**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | ARM® Cortex®-M4 |
| Core Size | 32-Bit Single-Core |
| Speed | 120MHz |
| Connectivity | CANbus, Ethernet, IrDA, MMC/SD, SPI, UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, DMA, POR, PWM, WDT |
| Number of I/O | 79 |
| Program Memory Size | 1MB (1M x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 128K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.62V ~ 3.6V |
| Data Converters | A/D 16x12b; D/A 2x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 105°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 100-LQFP |
| Supplier Device Package | 100-LQFP (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atsam4e16ca-anr |

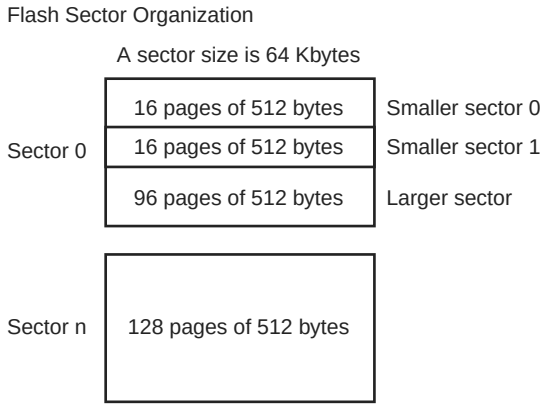
For sector 0:

- The smaller sector 0 has 16 pages of 512 bytes
- The smaller sector 1 has 16 pages of 512 bytes
- The larger sector has 96 pages of 512 bytes

From Sector 1 to n:

The rest of the array is composed of 64 Kbyte sector of each 128 pages of 512 bytes. Refer to Figure 7-3.

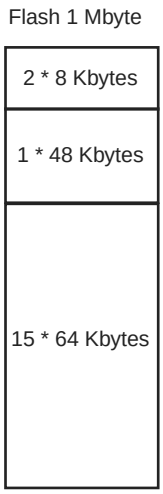
**Figure 7-3. Flash Sector Organization**




Flash size varies by product. The Flash size of SAM4E device is 1024 Kbytes.

Refer to Figure 7-4 for the organization of the Flash following its size.

**Figure 7-4. Flash Size**



#### 11.6.5.2 AND, ORR, EOR, BIC, and ORN

Logical AND, OR, Exclusive OR, Bit Clear, and OR NOT.

Syntax

```
op{S}{cond} {Rd,} Rn, Operand2
```

where:

op is one of:

AND logical AND.

ORR logical OR, or bit set.

EOR logical Exclusive OR.

BIC logical AND NOT, or bit clear.

ORN logical OR NOT.

S is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation, see “Conditional Execution” .

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

Rd is the destination register.

Rn is the register holding the first operand.

Operand2 is a flexible second operand. See “Flexible Second Operand” for details of the options.

Operation

The AND, EOR, and ORR instructions perform bitwise AND, Exclusive OR, and OR operations on the values in *Rn* and *Operand2*.

The BIC instruction performs an AND operation on the bits in *Rn* with the complements of the corresponding bits in the value of *Operand2*.

The ORN instruction performs an OR operation on the bits in *Rn* with the complements of the corresponding bits in the value of *Operand2*.

Restrictions

Do not use SP and do not use PC.

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

```
AND      R9, R2, #0xFF00
ORREQ    R2, R0, R5
ANDS     R9, R8, #0x19
EORS     R7, R11, #0x18181818
BIC      R0, R1, #0xab
ORN       R7, R11, R14, ROR #4
ORNS     R7, R11, R14, ASR #32
```

#### 11.6.12.10 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)
```

#### 11.9.1.7 Configuration and Control Register

**Name:** SCB_CCR

**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 |
| – | – | – | – | – | – | STKALIGN | BFHFNMIGN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | DIV_0_TRP | UNALIGN_TRP | – | USERSETMPEND | NONBASETHRDENA |

The SCB_CCR controls the entry to the Thread mode and enables the handlers for NMI, hard fault and faults escalated by FAULTMASK to ignore BusFaults. It also enables the division by zero and unaligned access trapping, and the access to the NVIC_STIR by unprivileged software (see “Software Trigger Interrupt Register” ).

- **STKALIGN: Stack Alignment**

Indicates the stack alignment on exception entry:

0: 4-byte aligned.

1: 8-byte aligned.

On exception entry, the processor uses bit [9] of the stacked PSR to indicate the stack alignment. On return from the exception, it uses this stacked bit to restore the correct stack alignment.

- **BFHFNMIGN: Bus Faults Ignored**

Enables handlers with priority -1 or -2 to ignore data bus faults caused by load and store instructions. This applies to the hard fault and FAULTMASK escalated handlers:

0: Data bus faults caused by load and store instructions cause a lock-up.

1: Handlers running at priority -1 and -2 ignore data bus faults caused by load and store instructions.

Set this bit to 1 only when the handler and its data are in absolutely safe memory. The normal use of this bit is to probe system devices and bridges to detect control path problems and fix them.

- **DIV_0_TRP: Division by Zero Trap**

Enables faulting or halting when the processor executes an SDIV or UDIV instruction with a divisor of 0:

0: Do not trap divide by 0.

1: Trap divide by 0.

When this bit is set to 0, a divide by zero returns a quotient of 0.

- **UNALIGN_TRP: Unaligned Access Trap**

Enables unaligned access traps:

0: Do not trap unaligned halfword and word accesses.

1: Trap unaligned halfword and word accesses.

If this bit is set to 1, an unaligned access generates a usage fault.

- **BERR: Bit Error (automatically cleared by reading CAN_SR)**

0: No bit error occurred during a previous transfer.

1: A bit error occurred during a previous transfer.

A bit error is set when the bit value monitored on the line is different from the bit value sent.

- **RBSY: Receiver Busy**

0: CAN receiver is not receiving a frame.

1: CAN receiver is receiving a frame.

Receiver busy. This status bit is set by hardware while CAN receiver is acquiring or monitoring a frame (remote, data, overload or error frame). It is automatically reset when CAN is not receiving.

- **TBSY: Transmitter Busy**

0: CAN transmitter is not transmitting a frame.

1: CAN transmitter is transmitting a frame.

Transmitter busy. This status bit is set by hardware while CAN transmitter is generating a frame (remote, data, overload or error frame). It is automatically reset when CAN is not transmitting.

- **OVLSY: Overload busy**

0: CAN transmitter is not transmitting an overload frame.

1: CAN transmitter is transmitting a overload frame.

It is automatically reset when the bus is not transmitting an overload frame.

**Figure 33-11. Parallel Capture Mode Waveforms (DSIZE = 2, ALWYS = 1, HALFS = 0)**

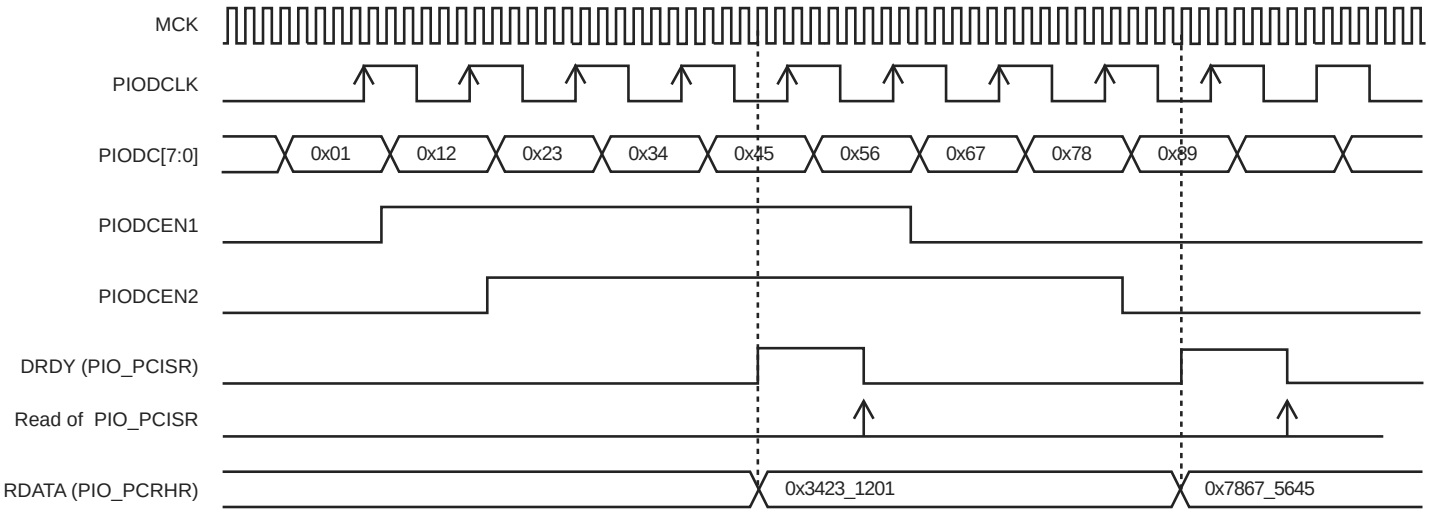
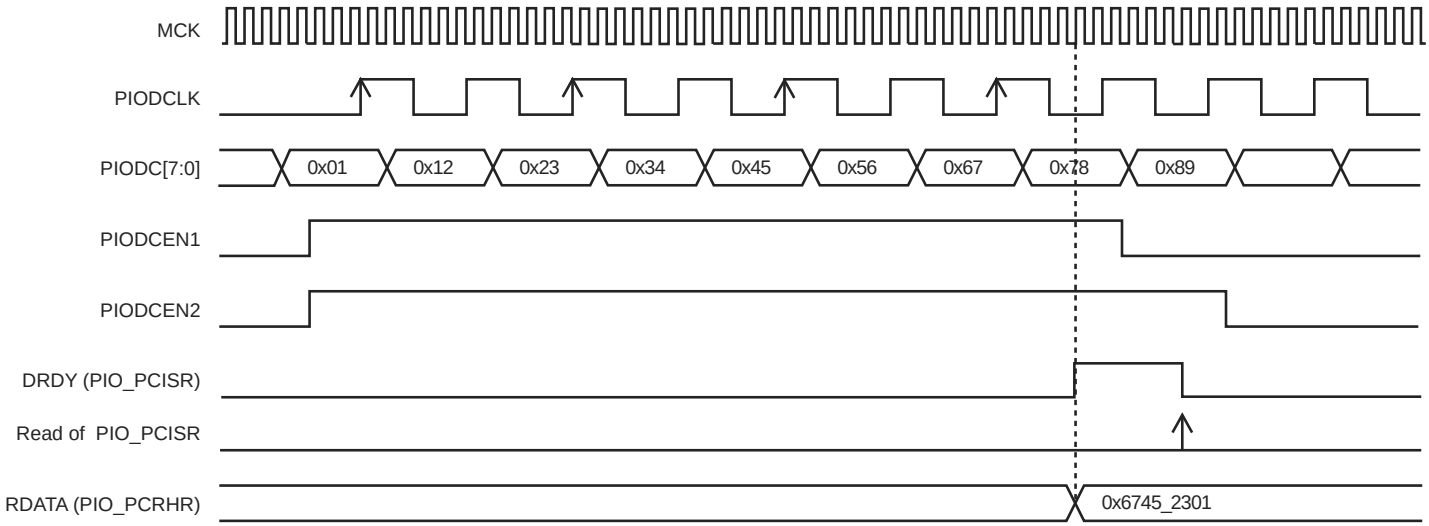



**Figure 33-12. Parallel Capture Mode Waveforms (DSIZE = 2, ALWYS = 0, HALFS = 1, FRSTS = 0)**



### 33.6.33 PIO Output Write Enable Register

**Name:** PIO_OWER

**Address:** 0x400E0EA0 (PIOA), 0x400E10A0 (PIOB), 0x400E12A0 (PIOC), 0x400E14A0 (PIOD), 0x400E16A0 (PIOE)

**Access:** Write-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 |

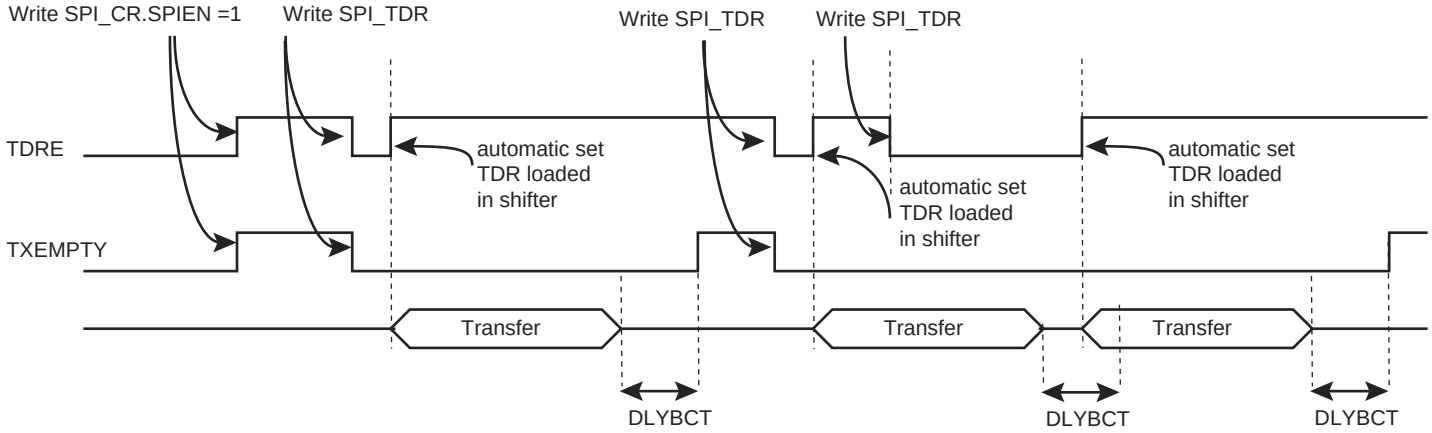
This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Output Write Enable**

0: No effect.

1: Enables writing PIO_ODSR for the I/O line.

**Figure 34-5. 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.

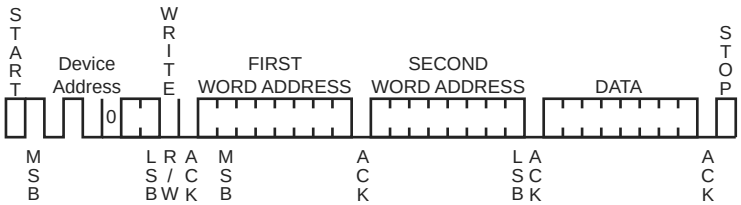
If the SPI_RDR has not been read before new data is received, the Overrun Error (OVRES) bit in the SPI_SR is set. As long as this flag is set, data is loaded in the SPI_RDR. The user has to read the SPI_SR to clear the OVRES bit.

Figure 34-6 shows a block diagram of the SPI when operating in Master mode. Figure 34-7 shows a flow chart describing how transfers are handled.

1. Program IADRSZ = 1,
2. Program DADR with 1 1 1 1 0 b1 b2 (b1 is the MSB of the 10-bit address, b2, etc.)
3. Program TWI_IADR with b3 b4 b5 b6 b7 b8 b9 b10 (b10 is the LSB of the 10-bit address)

Figure 35-12 below shows a byte write to a memory device. This demonstrates the use of internal addresses to access the device.

**Figure 35-12. Internal Address Usage**




### 35.7.3.6 Using the Peripheral DMA Controller (PDC)

The use of the PDC significantly reduces the CPU load.

To ensure correct implementation, proceed as follows.

*Data Transmit with the PDC*

1. Initialize the transmit PDC (memory pointers, transfer size - 1).
2. Configure the master (DADR, CKDIV, MREAD = 0, etc.)
3. Start the transfer by setting the PDC TXTEN bit.
4. Wait for the PDC ENDTX Flag either by using the polling method or ENDTX interrupt.
5. Disable the PDC by setting the PDC TXTDIS bit.
6. Wait for the TXRDY flag in TWI_SR.
7. Set the STOP bit in TWI_CR.
8. Write the last character in TWI_THR.
9. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWI_SR.

*Data Receive with the PDC*

The PDC transfer size must be defined with the buffer size minus 2. The two remaining characters must be managed without PDC to ensure that the exact number of bytes are received regardless of system bus latency conditions encountered during the end of buffer transfer period.

In Slave mode, the number of characters to receive must be known in order to configure the PDC.

1. Initialize the receive PDC (memory pointers, transfer size - 2).
2. Configure the master (DADR, CKDIV, MREAD = 1, etc.)
3. Set the PDC RXTEN bit.
4. (Master Only) Write the START bit in the TWI_CR to start the transfer.
5. Wait for the PDC ENDRX Flag either by using polling method or ENDRX interrupt.
6. Disable the PDC by setting the PDC RXTDIS bit.
7. Wait for the RXRDY flag in TWI_SR.
8. Set the STOP bit in TWI_CR.
9. Read the penultimate character in TWI_RHR.
10. Wait for the RXRDY flag in TWI_SR.
11. Read the last character in TWI_RHR.
12. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWI_SR.

transmission (even if US_THR has been written) while the receiver side is not ready (character not read). When WRDBT equals 0, the character is transmitted whatever the receiver status. If WRDBT is set to 1, the transmitter waits for the Receive Holding register (US_RHR) to be read before transmitting the character (RXRDY flag cleared), thus preventing any overflow (character loss) on the receiver side.

The chip select line is de-asserted for a period equivalent to three bits between the transmission of two data.

The transmitter reports two status bits in US_CSR: TXRDY (Transmitter Ready), which indicates that US_THR is empty and TXEMPTY, which indicates that all the characters written in US_THR have been processed. When the current character processing is completed, the last character written in US_THR is transferred into the Shift register of the transmitter and US_THR becomes empty, thus TXRDY rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in US_THR while TXRDY is low has no effect and the written character is lost.

If the USART is in SPI Slave mode and if a character must be sent while the US_THR is empty, the UNRE (Underrun Error) bit is set. The TXD transmission line stays at high level during all this time. The UNRE bit is cleared by writing a 1 to the RSTSTA (Reset Status) bit in US_CR.

In SPI Master mode, the slave select line (NSS) is asserted at low level one $t_{bit}$ ($t_{bit}$ being the nominal time required to transmit a bit) before the transmission of the MSB bit and released at high level one $t_{bit}$ after the transmission of the LSB bit. So, the slave select line (NSS) is always released between each character transmission and a minimum delay of three $t_{bit}$ always inserted. However, in order to address slave devices supporting the CSAAT mode (Chip Select Active After Transfer), the slave select line (NSS) can be forced at low level by writing a 1 to the RCS bit in the US_CR. The slave select line (NSS) can be released at high level only by writing a 1 to the FCS bit in the US_CR (for example, when all data have been transferred to the slave device).

In SPI Slave mode, the transmitter does not require a falling edge of the slave select line (NSS) to initiate a character transmission but only a low level. However, this low level must be present on the slave select line (NSS) at least one $t_{bit}$ before the first serial clock cycle corresponding to the MSB bit.

#### 37.6.8.6 Character Reception

When a character reception is completed, it is transferred to the Receive Holding register (US_RHR) and the RXRDY bit in the Status register (US_CSR) rises. If a character is completed while RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing a 1 to the RSTSTA (Reset Status) bit in the US_CR.

To ensure correct behavior of the receiver in SPI Slave mode, the master device sending the frame must ensure a minimum delay of one $t_{bit}$ between each character transmission. The receiver does not require a falling edge of the slave select line (NSS) to initiate a character reception but only a low level. However, this low level must be present on the slave select line (NSS) at least one $t_{bit}$ before the first serial clock cycle corresponding to the MSB bit.

#### 37.6.8.7 Receiver Timeout

Because the receiver baud rate clock is active only during data transfers in SPI mode, a receiver timeout is impossible in this mode, whatever the time-out value is (field TO) in the US_RTOR.

### 37.6.9 Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In Loopback mode, the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

#### 37.6.9.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

### 37.7.7 USART Interrupt Disable Register

**Name:** US_IDR

**Address:** 0x400A000C (0), 0x400A400C (1)

**Access:** Write-only

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

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | CTSIC | DCDIC | DSRIC | RIIC |

| 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 37.7.8 ”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**
- **RIIC: Ring Indicator Input Change Disable**

### 39.7.38 PWM Comparison x Mode Register

**Name:** PWM_CMPMx

**Access:** Read/Write

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

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| CUPRCNT | | | | CUPR | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CPRCNT | | | | CPR | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CTR | | | | – | – | – | CEN |

- **CEN: Comparison x Enable**

0: The comparison x is disabled and can not match.

1: The comparison x is enabled and can match.

- **CTR: Comparison x Trigger**

The comparison x is performed when the value of the comparison x period counter (CPRCNT) reaches the value defined by CTR.

- **CPR: Comparison x Period**

CPR defines the maximum value of the comparison x period counter (CPRCNT). The comparison x value is performed periodically once every CPR+1 periods of the channel 0 counter.

- **CPRCNT: Comparison x Period Counter**

Reports the value of the comparison x period counter.

Note: The field CPRCNT is read-only

- **CUPR: Comparison x Update Period**

Defines the time between each update of the comparison x mode and the comparison x value. This time is equal to CUPR+1 periods of the channel 0 counter.

- **CUPRCNT: Comparison x Update Period Counter**

Reports the value of the comparison x update period counter.

Note: The field CUPRCNT is read-only

### 39.7.46 PWM Channel Dead Time Register

**Name:** PWM_DTx [x=0..3]

**Access:** Read/Write

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

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

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

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

This register can only be written if bits WPSWS4 and WPHWS4 are cleared in the PWM Write Protection Status Register.

Only the first 12 bits (dead-time counter size) of fields DTH and DTL are significant.

- **DTH: Dead-Time Value for PWMHx Output**

Defines the dead-time value for PWMHx output. This value must be defined between 0 and the value (CPRD – CDTY) (PWM_CPRDx and PWM_CDTYx).

- **DTL: Dead-Time Value for PWMLx Output**

Defines the dead-time value for PWMLx output. This value must be defined between 0 and CDTY (PWM_CDTYx).

## 40.14 High Speed MultiMedia Card Interface (HSMCI) User Interface

**Table 40-8. Register Mapping**

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Control Register | HSMCI_CR | Write-only | – |
| 0x04 | Mode Register | HSMCI_MR | Read/Write | 0x0 |
| 0x08 | Data Timeout Register | HSMCI_DTOR | Read/Write | 0x0 |
| 0x0C | SD/SDIO Card Register | HSMCI_SDCR | Read/Write | 0x0 |
| 0x10 | Argument Register | HSMCI_ARGR | Read/Write | 0x0 |
| 0x14 | Command Register | HSMCI_CMDR | Write-only | – |
| 0x18 | Block Register | HSMCI_BLKR | Read/Write | 0x0 |
| 0x1C | Completion Signal Timeout Register | HSMCI_CSTOR | Read/Write | 0x0 |
| 0x20 | Response Register(1) | HSMCI_RSPR | Read-only | 0x0 |
| 0x24 | Response Register(1) | HSMCI_RSPR | Read-only | 0x0 |
| 0x28 | Response Register(1) | HSMCI_RSPR | Read-only | 0x0 |
| 0x2C | Response Register(1) | HSMCI_RSPR | Read-only | 0x0 |
| 0x30 | Receive Data Register | HSMCI_RDR | Read-only | 0x0 |
| 0x34 | Transmit Data Register | HSMCI_TDR | Write-only | – |
| 0x38–0x3C | Reserved | – | – | – |
| 0x40 | Status Register | HSMCI_SR | Read-only | 0xC0E5 |
| 0x44 | Interrupt Enable Register | HSMCI_IER | Write-only | – |
| 0x48 | Interrupt Disable Register | HSMCI_IDR | Write-only | – |
| 0x4C | Interrupt Mask Register | HSMCI_IMR | Read-only | 0x0 |
| 0x50 | Reserved | – | – | – |
| 0x54 | Configuration Register | HSMCI_CFG | Read/Write | 0x00 |
| 0x58–0xE0 | Reserved | – | – | – |
| 0xE4 | Write Protection Mode Register | HSMCI_WPMR | Read/Write | – |
| 0xE8 | Write Protection Status Register | HSMCI_WPSR | Read-only | – |
| 0xEC–0xFC | Reserved | – | – | – |
| 0x100–0x128 | Reserved for PDC registers | – | – | – |
| 0x12C–0x1FC | Reserved | – | – | – |
| 0x200 | FIFO Memory Aperture0 | HSMCI_FIFO0 | Read/Write | 0x0 |
| ... | ... | ... | ... | ... |
| 0x5FC | FIFO Memory Aperture255 | HSMCI_FIFO255 | Read/Write | 0x0 |

Notes: 1. The Response Register can be read by N accesses at the same HSMCI_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

### 42.8.47 GMAC PTP Peer Event Frame Received Nanoseconds Register

**Name:** GMAC_PEFRN

**Address:** 0x400341FC

**Access:** Read-only

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

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

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

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

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Nanoseconds Register holds when the SFD of a PTP receive primary event crosses the MII interface. An interrupt is issued when the register is updated.

### 43.7.20 AFEC Channel Data Register

**Name:** AFEC_CDR

**Address:** 0x400B0068 (0), 0x400B4068 (1)

**Access:** Read-only

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

- **DATA: Converted Data**

Returns the AFE conversion data corresponding to channel CSEL (configured in the AFEC Channel Selection Register).

At the end of a conversion, the converted data is loaded into one of the 16 internal registers (one for each channel) and remains in this internal register until a new conversion is completed on the same channel index. The AFEC_CDR together with AFEC_CSELR allows to multiplex all the internal channel data registers.

The data carried on AFEC_CDR is valid only if AFEC_CHSR.CH*x* bit is set (where *x* = AFEC_CSELR.CSEL field value).

### 44.7.1 DACC Control Register

**Name:** DACC_CR

**Address:** 0x400B8000

**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 |
| – | – | – | – | – | – | – | SWRST |

- **SWRST: Software Reset**

0: No effect

1: Resets the DACC, simulating a hardware reset

**Table 46-59. SMC Read Signals - NCS Controlled (READ_MODE = 0) (Continued)**

| | | | | | | |
|---|---|---|---|---|---|---|
| $SMC_{14}$ | NCS Pulse Width | NCS_RD_PULSE length × $t_{CPMCK}$ - 7.6 | NCS_RD_PULSE length × $t_{CPMCK}$ - 6.7 | — | — | ns |

### 46.11.5.2 Write Timings

**Table 46-60. SMC Write Signals - NWE Controlled (WRITE_MODE = 1)**

| | VDDIO Supply | 1.8V Domain | 3.3V Domain | 1.8V Domain | 3.3V Domain | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | | **Max** | | **Unit** |
| **HOLD or NO HOLD Settings (NWE_HOLD ≠ 0, NWE_HOLD = 0)** | | | | | | |
| $SMC_{15}$ | Data Out Valid before NWE High | NWE_PULSE × $t_{CPMCK}$ - 6.2 | NWE_PULSE × $t_{CPMCK}$ - 5.9 | — | — | ns |
| $SMC_{16}$ | NWE Pulse Width | NWE_PULSE × $t_{CPMCK}$ - 7.0 | NWE_PULSE × $t_{CPMCK}$ - 6.1 | — | — | ns |
| $SMC_{17}$ | A0–A22 valid before NWE low | NWE_SETUP × $t_{CPMCK}$ - 6.6 | NWE_SETUP × $t_{CPMCK}$ - 6.2 | — | — | ns |
| $SMC_{18}$ | NCS low before NWE high | (NWE_SETUP - NCS_RD_SETUP + NWE_PULSE) × $t_{CPMCK}$ - 5.9 | (NWE_SETUP - NCS_RD_SETUP + NWE_PULSE) × $t_{CPMCK}$ - 5.5 | — | — | ns |
| **HOLD Settings (NWE_HOLD ≠ 0)** | | | | | | |
| $SMC_{19}$ | NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 change | NWE_HOLD × $t_{CPMCK}$ - 9.4 | NWE_HOLD × $t_{CPMCK}$ - 7.6 | — | — | ns |
| $SMC_{20}$ | NWE High to NCS Inactive (1) | (NWE_HOLD - NCS_WR_HOLD) × $t_{CPMCK}$ - 6.0 | (NWE_HOLD - NCS_WR_HOLD) × $t_{CPMCK}$ - 5.6 | — | — | ns |
| **NO HOLD Settings (NWE_HOLD = 0)** | | | | | | |
| $SMC_{21}$ | NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25, NCS change(1) | 3.3 | 3.2 | — | — | ns |

Notes: 1. Hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “NCS_WR_HOLD length” or “NWE_HOLD length”.

**Table 46-61. SMC Write NCS Controlled (WRITE_MODE = 0)**

| | VDDIO Supply | 1.8V Domain | 3.3V Domain | 1.8V Domain | 3.3V Domain | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | | **Max** | | **Unit** |
| $SMC_{22}$ | Data Out Valid before NCS High | NCS_WR_PULSE × $t_{CPMCK}$ - 6.3 | NCS_WR_PULSE × $t_{CPMCK}$ - 6.0 | — | — | ns |
| $SMC_{23}$ | NCS Pulse Width | NCS_WR_PULSE × $t_{CPMCK}$ - 7.6 | NCS_WR_PULSE × $t_{CPMCK}$ - 6.7 | — | — | ns |
| $SMC_{24}$ | A0–A22 valid before NCS low | NCS_WR_SETUP × $t_{CPMCK}$ - 6.7 | NCS_WR_SETUP × $t_{CPMCK}$ - 6.3 | — | — | ns |

**Table 46-61. SMC Write NCS Controlled (WRITE_MODE = 0)**

| | VDDIO Supply | 1.8V Domain | 3.3V Domain | 1.8V Domain | 3.3V Domain | |
|---|---|---|---|---|---|---|
| **Symbol** | **Parameter** | **Min** | | **Max** | | **Unit** |
| $SMC_{25}$ | NWE low before NCS high | (NCS_WR_SETUP - NWE_SETUP + NCS pulse) × $t_{CPMCK}$ - 5.6 | (NCS_WR_SETUP - NWE_SETUP + NCS pulse) × $t_{CPMCK}$ - 5.3 | — | — | ns |
| $SMC_{26}$ | NCS High to Data Out, A0–A25, change | NCS_WR_HOLD × $t_{CPMCK}$ - 10.6 | NCS_WR_HOLD × $t_{CPMCK}$ - 9.0 | — | — | ns |
| $SMC_{27}$ | NCS High to NWE Inactive | (NCS_WR_HOLD - NWE_HOLD) × $t_{CPMCK}$ - 7.0 | (NCS_WR_HOLD - NWE_HOLD) × $t_{CPMCK}$ - 6.8 | — | — | ns |

**Figure 46-25. SMC Timings - NCS Controlled Read and Write**

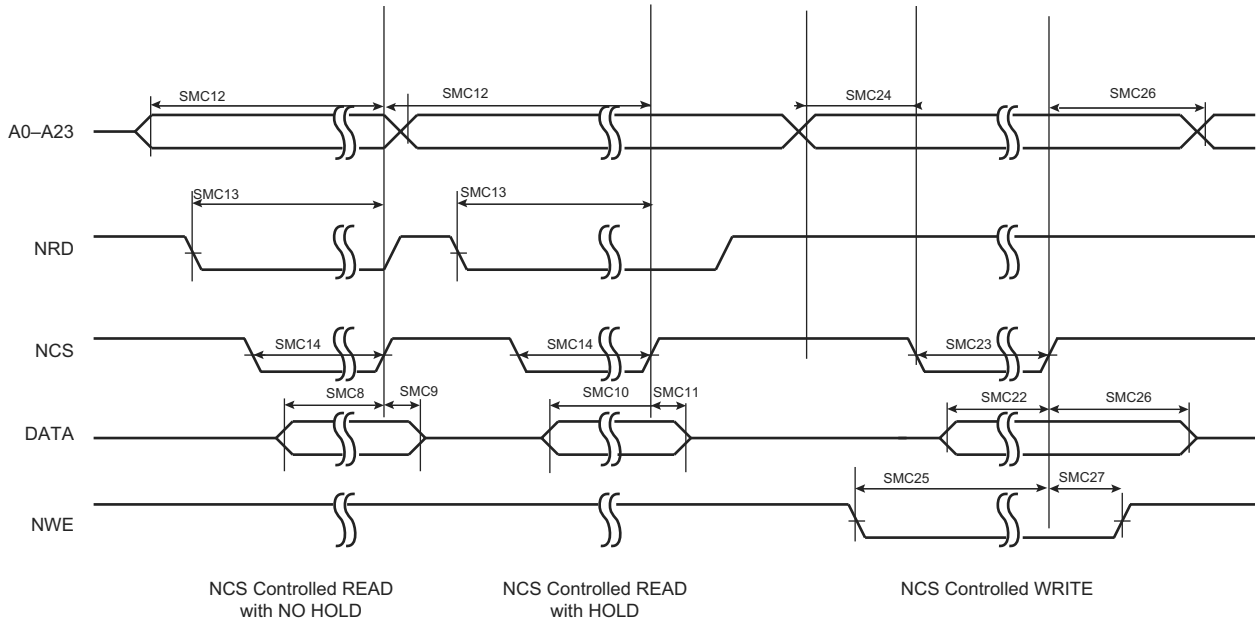


**Table 49-1. Ordering Codes for SAM4E Devices (Continued)**

| Ordering Code | MRL | Flash (Kbytes) | RAM (Kbytes) | Package | Carrier Type | Operating Temperature Range |
|---|---|---|---|---|---|---|
| ATSAM4E8CA-AU | A | 512 | 128 | LQFP100 | Tray | Industrial (-40°C to 85°C) |
| ATSAM4E8CA-AUR | | | | | Reel | |
| ATSAM4E8CA-AN | A | | | | Tray | Industrial (-40°C to 105°C) |
| ATSAM4E8CA-ANR | | | | | Reel | |
| ATSAM4E8CB-AN | B | | | | Tray | |
| ATSAM4E8CB-ANR | | | | | Reel | |