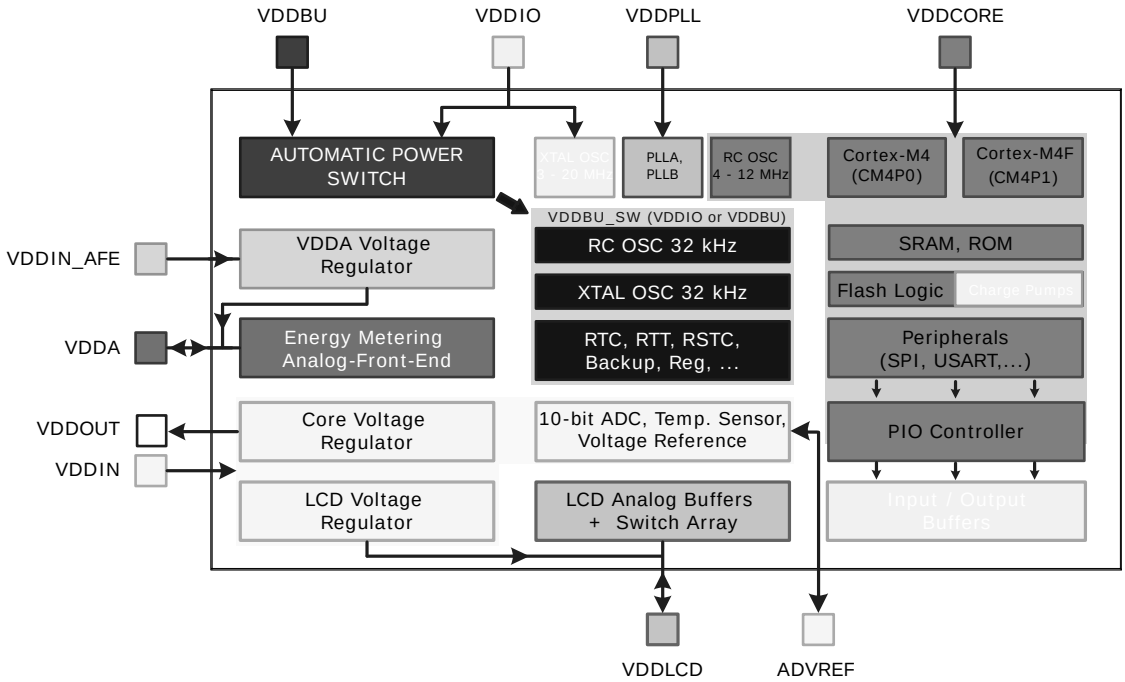



**Table 5-1. Power Supply Voltage Ranges(1)**

| Power Supply | Range | Comments |
|---|---|---|
| VDDIO | 1.6V to 3.6V | Flash memory charge pumps supply for erase and program operations, and read operation.<br>Input/Output buffers supply.<br>EMAFE digital functions supply.<br>Restrictions on range may apply. Refer to Section 46. “Electrical Characteristics”. |
| VDDBU(2) | 1.6V to 3.6V | Backup area power supply.<br>VDDBU is automatically disconnected when VDDIO is present (> 1.9V). |
| VDDIN | 1.6V to 3.6V | Core voltage regulator supply, LCD voltage regulator supply, ADC and programmable voltage reference supply.<br>Restrictions on range may apply. Refer to Section 46. “Electrical Characteristics”. |
| VDDLCD | 2.5V to 3.6V | LCD voltage regulator output.<br>External LCD power supply input (LCD regulator not used).<br>VDDIO/VDDIN must be supplied when the LCD Controller is used. |

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

| Mnemonic | Operands | Description | Flags |
|---|---|---|---|
| LDMDB, LDMEA | Rn{!}, reglist | Load Multiple registers, decrement before | – |
| LDMFD, LDMIA | Rn{!}, reglist | Load Multiple registers, increment after | – |
| LDR | Rt, [Rn, #offset] | Load Register with word | – |
| LDRB, LDRBT | Rt, [Rn, #offset] | Load Register with byte | – |
| LDRD | Rt, Rt2, [Rn, #offset] | Load Register with two bytes | – |
| LDREX | Rt, [Rn, #offset] | Load Register Exclusive | – |
| LDREXB | Rt, [Rn] | Load Register Exclusive with byte | – |
| LDREXH | Rt, [Rn] | Load Register Exclusive with halfword | – |
| LDRH, LDRHT | Rt, [Rn, #offset] | Load Register with halfword | – |
| LDRSB, DRSBT | Rt, [Rn, #offset] | Load Register with signed byte | – |
| LDRSH, LDRSHT | Rt, [Rn, #offset] | Load Register with signed halfword | – |
| LDRT | Rt, [Rn, #offset] | Load Register with word | – |
| LSL, LSLS | Rd, Rm, <Rs\|#n> | Logical Shift Left | N,Z,C |
| LSR, LSRS | Rd, Rm, <Rs\|#n> | Logical Shift Right | N,Z,C |
| MLA | Rd, Rn, Rm, Ra | Multiply with Accumulate, 32-bit result | – |
| MLS | Rd, Rn, Rm, Ra | Multiply and Subtract, 32-bit result | – |
| MOV, MOVS | Rd, Op2 | Move | N,Z,C |
| MOVT | Rd, #imm16 | Move Top | – |
| MOVW, MOV | Rd, #imm16 | Move 16-bit constant | N,Z,C |
| MRS | Rd, spec_reg | Move from special register to general register | – |
| MSR | spec_reg, Rm | Move from general register to special register | N,Z,C,V |
| MUL, MULS | {Rd,} Rn, Rm | Multiply, 32-bit result | N,Z |
| MVN, MVNS | Rd, Op2 | Move NOT | N,Z,C |
| NOP | – | No Operation | – |
| ORN, ORNS | {Rd,} Rn, Op2 | Logical OR NOT | N,Z,C |
| ORR, ORRS | {Rd,} Rn, Op2 | Logical OR | N,Z,C |
| PKHTB, PKHBT | {Rd,} Rn, Rm, Op2 | Pack Halfword | – |
| POP | reglist | Pop registers from stack | – |
| PUSH | reglist | Push registers onto stack | – |
| QADD | {Rd,} Rn, Rm | Saturating double and Add | Q |
| QADD16 | {Rd,} Rn, Rm | Saturating Add 16 | – |
| QADD8 | {Rd,} Rn, Rm | Saturating Add 8 | – |
| QASX | {Rd,} Rn, Rm | Saturating Add and Subtract with Exchange | – |
| QDADD | {Rd,} Rn, Rm | Saturating Add | Q |
| QDSUB | {Rd,} Rn, Rm | Saturating double and Subtract | Q |
| QSAX | {Rd,} Rn, Rm | Saturating Subtract and Add with Exchange | – |
| QSUB | {Rd,} Rn, Rm | Saturating Subtract | Q |

#### 12.6.4.8 LDREX and STREX

Load and Store Register Exclusive.

Syntax

```
LDREX{cond} Rt, [Rn {, #offset}]
STREX{cond} Rd, Rt, [Rn {, #offset}]
LDREXB{cond} Rt, [Rn]
STREXB{cond} Rd, Rt, [Rn]
LDREXH{cond} Rt, [Rn]
STREXH{cond} Rd, Rt, [Rn]
```

where:

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

Rd is the destination register for the returned status.

Rt is the register to load or store.

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

offset is an optional offset applied to the value in *Rn*.

If *offset* is omitted, the address is the value in *Rn*.

Operation

LDREX, LDREXB, and LDREXH load a word, byte, and halfword respectively from a memory address.

STREX, STREXB, and STREXH attempt to store a word, byte, and halfword respectively to a memory address. The address used in any Store-Exclusive instruction must be the same as the address in the most recently executed Load-exclusive instruction. The value stored by the Store-Exclusive instruction must also have the same data size as the value loaded by the preceding Load-exclusive instruction. This means software must always use a Load-exclusive instruction and a matching Store-Exclusive instruction to perform a synchronization operation, see “Synchronization Primitives”.

If an Store-Exclusive instruction performs the store, it writes 0 to its destination register. If it does not perform the store, it writes 1 to its destination register. If the Store-Exclusive instruction writes 0 to the destination register, it is guaranteed that no other process in the system has accessed the memory location between the Load-exclusive and Store-Exclusive instructions.

For reasons of performance, keep the number of instructions between corresponding Load-Exclusive and Store-Exclusive instruction to a minimum.

The result of executing a Store-Exclusive instruction to an address that is different from that used in the preceding Load-Exclusive instruction is unpredictable.

Restrictions

In these instructions:

- Do not use PC
- Do not use SP for *Rd* and *Rt*
- For STREX, *Rd* must be different from both *Rt* and *Rn*
- The value of *offset* must be a multiple of four in the range 0–1020.

Condition Flags

If S is specified, these instructions:

- Update the N and Z flags according to the result
- Can update the C flag during the calculation of *Operand2*, see “Flexible Second Operand”
- Do not affect the V flag.

Examples

```
MOVS  R11, #0x000B    ; Write value of 0x000B to R11, flags get updated
MOV   R1, #0xFA05     ; Write value of 0xFA05 to R1, flags are not updated
MOVS  R10, R12        ; Write value in R12 to R10, flags get updated
MOV   R3, #23         ; Write value of 23 to R3
MOV   R8, SP          ; Write value of stack pointer to R8
MVNS  R2, #0xF        ; Write value of 0xFFFFFFF0 (bitwise inverse of 0xF)
                      ; to the R2 and update flags.
```

#### 12.6.5.7 MOVT

Move Top.

Syntax

```
MOVT{cond} Rd, #imm16
```

where:

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

Rd is the destination register.

imm16 is a 16-bit immediate constant.

Operation

MOVT writes a 16-bit immediate value, *imm16*, to the top halfword, *Rd*[31:16], of its destination register. The write does not affect *Rd*[15:0].

The MOV, MOVT instruction pair enables to generate any 32-bit constant.

Restrictions

*Rd* must not be SP and must not be PC.

Condition Flags

This instruction does not change the flags.

Examples

```
MOVT   R3, #0xF123 ; Write 0xF123 to upper halfword of R3, lower halfword
                   ; and APSR are unchanged.
```

#### 12.6.5.11 SHASX and SHSAX

Signed Halving Add and Subtract with Exchange and Signed Halving Subtract and Add with Exchange.

Syntax

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

where:

op is any of:

SHASX Add and Subtract with Exchange and Halving.

SHSAX Subtract and Add with Exchange and Halving.

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 SHASX instruction:

1. Adds the top halfword of the first operand with the bottom halfword of the second operand.
2. Writes the halfword result of the addition to the top halfword of the destination register, shifted by one bit to the right causing a divide by two, or halving.
3. Subtracts the top halfword of the second operand from the bottom highword of the first operand.
4. Writes the halfword result of the division in the bottom halfword of the destination register, shifted by one bit to the right causing a divide by two, or halving.

The SHSAX instruction:

1. Subtracts the bottom halfword of the second operand from the top highword of the first operand.
2. Writes the halfword result of the addition to the bottom halfword of the destination register, shifted by one bit to the right causing a divide by two, or halving.
3. Adds the bottom halfword of the first operand with the top halfword of the second operand.
4. Writes the halfword result of the division in the top halfword of the destination register, shifted by one bit to the right causing a divide by two, or halving.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the condition code flags.

#### 12.8.3.3 Interrupt Set-pending Registers

**Name:** NVIC_ISPRx [x=0..7]

**Access:** Read/Write

**Reset:** 0x000000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| SETPEND | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| SETPEND | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SETPEND | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SETPEND | | | | | | | |

These registers force interrupts into the pending state, and show which interrupts are pending.

- **SETPEND: Interrupt Set-pending**

Write:

0: No effect.

1: Changes the interrupt state to pending.

Read:

0: Interrupt is not pending.

1: Interrupt is pending.

Notes:
1. Writing a 1 to an ISPR bit corresponding to an interrupt that is pending has no effect.
2. Writing a 1 to an ISPR bit corresponding to a disabled interrupt sets the state of that interrupt to pending.

#### 12.9.1.4 Vector Table Offset Register

**Name:** SCB_VTOR

**Access:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| TBLOFF | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| TBLOFF | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| TBLOFF | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TBLOFF | – | – | – | – | – | – | – |

The SCB_VTOR indicates the offset of the vector table base address from memory address 0x00000000.

- **TBLOFF: Vector Table Base Offset**

It contains bits [29:7] of the offset of the table base from the bottom of the memory map.

Bit [29] determines whether the vector table is in the code or SRAM memory region:

0: Code.

1: SRAM.

It is sometimes called the TBLBASE bit.

Note: When setting TBLOFF, the offset must be aligned to the number of exception entries in the vector table. Configure the next statement to give the information required for your implementation; the statement reminds the user of how to determine the alignment requirement. The minimum alignment is 32 words, enough for up to 16 interrupts. For more interrupts, adjust the alignment by rounding up to the next power of two. For example, if 21 interrupts are required, the alignment must be on a 64-word boundary because the required table size is 37 words, and the next power of two is 64.

Table alignment requirements mean that bits[6:0] of the table offset are always zero.

#### 12.9.1.9 System Handler Priority Register 1

**Name:** SCB_SHPR1

**Access:** Read/Write

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

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| PRI_6 | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PRI_5 | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PRI_4 | | | | | | | |

- **PRI_6: Priority**

Priority of system handler 6, UsageFault.

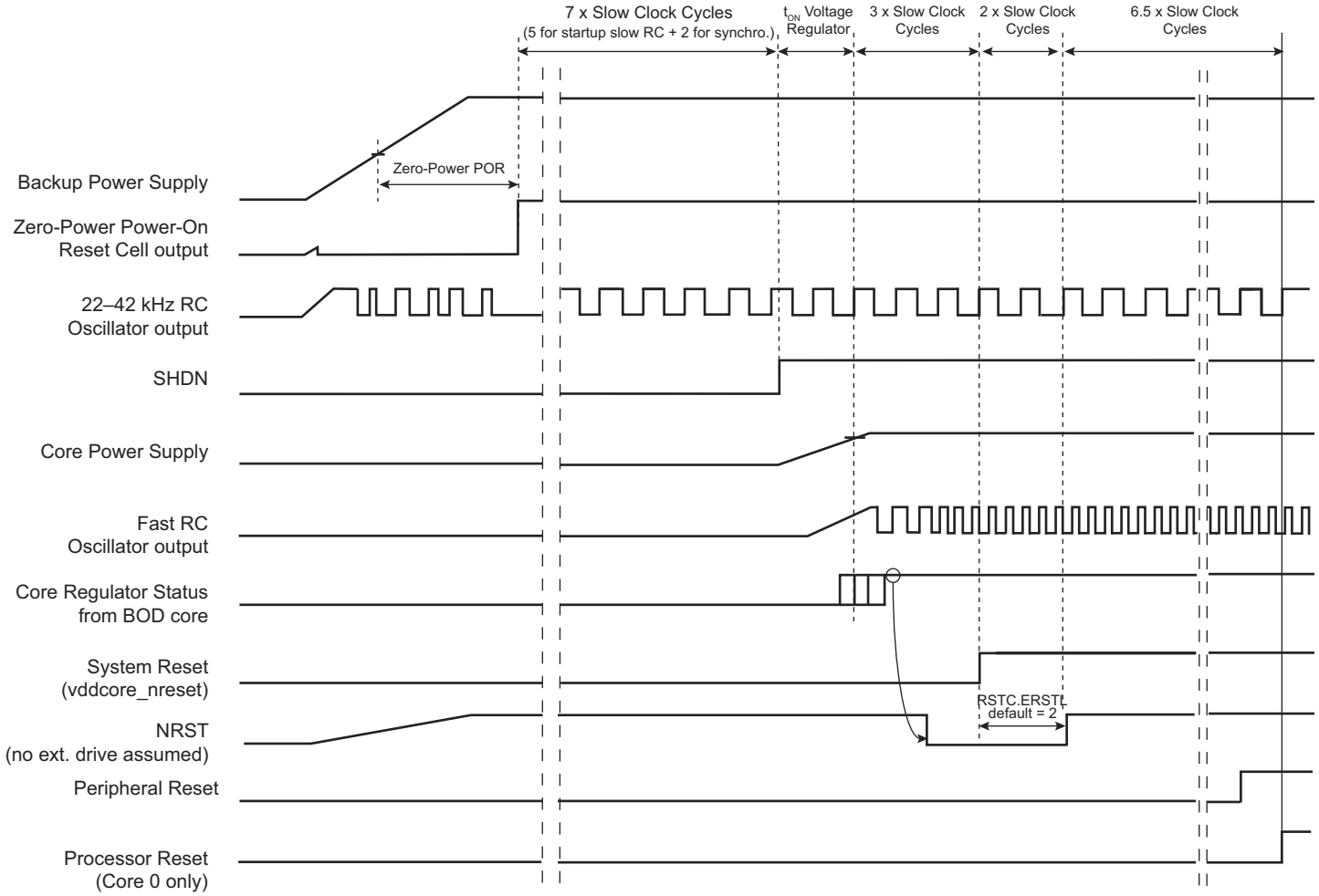
- **PRI_5: Priority**

Priority of system handler 5, BusFault.

- **PRI_4: Priority**

Priority of system handler 4, MemManage.

**Figure 20-3. Raising the VDDBU_SW Power Supply**




Note: After processor reset rising, the core starts fetching instructions from Flash at 4 MHz.

#### 20.4.7.2 SHDN Output Pin

The SHDN pin is designed to drive the enable pin of an external voltage regulator. This pin is controlled by the VROFF bit in SUPC_CR. When the device goes into Backup mode (bit VROFF set), the SHDN pin is asserted low. Upon a wakeup event, the SHDN pin is released (VDDBU level).

### 20.4.8 System Reset

The SUPC manages the system reset signal (vddcore_nreset) to the Reset Controller, as described in Section 20.4.7 ”Backup Power Supply Reset”. The system reset signal is normally asserted before shutting down the core power supply and released as soon as the core power supply is correctly regulated.

There are two additional sources which can be programmed to activate the system reset signal:

- a supply monitor detection
- a brownout detection

#### 20.4.8.1 Supply Monitor Reset

The supply monitor can generate a reset of the system. This can be enabled by setting the SMRSTEN bit in SUPC_SMMR.

The output of the supply monitor is synchronized on SLCK. If SMRSTEN is set and if a supply monitor detection occurs, the system reset is asserted one or two slow clock cycles after the detection.

# 26. Bus Matrix (MATRIX)

## 26.1 Description

The Bus Matrix implements a multi-layer AHB, based on the AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system, thus increasing the overall bandwidth. The Bus Matrix interconnects AHB masters to AHB slaves. The normal latency to connect a master to a slave is one cycle except for the default master of the accessed slave which is connected directly (zero cycle latency).

## 26.2 Embedded Characteristics

- One Decoder for Each Master
- Support for Long Bursts of 32, 64 and 128 Beats and Up to the 256-beat Word Burst AHB Limit
- Enhanced Programmable Mixed Arbitration for Each Slave
  - Round-robin
  - Fixed Priority
  - Latency Quality of Service
- Programmable Default Master for Each Slave
  - No Default Master
  - Last Accessed Default Master
  - Fixed Default Master
- Deterministic Maximum Access Latency for Masters
- Zero or One Cycle Arbitration Latency for the First Access of a Burst
- Bus Lock Forwarding to Slaves
- Master Number Forwarding to Slaves
- Write Protection of User Interface Registers

### 26.2.1 Matrix 0

#### 26.2.1.1 Matrix 0 Masters

The Bus Matrix 0, which corresponds to the sub-system 0 (Core 0 - CM4P0), manages the masters listed in Table 26-1. Each master can perform an access to an available slave concurrently with other masters.

Each master has its own specifically-defined decoder. In order to simplify the addressing, all the masters have the same decodings.

**Table 26-1. List of Bus Matrix Masters**

| | |
|---|---|
| Master 0 | Cortex-M4 Instruction/Data (CM4P0 I/D Bus) |
| Master 1 | Cortex-M4 System (CM4P0 S Bus) |
| Master 2 | Peripheral DMA Controller 0 (PDC0) |
| Master 3 | Integrity Check Module (ICM) |
| Master 4 | Matrix1 |
| Master 5 | EBI Matrix1 |
| Master 6 | CMCC0 |

**Table 30-3. 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”.

### 30.18.18PMC Fast Startup Polarity Register

**Name:** PMC_FSPR

**Address:** 0x400E0474

**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 |
|---|---|---|---|---|---|---|---|
| FSTP15 | FSTP14 | FSTP13 | FSTP12 | FSTP11 | FSTP10 | FSTP9 | FSTP8 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FSTP7 | FSTP6 | FSTP5 | FSTP4 | FSTP3 | FSTP2 | FSTP1 | FSTP0 |

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

- **FSTPx: Fast Startup Input Polarityx**

Defines the active polarity of the corresponding wake-up input. If the corresponding wake-up input is enabled and at the FSTP level, it enables a fast restart signal.

The glitch filters are controlled by the Input Filter Enable Register (PIO_IFER), the Input Filter Disable Register (PIO_IFDR) and the Input Filter Status Register (PIO_IFSR). Writing PIO_IFER and PIO_IFDR respectively sets and clears bits in PIO_IFSR. This last register enables the glitch filter on the I/O lines.

When the glitch and/or debouncing filter is enabled, it does not modify the behavior of the inputs on the peripherals. It acts only on the value read in PIO_PDSR and on the input change interrupt detection. The glitch and debouncing filters require that the peripheral clock is enabled.

**Figure 32-4. Input Glitch Filter Timing**

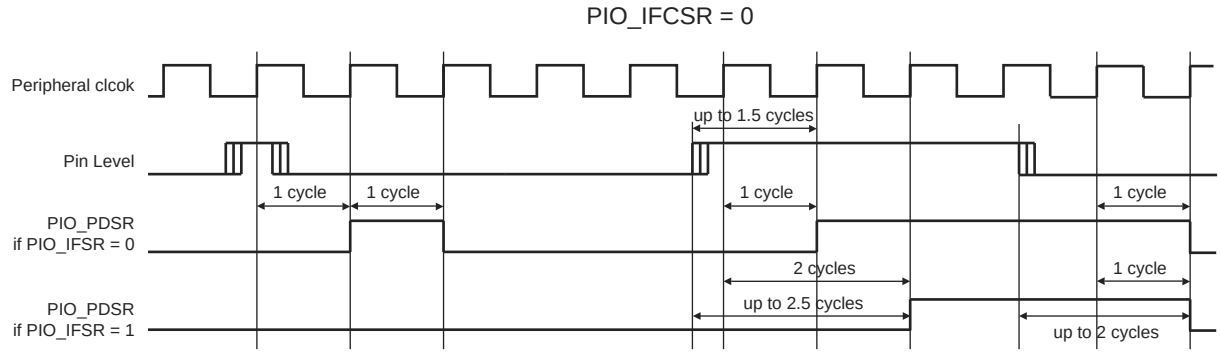
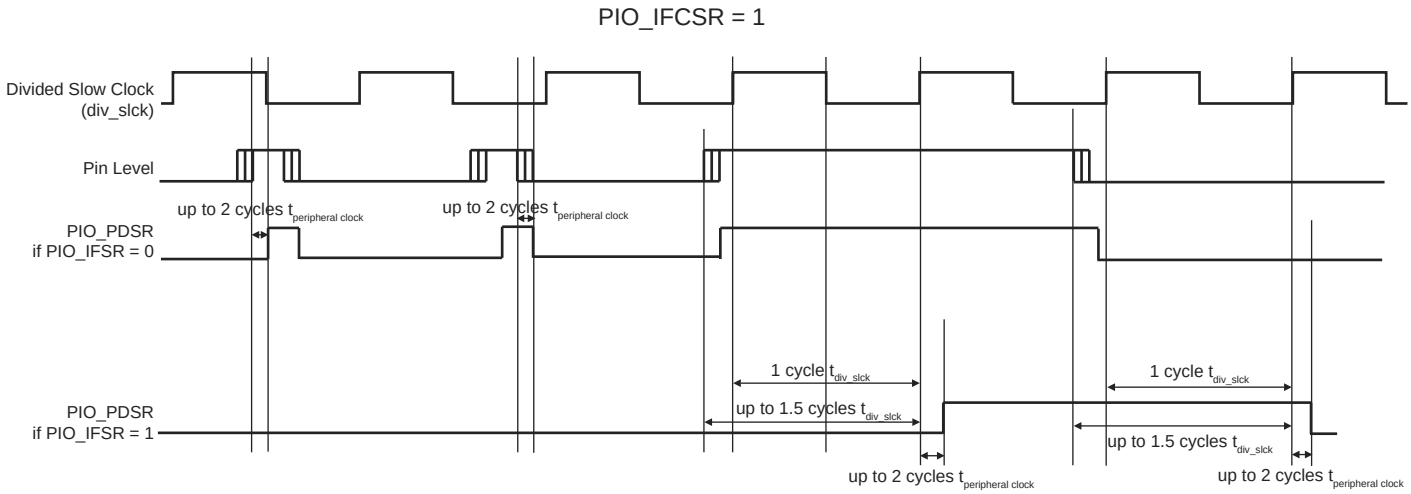



**Figure 32-5. Input Debouncing Filter Timing**




### 32.5.10 Input Edge/Level Interrupt

The PIO Controller can be programmed to generate an interrupt when it detects an edge or a level on an I/O line. The Input Edge/Level interrupt is controlled by writing the Interrupt Enable Register (PIO_IER) and the Interrupt Disable Register (PIO_IDR), which enable and disable the input change interrupt respectively by setting and clearing the corresponding bit in the Interrupt Mask Register (PIO_IMR). As input change detection is possible only by comparing two successive samplings of the input of the I/O line, the peripheral clock must be enabled. The Input Change interrupt is available regardless of the configuration of the I/O line, i.e., configured as an input only, controlled by the PIO Controller or assigned to a peripheral function.

By default, the interrupt can be generated at any time an edge is detected on the input.

Some additional interrupt modes can be enabled/disabled by writing in the Additional Interrupt Modes Enable Register (PIO_AIMER) and Additional Interrupt Modes Disable Register (PIO_AIMDR). The current state of this selection can be read through the Additional Interrupt Modes Mask Register (PIO_AIMMR).

### 36.7.11 USART Channel Status Register

**Name:** US_CSR

**Address:** 0x40024014 (0), 0x40028014 (1), 0x4002C014 (2), 0x40030014 (3), 0x40034014 (4)

**Access:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | MANERR |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CTS | – | – | – | CTSIC | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | NACK | RXBUFF | TXBUFE | ITER | TXEMPTY | TIMEOUT |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | RXBRK | TXRDY | RXRDY |

For SPI specific configuration, see Section 36.7.12 ”USART Channel Status Register (SPI_MODE)”.

- **RXRDY: Receiver Ready (cleared by reading US_RHR)**

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

- **TXRDY: Transmitter Ready (cleared by writing US_THR)**

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested, or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **RXBRK: Break Received/End of Break (cleared by writing a one to bit US_CR.RSTSTA)**

0: No break received or end of break detected since the last RSTSTA.

1: Break received or end of break detected since the last RSTSTA.

- **ENDRX: End of RX Buffer (cleared by writing US_RCR or US_RNCR)**

0: The Receive Counter Register has not reached 0 since the last write in US_RCR or US_RNCR(1).

1: The Receive Counter Register has reached 0 since the last write in US_RCR or US_RNCR(1).

- **ENDTX: End of TX Buffer (cleared by writing US_TCR or US_TNCR)**

0: The Transmit Counter Register has not reached 0 since the last write in US_TCR or US_TNCR(1).

1: The Transmit Counter Register has reached 0 since the last write in US_TCR or US_TNCR(1).

- **OVRE: Overrun Error (cleared by writing a one to bit US_CR.RSTSTA)**

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

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

### 37.7.4 TC Stepper Motor Mode Register

**Name:** TC_SMMRx [x=0..2]

**Address:** 0x40010008 (0)[0], 0x40010048 (0)[1], 0x40010088 (0)[2], 0x40014008 (1)[0], 0x40014048 (1)[1], 0x40014088 (1)[2]

**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 |
| – | – | – | – | – | – | DOWN | GCEN |

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

- **GCEN: Gray Count Enable**

0: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by internal counter of channel x.

1: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by a 2-bit gray counter.

- **DOWN: Down Count**

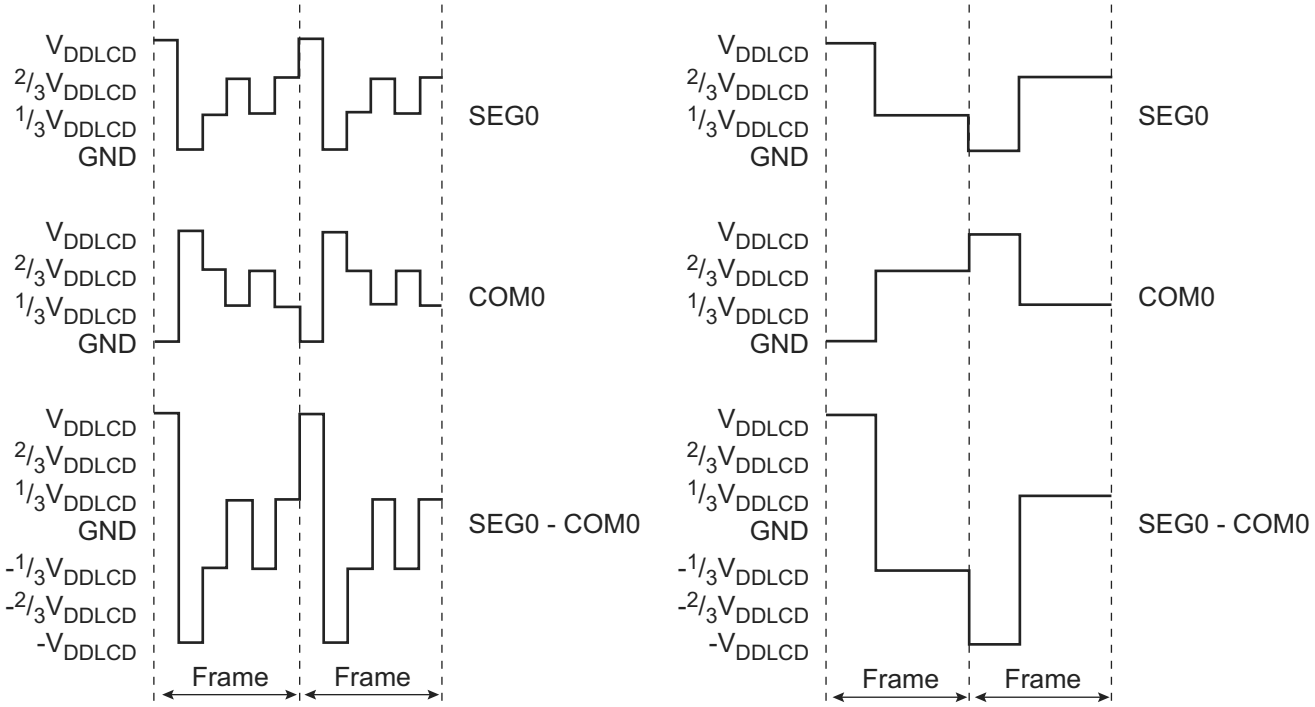
0: Up counter.

1: Down counter.

#### 39.6.2.5 Low Power Waveform

To reduce toggle activity and hence power consumption, a low power waveform can be selected by writing LPMODE to one. The default and low power waveform is shown in Figure 39-7 for 1/3 duty and 1/3 bias. For other selections of duty and bias, the effect is similar.

**Figure 39-7. Default and Low Power Waveform**




Note: Refer to the LCD specification to verify that low power waveforms are supported.

#### 39.6.2.6 Frame Rate

The Frame Rate register (SLCDC_FRR) enables the generation of the frequency used by the SLCDC. It is done by a prescaler (division by 8, 16, 32, 64, 128, 256, 512 and 1024) followed by a finer divider (division by 1, 2, 3, 4, 5, 6, 7 or 8).

To calculate the needed frame frequency, the equation below must be used:

$$f_{frame} = \frac{fSLCK}{(PRESC \cdot DIV \cdot NCOM)}$$

where:

$f_{SLCK}$ = slow clock frequency

$f_{frame}$ = frame frequency

PRESC = prescaler value (8, 16, 32, 64, 128, 256, 512 or 1024)

DIV = divider value (1, 2, 3, 4, 5, 6, 7, or 8)

NCOM = depends of number of commons and is defined in Table 39-5.

NCOM is automatically provided by the SLCDC.

For example, if COMSEL is programmed to 0 (1 common terminal on display device), the SLCDC introduces a divider by 16 so that NCOM = 16. If COMSEL is programmed to 3 (3 common terminals on display device), the

#### 43.5.2.3 ICM Region Control Structure Member

**Name:** ICM_RCTRL

**Address**: ICM_DSCR+0x008+RID*(0x10)

**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 |
| TRSIZE | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TRSIZE | | | | | | | |

- **TRSIZE: Transfer Size for the Current Chunk of Data**

ICM performs a transfer of (TRSIZE + 1) blocks of 512 bits.

*Master Read Mode*

$$f_{SPCK}Max = \frac{1}{SPI_0(orSPI_3) + t_{valid}}$$

$t_{valid}$ is the slave time response to output data after detecting an SPCK edge.

For a non-volatile memory with $t_{valid}$ (or $t_v$) = 5 ns, $f_{SPCK}$max = 40 MHz at $V_{DDIO}$ = 3.3 V.

*Slave Read Mode*

In Slave mode, SPCK is the input clock for the SPI. The max SPCK frequency is given by setup and hold timings $SPI_7$/$SPI_8$(or $SPI_{10}$/$SPI_{11}$). Since this gives a frequency well above the pad limit, the limit in Slave Read mode is given by the SPCK pad.

*Slave Write Mode*

$$f_{SPCK}Max = \frac{1}{2x(SPI_{6max}(orSPI_{9max}) + t_{setup})}$$

$t_{setup}$ is the setup time from the master before sampling data.

### 46.4.4 SMC Timings

Timings are given in the following domains:

- 1.8V domain: VDDIO from 1.65V to 1.95V, maximum external capacitor = 10 pF
- 3.3V domain: VDDIO from 2.85V to 3.6V, maximum external capacitor = 10 pF

Timings are given assuming a capacitance load on data, control and address pads.

In the tables that follow, $t_{CPMCK}$ is the MCK period.

#### 46.4.4.1 Read Timings

**Table 46-11. SMC Read Signals - NRD Controlled (READ_MODE = 1)**

| Symbol | Parameter | Min | | Max | | Unit |
|---|---|---|---|---|---|---|
| | VDDIO Supply | 1.8V(1) | 3.3V(2) | – | – | |
| **NO HOLD SETTINGS (nrd hold = 0)** | | | | | | |
| $SMC_1$ | Data setup before NRD high | 18 | 18 | – | – | ns |
| $SMC_2$ | Data hold after NRD high | -6.7 | -6.7 | – | – | ns |
| **HOLD SETTINGS (nrd hold ≠ 0)** | | | | | | |
| $SMC_3$ | Data setup before NRD high | 11.7 | 11.7 | – | – | ns |
| $SMC_4$ | Data hold after NRD high | -6.5 | -6.5 | – | – | ns |
| **HOLD or NO HOLD SETTINGS (nrd hold ≠ 0, nrd hold = 0)** | | | | | | |
| $SMC_5$ | NBS0/A0, NBS1, A1 - A23 Valid before NRD high | (nrd setup + nrd pulse) * $t_{CPMCK}$ - 5.2 | (nrd setup + nrd pulse) * $t_{CPMCK}$ - 5.2 | – | – | ns |
| $SMC_6$ | NCS low before NRD high | (nrd setup + nrd pulse - ncs rd setup) * $t_{CPMCK}$ - 1.1 | (nrd setup + nrd pulse - ncs rd setup) * $t_{CPMCK}$ - 1.1 | – | – | ns |
| $SMC_7$ | NRD pulse width | nrd pulse * $t_{CPMCK}$ - 3.2 | nrd pulse * $t_{CPMCK}$ - 3.2 | – | – | ns |

Notes: 1. 1.8V domain: VDDIO from 1.65V to 1.95V, maximum external capacitor = 10 pF.
2. 3.3V domain: VDDIO from 3.0V to 3.6V, maximum external capacitor = 10 pF.

**Table 46-12. SMC Read Signals - NCS Controlled (READ_MODE= 0)**

| Symbol | Parameter | Min | | Max | | Unit |
|---|---|---|---|---|---|---|
| | VDDIO supply | 1.8V(1) | 3.3V(2) | – | – | |
| **NO HOLD SETTINGS (ncs rd hold = 0)** | | | | | | |
| $SMC_8$ | Data setup before NCS high | 31.3 | 31.3 | – | – | ns |
| $SMC_9$ | Data hold after NCS high | -6.9 | -6.9 | – | – | ns |
| **HOLD SETTINGS (ncs rd hold ≠ 0)** | | | | | | |
| $SMC_{10}$ | Data setup before NCS high | 23 | 23 | – | – | ns |
| $SMC_{11}$ | Data hold after NCS high | -6.6 | -6.6 | – | – | ns |
| **HOLD or NO HOLD SETTINGS (ncs rd hold ≠ 0, ncs rd hold = 0)** | | | | | | |
| $SMC_{12}$ | NBS0/A0, NBS1, A1–A23 valid before NCS high | (ncs rd setup + ncs rd pulse)* $t_{CPMCK}$ - 4.9 | (ncs rd setup + ncs rd pulse)* $t_{CPMCK}$ - 4.9 | – | – | ns |
| $SMC_{13}$ | NRD low before NCS high | (ncs rd setup + ncs rd pulse - nrd setup)* $t_{CPMCK}$ - 1.5 | (ncs rd setup + ncs rd pulse - nrd setup)* $t_{CPMCK}$ - 1.5 | – | – | ns |
| $SMC_{14}$ | NCS pulse width | ncs rd pulse length * $t_{CPMCK}$ - 5.4 | ncs rd pulse length * $t_{CPMCK}$ - 5.4 | – | – | ns |

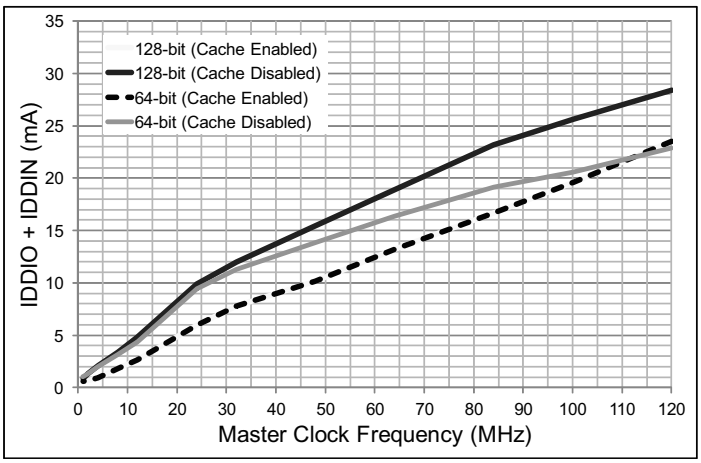
Notes: 1. 1.8V domain: VDDIO from 1.65V to 1.95V, maximum external capacitor = 10 pF.
2. 3.3V domain: VDDIO from 3.0V to 3.6V, maximum external capacitor = 10 pF.

**Table 46-71. SAM4CM32Test Setup 5 Current Consumption**

| Clock (MHz) | 128-bit Flash Access | | | | | | 64-bit Flash Access | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cache Enabled | | | Cache Disabled | | | Cache Enabled | | | Cache Disabled | | | |
| | IDD_IN (AMP1) | IDD_I0 (AMP2) | IDD_CORE (AMP3) | IDD_IN (AMP1) | IDD_I0 (AMP2) | IDD_CORE (AMP3) | IDD_IN (AMP1) | IDD_I0 (AMP2) | IDD_CORE (AMP3) | IDD_IN (AMP1) | IDD_I0 (AMP2) | IDD_CORE (AMP3) | |
| 120 | 26.6 | 0.40 | 23.3 | 29.9 | 2.1 | 26.6 | 26.3 | 0.42 | 23.1 | 24.4 | 1.6 | 21.2 | mA |
| 100 | 22.4 | 0.39 | 19.7 | 27.8 | 2.0 | 24.5 | 22.7 | 0.40 | 20.0 | 22.0 | 1.7 | 19.4 | |
| 84 | 18.9 | 0.38 | 16.7 | 24.1 | 1.9 | 21.9 | 19.4 | 0.39 | 17.1 | 19.9 | 1.7 | 17.7 | |
| 64 | 14.7 | 0.36 | 13.0 | 19.7 | 1.8 | 18.0 | 14.6 | 0.36 | 13.0 | 16.7 | 1.6 | 15.1 | |
| 48 | 11.6 | 0.34 | 10.4 | 15.3 | 1.6 | 14.0 | 11.6 | 0.34 | 10.4 | 14.4 | 1.5 | 13.2 | |
| 32 | 9.0 | 0.33 | 7.3 | 11.8 | 1.4 | 10.2 | 8.9 | 0.32 | 7.3 | 11.0 | 1.4 | 9.4 | |
| 24 | 6.5 | 0.32 | 5.3 | 10.0 | 1.4 | 8.7 | 6.5 | 0.31 | 5.2 | 8.2 | 1.3 | 7.5 | |
| 12 | 2.7 | 0.08 | 2.7 | 4.8 | 1.23 | 4.8 | 2.7 | 0.08 | 2.7 | 3.9 | 1.1 | 3.9 | |
| 8 | 1.9 | 0.06 | 1.8 | 2.9 | 0.99 | 2.9 | 2.2 | 0.06 | 2.2 | 3.0 | 1.1 | 2.9 | |
| 4 | 1.03 | 0.04 | 1.01 | 1.9 | 0.64 | 1.9 | 1.36 | 0.05 | 1.35 | 1.8 | 0.9 | 1.8 | |
| 2 | 0.95 | 0.03 | 0.94 | 1.23 | 0.45 | 1.24 | 0.95 | 0.04 | 0.94 | 0.86 | 0.71 | 0.86 | |
| 1 | 0.75 | 0.02 | 0.73 | 0.56 | 0.23 | 0.54 | 0.75 | 0.03 | 0.74 | 0.85 | 0.32 | 0.84 | |

**Figure 46-31. Typical Current Consumption in Active Mode (Test Setup 5)**

SAM4Cx4/8/16




SAM4Cx32