**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 | 52 |
| 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/atsam4cmp16ca-aur |

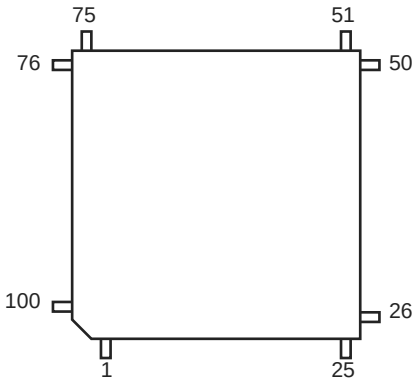
# 4. Package and Pinout

## 4.1 100-lead LQFP Package Outline

The 100-lead LQFP package has a 0.5 mm ball pitch and respects Green standards.

Figure 4-1 shows the orientation of the 100-lead LQFP package. Refer to Figure 47-1 “100-lead LQFP Package Drawing”.

**Figure 4-1. Orientation of the 100-lead LQFP Package**



#### 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)
```

- **MLSPERR: MemManage During Lazy State Preservation**

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

0: No MemManage fault occurred during the floating-point lazy state preservation.

1: A MemManage fault occurred during the floating-point lazy state preservation.

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

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

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

1: SCB_MMFAR holds a valid fault address.

If a memory management fault occurs and is escalated to a hard fault because of priority, the hard fault handler must set this bit to 0. This prevents problems on return to a stacked active memory management fault handler whose SCB_MMFAR value has been overwritten.

- **IBUSERR: Instruction Bus Error**

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

0: No instruction bus error.

1: Instruction bus error.

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

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

- **PRECISERR: Precise Data Bus Error**

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

0: No precise data bus error.

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

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

- **IMPRECISERR: Imprecise Data Bus Error**

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

0: No imprecise data bus error.

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

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

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

- **UNSTKERR: Bus Fault on Unstacking for a Return From Exception**

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

0: No unstacking fault.

1: Unstack for an exception return has caused one or more bus faults.

# 17. Real-time Clock (RTC)

## 17.1 Description

The Real-time Clock (RTC) peripheral is designed for very low power consumption. For optimal functionality, the RTC requires an accurate external 32.768 kHz clock, which can be provided by a crystal oscillator.

It combines a complete time-of-day clock with alarm and a Gregorian or Persian calendar, complemented by a programmable periodic interrupt. The alarm and calendar registers are accessed by a 32-bit data bus.

The time and calendar values are coded in binary-coded decimal (BCD) format. The time format can be 24-hour mode or 12-hour mode with an AM/PM indicator.

Updating time and calendar fields and configuring the alarm fields are performed by a parallel capture on the 32-bit data bus. An entry control is performed to avoid loading registers with incompatible BCD format data or with an incompatible date according to the current month/year/century.

A clock divider calibration circuitry can be used to compensate for crystal oscillator frequency variations.

An RTC output can be programmed to generate several waveforms, including a prescaled clock derived from 32.768 kHz.

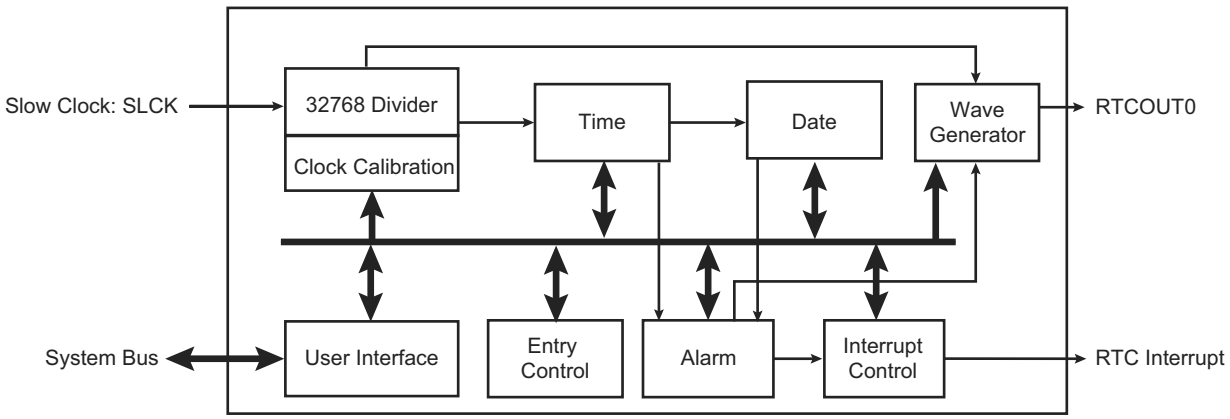
Timestamping capability reports the first and last occurrences of tamper events.

## 17.2 Embedded Characteristics

- Full Asynchronous Design for Ultra Low Power Consumption
- Gregorian and Persian Modes Supported
- Programmable Periodic Interrupt
- Safety/security Features:
  - Valid Time and Date Programming Check
  - On-The-Fly Time and Date Validity Check
- Counters Calibration Circuitry to Compensate for Crystal Oscillator Variations
- Waveform Generation
- Tamper Timestamping Registers
- Register Write Protection

## 17.3 Block Diagram

**Figure 17-1. Real-time Clock Block Diagram**



#### 27.11.3.2 Slow Clock Mode Transition

A Reload Configuration Wait State is also inserted when the Slow Clock mode is entered or exited, after the end of the current transfer (see Section 27.14 ”Slow Clock Mode”).

### 27.11.4 Read to Write Wait State

Due to an internal mechanism, a wait cycle is always inserted between consecutive read and write SMC accesses.

This wait cycle is referred to as a read to write wait state in this document.

This wait cycle is applied in addition to chip select and reload user configuration wait states when they are to be inserted. See Figure 27-15.

## 27.12 Data Float Wait States

Some memory devices are slow to release the external bus. For such devices, it is necessary to add wait states (data float wait states) after a read access:

- before starting a read access to a different external memory
- before starting a write access to the same device or to a different external one.

The Data Float Output Time ($t_{DF}$) for each external memory device is programmed in the TDF_CYCLES field of the SMC_MODE register for the corresponding chip select. The value of TDF_CYCLES indicates the number of data float wait cycles (between 0 and 15) before the external device releases the bus, and represents the time allowed for the data output to go to high impedance after the memory is disabled.

Data float wait states do not delay internal memory accesses. Hence, a single access to an external memory with long $t_{DF}$ will not slow down the execution of a program from internal memory.

The data float wait states management depends on the READ_MODE and the TDF_MODE fields of the SMC_MODE register for the corresponding chip select.

### 27.12.1 READ_MODE

Setting the READ_MODE to 1 indicates to the SMC that the NRD signal is responsible for turning off the tri-state buffers of the external memory device. The Data Float Period then begins after the rising edge of the NRD signal and lasts TDF_CYCLES MCK cycles.

When the read operation is controlled by the NCS signal (READ_MODE = 0), the TDF field gives the number of MCK cycles during which the data bus remains busy after the rising edge of NCS.

Figure 27-19 illustrates the Data Float Period in NRD-controlled mode (READ_MODE =1), assuming a data float period of 2 cycles (TDF_CYCLES = 2). Figure 27-20 shows the read operation when controlled by NCS (READ_MODE = 0) and the TDF_CYCLES parameter equals 3.

## 28.4 Functional Description

### 28.4.1 Configuration

The PDC channel user interface enables the user to configure and control data transfers for each channel. The user interface of each PDC channel is integrated into the associated peripheral user interface.

The user interface of a serial peripheral, whether it is full- or half-duplex, contains four 32-bit pointers (RPR, RNPR, TPR, TNPR) and four 16-bit counter registers (RCR, RNCR, TCR, TNCR). However, the transmit and receive parts of each type are programmed differently: the transmit and receive parts of a full-duplex peripheral can be programmed at the same time, whereas only one part (transmit or receive) of a half-duplex peripheral can be programmed at a time.

32-bit pointers define the access location in memory for the current and next transfer, whether it is for read (transmit) or write (receive). 16-bit counters define the size of the current and next transfers. It is possible, at any moment, to read the number of transfers remaining for each channel.

The PDC has dedicated status registers which indicate if the transfer is enabled or disabled for each channel. The status for each channel is located in the associated peripheral status register. Transfers can be enabled and/or disabled by setting TXTEN/TXTDIS and RXTEN/RXTDIS in the peripheral's Transfer Control register.

At the end of a transfer, the PDC channel sends status flags to its associated peripheral. These flags are visible in the peripheral Status register (ENDRX, ENDTX, RXBUFF, and TXBUFE). Refer to Section 28.4.3 and to the associated peripheral user interface.

The peripheral where a PDC transfer is configured must have its peripheral clock enabled. The peripheral clock must be also enabled to access the PDC register set associated to this peripheral.

### 28.4.2 Memory Pointers

Each full-duplex peripheral is connected to the PDC by a receive channel and a transmit channel. Both channels have 32-bit memory pointers that point to a receive area and to a transmit area, respectively, in the target memory.

Each half-duplex peripheral is connected to the PDC by a bidirectional channel. This channel has two 32-bit memory pointers, one for current transfer and the other for next transfer. These pointers point to transmit or receive data depending on the operating mode of the peripheral.

Depending on the type of transfer (byte, half-word or word), the memory pointer is incremented respectively by 1, 2 or 4 bytes.

If a memory pointer address changes in the middle of a transfer, the PDC channel continues operating using the new address.

### 28.4.3 Transfer Counters

Each channel has two 16-bit counters, one for the current transfer and the one for the next transfer. These counters define the size of data to be transferred by the channel. The current transfer counter is decremented first as the data addressed by the current memory pointer starts to be transferred. When the current transfer counter reaches zero, the channel checks its next transfer counter. If the value of the next counter is zero, the channel stops transferring data and sets the appropriate flag. If the next counter value is greater than zero, the values of the next pointer/next counter are copied into the current pointer/current counter and the channel resumes the transfer, whereas next pointer/next counter get zero/zero as values.At the end of this transfer, the PDC channel sets the appropriate flags in the Peripheral Status register.

The following list gives an overview of how status register flags behave depending on the counters' values:

- ENDRX flag is set when the PDC Receive Counter Register (PERIPH_RCR) reaches zero.
- RXBUFF flag is set when both PERIPH_RCR and the PDC Receive Next Counter Register (PERIPH_RNCR) reach zero.

The software can disable or enable the 4/8/12 MHz RC oscillator with the MOSCRCEN bit in the Clock Generator Main Oscillator Register (CKGR_MOR).

The output frequency of the RC oscillator can be selected among 4/8/12 MHz. The selection is done via the CKGR_MOR.MOSCRCF field. When changing the frequency selection, the MOSCRCS bit in the Power Management Controller Status Register (PMC_SR) is automatically cleared and MAINCK is stopped until the oscillator is stabilized. Once the oscillator is stabilized, MAINCK restarts and PMC_SR.MOSCRCS is set.

When disabling the main clock by clearing the CKGR_MOR.MOSCRCEN bit, the PMC_SR.MOSCRCS bit is automatically cleared, indicating the main clock is off.

Setting the MOSCRCS bit in the Power Management Controller Interrupt Enable Register (PMC_IER) can trigger an interrupt to the processor.

When main clock (MAINCK) is not used to drive the processor and frequency monitor (SLCK or PLLACK is used instead), it is recommended to disable the 4/8/12 MHz RC oscillator and 3 to 20 MHz crystal oscillator.

The CAL4, CAL8 and CAL12 values in the PMC Oscillator Calibration Register (PMC_OCR) are the default values set by Atmel during production. These values are stored in a specific Flash memory area different from the memory plane for code. These values cannot be modified by the user and cannot be erased by a Flash erase command or by the ERASE pin. Values written by the user application in PMC_OCR are reset after each power up or peripheral reset.

### 29.5.2 4/8/12 MHz RC Oscillator Clock Frequency Adjustment

It is possible for the user to adjust the 4/8/12 MHz RC oscillator frequency through PMC_OCR. By default, SEL4/8/12 bits are cleared, so the RC oscillator will be driven with Flash calibration bits which are programmed during chip production.

The user can adjust the trimming of the 4/8/12 MHz RC oscillator through this register. This can be used to compensate derating factors such as temperature and voltage, thus providing greater accuracy.

In order to calibrate the RC oscillator lower frequency, SEL4 bit must be set to 1 and a frequency value must be configured in the field CAL4. Likewise, SEL8/12 bit must be set to 1 and a trim value must be configured in the field CAL8/12 in order to adjust the other frequencies of the RC oscillator.

It is possible to adjust the RC oscillator frequency while operating from this clock. For example, when running on lowest frequency it is possible to change the CAL4 value if PMC_OCR.SEL4 bit is set.

At any time, it is possible to restart a measurement of the frequency of the selected clock via the RCMEAS bit in Main Clock Frequency Register (CKGR_MCFR). Thus, when CKGR_MCFR.MAINFRDY reads 1, another read access on CKGR_MCFR provides an image of the frequency on CKGR_MCFR.MAINF field. The software can calculate the error with an expected frequency and correct the CAL4 (or CAL8/CAL12) field accordingly. This may be used to compensate frequency drift due to derating factors such as temperature and/or voltage.

### 29.5.3 3 to 20 MHz Crystal or Ceramic Resonator-based Oscillator

After reset, the 3 to 20 MHz crystal or ceramic resonator-based oscillator is disabled and is not selected as the source of MAINCK.

As the source of MAINCK, the 3 to 20 MHz crystal or ceramic resonator-based oscillator provides a very precise frequency. The software enables or disables this oscillator in order to reduce power consumption via CKGR_MOR.MOSCXTEN.

When disabling this oscillator by clearing the CKGR_MOR.MOSCXTEN, PMC_SR.MOSCXTS is automatically cleared, indicating the 3 to 20 MHz crystal oscillator is off.

When enabling this oscillator, the user must initiate the start-up time counter. The start-up time depends on the characteristics of the external device connected to this oscillator.

### 32.6.44 PIO Fall/Rise - Low/High Status Register

**Name:** PIO_FRLHSR

**Address:** 0x400E0ED8 (PIOA), 0x400E10D8 (PIOB), 0x4800C0D8 (PIOC)

**Access:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

- **P0–P31: Edge/Level Interrupt Source Selection**

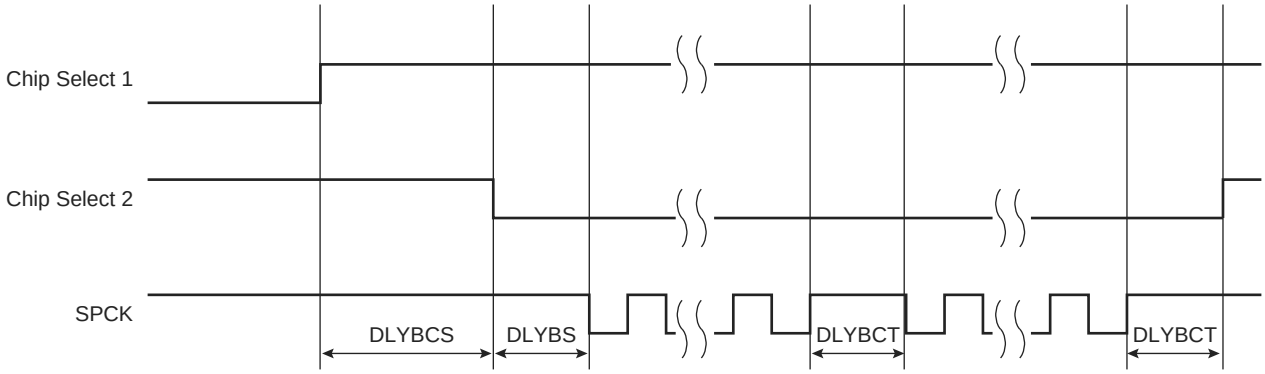
0: The interrupt source is a falling edge detection (if PIO_ELSR = 0) or low-level detection event (if PIO_ELSR = 1).

1: The interrupt source is a rising edge detection (if PIO_ELSR = 0) or high-level detection event (if PIO_ELSR = 1).

- Delay between consecutive transfers—independently programmable for each chip select by writing the DLYBCT field. The time required by the SPI slave device to process received data is managed through DLYBCT. This time depends on the SPI slave system activity.

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

**Figure 33-10. Programmable Delays**




#### 33.7.3.5 Peripheral Selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all NPCS signals are high before and after each transfer.

- **Fixed Peripheral Select Mode**: SPI exchanges data with only one peripheral.
  Fixed peripheral select mode is enabled by writing the PS bit to zero in the SPI_MR. In this case, the current peripheral is defined by the PCS field in the SPI_MR and the PCS field in the SPI_TDR has no effect.
- **Variable Peripheral Select Mode:** Data can be exchanged with more than one peripheral without having to reprogram the NPCS field in the SPI_MR.
  Variable peripheral select mode is enabled by setting the PS bit to 1 in the SPI_MR. The PCS field in the SPI_TDR is used to select the current peripheral. This means that the peripheral selection can be defined for each new data. The value to write in the SPI_TDR has the following format:

  [xxxxxxx(7-bit) + LASTXFER(1-bit)[1]+ xxxx(4-bit) + PCS (4-bit) + DATA (8 to 16-bit)] with PCS equals the chip select to assert, as defined in Section 33.8.4 “SPI Transmit Data Register” and LASTXFER bit at 0 or 1 depending on the CSAAT bit.

Note: 1. Optional

CSAAT, LASTXFER and CSNAAT bits are discussed in Section 33.7.3.9 “Peripheral Deselection with PDC”.

If LASTXFER is used, the command must be issued after writing the last character. Instead of LASTXFER, the user can use the SPIDIS command. After the end of the PDC transfer, it is necessary to wait for the TXEMPTY flag and then write SPIDIS into the SPI Control Register (SPI_CR). This does not change the configuration register values). The NPCS is disabled after the last character transfer. Then, another PDC transfer can be started if the SPIEN has previously been written in the SPI_CR.

#### 33.7.3.6 SPI Peripheral DMA Controller (PDC)

In both Fixed and Variable peripheral select modes, the Peripheral DMA Controller (PDC) can be used to reduce processor overhead.

The fixed peripheral selection allows buffer transfers with a single peripheral. Using the PDC is an optimal means, as the size of the data transfer between the memory and the SPI is either 8 bits or 16 bits. However, if the peripheral selection is modified, the SPI_MR must be reprogrammed.

*General Call*

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

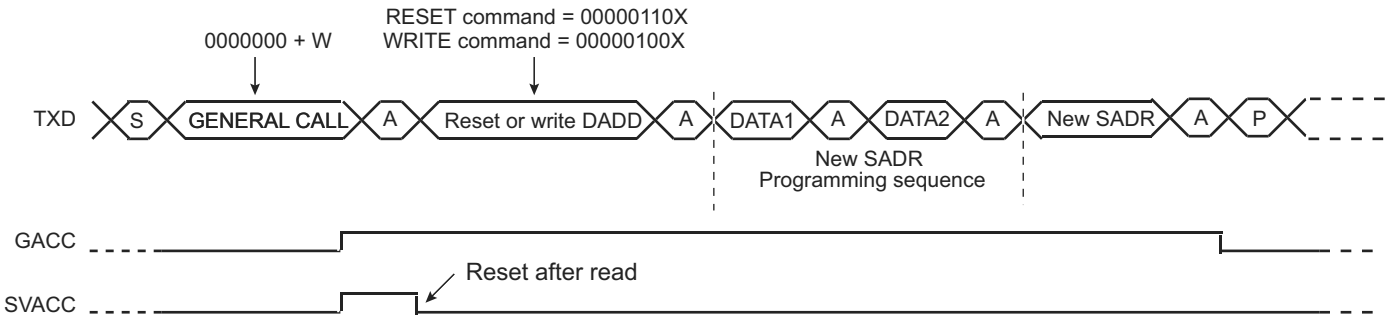
If a GENERAL CALL is detected, GACC is set.

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

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

Figure 34-27 describes the GENERAL CALL access.

**Figure 34-27. Master Performs a General Call**




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

*Clock Synchronization/Stretching*

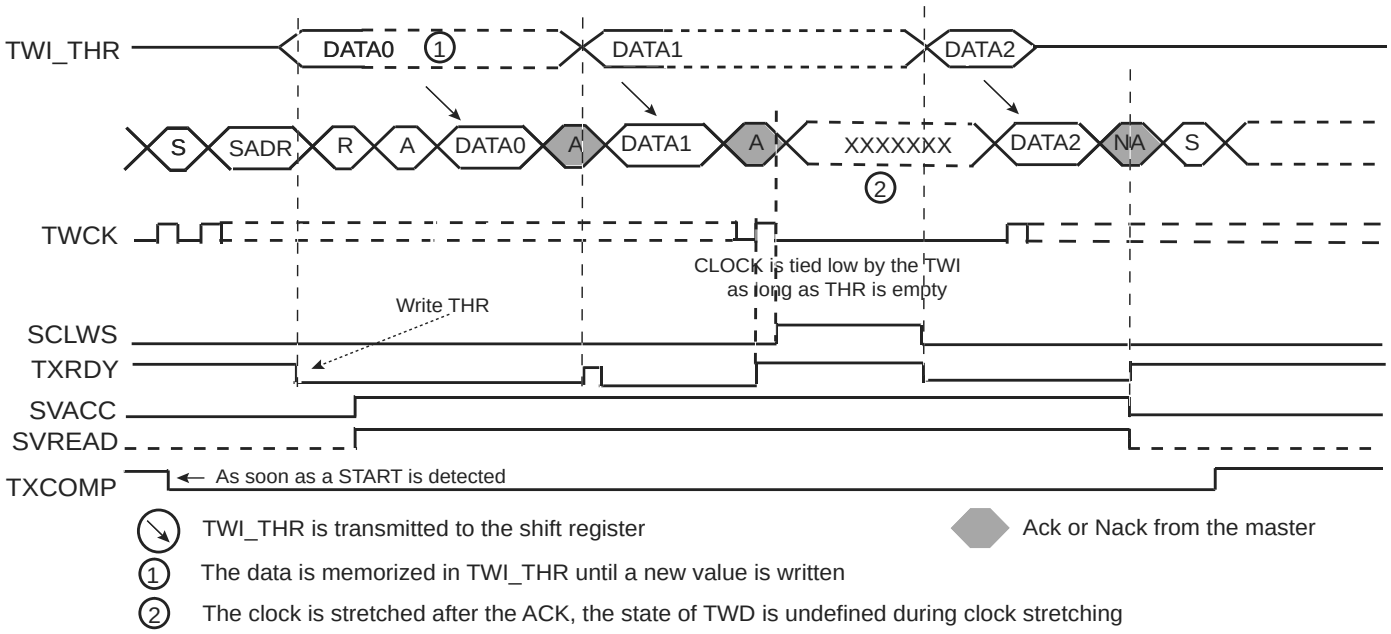
In both Read and Write modes, it may occur that TWI_THR/TWI_RHR buffer is not filled /emptied before transmission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching/synchronization mechanism is implemented.

**Clock Stretching in Read Mode**

The clock is tied low during the acknowledge phase if the internal shifter is empty and if a STOP or REPEATED START condition was not detected. It is tied low until the internal shifter is loaded.

Figure 34-28 describes clock stretching in Read mode.

**Figure 34-28. Clock Stretching in Read Mode**




Notes:
1. TXRDY is reset when data has been written in the TWI_THR to the internal shifter and set when this data has been acknowledged or non acknowledged.
2. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
3. SCLWS is automatically set when the clock stretching mechanism is started.

### 36.7.5 USART Interrupt Enable Register

**Name:** US_IER

**Address:** 0x40024008 (0), 0x40028008 (1), 0x4002C008 (2), 0x40030008 (3), 0x40034008 (4)

**Access:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | MANE |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | - | 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 36.7.6 ”USART Interrupt Enable Register (SPI_MODE)”.

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

0: No effect

1: Enables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **RXBRK: Receiver Break Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable (available in all USART modes of operation)**
- **ENDTX: End of Transmit Buffer Interrupt Enable (available in all USART modes of operation)**
- **OVRE: Overrun Error Interrupt Enable**
- **FRAME: Framing Error Interrupt Enable**
- **PARE: Parity Error Interrupt Enable**
- **TIMEOUT: Time-out Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **ITER: Max number of Repetitions Reached Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable (available in all USART modes of operation)**
- **RXBUFF: Receive Buffer Full Interrupt Enable (available in all USART modes of operation)**
- **NACK: Non Acknowledge Interrupt Enable**

- **CTSIC: Clear to Send Input Change Interrupt Enable**
- **MANE: Manchester Error Interrupt Enable**

### 36.7.7 USART Interrupt Disable Register

**Name:** US_IDR

**Address:** 0x4002400C (0), 0x4002800C (1), 0x4002C00C (2), 0x4003000C (3), 0x4003400C (4)

**Access:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | MANE |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | - | 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 36.7.8 ”USART Interrupt Disable Register (SPI_MODE)”.

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

0: No effect

1: Disables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **RXBRK: Receiver Break Interrupt Disable**
- **ENDRX: End of Receive Buffer Transfer Interrupt Disable (available in all USART modes of operation)**
- **ENDTX: End of Transmit Buffer Interrupt Disable (available in all USART modes of operation)**
- **OVRE: Overrun Error Interrupt Enable**
- **FRAME: Framing Error Interrupt Disable**
- **PARE: Parity Error Interrupt Disable**
- **TIMEOUT: Time-out Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **ITER: Max Number of Repetitions Reached Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable (available in all USART modes of operation)**
- **RXBUFF: Receive Buffer Full Interrupt Disable (available in all USART modes of operation)**
- **NACK: Non Acknowledge Interrupt Disable**

- **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.

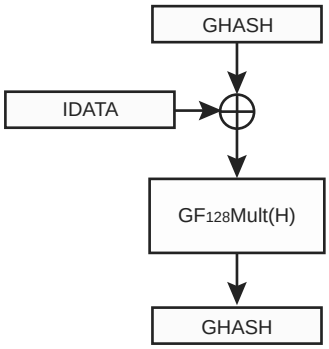
- **MTIOB: TIOB Mirror**

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.

**Processing a Message with only AAD (GHASH$_H$)**

**Figure 42-7. Single GHASH$_H$ Block Diagram (AADLEN ≤ 0x10 and CLEN = 0)**




It is possible to process a message with only *AAD* setting the CLEN field to '0' in the AES_CLENR, this can be used for $J_0$ generation when len(*IV*) ≠ 96 for instance.

Example: Processing $J_0$ when len(*IV*) ≠ 96

To process $J_0$ = GHASH$_H$(*IV* || $0^{s+64}$ || [len(*IV*)]$_{64}$), the sequence is as follows:

1. In AES_MR set OPMOD to GCM and GTAGEN to '0' (configuration as usual for the rest).
2. Set KEYW in AES_KEYWRx and wait until DATRDY bit of AES_ISR is set (GCM hash subkey generation complete); use interrupt if needed. After the GCM hash subkey generation is complete the GCM hash subkey can be read or overwritten with specific value in the AES_GCMHRx (see Section 42.4.6.2 "Key Writing and Automatic Hash Subkey Calculation" for details).
3. Set AADLEN field with 'len(*IV* || $0^{s+64}$ || [len(*IV*)]$_{64}$)' in AES_AADLENR and CLEN field to '0' in AES_CLENR. This will allow running a GHASH$_H$ only.
4. Fill the IDATA field of AES_IDATARx with the message to process (*IV* || $0^{s+64}$ || [len(*IV*)]$_{64}$) according to the SMOD configuration used. If Manual Mode or Auto Mode is used, the DATRDY bit indicates when a GHASH$_H$ step is over (use interrupt if needed).
5. Read the GHASH field of AES_GHASHRx to obtain the $J_0$ value.

Note: The GHASH value can be overwritten at any time by writing the GHASH field value of AES_GHASHRx, used to perform a GHASH$_H$ with an initial value for GHASH (write GHASH field between step 3 and step 4 in this case).

**Processing a Single GF$_{128}$ Multiplication**

The AES can also be used to process a single multiplication in the Galois field on 128 bits (GF$_{128}$) using a single GHASH$_H$ with custom *H* value (see Figure 42-7).

To run a GF$_{128}$ multiplication (A x B), the sequence is as follows:

1. In AES_MR set OPMOD to GCM and GTAGEN to '0' (configuration as usual for the rest).
2. Set AADLEN field with 0x10 (16 bytes) in AES_AADLENR and CLEN field to '0' in AES_CLENR. This will allow running a single GHASH$_H$.
3. Fill the H field of the AES_GCMHRx with B value.
4. Fill the IDATA field of AES_IDATARx with the A value according to the SMOD configuration used. If Manual Mode or Auto Mode is used, the DATRDY bit indicates when a GHASH$_H$ computation is over (use interrupt if needed).
5. Read the GHASH field of AES_GHASHRx to obtain the result.

Note: The GHASH field of AES_GHASHRx can be initialized with a value C between step 3 and step 4 to run a ((A XOR C) x B) GF$_{128}$ multiplication.

### 45.6.2 TRNG Interrupt Enable Register

**Name:** TRNG_IER

**Address:** 0x40048010

**Access:** Write-only

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

- **DATRDY: Data Ready Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

**Table 46-4. Recommended Operating Conditions on Power Supply Inputs at Powerup**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $RR_{VDDBU}$ | Rise Rate on VDDBU | (1) | 660 | – | 300k | V/s |
| $V_{ST\_VDDBU}$ | VDDBU voltage at powerup | (1) | 3.0 | – | – | V |
| $V_{ST\_VDDIO}$ | VDDIO and VDDIN voltage at powerup | – | 3.0 | – | – | V |
| $V_{VDDIO\_VDDBU}$ | Voltage on VDDIO and VDDIN while VDDBU < 1.6V | (1) | – | – | $V_{VDDBU}$ | V |
| $RR_{VDDIO}$ | Rise Rate on VDDIO and VDDIN | – | 330 | – | 300k | V/s |

Note: 1. Applies whenever VDDBU is restarted from below 1.35V.

### 46.2.3 Recommended Operating Conditions on Input Pins

**Table 46-5. Recommended Operating Conditions on Input Pins**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $AD[x]_{IN}$ | Input voltage range on 10-bit ADC analog inputs | On AD[0..x] | 0 | – | Min (VDDIN, VDDIO) | V |
| $EMAFE_{IN}$ | Input voltage range on EMAFE input pins | On $I_{P\{0,1,2,3\}}$, $I_{N\{0,1,2,3\}}$ and $V_{P\{1,2,3\}}$ | -0.25 | – | 0.25 | V |
| $V_{GPIO\_IN}$ | Input voltage range on GPIOs referenced to VDDIO | On any pin configured as a digital input | 0 | – | VDDIO | V |
| $V_{VDDBU\_IN}$ | Input voltage range on inputs referenced to VDDBU | On FWUP, TMP0 and XIN32 inputs | 0 | – | VDDBU | V |

### 46.2.4 Recommended Thermal Operating Conditions

**Table 46-6. Recommended Thermal Operating Conditions**

| Symbol | Parameter | Conditions | | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| $T_A$ | Ambient temperature range | – | | -40 | – | +85 | °C |
| $T_J$ | Junction temperature range | – | | -40 | – | 100 | |
| $R_{JA}$ | Junction-to-ambient thermal resistance | LQFP100 (SAM4CM16/8/4) | | – | 43 | – | °C/W |
| | | LQFP100 (SAM4CM32) | | – | 41 | – | |
| $P_D$ | Power dissipation | LQFP100 (SAM4CM16/8/4) | $T_A$ = 70°C | – | – | 700 | mW |
| | | | $T_A$ = 85°C | – | – | 350 | |
| | | LQFP100 (SAM4CM32) | $T_A$ = 70°C | – | – | 730 | |
| | | | $T_A$ = 85°C | – | – | 365 | |

## 46.3 Electrical Parameters Usage

The tables that follow further on in Section 46.4 “I/O Characteristics”, Section 46.5 “Embedded Analog Peripherals Characteristics”, Section 46.6 “Embedded Flash Characteristics”, and Section 46.7 “Power Supply Current Consumption” define the limiting values for several electrical parameters. Unless otherwise noted, these values are valid over the ambient temperature range $T_A$= [-40°C + 85°C]. Note that these limits may be affected by the board on which the MCU is mounted. Particularly, noisy supply and ground conditions must be avoided and care must be taken to provide:

- a PCB with a low impedance ground plane (unbroken ground planes are strongly recommended)

### 46.4.3 SPI Characteristics

In Figure 46-4 "SPI Master Mode with (CPOL= NCPHA = 0) or (CPOL= NCPHA= 1)" and Figure 46-5 "SPI Master Mode with (CPOL = 0 and NCPHA=1) or (CPOL=1 and NCPHA= 0)" below, the MOSI line shifting edge is represented with a hold time equal to 0. However, it is important to note that for this device, the MISO line is sampled prior to the MOSI line shifting edge. As shown in Figure 46-3 "MISO Capture in Master Mode", the device sampling point extends the propagation delay ($t_p$) for slave and routing delays to more than half the SPI clock period, whereas the common sampling point allows only less than half the SPI clock period.

As an example, an SPI Slave working in Mode 0 can be safely driven if the SPI Master is configured in Mode 0.

**Figure 46-3. MISO Capture in Master Mode**

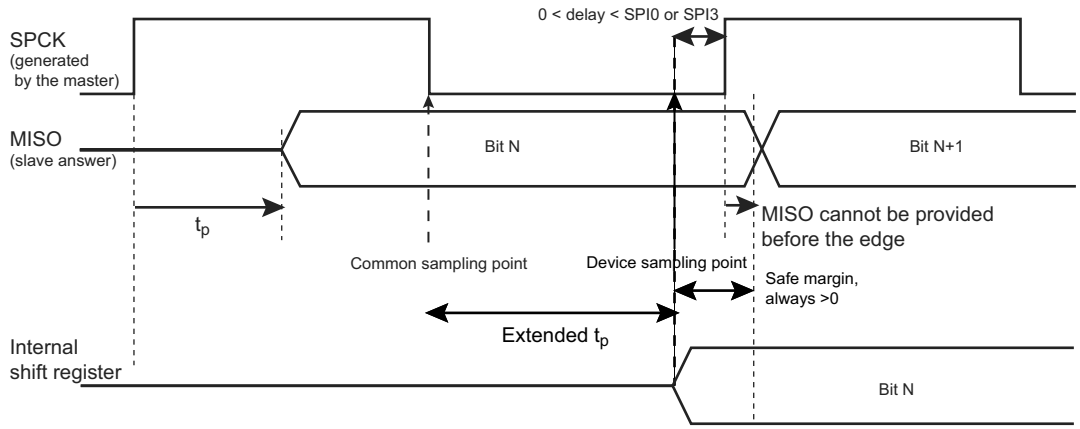



**Figure 46-4. SPI Master Mode with (CPOL= NCPHA = 0) or (CPOL= NCPHA= 1)**

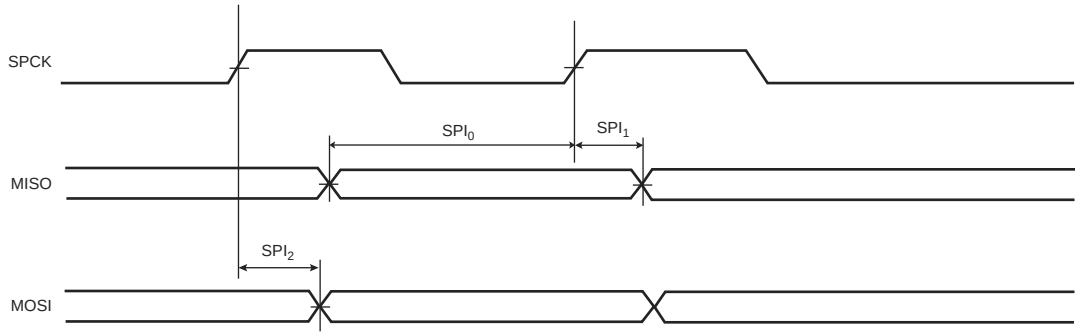



**Figure 46-5. SPI Master Mode with (CPOL = 0 and NCPHA=1) or (CPOL=1 and NCPHA= 0)**