Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	ARM® Cortex®-M4/M4F
Core Size	32-Bit Dual-Core
Speed	120MHz
Connectivity	EBI/EMI, I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, LCD, POR, PWM, WDT
Number of I/O	52
Program Memory Size	512KB (512K 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/atsam4cmp8cc-au

12.4 Cortex-M4 Models

12.4.1 Programmers Model

This section describes the Cortex-M4 programmers model. In addition to the individual core register descriptions, it contains information about the processor modes and privilege levels for software execution and stacks.

12.4.1.1 Processor Modes and Privilege Levels for Software Execution

The processor *modes* are:

- Thread mode
  Used to execute application software. The processor enters the Thread mode when it comes out of reset.
- Handler mode
  Used to handle exceptions. The processor returns to the Thread mode when it has finished exception processing.

The *privilege levels* for software execution are:

- Unprivileged
  The software:
  - Has limited access to the MSR and MRS instructions, and cannot use the CPS instruction
  - Cannot access the System Timer, NVIC, or System Control Block
  - Might have a restricted access to memory or peripherals.

Unprivileged software executes at the unprivileged level.

- Privileged
  The software can use all the instructions and has access to all resources. *Privileged software* executes at the privileged level.

In Thread mode, the Control Register controls whether the software execution is privileged or unprivileged, see “Control Register”. In Handler mode, software execution is always privileged.

Only privileged software can write to the Control Register to change the privilege level for software execution in Thread mode. Unprivileged software can use the SVC instruction to make a *supervisor call* to transfer control to privileged software.

12.4.1.2 Stacks

The processor uses a full descending stack. This means the stack pointer holds the address of the last stacked item in memory When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks, the *main stack* and the *process stack*, with a pointer for each held in independent registers, see “Stack Pointer”.

In Thread mode, the Control Register controls whether the processor uses the main stack or the process stack, see “Control Register”.

In Handler mode, the processor always uses the main stack.

The options for processor operations are:

Table 12-1. Summary of Processor Mode, Execution Privilege Level, and Stack Use Options

Processor Mode	Used to Execute	Privilege Level for Software Execution	Stack Used
Thread	Applications	Privileged or unprivileged(1)	Main stack or process stack(1)
Handler	Exception handlers	Always privileged	Main stack

Note: 1. See “Control Register”.

12.6.5.1 ADD, ADC, SUB, SBC, and RSB

Add, Add with carry, Subtract, Subtract with carry, and Reverse Subtract.

Syntax

```
op{S}{cond} {Rd,} Rn, Operand2
op{cond} {Rd,} Rn, #imm12                ; ADD and SUB only
```

where:

op is one of:

ADD Add.

ADC Add with Carry.

SUB Subtract.

SBC Subtract with Carry.

RSB Reverse Subtract.

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. If *Rd* is omitted, the destination register is *Rn*.

Rn is the register holding the first operand.

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

imm12 is any value in the range 0–4095.

Operation

The ADD instruction adds the value of *Operand2* or *imm12* to the value in *Rn*.

The ADC instruction adds the values in *Rn* and *Operand2*, together with the carry flag.

The SUB instruction subtracts the value of *Operand2* or *imm12* from the value in *Rn*.

The SBC instruction subtracts the value of *Operand2* from the value in *Rn*. If the carry flag is clear, the result is reduced by one.

The RSB instruction subtracts the value in *Rn* from the value of *Operand2*. This is useful because of the wide range of options for *Operand2*.

Use ADC and SBC to synthesize multiword arithmetic, see “*Multiword arithmetic examples”* below.

See also “ADR”.

Note: ADDW is equivalent to the ADD syntax that uses the *imm12* operand. SUBW is equivalent to the SUB syntax that uses the *imm12* operand.

Restrictions

In these instructions:

- *Operand2* must not be SP and must not be PC
- *Rd* can be SP only in ADD and SUB, and only with the additional restrictions:
  - *Rn* must also be SP
  - Any shift in *Operand2* must be limited to a maximum of 3 bits using LSL
- *Rn* can be SP only in ADD and SUB

12.6.5.19 UHASX and UHSAX

Unsigned Halving Add and Subtract with Exchange and Unsigned Halving Subtract and Add with Exchange.

Syntax

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

where:

op is one of:

UHASX Add and Subtract with Exchange and Halving.

UHSAX Subtract and Add with Exchange and Halving.

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

Rd is the destination register.

Rn, Rm are registers holding the first and second operands.

Operation

The UHASX instruction:

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

The UHSAX instruction:

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

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the condition code flags.

Examples

```
UHASX  R7, R4, R2 ; Adds top halfword of R4 with bottom halfword of R2
                  ; and writes halved result to top halfword of R7
                  ; Subtracts top halfword of R2 from bottom halfword of
                  ; R7 and writes halved result to bottom halfword of R7
UHSAX  R0, R3, R5 ; Subtracts bottom halfword of R5 from top halfword of
                  ; R3 and writes halved result to top halfword of R0
                  ; Adds top halfword of R5 to bottom halfword of R3 and
                  ; writes halved result to bottom halfword of R0.
```

If the user disables a system handler and the corresponding fault occurs, the processor treats the fault as a hard fault.

The user can write to this register to change the pending or active status of system exceptions. An OS kernel can write to the active bits to perform a context switch that changes the current exception type.

12.12.2 Floating Point Unit (FPU) User Interface

Table 12-42. Floating Point Unit (FPU) Register Mapping

Offset	Register	Name	Access	Reset
0xE000ED88	Coprocessor Access Control Register	CPACR	Read/Write	0x00000000
0xE000EF34	Floating-point Context Control Register	FPCCR	Read/Write	0xC0000000
0xE000EF38	Floating-point Context Address Register	FPCAR	Read/Write	–
–	Floating-point Status Control Register	FPSCR	Read/Write	–
0xE000E01C	Floating-point Default Status Control Register	FPDSCR	Read/Write	0x00000000

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.

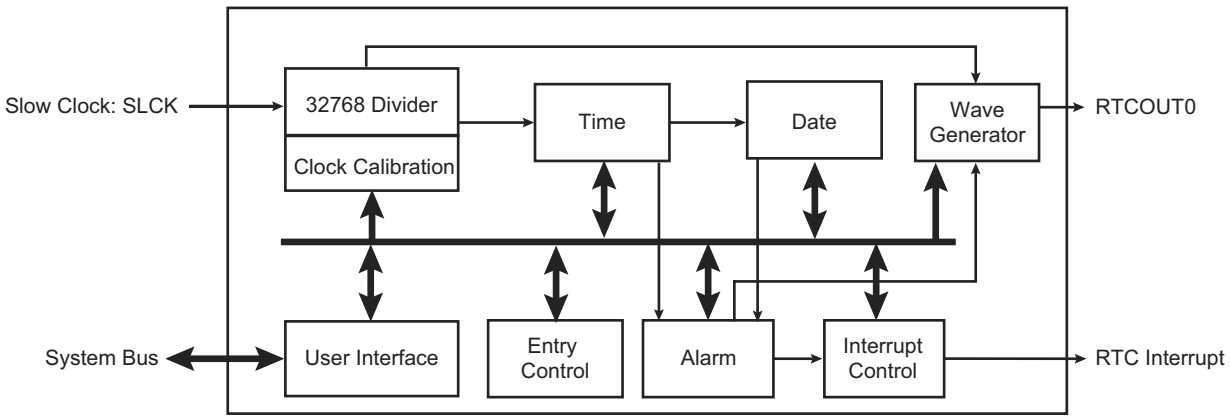
Timestamping capability reports the first and last occurrences of tamper events.

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



- **TDERR: Time and/or Date Free Running Error**

Value	Name	Description
0	CORRECT	The internal free running counters are carrying valid values since the last read of the Status Register (RTC_SR).
1	ERR_TIMEDATE	The internal free running counters have been corrupted (invalid date or time, non-BCD values) since the last read and/or they are still invalid.

20. Supply Controller (SUPC)

20.1 Description

The Supply Controller (SUPC) controls the supply voltages of the system and manages the backup Low-power mode. In this mode, current consumption is reduced to less than 1 µA (typ) for backup power retention. Exit from this mode is possible on multiple wakeup sources. The SUPC also generates the slow clock by selecting either the low-power RC oscillator or the low-power crystal oscillator.

20.2 Embedded Characteristics

- Manages VDDCORE and the Backup Low-power Mode by Controlling the Embedded Voltage Regulator
- Manages the LCD Power Supply VDDLCD and the Backup Low-power Mode by Controlling the Embedded LCD Voltage Regulator
- A Supply Monitor Detection on VDDIO or a Brownout Detection on VDDCORE Triggers a System Reset
- A Zero-power Power-on Reset on VDDBU_SW Triggers a System Reset
- Generates the Slow Clock SLCK by Selecting Either the 32 kHz Low-power RC Oscillator or the 32 kHz Low-power Crystal Oscillator
- Supports Multiple Wakeup Sources for Exit from Backup Low-power Mode
  - Force Wakeup Pin, with Programmable Debouncing
  - Up to 13 Wakeup Inputs (including Tamper Inputs), with Programmable Debouncing
  - Real-time Clock Alarm
  - Real-time Timer Alarm
  - Supply Monitor Detection on VDDIO, with Programmable Scan Period and Voltage Threshold

The erase sequence is the following:

1. Erase starts as soon as one of the erase commands and the FARG field are written in EEFC_FCR.
   – For the EPA command, the two lowest bits of the FARG field define the number of pages to be erased (FARG[1:0]):

Table 22-4. EEFC_FCR.FARG Field for EPA Command

FARG[1:0]	Number of pages to be erased with EPA command
0	4 pages (only valid for small 8 KB sectors)
1	8 pages
2	16 pages
3	32 pages (not valid for small 8 KB sectors)

2. When erasing is completed, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.

Three errors can be detected in EEFC_FSR after an erasing sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Lock Error: At least one page to be erased belongs to a locked region. The erase command has been refused, no page has been erased. A command must be run previously to unlock the corresponding region.
- Flash Error: At the end of the erase period, the EraseVerify test of the Flash memory has failed.

22.4.3.4 Lock Bit Protection

Lock bits are associated with several pages in the embedded Flash memory plane. This defines lock regions in the embedded Flash memory plane. They prevent writing/erasing protected pages.

The lock sequence is the following:

1. Execute the ‘Set Lock Bit’ command by writing EEFC_FCR.FCMD with the SLB command and EEFC_FCR.FARG with a page number to be protected.
2. When the locking completes, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.
3. The result of the SLB command can be checked running a ‘Get Lock Bit’ (GLB) command.

Note: The value of the FARG argument passed together with SLB command must not exceed the higher lock bit index available in the product.

Two errors can be detected in EEFC_FSR after a programming sequence:

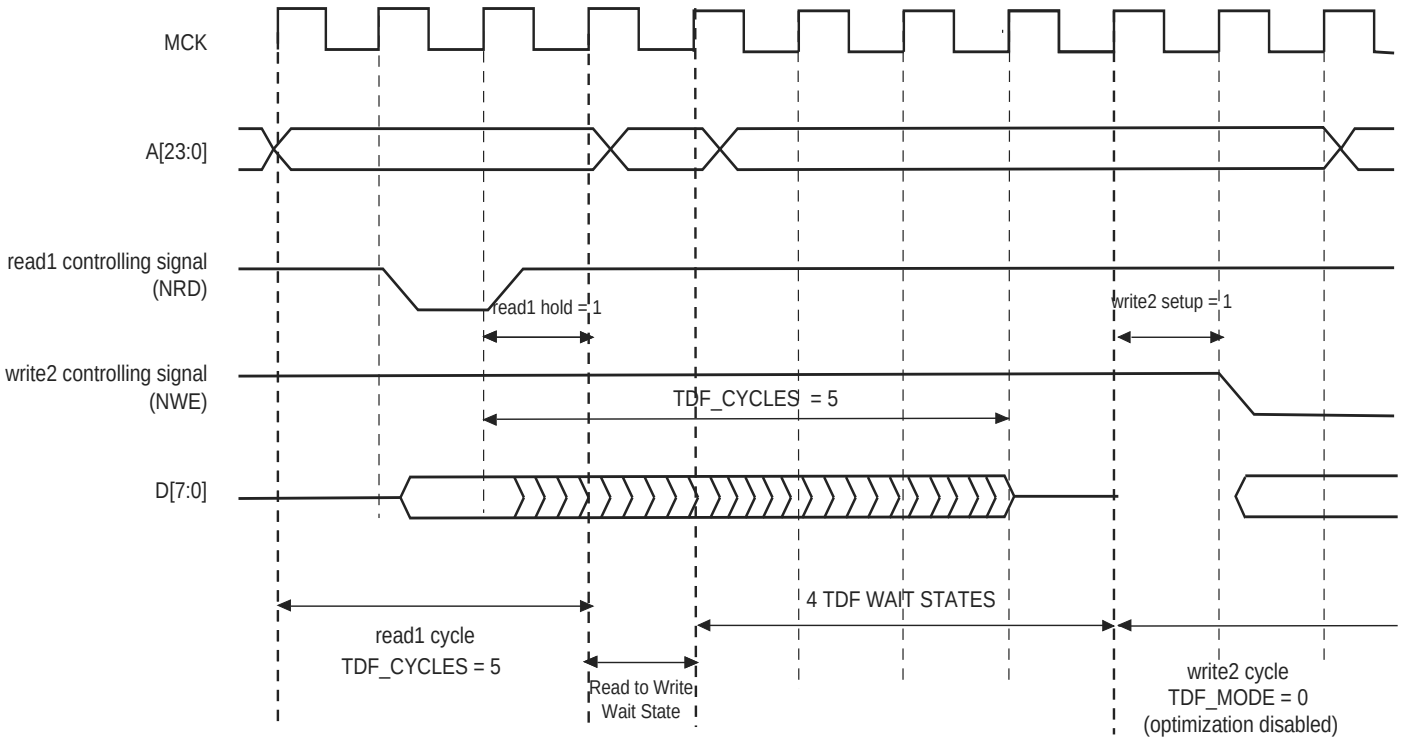
- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the EraseVerify or WriteVerify test of the Flash memory has failed.

It is possible to clear lock bits previously set. After the lock bits are cleared, the locked region can be erased or programmed. The unlock sequence is the following:

1. Execute the ‘Clear Lock Bit’ command by writing EEFC_FCR.FCMD with the CLB command and EEFC_FCR.FARG with a page number to be unprotected.
2. When the unlock completes, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.

Note: The value of the FARG argument passed together with CLB command must not exceed the higher lock bit index available in the product.

Figure 27-24. TDF Mode = 0: TDF wait states between read and write accesses on the same chip select



27.13 External Wait

Any access can be extended by an external device using the NWAIT input signal of the SMC. The EXNW_MODE field of the SMC_MODE register on the corresponding chip select must be set either to “10” (Frozen mode) or “11” (Ready mode). When the EXNW_MODE is set to “00” (disabled), the NWAIT signal is simply ignored on the corresponding chip select. The NWAIT signal delays the read or write operation in regards to the read or write controlling signal, depending on the Read and Write modes of the corresponding chip select.

27.13.1 Restriction

When one of the EXNW_MODE is enabled, **it is mandatory to program at least one hold cycle for the read/write controlling signal. For that reason, the NWAIT signal cannot be used in Page mode (Section 27.15 ”Asynchronous Page Mode”), or in Slow clock mode (Section 27.14 ”Slow Clock Mode”).**

The NWAIT signal is assumed to be a response of the external device to the read/write request of the SMC. Then NWAIT is examined by the SMC only in the pulse state of the read or write controlling signal. The assertion of the NWAIT signal outside the expected period has no impact on SMC behavior.

27.16 Static Memory Controller (SMC) User Interface

The SMC is programmed using the registers listed in Table 27-9. For each chip select, a set of four registers is used to program the parameters of the external device connected on it. In Table 27-9, “CS_number” denotes the chip select number. 16 bytes (0x10) are required per chip select.

The user must complete writing the configuration by writing any one of the SMC_MODE registers.

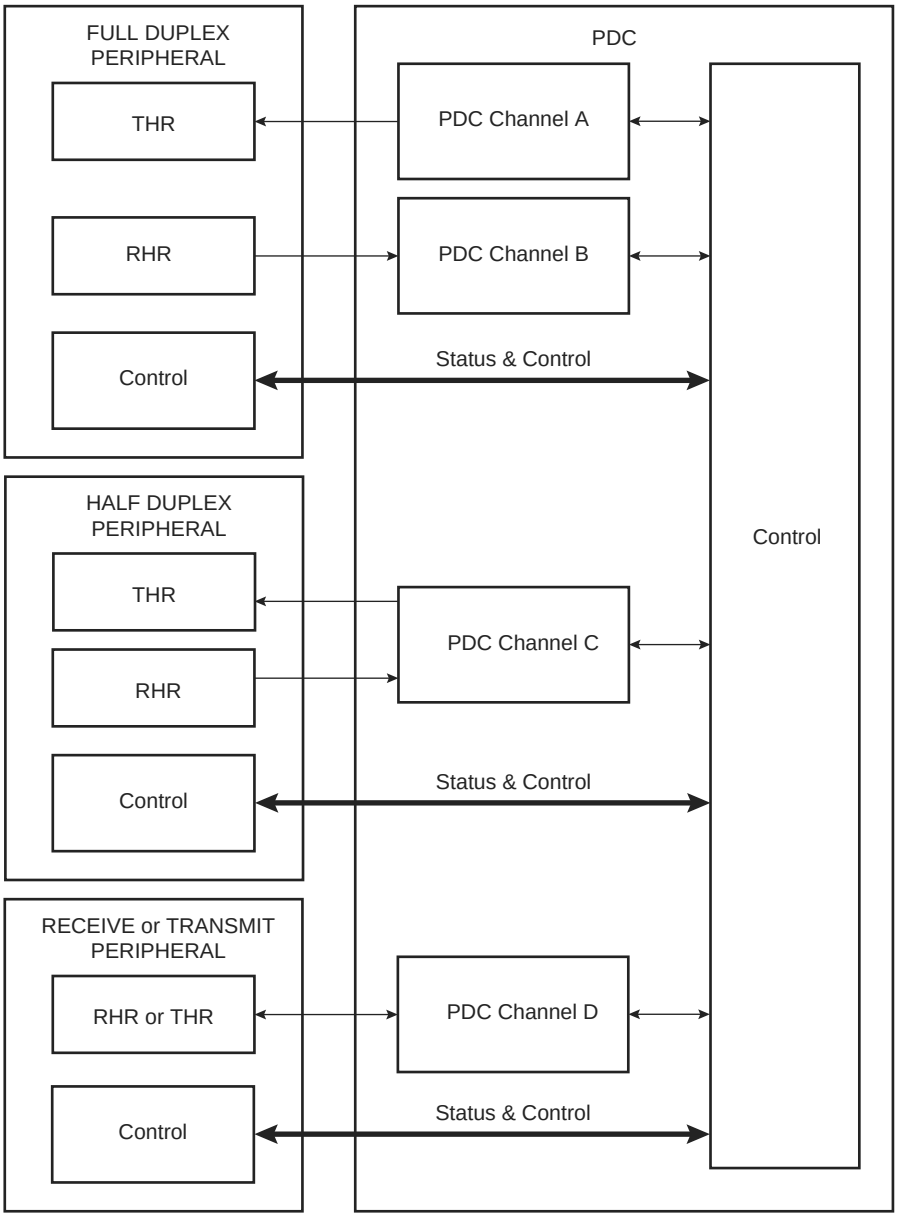
Table 27-9. Register Mapping

Offset	Register	Name	Access	Reset
0x10 x CS_number + 0x00	SMC Setup Register	SMC_SETUP	Read/Write	0x01010101
0x10 x CS_number + 0x04	SMC Pulse Register	SMC_PULSE	Read/write	0x01010101
0x10 x CS_number + 0x08	SMC Cycle Register	SMC_CYCLE	Read/Write	0x00030003
0x10 x CS_number + 0x0C	SMC MODE Register	SMC_MODE	Read/Write	0x10000003
0x80	SMC OCMS MODE Register	SMC_OCMS	Read/Write	0x00000000
0x84	SMC OCMS KEY1 Register	SMC_KEY1	Write Once	0x00000000
0x88	SMC OCMS KEY2 Register	SMC_KEY2	Write Once	0x00000000
0xE4	SMC Write Protection Mode Register	SMC_WPMR	Read/Write	0x00000000
0xE8	SMC Write Protection Status Register	SMC_WPSR	Read-only	0x00000000
0xEC-0xFC	Reserved	–	–	–

Notes: 1. All unlisted offset values are considered as ‘reserved’.

28.3 Block Diagram

Figure 28-1. Block Diagram



Write Sequence

In the case of a write sequence (SVREAD is low), the RXRDY (Receive Holding Register Ready) flag is set as soon as a character has been received in the TWI_RHR (TWI Receive Holding Register). RXRDY is reset when reading the TWI_RHR.

TWI continues receiving data until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the write sequence TXCOMP flag is set and SVACC reset.

See Figure 34-26.

Clock Synchronization Sequence

If TWI_RHR is not read in time, the TWI performs a clock synchronization.

Clock synchronization information is given by the bit SCLWS (Clock Wait State).

See Figure 34-29.

Clock Stretching Sequence

If TWI_THR is not written in time, the TWI performs a clock stretching.

Clock stretching information is given by the bit SCLWS (Clock Wait State).

See Figure 34-28.

General Call

In the case where a GENERAL CALL is performed, the GACC (General Call Access) flag is set.

After GACC is set, the user must interpret the meaning of the GENERAL CALL and decode the new address programming sequence.

See Figure 34-27.

34.7.5.5 Data Transfer

Read Operation

The Read mode is defined as a data requirement from the master.

After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

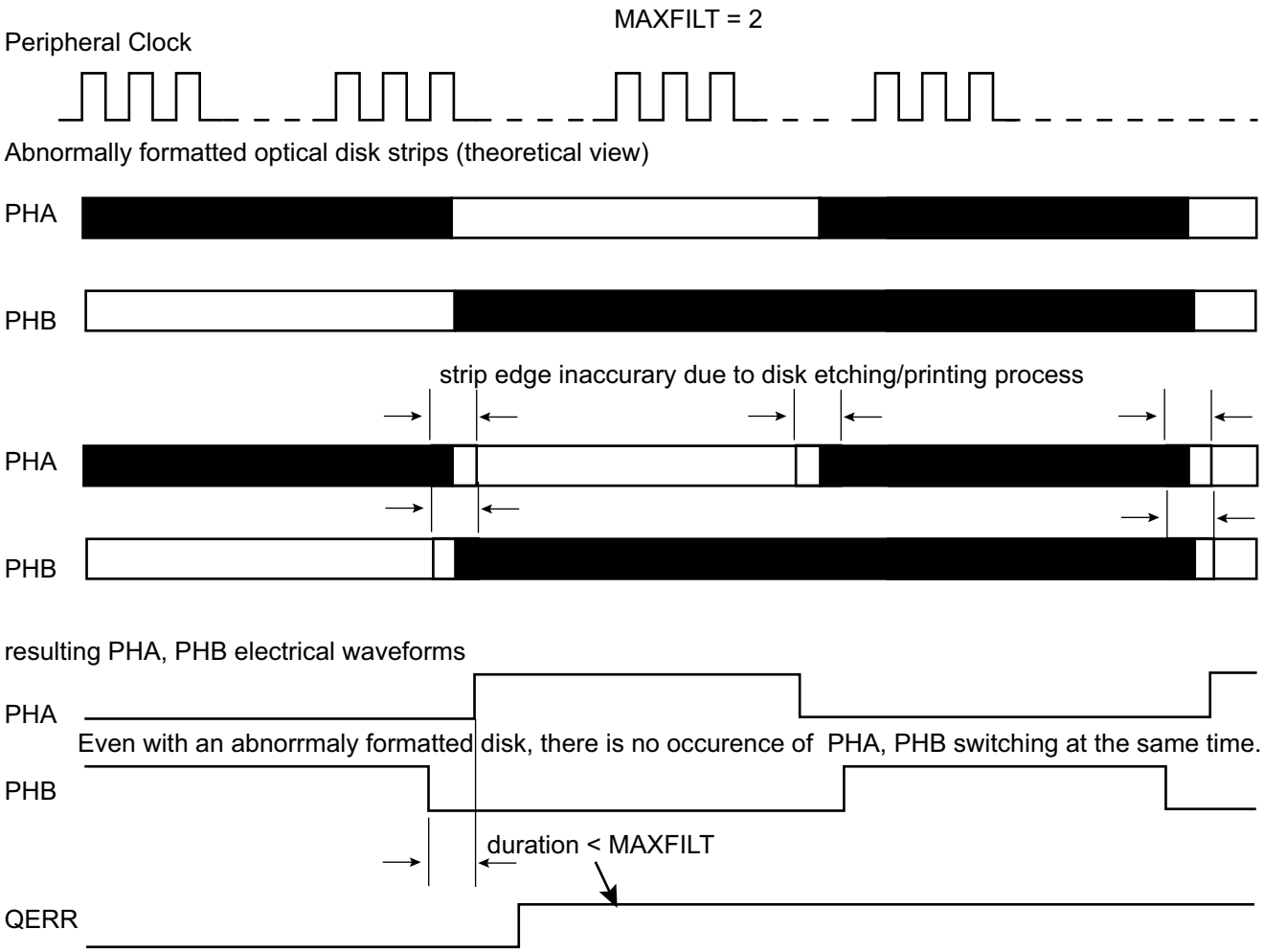
Until a STOP or REPEATED START condition is detected, TWI continues sending data loaded in the TWI_THR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 34-25 describes the write operation.

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 37-19. 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.

37.6.14.4 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.

40.7.10 ADC Interrupt Mask Register

Name: ADC_IMR

Address: 0x4003802C

Access: Read-only

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

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

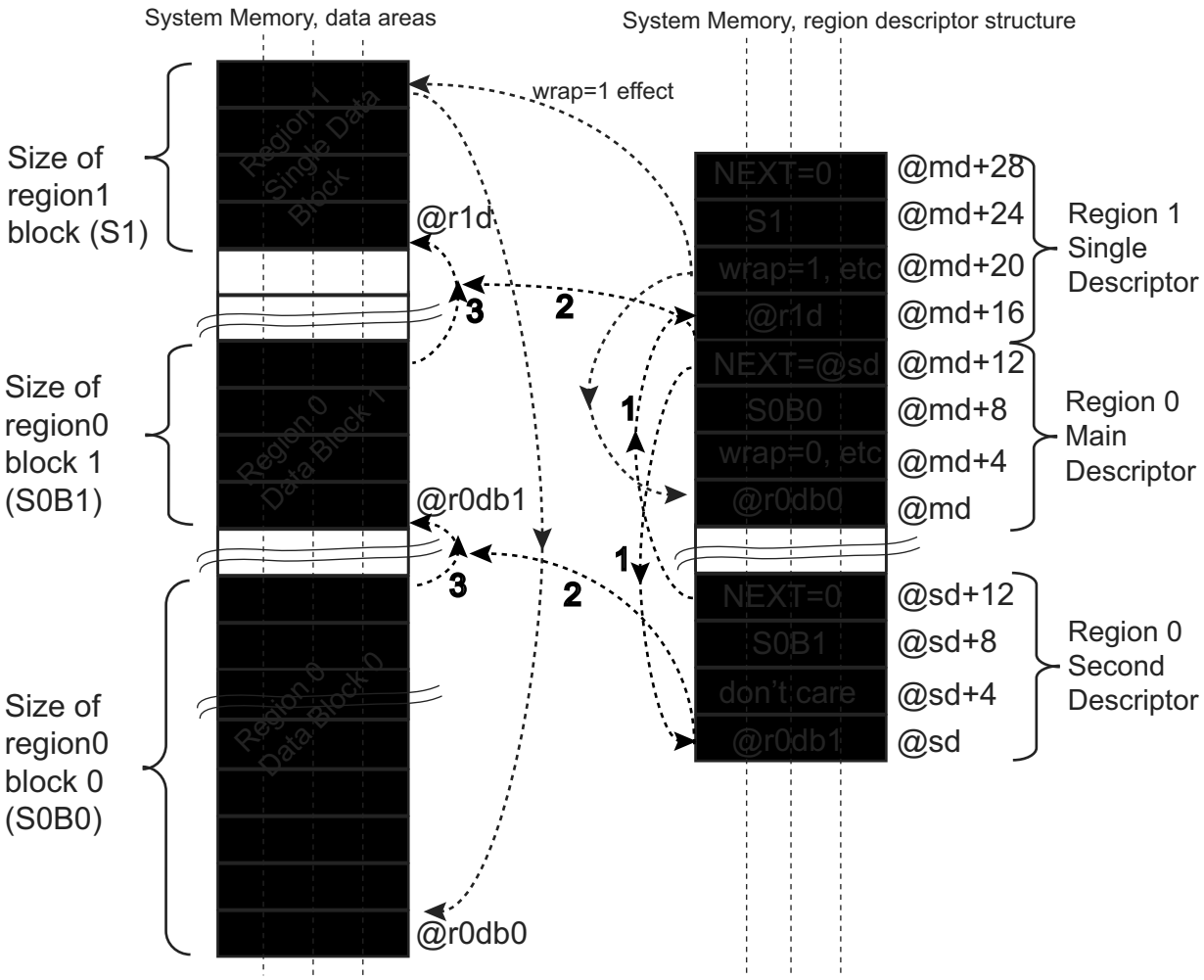
0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

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

Figure 43-5 shows an example of the mandatory ICM settings to monitor three memory data blocks of the system memory (defined as two regions) with one region being not contiguous (two separate areas) and one contiguous memory area. For each said region, the SHA algorithm may be independently selected (different for each region). The wrap allows continuous monitoring.

Figure 43-5. Example: Monitoring of 3 Memory Data Blocks (Defined as 2 Regions)



43.5.2 ICM Region Descriptor Structure

The ICM Region Descriptor Area is a contiguous area of system memory that the controller and the processor can access. When the ICM controller is activated, the controller performs a descriptor fetch operation at *(ICM_DSCR) address. If the Main List contains more than one descriptor (i.e., more than one region is to be monitored), the fetch address is *(ICM_DSCR) + (RID<<4) where RID is the region identifier.

Table 43-2. Region Descriptor Structure (Main List)

Offset	Structure Member	Name
ICM_DSCR+0x000+RID*(0x10)	ICM Region Start Address	ICM_RADDR
ICM_DSCR+0x004+RID*(0x10)	ICM Region Configuration	ICM_RCFG
ICM_DSCR+0x008+RID*(0x10)	ICM Region Control	ICM_RCTRL
ICM_DSCR+0x00C+RID*(0x10)	ICM Region Next Address	ICM_RNEXT

44.2 Product Dependencies

44.2.1 Power Management

The CPKCC is not continuously clocked. The CPCKCC interface is clocked through the Power Management Controller (PMC).

44.2.2 Interrupt Sources

The CPKCC has an interrupt line connected to the Nested Vector Interrupt Controller (NVIC). Handling interrupts requires programming the NVIC before configuring the CPKCC.

44.3 Functional Description

The CPKCC macrocell is managed by the CPKCL Library available in the ROM memory of the microcontroller. The user interface of the CPKCC is not described in this chapter.

The usage description of the CPKCC and its associated Library is provided in a separate document. Contact an Atmel Sales Representative for further details.

51. SAM4CM16/8/4 Errata Revision B (MRL B) Parts

51.1 Device Identification

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

Table 51-1. Device List

Device Marking	Chip ID
ATSAM4CMP16CB-AU	0xA64C_0CE1
ATSAM4CMP16CB-AUR	0xA64C_0CE1
ATSAM4CMP8CB-AU	0xA64C_0AE1
ATSAM4CMP8CB-AUR	0xA64C_0AE1
ATSAM4CMS16CB-AU	0xA64C_0CE1
ATSAM4CMS16CB-AUR	0xA64C_0CE1
ATSAM4CMS8CB-AU	0xA64C_0AE1
ATSAM4CMS8CB-AUR	0xA64C_0AE1
ATSAM4CMS4CB-AU	0xA64C_0CE5
ATSAM4CMS4CB-AUR	0xA64C_0CE5

51.2 Supply Controller (SUPC)

51.2.1 SUPC: Supply Monitor (SM) on VDDIO

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

Problem Fix/Workaround

Use the Supply Monitor in Continuous mode only.

51.2.2 SUPC: Core Voltage Regulator Standby Mode Control

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

Problem Fix/Workaround

None. Do not use the ONREG Bit.

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

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

Problem Fix/Workaround

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

Table 55-1. SAM4CM Datasheet Rev. 11203E Revision History

Doc. Rev. 11203E	Changes
24-Oct-16 (cont'd)	Section 17. "Real-time Clock (RTC)" Updated Section 17.2 "Embedded Characteristics" Reworked Section 17.5.6 "Updating Time/Calendar" Table 17-2 "Register Mapping": added offset 0xCC as reserved Section 17.6.1 "RTC Control Register": updated UPDTIM, UPDCAL and CALEVSEL bit description Deleted "All non-significant bits read zero." from Section 17.6.3 "RTC Time Register", Section 17.6.4 "RTC Calendar Register", Section 17.6.13 "RTC TimeStamp Time Register 0", Section 17.6.14 "RTC TimeStamp Time Register 1", Section 17.6.15 "RTC TimeStamp Date Register" and Section 17.6.16 "RTC TimeStamp Source Register" Added sentence on register report of timestamp to Section 17.6.13 "RTC TimeStamp Time Register 0", Section 17.6.14 "RTC TimeStamp Time Register 1" and Section 17.6.15 "RTC TimeStamp Date Register"
	Section 19. "Reinforced Safety Watchdog Timer (RSWDT)" Removed all references to interrupt by modifying: - Figure 19-1 "Reinforced Safety Watchdog Timer Block Diagram" - Section 19.2 "Embedded Characteristics" - Section 19.4 "Functional Description" - Section 19.5.2 "Reinforced Safety Watchdog Timer Mode Register" (bit WDFIEN now reserved)
	Section 20. "Supply Controller (SUPC)" Corrected all occurrences of "SCLK" to "SLCK" Section 20.2 "Embedded Characteristics": bullet "A Supply Monitor Detection on VDDBU_SW Triggers a System Reset" changed to "A Zero-power Power-on-reset on VDDBU_SW Triggers a System Reset" Section 20.4.8.1 "Supply Monitor Reset": updated second paragraph Section 20.4.9.1 "Force Wakeup": replaced "the FWUP pin must be low during at least one slow clock period to wake up the system" with "the FWUP pin must transition from high to low and remain low during at least one slow clock period to wake up the core power supply" Section 20.5 "Register Write Protection": removed "Supply Controller Wake-up Inputs Register" from list of protectable registers Added sentence about write protection in Section 20.6.3 "Supply Controller Control Register", Section 20.6.4 "Supply Controller Supply Monitor Mode Register", Section 20.6.5 "Supply Controller Mode Register" and Section 20.6.6 "Supply Controller Wakeup Mode Register" Section 20.6.9 "System Controller Write Protection Mode Register": updated WPEN bit description Figure 20-2 "Supply Monitor Status Bit and Associated Interrupt": added SMOS waveform Changed "Low Level Detect" to "Negative Edge Detector" in Figure 20-4 "SAM4CM16/8/4 Wakeup Sources" and Figure 20-5 "SAM4CM32 Wakeup Sources"
	Section 21. "General Purpose Backup Registers (GPBR)" Section 21.1 "Description": reworded parts related to tamper pins Section 21.2 "Embedded Characteristics": added "Immediate Clear on Tamper Event" characteristic
	Section 23. "Fast Flash Programming Interface (FFPI)" Table 23-1 "Signal Description List": updated XIN information Section 23.3.3 "Entering Parallel Programming Mode": deleted note on device clocking. Figure 23-1 "16-bit Parallel Programming Interface": changed input source for XIN
	Section 31. "Chip Identifier (CHIPID)" Updated Table 31-1 "SAM4CM Chip ID Registers" with: • MRL-B for SAM4CMP32C and SAM4CMS32C • MRL-C for SAM4CMP16C, SAM4CMP8C, SAM4CMS16C, SAM4CMS8C and SAM4CMS4C