**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/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 | 52 |
| Program Memory Size | 2MB (2M x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 256K 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/atsam4cmp32cb-au |

5. Refer to Table 3-1 “Signal Description List”. Some anti-tamper pin pads are VDDIO-powered.
6. Fast RC Oscillator set to 4 MHz frequency.
7. Refer to PIO Controller Multiplexing tables in Section 11.4 “Peripheral Signal Multiplexing on I/O Lines”.

## 5.6 Wake-up Sources

Wake-up events allow the device to exit Backup mode. When a wake-up event is detected, the Supply Controller performs a sequence which automatically reenables the core power supply and all digital logic.

## 5.7 Fast Start-up

The SAM4CM allows the processor to restart in a few microseconds while the processor is in Wait mode or in Sleep mode. A fast start-up occurs upon detection of one of the wake-up inputs.

The fast restart circuitry is fully asynchronous and provides a fast start-up signal to the Power Management Controller. As soon as the fast start-up signal is asserted, the PMC automatically restarts the embedded 4/8/12 MHz Fast RC oscillator, switches the master clock on this 4 MHz clock and re-enables the processor clock.

# 6. Input/Output Lines

The SAM4CM has two types of input/output (I/O) lines—general-purpose I/Os (GPIO) and system I/Os. GPIOs have alternate functionality due to multiplexing capabilities of the PIO controllers. The same PIO line can be used whether in I/O mode or by the multiplexed peripheral. System I/Os include pins such as test pins, oscillators, erase or analog inputs.

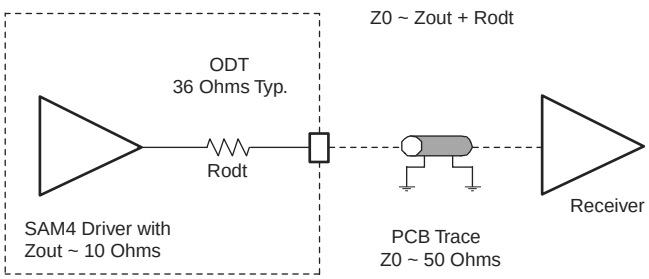
## 6.1 General-Purpose I/O Lines

General-purpose I/O (GPIO) lines are managed by PIO Controllers. All I/Os have several input or output modes such as pull-up or pull-down, input Schmitt triggers, multi-drive (open-drain), glitch filters, debouncing or input change interrupt. Programming of these modes is performed independently for each I/O line through the PIO controller user interface. Refer to Section 32. “Parallel Input/Output Controller (PIO)” for details.

The input/output buffers of the PIO lines are supplied through VDDIO power supply rail when used as GPIOs. When used as extra functions such as LCD or Analog modes, GPIO lines have either VDDLCD or VDDIN voltage range.

Each I/O line embeds an ODT (On-die Termination), shown in Figure 6-1 below. ODT consists of an internal series resistor termination scheme for impedance matching between the driver output (SAM4CM) and the PCB trace impedance preventing signal reflection. The series resistor helps to reduce IOs switching current (di/dt) thereby reducing EMI. It also decreases overshoot and undershoot (ringing) due to inductance of interconnect between devices or between boards. Finally, ODT helps diminish signal integrity issues.

**Figure 6-1. On-die Termination**



## 6.4 NRST Pin

The NRST pin is bidirectional. It is handled by the on-chip reset controller and can be driven low to provide a reset signal to the external components, or asserted low externally to reset the microcontroller. It resets the core and the peripherals, with the exception of the Backup region (RTC, RTT and Supply Controller). There is no constraint on the length of the reset pulse, and the Reset Controller can guarantee a minimum pulse length. The NRST pin integrates a permanent pull-up resistor to VDDIO of about 100 kΩ. By default, the NRST pin is configured as an input.

## 6.5 TMPx Pins: Anti-tamper Pins

Anti-tamper pins detect intrusion—for example, into a smart meter case. Upon detection through a tamper switch, automatic, asynchronous and immediate clear of registers in the backup area, and time stamping in the RTC are performed. Anti-tamper pins can be used in all modes. Date and number of tampering events are stored automatically. Anti-tampering events can be programmed so that half of the General-purpose Backup Registers (GPBR) are erased automatically. The TMP1 signal is referred to VDDIO, meaning that it is effective only if VDDIO is supplied, whereas TMP0 is in the VDDBU domain.

## 6.6 RTCOUT0 Pin

The RTCOUT0 pin shared in the PIO (supplied by VDDIO) can be used to generate waveforms from the RTC in order to take advantage of the RTC inherent prescalers while the RTC is the only powered circuitry (Low-power mode, Backup mode) or in any active mode. Entering Backup or low-power operating modes does not affect the waveform generation outputs (VDDIO still must be supplied). Anti-tampering pin detection can be synchronized with this signal.

Note: To use the RTCOUT0 signal during application development using JTAG-ICE interface, the programmer must use Serial Wire Debug (SWD) mode. In this case, the TDO pin is not used as a JTAG signal by the ICE interface.

## 6.7 Shutdown (SHDN) Pin

The SHDN pin designates the Backup mode of operation. When the device is in Backup mode, SHDN = 0. In any other mode, SHDN = 1 (VDDBU). This pin is designed to control the enable pin of the main external voltage regulator. When the device enters Backup mode, the SHDN pin disables the external voltage regulator and, upon the wake-up event, it re-enables the voltage regulator.

The SHDN pin is asserted low when the VROFF bit in the Supply Controller Control Register (SUPC_CR) is set to 1.

## 6.8 Force Wake-up (FWUP) Pin

The FWUP pin can be used as a wake-up source in all low-power modes as it is supplied by VDDBU.

## 6.9 ERASE Pin

The ERASE pin is used to reinitialize the Flash content (and some of its NVM bits) to an erased state (all bits read as logic level 1). The ERASE pin and the ROM code ensure an in-situ reprogrammability of the Flash content without the use of a debug tool. When the security bit is activated, the ERASE pin provides the capability to reprogram the Flash content. The ERASE pin integrates a pull-down resistor of about 100 kΩ into GND, so that it can be left unconnected for normal operations.

This pin is debounced by SLCK to improve the glitch tolerance. When the ERASE pin is tied high during less than 100 ms, it is not taken into account. The pin must be tied high during more than 220 ms to perform a Flash erase operation.

The ERASE pin is a system I/O pin and can be used as a standard I/O. At start-up, the ERASE pin is not configured as a PIO pin. If the ERASE pin is used as a standard I/O, the start-up level of this pin must be low to

#### 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.

#### 12.6.9.2 SBFX and UBFX

Signed Bit Field Extract and Unsigned Bit Field Extract.

Syntax

```
SBFX{cond} Rd, Rn, #lsb, #width
UBFX{cond} Rd, Rn, #lsb, #width
```

where:

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

Rd — is the destination register.

Rn — is the source register.

lsb — is the position of the least significant bit of the bitfield. *lsb* must be in the range 0 to 31.

width — is the width of the bitfield and must be in the range 1 to 32-*lsb*.

Operation

SBFX extracts a bitfield from one register, sign extends it to 32 bits, and writes the result to the destination register.

UBFX extracts a bitfield from one register, zero extends it to 32 bits, and writes the result to the destination register.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the flags.

Examples

```
SBFX  R0, R1, #20, #4  ; Extract bit 20 to bit 23 (4 bits) from R1 and sign
                       ; extend to 32 bits and then write the result to R0.
UBFX  R8, R11, #9, #10 ; Extract bit 9 to bit 18 (10 bits) from R11 and zero
                       ; extend to 32 bits and then write the result to R8.
```

#### 12.6.11.15 VMOV Scalar to ARM Core Register

Transfers one word of a doubleword floating-point register to an ARM core register.

Syntax

```
VMOV{cond} Rt, Dn[x]
```

where:

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

Rt — is the destination ARM core register.

Dn — is the 64-bit doubleword register.

x — Specifies which half of the doubleword register to use:
- If *x* is 0, use lower half of doubleword register
- If *x* is 1, use upper half of doubleword register.

Operation

This instruction transfers:

- One word from the upper or lower half of a doubleword floating-point register to an ARM core register.

Restrictions

*Rt* cannot be PC or SP.

Condition Flags

These instructions do not change the flags.

#### 12.6.12.9 SEV

Send Event.

Syntax

`SEV{cond}`

where:

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

Operation

SEV is a hint instruction that causes an event to be signaled to all processors within a multiprocessor system. It also sets the local event register to 1, see “Power Management”.

Condition Flags

This instruction does not change the flags.

Examples

```
SEV ; Send Event
```

#### 12.6.12.10 SVC

Supervisor Call.

Syntax

`SVC{cond} #imm`

where:

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

imm is an expression evaluating to an integer in the range 0-255 (8-bit value).

Operation

The SVC instruction causes the SVC exception.

*imm* is ignored by the processor. If required, it can be retrieved by the exception handler to determine what service is being requested.

Condition Flags

This instruction does not change the flags.

Examples

```
SVC  0x32  ; Supervisor Call (SVC handler can extract the immediate value
           ; by locating it via the stacked PC)
```

- **SYSRESETREQ: System Reset Request**

0: No system reset request.

1: Asserts a signal to the outer system that requests a reset.

This is intended to force a large system reset of all major components except for debug. This bit reads as 0.

- **VECTCLRACTIVE: Reserved for Debug use**

This bit reads as 0. When writing to the register, write a 0 to this bit, otherwise the behavior is unpredictable.

- **VECTRESET: Reserved for Debug use**

This bit reads as 0. When writing to the register, write a 0 to this bit, otherwise the behavior is unpredictable.

### 13.7.8 ID Code Register

Access: Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| VERSION | | | | PART NUMBER | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| PART NUMBER | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PART NUMBER | | | | MANUFACTURER IDENTITY | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MANUFACTURER IDENTITY | | | | | | | 1 |

- **VERSION[31:28]: Product Version Number**

Set to 0x0.

- **PART NUMBER[27:12]: Product Part Number**

| Chip Name | Chip ID |
|---|---|
| SAM4CM | 0x05B34 |

- **MANUFACTURER IDENTITY[11:1]**

Set to 0x01F.

- **Bit[0] Required by IEEE Std. 1149.1**

Set to 0x1.

| Chip Name | JTAG ID Code |
|---|---|
| SAM4CM | 0x05B3_403F |

## 16.4 Functional Description

The programmable 16-bit prescaler value can be configured through the RTPRES field in the “Real-time Timer Mode Register” (RTT_MR).

Configuring the RTPRES field value to 0x8000 (default value) corresponds to feeding the real-time counter with a 1Hz signal (if the slow clock is 32.768 kHz). The 32-bit counter can count up to $2^{32}$ seconds, corresponding to more than 136 years, then roll over to 0. Bit RTTINC in the “Real-time Timer Status Register” (RTT_SR) is set each time there is a prescaler roll-over (see Figure 16-2)

The real-time 32-bit counter can also be supplied by the 1Hz RTC clock. This mode is interesting when the RTC 1Hz is calibrated (CORRECTION field $\neq 0$ in RTC_MR) in order to guaranty the synchronism between RTC and RTT counters.

Setting the RTC1HZ bit in the RTT_MR drives the 32-bit RTT counter from the 1Hz RTC clock. In this mode, the RTPRES field has no effect on the 32-bit counter.

The prescaler roll-over generates an increment of the real-time timer counter if RTC1HZ = 0. Otherwise, if RTC1HZ = 1, the real-time timer counter is incremented every second. The RTTINC bit is set independently from the 32-bit counter increment.

The real-time timer can also be used as a free-running timer with a lower time-base. The best accuracy is achieved by writing RTPRES to 3 in RTT_MR.

Programming RTPRES to 1 or 2 is forbidden.

If the RTT is configured to trigger an interrupt, the interrupt occurs two slow clock cycles after reading the RTT_SR. To prevent several executions of the interrupt handler, the interrupt must be disabled in the interrupt handler and re-enabled when the RTT_SR is cleared.

The CRTV field can be read at any time in the “Real-time Timer Value Register” (RTT_VR). As this value can be updated asynchronously with the Master Clock, the CRTV field must be read twice at the same value to read a correct value.

The current value of the counter is compared with the value written in the “Real-time Timer Alarm Register” (RTT_AR). If the counter value matches the alarm, the ALMS bit in the RTT_SR is set. The RTT_AR is set to its maximum value (0xFFFF_FFFF) after a reset.

The ALMS flag is always a source of the RTT alarm signal that may be used to exit the system from low power modes (see Figure 16-1).

The alarm interrupt must be disabled (ALMIEN must be cleared in RTT_MR) when writing a new ALMV value in the RTT_AR.

The RTTINC bit can be used to start a periodic interrupt, the period being one second when the RTPRES field value = 0x8000 and the slow clock = 32.768 kHz.

The RTTINCIEN bit must be cleared prior to writing a new RTPRES value in the RTT_MR.

Reading the RTT_SR automatically clears the RTTINC and ALMS bits.

Writing the RTTRST bit in the RTT_MR immediately reloads and restarts the clock divider with the new programmed value. This also resets the 32-bit counter.

When not used, the Real-time Timer can be disabled in order to suppress dynamic power consumption in this module. This can be achieved by setting the RTTDIS bit in the RTT_MR.

## 20.6 Supply Controller (SUPC) User Interface

The user interface of the SUPC is part of the System Controller user interface.

### 20.6.1 System Controller (SYSC) User Interface

**Table 20-1. System Controller Peripheral Offsets**

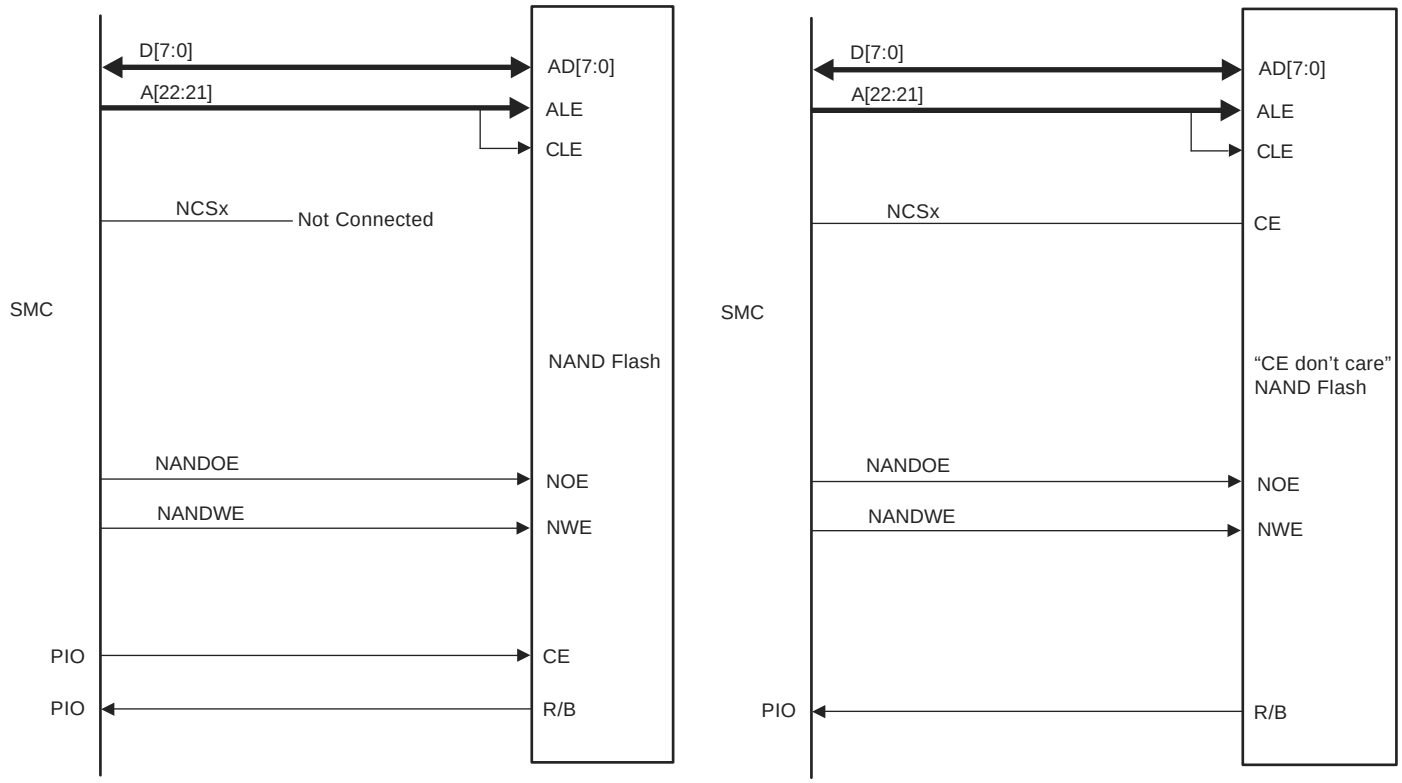
| Offset | System Controller Peripheral | Name |
|---|---|---|
| 0x00–0x0c | Reset Controller | RSTC |
| 0x10–0x2C | Supply Controller | SUPC |
| 0x30–0x3C | Real Time Timer | RTT |
| 0x50–0x5C | Watchdog Timer | WDT |
| 0x60–0x8C | Real Time Clock | RTC |
| 0x90–0xDC | General Purpose Backup Register | GPBR |
| 0xE0 | Reserved | – |
| 0xE4 | Write Protection Mode Register | SYSC_WPMR |
| 0xE8–0xF8 | Reserved | – |
| 0xFC | Reserved | – |
| 0x100–0x10C | Reinforced Safety Watchdog Timer | RSWDT |
| 0x110–0x124 | Time Stamping Registers | RTC |

### 20.6.2 Supply Controller (SUPC) User Interface

**Table 20-2. Register Mapping**

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Supply Controller Control Register | SUPC_CR | Write-only | – |
| 0x04 | Supply Controller Supply Monitor Mode Register | SUPC_SMMR | Read/Write | 0x0000_0000 |
| 0x08 | Supply Controller Mode Register | SUPC_MR | Read/Write | 0x0000_DA00 |
| 0x0C | Supply Controller Wakeup Mode Register | SUPC_WUMR | Read/Write | 0x0000_0000 |
| 0x10 | Supply Controller Wakeup Inputs Register | SUPC_WUIR | Read/Write | 0x0000_0000 |
| 0x14 | Supply Controller Status Register | SUPC_SR | Read-only | 0x0000_0000 |
| 0x18 | Reserved | – | – | – |
| 0xFC | Reserved | – | – | – |

**Figure 27-6. Standard and “CE don’t care” NAND Flash Application Examples**




## 27.8 Application Example

### 27.8.1 Implementation Examples

Hardware configurations are given for illustration only. The user should refer to the manufacturer web site to check for memory device availability.

For hardware implementation examples, refer to the evaluation kit schematics for this microcontroller, which show examples of a connection to an LCD module and NAND Flash.

### 28.5.4 Transmit Counter Register

**Name:** PERIPH_TCR

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

- **TXCTR: Transmit Counter Register**

TXCTR must be set to transmit buffer size.

When a half-duplex peripheral is connected to the PDC, RXCTR = TXCTR.

0: Stops peripheral data transfer to the transmitter.

1–65535: Starts peripheral data transfer if the corresponding channel is active.

5. Check the main clock frequency:

   This main clock frequency can be measured via CKGR_MCFR.

   Read CKGR_MCFR until the MAINFRDY field is set, after which the user can read the MAINF field in CKGR_MCFR by performing an additional read. This provides the number of main clock cycles that have been counted during a period of 16 slow clock cycles.

   If MAINF = 0, switch the MAINCK to the 4/8/12 MHz RC Oscillator by clearing MOSCSEL in CKGR_MOR. If MAINF ≠ 0, proceed to Step 6.

6. Set PLLx and Divider (if not required, proceed to Step 7.):

   In the names PLLx, DIVx, MULx, LOCKx, PLLxCOUNT, and CKGR_PLLxR, 'x' represents A or B.

   All parameters needed to configure PLLx and the divider are located in CKGR_PLLxR.

   The DIVx field is used to control the divider itself. This parameter can be programmed between 0 and 127. Divider output is divider input divided by DIVx parameter. By default, DIVx field is cleared which means that the divider and PLLx are turned off.

   The MULx field is the PLLx multiplier factor. This parameter can be programmed between 0 and 254. If MULx is cleared, PLLx will be turned off, otherwise the PLLx output frequency is PLLx input frequency multiplied by (MULx + 1).

   The PLLxCOUNT field specifies the number of slow clock cycles before the LOCKx bit is set in the PMC_SR after CKGR_PLLxR has been written.

   Once CKGR_PLLxR has been written, the user must wait for the LOCKx bit to be set in the PMC_SR. This can be done either by polling LOCKx in PMC_SR or by waiting for the interrupt line to be raised if the associated interrupt source (LOCKx) has been enabled in PMC_IER. All fields in CKGR_PLLxR can be programmed in a single write operation. If at some stage one of the following parameters, MULx or DIVx is modified, the LOCKx bit goes low to indicate that PLLx is not yet ready. When PLLx is locked, LOCKx is set again. The user must wait for the LOCKx bit to be set before using the PLLx output clock.

7. Select the master clock and processor clock

   The master clock and the processor clock are configurable via PMC_MCKR.

   The CSS field is used to select the clock source of the master clock and processor clock dividers. By default, the selected clock source is the main clock.

   The PRES field is used to define the processor clock and master clock prescaler. The user can choose between different values (1, 2, 3, 4, 8, 16, 32, 64). Prescaler output is the selected clock source frequency divided by the PRES value.

   Once the PMC_MCKR has been written, the user must wait for the MCKRDY bit to be set in the PMC_SR. This can be done either by polling MCKRDY in PMC_SR or by waiting for the interrupt line to be raised if the associated interrupt source (MCKRDY) has been enabled in PMC_IER. PMC_MCKR must not be programmed in a single write operation. The programming sequence for PMC_MCKR is as follows:

- If a new value for CSS field corresponds to PLL clock,
  - Program the PRES field in PMC_MCKR.
  - Wait for the MCKRDY bit to be set in PMC_SR.
  - Program the CSS field in PMC_MCKR.
  - Wait for the MCKRDY bit to be set in PMC_SR.
- If a new value for CSS field corresponds to main clock or slow clock,
  - Program the CSS field in PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
  - Program the PRES field in PMC_MCKR.

### 33.7.3 Master Mode Operations

When configured in Master mode, the SPI operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register (SPI_TDR) and the Receive Data Register (SPI_RDR), and a single shift register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer starts when the processor writes to the SPI_TDR. The written data is immediately transferred in the Shift register and the transfer on the SPI bus starts. While the data in the Shift register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift register. Data cannot be loaded in the SPI_RDR without transmitting data. If there is no data to transmit, dummy data can be used (SPI_TDR filled with ones). When the SPI_MR.WDRBT bit is set, new data cannot be transmitted if the SPI_RDR has not been read. If Receiving mode is not required, for example when communicating with a slave receiver only (such as an LCD), the receive status flags in the SPI Status register (SPI_SR) can be discarded.

Before writing the SPI_TDR, the PCS field in the SPI_MR must be set in order to select a slave.

If new data is written in the SPI_TDR during the transfer, it is kept in the SPI_TDR until the current transfer is completed. Then, the received data is transferred from the Shift register to the SPI_RDR, the data in the SPI_TDR is loaded in the Shift register and a new transfer starts.

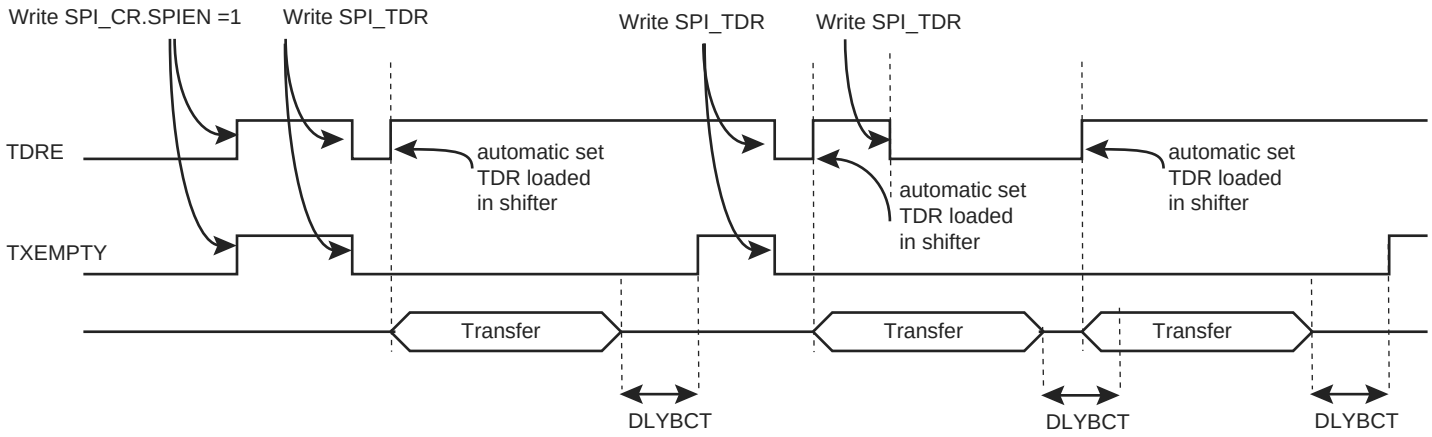
As soon as the SPI_TDR is written, the Transmit Data Register Empty (TDRE) flag in the SPI_SR is cleared. When the data written in the SPI_TDR is loaded into the Shift register, the TDRE flag in the SPI_SR is set. The TDRE bit is used to trigger the Transmit PDC channel.

See Figure 33-5.

The end of transfer is indicated by the TXEMPTY flag in the SPI_SR. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of this delay. The peripheral clock can be switched off at this time.

Note: When the SPI is enabled, the TDRE and TXEMPTY flags are set.

**Figure 33-5. TDRE and TXEMPTY flag behavior**




The transfer of received data from the Shift register to the SPI_RDR is indicated by the Receive Data Register Full (RDRF) bit in the SPI_SR. When the received data is read, the RDRF bit is cleared.

If the SPI_RDR has not been read before new data is received, the Overrun Error (OVRES) bit in the SPI_SR is set. As long as this flag is set, data is loaded in the SPI_RDR. The user has to read the SPI_SR to clear the OVRES bit.

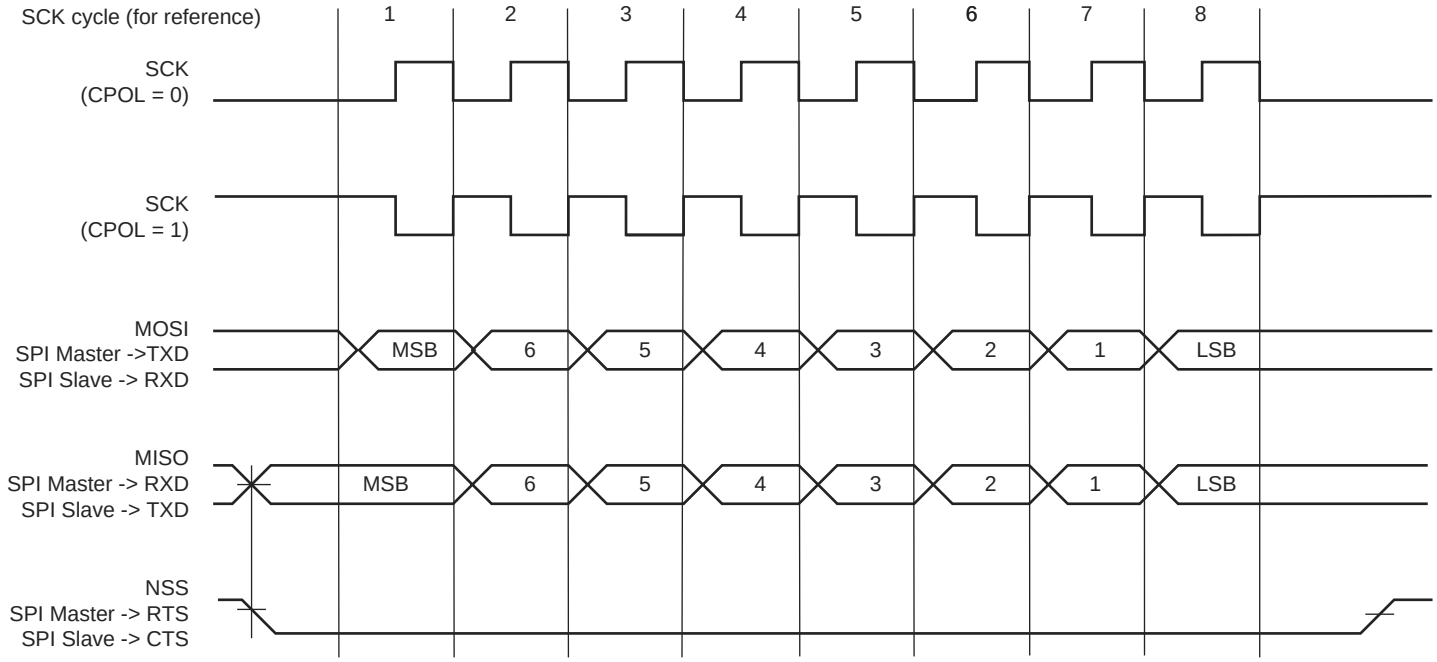
Figure 33-6 shows a block diagram of the SPI when operating in Master mode. Figure 33-7 shows a flow chart describing how transfers are handled.

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the US_MR. The clock phase is programmed with the CPHA bit. These two parameters determine the edges of the clock signal upon which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

**Table 36-13. SPI Bus Protocol Mode**

| SPI Bus Protocol Mode | CPOL | CPHA |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |

**Figure 36-37. SPI Transfer Format (CPHA = 1, 8 bits per transfer)**

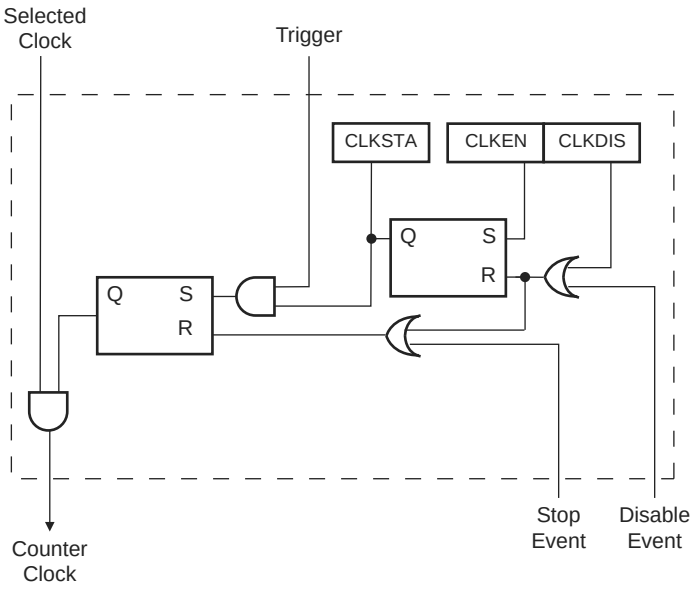


### 37.6.4 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 37-4.

- The clock can be enabled or disabled by the user with the CLKEN and the CLKDIS commands in the TC Channel Control Register (TC_CCR). In Capture mode it can be disabled by an RB load event if LDBDIS is set to 1 in the TC_CMR. In Waveform mode, it can be disabled by an RC Compare event if CPCDIS is set to 1 in TC_CMR. When disabled, the start or the stop actions have no effect: only a CLKEN command in the TC_CCR can re-enable the clock. When the clock is enabled, the CLKSTA bit is set in the TC_SR.
- The clock can also be started or stopped: a trigger (software, synchro, external or compare) always starts the clock. The clock can be stopped by an RB load event in Capture mode (LDBSTOP = 1 in TC_CMR) or an RC compare event in Waveform mode (CPCSTOP = 1 in TC_CMR). The start and the stop commands are effective only if the clock is enabled.

**Figure 37-4. Clock Control**




### 37.6.5 Operating Modes

Each channel can operate independently in two different modes:

- Capture mode provides measurement on signals.
- Waveform mode provides wave generation.

The TC operating mode is programmed with the WAVE bit in the TC_CMR.

In Capture mode, TIOA and TIOB are configured as inputs.

In Waveform mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

### 37.6.6 Trigger

A trigger resets the counter and starts the counter clock. Three types of triggers are common to both modes, and a fourth external trigger is available to each mode.

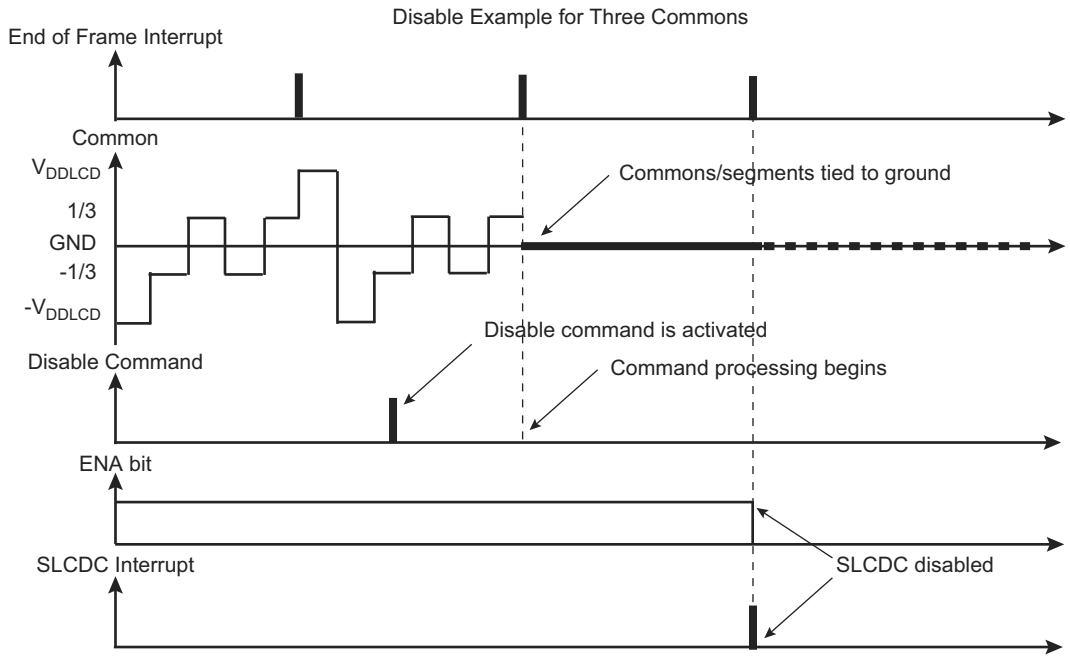
Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

- **CPD: Channel Update Period**

0 = Writing to the PWM_CUPDx will modify the duty cycle at the next period start event.

1 = Writing to the PWM_CUPDx will modify the period at the next period start event.

**Figure 39-10. Disabling Sequence**

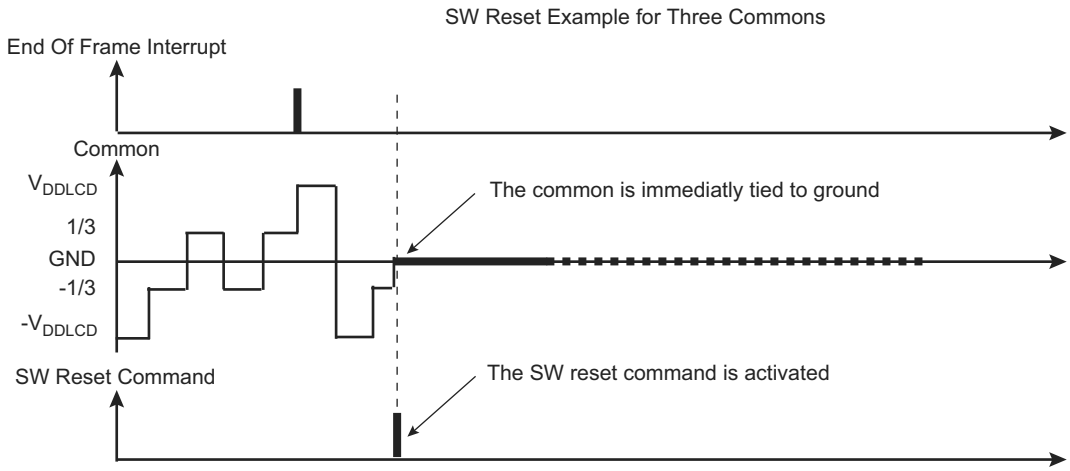



### 39.6.7.2 Software Reset

When the SLCDC software reset command is activated during a frame, it is immediately processed and all commons and segments are tied to ground.

Note that in the case of a software reset, the disable interrupt is not asserted.

**Figure 39-11. Software Reset**



### 40.7.9 ADC Interrupt Disable Register

**Name:** ADC_IDR

**Address:** 0x40038028

**Access:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | RXBUFF | ENDRX | COMPE | GOVRE | DRDY |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | TEMPCHG | – | – | - |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EOC7 | EOC6 | – | – | EOC3 | EOC2 | EOC1 | EOC0 |

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

0: No effect.

1: Disables the corresponding interrupt.

- **EOCx: End of Conversion Interrupt Disable x**
- **TEMPCHG: Temperature Change Interrupt Disable**
- **DRDY: Data Ready Interrupt Disable**
- **GOVRE: General Overrun Error Interrupt Disable**
- **COMPE: Comparison Event Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**

# 52. SAM4CM16/8/4 Errata Revision C (MRL C) Parts

## 52.1 Device Identification

The following errata apply to the devices listed in Table 51-1.

**Table 52-1. Device List**

| Device Marking | Chip ID |
|---|---|
| ATSAM4CMP16CC-AU | 0xA64C_0CE2 |
| ATSAM4CMP16CC-AUR | 0xA64C_0CE2 |
| ATSAM4CMP8CC-AU | 0xA64C_0AE2 |
| ATSAM4CMP8CC-AUR | 0xA64C_0AE2 |
| ATSAM4CMS16CC-AU | 0xA64C_0CE2 |
| ATSAM4CMS16CC-AUR | 0xA64C_0CE2 |
| ATSAM4CMS8CC-AU | 0xA64C_0AE2 |
| ATSAM4CMS8CC-AUR | 0xA64C_0AE2 |
| ATSAM4CMS4CC-AU | 0xA64C_0CE6 |
| ATSAM4CMS4CC-AUR | 0xA64C_0CE6 |

## 52.2 Supply Controller (SUPC)

### 52.2.1 SUPC: Supply Monitor (SM) on VDDIO

The Supply Monitor (SM) Sampling mode reducing the average current consumption on VDDIO is not functional.

**Problem Fix/Workaround**

Use the Supply Monitor in Continuous mode only.

### 52.2.2 SUPC: Core Voltage Regulator Standby Mode Control

The Core Voltage Regulator Standby mode controlled by the ONREG bit in SUPC_MR is not functional. This does not prevent to power VDDCORE and VDDPL by using an external voltage regulator.

**Problem Fix/Workaround**

None. Do not use the ONREG Bit.

### 52.2.3 SUPC: Core Brownout Detector. Unpredictable Behavior if BOD is Disabled, VDDCORE is Lost and VDDIO is Powered

In Active mode or in Wait mode, if the Brownout Detector (BOD) is disabled (SUPC_MR: BODDIS=1) and power is lost on VDDCORE while VDDIO is powered, the device can be reset incorrectly and its behavior becomes then unpredictable.

**Problem Fix/Workaround**

When the Brownout Detector is disabled in Active or in Wait mode, VDDCORE must be always powered.