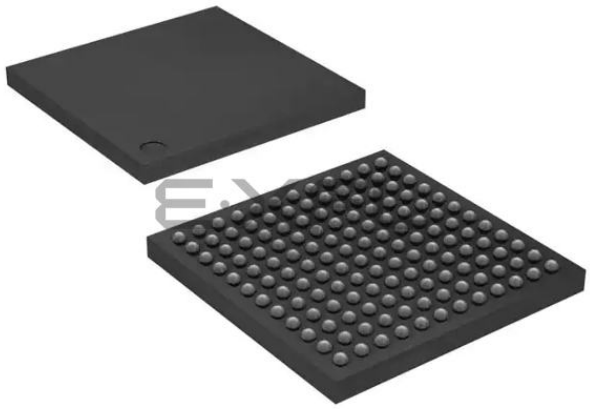


**Welcome to E-XFL.COM**

## 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®-M3 |
| Core Size | 32-Bit Single-Core |
| Speed | 84MHz |
| Connectivity | CANbus, EBI/EMI, Ethernet, I²C, IrDA, LINbus, Memory Card, SPI, SSC, UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT |
| Number of I/O | 103 |
| Program Memory Size | 512KB (512K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 100K 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 ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 144-LFBGA |
| Supplier Device Package | 144-BGA (13x13) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atsam3x8ea-cu |

## 10.7 Fault handling

Faults are a subset of the exceptions, see “Exception model” on page 70. The following generate a fault:

- a bus error on:
  - an instruction fetch or vector table load
  - a data access
- an internally-detected error such as an undefined instruction or an attempt to change state with a BX instruction
- attempting to execute an instruction from a memory region marked as *Non-Executable* (XN).
- an MPU fault because of a privilege violation or an attempt to access an unmanaged region.

### 10.7.1 Fault types

Table 10-11 shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates that the fault has occurred. See “Configurable Fault Status Register” on page 182 for more information about the fault status registers.

**Table 10-11. Faults**

| Fault | Handler | Bit name | Fault status register |
|---|---|---|---|
| Bus error on a vector read | Hard fault | VECTTBL | “Hard Fault Status Register” on page 188 |
| Fault escalated to a hard fault | | FORCED | |
| MPU mismatch: | Memory management fault | - | - |
| on instruction access | | IACCVIOL (1) | “Memory Management Fault Address Register” on page 189 |
| on data access | | DACCVIOL | |
| during exception stacking | | MSTKERR | |
| during exception unstacking | | MUNSKERR | |
| Bus error: | Bus fault | - | - |
| during exception stacking | | STKERR | “Bus Fault Status Register” on page 184 |
| during exception unstacking | | UNSTKERR | |
| during instruction prefetch | | IBUSERR | |
| Precise data bus error | | PRECISERR | |
| Imprecise data bus error | | IMPRECISERR | |
| Attempt to access a coprocessor | Usage fault | NOCP | “Usage Fault Status Register” on page 186 |
| Undefined instruction | | UNDEFINSTR | |
| Attempt to enter an invalid instruction set state (2) | | INVSTATE | |
| Invalid EXC_RETURN value | | INVPC | |
| Illegal unaligned load or store | | UNALIGNED | |
| Divide By 0 | | DIVBYZERO | |

1. Occurs on an access to an XN region even if the MPU is disabled.
2. Attempting to use an instruction set other than the Thumb instruction set.

### 10.18.5 ISB

Instruction Synchronization Barrier.

#### 10.18.5.1 Syntax

```
ISB{cond}
```

where:

cond is an optional condition code, see “Conditional execution” on page 91.

#### 10.18.5.2 Operation

ISB acts as an instruction synchronization barrier. It flushes the pipeline of the processor, so that all instructions following the ISB are fetched from memory again, after the ISB instruction has been completed.

#### 10.18.5.3 Condition flags

This instruction does not change the flags.

#### 10.18.5.4 Examples

```
ISB  ; Instruction Synchronisation Barrier
```

### 10.21.14 Bus Fault Address Register

The BFAR contains the address of the location that generated a bus fault. See the register summary in Table 10-30 on page 165 for its attributes. The bit assignments are:

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

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

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

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

- **ADDRESS**

When the BFARVALID bit of the BFSR is set to 1, this field holds the address of the location that generated the bus fault

When an unaligned access faults the address in the BFAR is the one requested by the instruction, even if it is not the address of the fault.

Flags in the BFSR indicate the cause of the fault, and whether the value in the BFAR is valid. See “Bus Fault Status Register” on page 184.

# 18. Enhanced Embedded Flash Controller (EEFC)

## 18.1 Description

The Enhanced Embedded Flash Controller (EEFC) ensures the interface of the Flash block with the 32-bit internal bus.

Its 128-bit or 64-bit wide memory interface increases performance. It also manages the programming, erasing, locking and unlocking sequences of the Flash using a full set of commands. One of the commands returns the embedded Flash descriptor definition that informs the system about the Flash organization, thus making the software generic.

## 18.2 Embedded Characteristics

- Interface of the Flash Block with the 32-bit Internal Bus
- Increases Performance in Thumb2® Mode with 128-bit or -64 bit Wide Memory Interface up to 23 MHz
- 16 Lock Bits, Each Protecting a Lock Region
- 3 General-purpose GPNVM Bits
- One-by-one Lock Bit Programming
- Commands Protected by a Keyword
- Erases the Entire Flash
- Possibility of Erasing before Programming
- Locking and Unlocking Operations
- Consecutive Programming and Locking Operations
- Possibility to read the Calibration Bits

## 18.3 Product Dependencies

### 18.3.1 Power Management

The Enhanced Embedded Flash Controller (EEFC) is continuously clocked. The Power Management Controller has no effect on its behavior.

### 18.3.2 Interrupt Sources

The Enhanced Embedded Flash Controller (EEFC) interrupt line is connected to the Nested Vectored Interrupt Controller (NVIC). Using the Enhanced Embedded Flash Controller (EEFC) interrupt requires the NVIC to be programmed first. The EEFC interrupt is generated only on FRDY bit rising.

**Table 18-1. Peripheral IDs**

| Instance | ID |
|---|---|
| EFC0 | 6 |
| EFC1 | 7 |

## 21.6 System I/O Configuration

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

## 21.7 Write Protect Registers

To prevent any single software error that may corrupt MATRIX behavior, the entire MATRIX address space from address offset 0x000 to 0x1FC can be write-protected by setting the WPEN bit in the MATRIX Write Protect Mode Register (MATRIX_WPMR).

If a write access to anywhere in the MATRIX address space from address offset 0x000 to 0x1FC is detected, then the WPVS flag in the MATRIX Write Protect Status Register (MATRIX_WPSR) is set and the WPVSRC field indicates in which register the write access has been attempted.

The WPVS flag is reset by writing the MATRIX Write Protect Mode Register (MATRIX_WPMR) with the appropriate access key, WPKEY.

The protected registers are:

“Bus Matrix Master Configuration Registers”

“Bus Matrix Slave Configuration Registers”

“Bus Matrix Priority Registers For Slaves”

“Bus Matrix Master Remap Control Register”

“System I/O Configuration Register”

### 31.7.27 PIO Glitch or Debouncing Input Filter Selection Status Register

**Name:** PIO_IFDGSR

**Address:** 0x400E0E88 (PIOA), 0x400E1088 (PIOB), 0x400E1288 (PIOC), 0x400E1488 (PIOD), 0x400E1688 (PIOE), 0x400E1888 (PIOF)

**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: Glitch or Debouncing Filter Selection Status**

0: The Glitch Filter is able to filter glitches with a duration < Tmck2.

1: The Debouncing Filter is able to filter pulses with a duration < Tdiv_slclk/2.

transferred through MISO and MOSI lines with the chip select configuration registers. This is not the optimal means in term of memory size for the buffers, but it provides a very effective means to exchange data with several peripherals without any intervention of the processor.

#### 32.7.3.7 Peripheral Chip Select Decoding

The user can program the SPI to operate with up to 15 peripherals by decoding the four Chip Select lines, NPCS0 to NPCS3 with 1 of up to 16 decoder/demultiplexer. This can be enabled by writing the PCSDEC bit at 1 in the Mode Register (SPI_MR).

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e., one NPCS line driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

When operating with decoding, the SPI directly outputs the value defined by the PCS field on NPCS lines of either the Mode Register or the Transmit Data Register (depending on PS).

As the SPI sets a default value of 0xF on the chip select lines (i.e. all chip select lines at 1) when not processing any transfer, only 15 peripherals can be decoded.

The SPI has only four Chip Select Registers, not 15. As a result, when decoding is activated, each chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Thus, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14. Figure 32-9 below shows such an implementation.

If the CSAAT bit is used, with or without the DMAC, the Mode Fault detection for NPCS0 line must be disabled. This is not needed for all other chip select lines since Mode Fault Detection is only on NPCS0.

**Figure 32-9. Chip Select Decoding Application Block Diagram: Single Master/Multiple Slave Implementation**

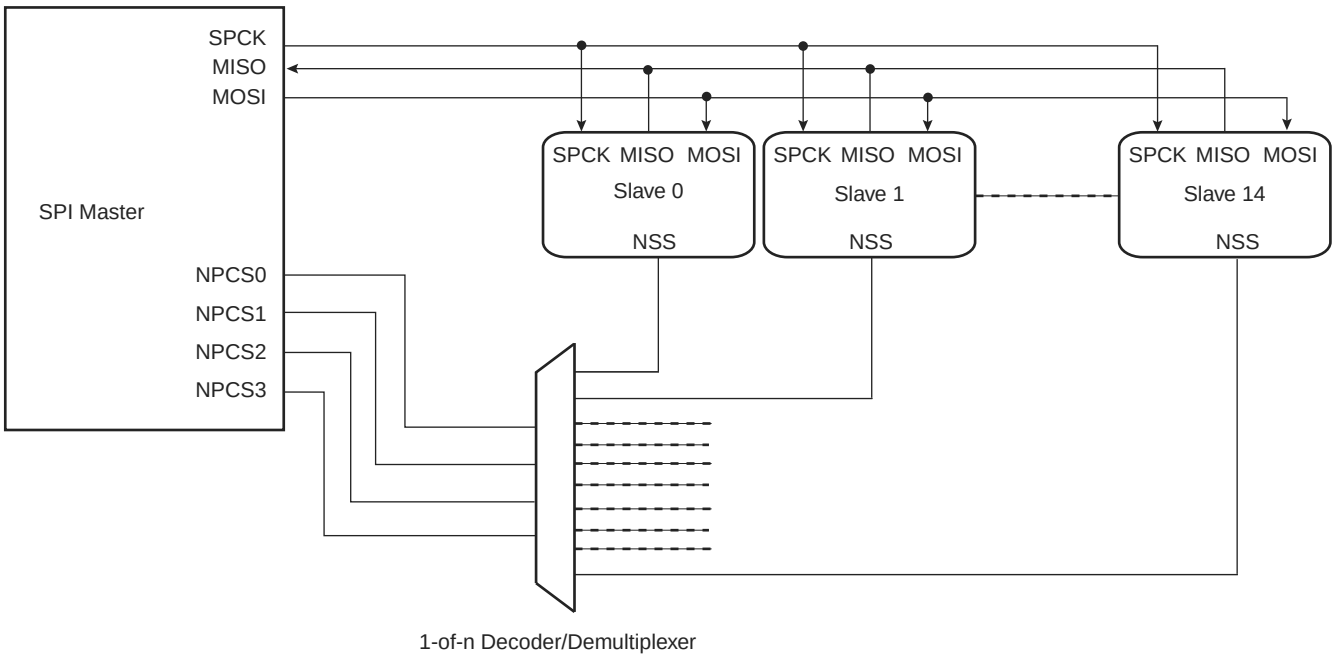


**Figure 33-21. Programmer Sends Data While the Bus is Busy**

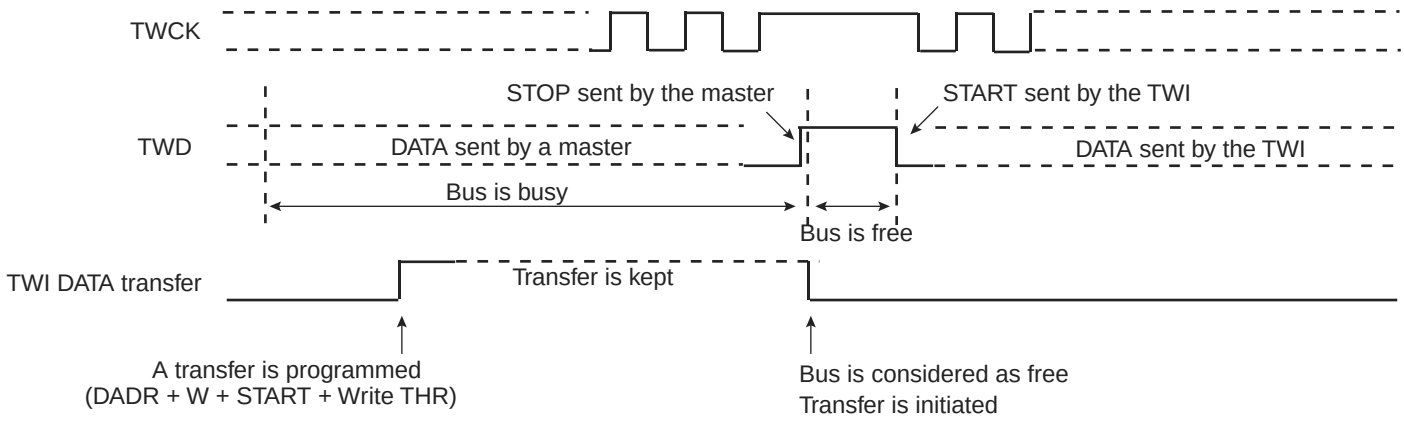
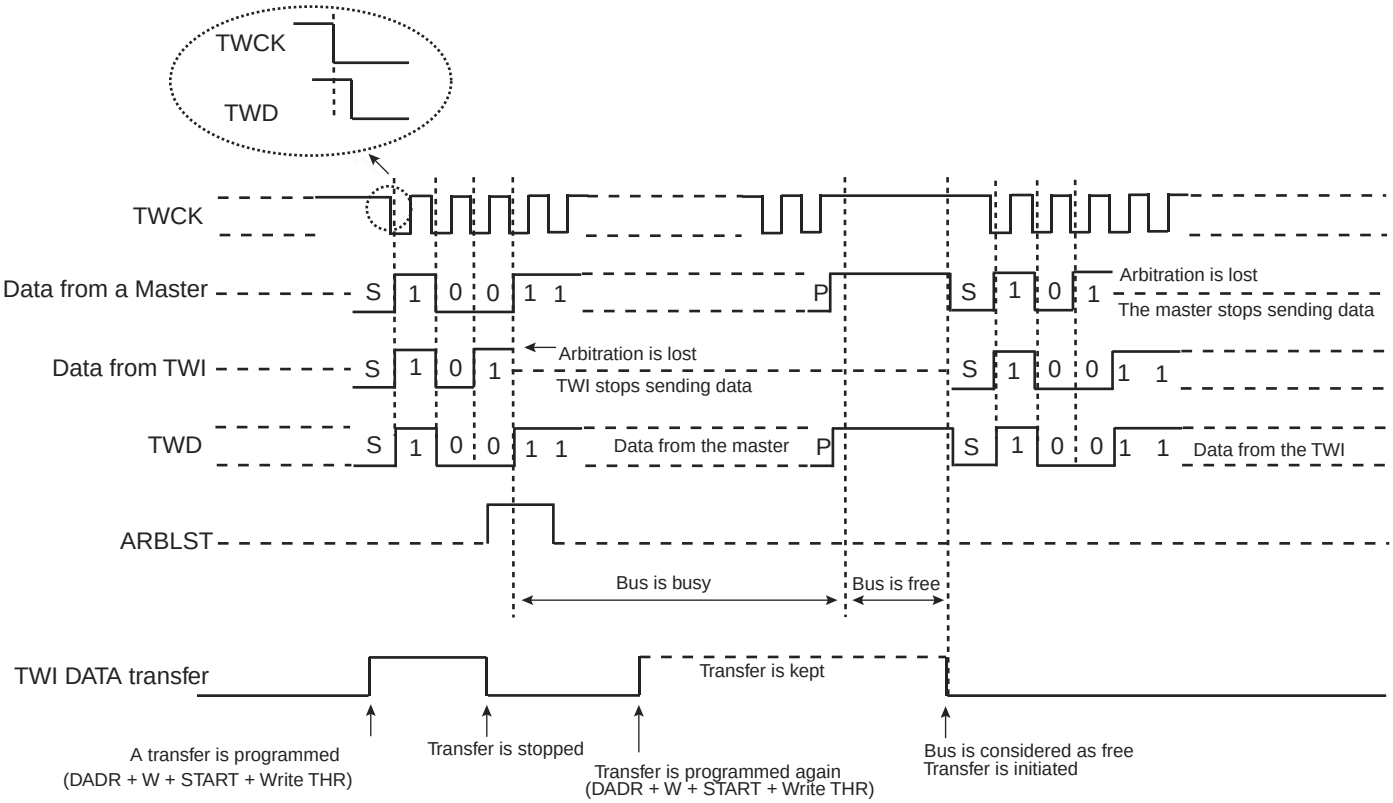



**Figure 33-22. Arbitration Cases**




The flowchart shown in Figure 33-23 on page 727 gives an example of read and write operations in Multi-master mode.

### 33.11.7 TWI Interrupt Enable Register

**Name:** TWI_IER

**Address:** 0x4008C024 (0), 0x40090024 (1)

**Access:** Write-only

**Reset:** 0x00000000

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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| TXBUFE | RXBUFF | ENDTX | ENDRX | EOSACC | SCL_WS | ARBLST | NACK |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | OVRE | GACC | SVACC | – | TXRDY | RXRDY | TXCOMP |

- **TXCOMP: Transmission Completed Interrupt Enable**
- **RXRDY: Receive Holding Register Ready Interrupt Enable**
- **TXRDY: Transmit Holding Register Ready Interrupt Enable**
- **SVACC: Slave Access Interrupt Enable**
- **GACC: General Call Access Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **NACK: Not Acknowledge Interrupt Enable**
- **ARBLST: Arbitration Lost Interrupt Enable**
- **SCL_WS: Clock Wait State Interrupt Enable**
- **EOSACC: End Of Slave Access Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

- **WKUPTYP: Wakeup Signal Type**

0: setting the bit LINWKUP in the control register sends a LIN 2.0 wakeup signal.

1: setting the bit LINWKUP in the control register sends a LIN 1.3 wakeup signal.

- **DLC: Data Length Control**

0 - 255: Defines the response data length if DLM=0,in that case the response data length is equal to DLC+1 bytes.

- **PDCM: PDC Mode**

0: The LIN mode register US_LINMR is not written by the PDC.

1: The LIN mode register US_LINMR (excepting that flag) is written by the PDC.

### 36.6.12 External Event/Trigger Conditions

An external event can be programmed to be detected on one of the clock sources (XC0, XC1, XC2) or TIOB. The external event selected can then be used as a trigger.

The EEVT parameter in TC_CMR selects the external trigger. The EEVTEDG parameter defines the trigger edge for each of the possible external triggers (rising, falling or both). If EEVTEDG is cleared (none), no external event is defined.

If TIOB is defined as an external event signal (EEVT = 0), TIOB is no longer used as an output and the compare register B is not used to generate waveforms and subsequently no IRQs. In this case the TC channel can only generate a waveform on TIOA.

When an external event is defined, it can be used as a trigger by setting bit ENETRG in the TC_CMR.

As in Capture mode, the SYNC signal and the software trigger are also available as triggers. RC Compare can also be used as a trigger depending on the parameter WAVSEL.

### 36.6.13 Output Controller

The output controller defines the output level changes on TIOA and TIOB following an event. TIOB control is used only if TIOB is defined as output (not as an external event).

The following events control TIOA and TIOB: software trigger, external event and RC compare. RA compare controls TIOA and RB compare controls TIOB. Each of these events can be programmed to set, clear or toggle the output as defined in the corresponding parameter in TC_CMR.

### 36.6.14 Quadrature Decoder

#### 36.6.14.1 Description

The quadrature decoder (QDEC) is driven by TIOA0, TIOB0, TIOB1 input pins and drives the timer/counter of channel 0 and 1. Channel 2 can be used as a time base in case of speed measurement requirements (refer to Figure 36-15).

When writing a 0 to bit QDEN of the TC_BMR, the QDEC is bypassed and the IO pins are directly routed to the timer counter function. See

TIOA0 and TIOB0 are to be driven by the two dedicated quadrature signals from a rotary sensor mounted on the shaft of the off-chip motor.

A third signal from the rotary sensor can be processed through pin TIOB1 and is typically dedicated to be driven by an index signal if it is provided by the sensor. This signal is not required to decode the quadrature signals PHA, PHB.

Field TCCLKS of TC_CMRx must be configured to select XC0 input (i.e., 0x101). Field TC0XC0S has no effect as soon as the QDEC is enabled.

Either speed or position/revolution can be measured. Position channel 0 accumulates the edges of PHA, PHB input signals giving a high accuracy on motor position whereas channel 1 accumulates the index pulses of the sensor, therefore the number of rotations. Concatenation of both values provides a high level of precision on motion system position.

In Speed mode, position cannot be measured but revolution can be measured.

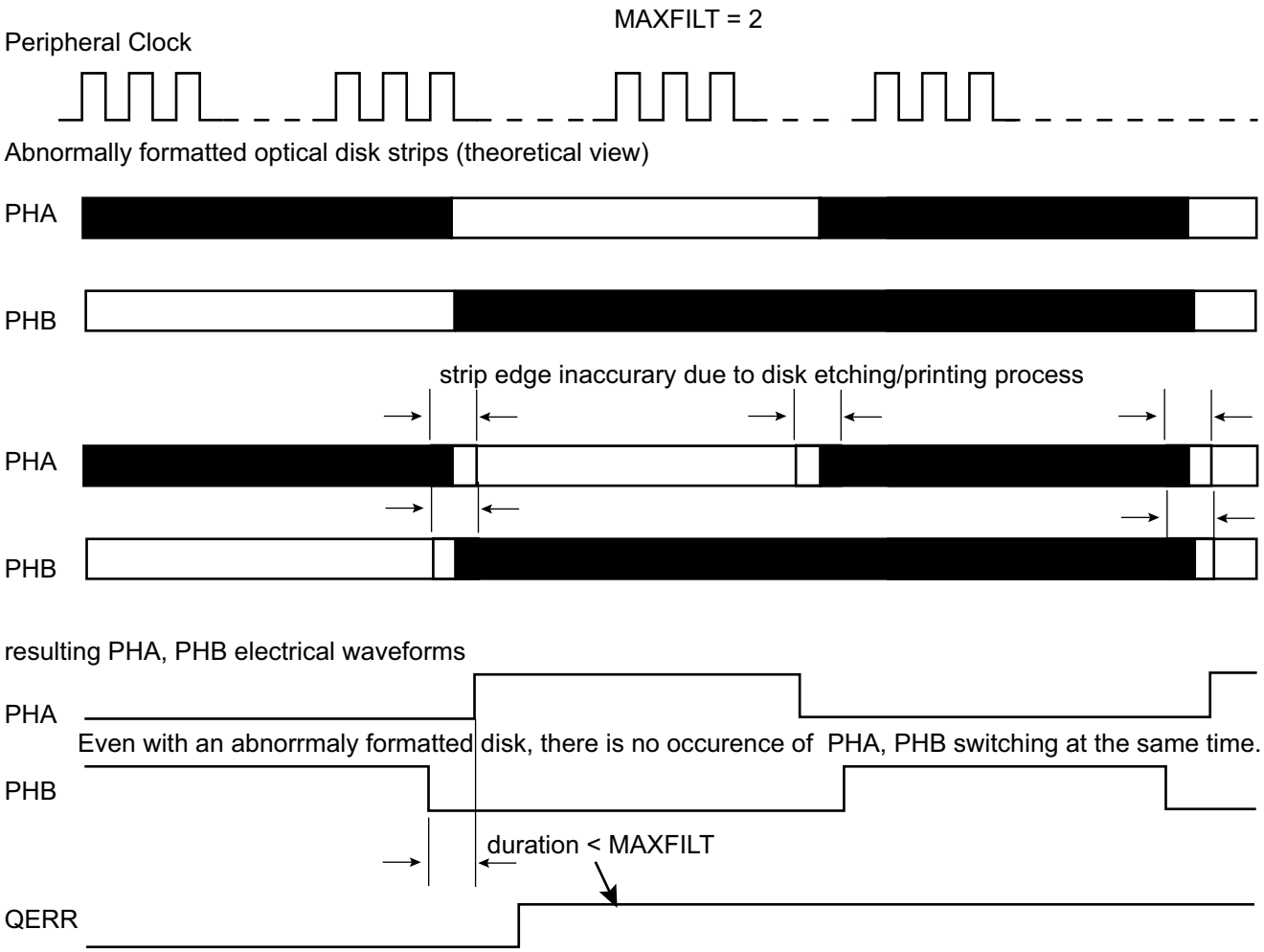
Inputs from the rotary sensor can be filtered prior to down-stream processing. Accommodation of input polarity, phase definition and other factors are configurable.

Interruptions can be generated on different events.

A compare function (using TC_RC) is available on channel 0 (speed/position) or channel 1 (rotation) and can generate an interrupt by means of the CPCS flag in the TC_SRx.

configurable and corresponds to (MAXFILT + 1) × $t_{peripheral\ clock}$ ns. After being filtered there is no reason to have two edges closer than (MAXFILT + 1) × $t_{peripheral\ clock}$ ns under normal mode of operation.

**Figure 36-19. Quadrature Error Detection**




MAXFILT must be tuned according to several factors such as the peripheral clock frequency, type of rotary sensor and rotation speed to be achieved.

#### 36.6.14.4 Position and Rotation Measurement

When the POSEN bit is set in the TC_BMR, the motor axis position is processed on channel 0 (by means of the PHA, PHB edge detections) and the number of motor revolutions are recorded on channel 1 if the IDX signal is provided on the TIOB1 input. The position measurement can be read in the TC_CV0 register and the rotation measurement can be read in the TC_CV1 register.

Channel 0 and 1 must be configured in Capture mode (TC_CMR0.WAVE = 0). ‘Rising edge’ must be selected as the External Trigger Edge (TC_CMR.ETRGEDG = 0x01) and ‘TIOA’ must be selected as the External Trigger (TC_CMR.ABETRG = 0x1).

In parallel, the number of edges are accumulated on timer/counter channel 0 and can be read on the TC_CV0 register.

Therefore, the accurate position can be read on both TC_CV registers and concatenated to form a 32-bit word.

The timer/counter channel 0 is cleared for each increment of IDX count value.

## 37.6 Product Dependencies

### 37.6.1 I/O Lines

The pins used for interfacing the High Speed MultiMedia Cards or SD Cards are multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the peripheral functions to HSMCI pins.

**Table 37-2. I/O Lines**

| Instance | Signal | I/O Line | Peripheral |
|---|---|---|---|
| HSMCI | MCCDA | PA20 | A |
| HSMCI | MCCDB | PE20 | B |
| HSMCI | MCCK | PA19 | A |
| HSMCI | MCDA0 | PA21 | A |
| HSMCI | MCDA1 | PA22 | A |
| HSMCI | MCDA2 | PA23 | A |
| HSMCI | MCDA3 | PA24 | A |
| HSMCI | MCDA4 | PD0 | B |
| HSMCI | MCDA5 | PD1 | B |
| HSMCI | MCDA6 | PD2 | B |
| HSMCI | MCDA7 | PD3 | B |
| HSMCI | MCDB0 | PE22 | B |
| HSMCI | MCDB1 | PE24 | B |
| HSMCI | MCDB2 | PE26 | B |
| HSMCI | MCDB3 | PE27 | B |

### 37.6.2 Power Management

The HSMCI is clocked through the Power Management Controller (PMC), so the programmer must first configure the PMC to enable the HSMCI clock.

### 37.6.3 Interrupt

The HSMCI interface has an interrupt line connected to the Nested Vector Interrupt Controller (NVIC).

Handling the HSMCI interrupt requires programming the NVIC before configuring the HSMCI.

**Table 37-3. Peripheral IDs**

| Instance | ID |
|---|---|
| HSMCI | 21 |

### 37.8.4 Write Operation

In write operation, the HSMCI Mode Register (HSMCI_MR) is used to define the padding value when writing non-multiple block size. If the bit PADV is 0, then 0x00 value is used when padding data, otherwise 0xFF is used.

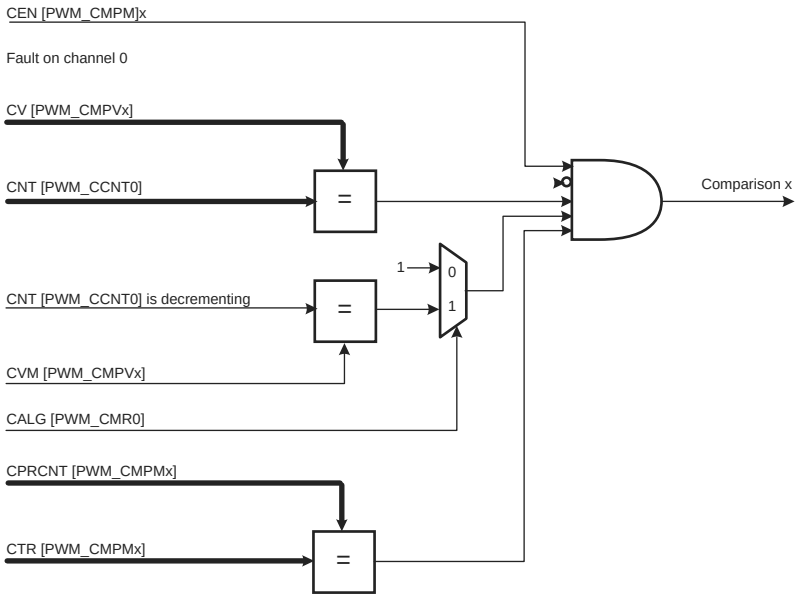
If set, the bit DMAEN in the HSMCI_DMA register enables DMA transfer.

The following flowchart (Figure 37-10) shows how to write a single block with or without use of DMA facilities. Polling or interrupt method can be used to wait for the end of write according to the contents of the Interrupt Mask Register (HSMCI_IMR).

### 38.6.3 PWM Comparison Units

The PWM provides 8 independent comparison units able to compare a programmed value with the current value of the channel 0 counter (which is the channel counter of all synchronous channels, Section 38.6.2.7 “Synchronous Channels”). These comparisons are intended to generate pulses on the event lines (used to synchronize ADC, see Section 38.6.4 “PWM Event Lines”), to generate software interrupts and to trigger PDC transfer requests for the synchronous channels (see “Method 3: Automatic write of duty-cycle values and automatic trigger of the update” on page 990).

**Figure 38-14. Comparison Unit Block Diagram**




The comparison x matches when it is enabled by the bit CEN in the “PWM Comparison x Mode Register” (PWM_CMPMx for the comparison x) and when the counter of the channel 0 reaches the comparison value defined by the field CV in “PWM Comparison x Value Register” (PWM_CMPVx for the comparison x). If the counter of the channel 0 is center aligned (CALG = 1 in “PWM Channel Mode Register” ), the bit CVM (in PWM_CMPVx) defines if the comparison is made when the counter is counting up or counting down (in left alignment mode CALG=0, this bit is useless).

If a fault is active on the channel 0, the comparison is disabled and cannot match (see Section 38.6.2.6 “Fault Protection”).

The user can define the periodicity of the comparison x by the fields CTR and CPR (in PWM_CMPVx). The comparison is performed periodically once every CPR+1 periods of the counter of the channel 0, when the value of the comparison period counter CPRCNT (in PWM_CMPMx) reaches the value defined by CTR. CPR is the maximum value of the comparison period counter CPRCNT. If CPR=CTR=0, the comparison is performed at each period of the counter of the channel 0.

The comparison x configuration can be modified while the channel 0 is enabled by using the “PWM Comparison x Mode Update Register” (PWM_CMPMUPDx registers for the comparison x). In the same way, the comparison x value can be modified while the channel 0 is enabled by using the “PWM Comparison x Value Update Register” (PWM_CMPVUPDx registers for the comparison x).

The update of the comparison x configuration and the comparison x value is triggered periodically after the comparison x update period. It is defined by the field CUPR in the PWM_CMPMx. The comparison unit has an update period counter independent from the period counter to trigger this update. When the value of the comparison update period counter CUPRCNT (in PWM_CMPMx) reaches the value defined by CUPR, the update

Data is typically transferred into and out of the FIFOs in bursts of four words. For receive, a bus request is asserted when the FIFO contains four words and has space for 28 more. For transmit, a bus request is generated when there is space for four words, or when there is space for 27 words if the next transfer is to be only one or two words.

Thus the bus latency must be less than the time it takes to load the FIFO and transmit or receive three words (112 bytes) of data.

At 100 Mbit/s, it takes 8960 ns to transmit or receive 112 bytes of data. In addition, six master clock cycles should be allowed for data to be loaded from the bus and to propagate through the FIFOs. For a 133 MHz master clock this takes 45 ns, making the bus latency requirement 8915 ns.

#### 41.4.2.2 Receive Buffers

Received frames, including CRC/FCS optionally, are written to receive buffers stored in memory. Each receive buffer is 128 bytes long. The start location for each receive buffer is stored in memory in a list of receive buffer descriptors at a location pointed to by the receive buffer queue pointer register. The receive buffer start location is a word address. For the first buffer of a frame, the start location can be offset by up to three bytes depending on the value written to bits 14 and 15 of the network configuration register. If the start location of the buffer is offset the available length of the first buffer of a frame is reduced by the corresponding number of bytes.

Each list entry consists of two words, the first being the address of the receive buffer and the second being the receive status. If the length of a receive frame exceeds the buffer length, the status word for the used buffer is written with zeroes except for the “start of frame” bit and the offset bits, if appropriate. Bit zero of the address field is written to one to show the buffer has been used. The receive buffer manager then reads the location of the next receive buffer and fills that with receive frame data. The final buffer descriptor status word contains the complete frame status. Refer to Table 41-1 for details of the receive buffer descriptor list.

**Table 41-1. Receive Buffer Descriptor Entry**

| Bit | Function |
|---|---|
| Word 0 | |
| 31:2 | Address of beginning of buffer |
| 1 | Wrap - marks last descriptor in receive buffer descriptor list. |
| 0 | Ownership - needs to be zero for the EMAC to write data to the receive buffer. The EMAC sets this to one once it has successfully written a frame to memory.<br>Software has to clear this bit before the buffer can be used again. |
| Word 1 | |
| 31 | Global all ones broadcast address detected |
| 30 | Multicast hash match |
| 29 | Unicast hash match |
| 28 | External address match |
| 27 | Reserved for future use |
| 26 | Specific address register 1 match |
| 25 | Specific address register 2 match |
| 24 | Specific address register 3 match |
| 23 | Specific address register 4 match |
| 22 | Type ID match |
| 21 | VLAN tag detected (i.e., type id of 0x8100) |
| 20 | Priority tag detected (i.e., type id of 0x8100 and null VLAN identifier) |

### 43.7.2 ADC Mode Register

**Name:** ADC_MR

**Address:** 0x400C0004

**Access:** Read-write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| USEQ | – | TRANSFER | | TRACKTIM | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ANACH | – | SETTLING | | STARTUP | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PRESCAL | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FREERUN | FWUP | SLEEP | LOWRES | TRGSEL | | | TRGEN |

This register can only be written if the WPEN bit is cleared in “ADC Write Protect Mode Register” on page 1353.

- **TRGEN: Trigger Enable**

| Value | Name | Description |
|---|---|---|
| 0 | DIS | Hardware triggers are disabled. Starting a conversion is only possible by software. |
| 1 | EN | Hardware trigger selected by TRGSEL field is enabled. |

- **TRGSEL: Trigger Selection**

| Value | Name | Description |
|---|---|---|
| 0 | ADC_TRIG0 | External : ADCTRG |
| 1 | ADC_TRIG1 | TIOA Output of the Timer Counter Channel 0 |
| 2 | ADC_TRIG2 | TIOA Output of the Timer Counter Channel 1 |
| 3 | ADC_TRIG3 | TIOA Output of the Timer Counter Channel 2 |
| 4 | ADC_TRIG4 | PWM Event Line 0 |
| 5 | ADC_TRIG5 | PWM Event Line 0 |
| 6 | ADC_TRIG6 | Reserved |
| 7 | – | Reserved |

- **LOWRES: Resolution**

| Value | Name | Description |
|---|---|---|
| 0 | BITS_12 | 12-bit resolution |
| 1 | BITS_10 | 10-bit resolution |

- **SLEEP: Sleep Mode**

| Value | Name | Description |
|---|---|---|
| 0 | NORMAL | Normal Mode: The ADC Core and reference voltage circuitry are kept ON between conversions |
| 1 | SLEEP | Sleep Mode: The ADC Core and reference voltage circuitry are OFF between conversions |

### 44.7.7 DACC Interrupt Enable Register

**Name:** DACC_IER

**Address:** 0x400C8024

**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 |
| – | – | – | – | TXBUFE | ENDTX | EOC | TXRDY |

- **TXRDY: Transmit Ready Interrupt Enable**
- **EOC: End of Conversion Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**

**Table 45-13. Active Power Consumption with VDDCORE @ 1.62V Running from Flash Memory or SRAM**

| | CoreMark | | | | | | |
|---|---|---|---|---|---|---|---|
| | 128-bit Flash Access(2) | | 64-bit Flash Access(2) | | SRAM | | |
| Core Clock (MHz) | AMP1 | AMP2(1) | AMP1 | AMP2(1) | AMP1 | AMP2(1) | Unit |
| 84 | 57.66 | 12.32 | 51.41 | 12.32 | 50.10 | 13.30 | mA |
| 72 | 51.46 | 12.32 | 46.18 | 12.25 | 43.60 | 13.30 | |
| 60 | 45.98 | 12.27 | 41.97 | 12.30 | 36.80 | 13.30 | |
| 48 | 40.90 | 12.28 | 36.08 | 12.34 | 30.10 | 13.30 | |
| 32 | 31.45 | 12.32 | 27.94 | 12.26 | 21.30 | 13.30 | |
| 24 | 24.73 | 12.39 | 22.32 | 12.30 | 16.50 | 13.30 | |
| 12 | 15.32 | 12.37 | 14.72 | 12.30 | 9.60 | 13.30 | |
| 8 | 12.16 | 12.28 | 11.65 | 12.32 | 7.50 | 13.30 | |
| 4 | 8.79 | 12.30 | 8.01 | 12.29 | 5.20 | 13.30 | |
| 2 | 10.57 | 12.32 | 9.57 | 12.28 | 5.50 | 13.30 | |
| 1 | 6.83 | 12.30 | 6.05 | 12.27 | 3.70 | 13.30 | |
| 0.5 | 4.59 | 12.26 | 4.30 | 12.31 | 2.80 | 13.30 | |
| 0.25 | 3.42 | 12.25 | 3.12 | 12.29 | 2.40 | 13.30 | |
| 0.125 | 2.57 | 12.30 | 2.47 | 12.27 | 2.20 | 13.30 | |
| 0.032 | 1.74 | 12.35 | 1.73 | 12.34 | 1.80 | 13.30 | |

Notes: 1. Internal voltage regulator not used. VDDIO power consumption only.
2. Flash Wait State (FWS) in EEFC_FMR adjusted versus Core Frequency

**Figure 45-11. Active Power Consumption with VDDCORE @ 1.62V Running from Flash Memory or SRAM**