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	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/atsam4cmp8cb-aur

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Voltage Reference	Comments
Universal Synchronous Asynchronous Receiver Transmitter - USARTx					
SCKx	USARTx Serial Clock	Digital I/O	–	VDDIO	–
TXDx	USARTx Transmit Data	Digital Output	–		–
RXDx	USARTx Receive Data	Digital Input	–		–
RTSx	USARTx Request To Send	Digital Output	–		–
CTSx	USARTx Clear To Send	Digital Input	–		–
Timer/Counter - TC					
TCLKx	TC Channel x External Clock Input	Digital Input	–	VDDIO	–
TIOAx	TC Channel x I/O Line A	Digital I/O	–		–
TIOBx	TC Channel x I/O Line B		–		–
Pulse Width Modulation Controller - PWMC					
PWMx	PWM Waveform Output for channel x	Digital Output	–	VDDIO	–
Serial Peripheral Interface - SPI					
SPI0_MISO	Master In Slave Out	Digital Input	–	VDDIO	–
SPI0_MOSI	Master Out Slave In	Digital Output	–		–
SPCK0	SPI Serial Clock		–		–
SPI0_NPCS0	SPI Peripheral Chip Select 0		Low		NPCS0 is also NSS for Slave mode
SPI0_NPCS1–SPI0_NPCS3	SPI Peripheral Chip Select	Output	Low		–
Segmented LCD Controller - SLCDC					
COM0–COM5	Common Terminals	Output	–	VDDIO	–
SEG0–SEG39	Segment Terminals		–		–
Two-wire Interface - TWI					
TWDx	TWIx Two-wire Serial Data	Digital I/O	–	VDDIO	–
TWCKx	TWIx Two-wire Serial Clock	Digital Output	–		–
Analog					
ADVREF	External Voltage Reference for ADC	Analog Input	–	VDDIN	–

12.4.1.13 Priority Mask Register

Name: PRIMASK

Access: Read/Write

Reset: 0x000000000

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
–							PRIMASK

The PRIMASK register prevents the activation of all exceptions with a configurable priority.

- **PRIMASK**

0: No effect

1: Prevents the activation of all exceptions with a configurable priority.

The Cortex-M4 includes an exclusive access monitor, that tags the fact that the processor has executed a Load-Exclusive instruction. If the processor is part of a multiprocessor system, the system also globally tags the memory locations addressed by exclusive accesses by each processor.

The processor removes its exclusive access tag if:

- It executes a CLREX instruction
- It executes a Store-Exclusive instruction, regardless of whether the write succeeds.
- An exception occurs. This means that the processor can resolve semaphore conflicts between different threads.

In a multiprocessor implementation:

- Executing a CLREX instruction removes only the local exclusive access tag for the processor
- Executing a Store-Exclusive instruction, or an exception, removes the local exclusive access tags, and all global exclusive access tags for the processor.

For more information about the synchronization primitive instructions, see “LDREX and STREX” and “CLREX”.

12.4.2.8 Programming Hints for the Synchronization Primitives

ISO/IEC C cannot directly generate the exclusive access instructions. CMSIS provides intrinsic functions for generation of these instructions:

Table 12-8. CMSIS Functions for Exclusive Access Instructions

Instruction	CMSIS Function
LDREX	uint32_t __LDREXW (uint32_t *addr)
LDREXH	uint16_t __LDREXH (uint16_t *addr)
LDREXB	uint8_t __LDREXB (uint8_t *addr)
STREX	uint32_t __STREXW (uint32_t value, uint32_t *addr)
STREXH	uint32_t __STREXH (uint16_t value, uint16_t *addr)
STREXB	uint32_t __STREXB (uint8_t value, uint8_t *addr)
CLREX	void __CLREX (void)

The actual exclusive access instruction generated depends on the data type of the pointer passed to the intrinsic function. For example, the following C code generates the required LDREXB operation:

```
__ldrex((volatile char *) 0xFF);
```

12.4.3 Exception Model

This section describes the exception model.

12.4.3.1 Exception States

Each exception is in one of the following states:

Inactive

The exception is not active and not pending.

Pending

The exception is waiting to be serviced by the processor.

An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.

Return

This occurs when the exception handler is completed, and:

- There is no pending exception with sufficient priority to be serviced
- The completed exception handler was not handling a late-arriving exception.

The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See “Exception Return” for more information.

Tail-chaining

This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.

Late-arriving

This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. Therefore the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

Exception Entry

An Exception entry occurs when there is a pending exception with sufficient priority and either the processor is in Thread mode, or the new exception is of a higher priority than the exception being handled, in which case the new exception preempts the original exception.

When one exception preempts another, the exceptions are nested.

Sufficient priority means that the exception has more priority than any limits set by the mask registers, see “Exception Mask Registers”. An exception with less priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred as *stacking* and the structure of eight data words is referred to as *stack frame*.

When using floating-point routines, the Cortex-M4 processor automatically stacks the architected floating-point state on exception entry. Figure 12-7 shows the Cortex-M4 stack frame layout when floating-point state is preserved on the stack as the result of an interrupt or an exception.

Note: Where stack space for floating-point state is not allocated, the stack frame is the same as that of ARMv7-M implementations without an FPU. Figure 12-7 shows this stack frame also.

12.5.2.2 Wakeup from WFE

The processor wakes up if:

- It detects an exception with sufficient priority to cause an exception entry
- It detects an external event signal. See “External Event Input”
- In a multiprocessor system, another processor in the system executes an SEV instruction.

In addition, if the SEVONPEND bit in the SCR is set to 1, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause an exception entry. For more information about the SCR, see “System Control Register”.

12.5.2.3 External Event Input

The processor provides an external event input signal. Peripherals can drive this signal, either to wake the processor from WFE, or to set the internal WFE event register to 1 to indicate that the processor must not enter sleep mode on a later WFE instruction. See “Wait for Event” for more information.

12.5.3 Power Management Programming Hints

ISO/IEC C cannot directly generate the WFI and WFE instructions. The CMSIS provides the following functions for these instructions:

```
void __WFE(void) // Wait for Event
void __WFI(void) // Wait for Interrupt
```

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.
```

12.6.11.5 VCVT between Floating-point and Fixed-point

Converts a value in a register from floating-point to and from fixed-point.

Syntax

```
VCVT{cond}.Td.F32 Sd, Sd, #fbits
VCVT{cond}.F32.Td Sd, Sd, #fbits
```

where:

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

Td is the data type for the fixed-point number. It must be one of:

S16 signed 16-bit value.
U16 unsigned 16-bit value.

S32 signed 32-bit value.
U32 unsigned 32-bit value.

Sd is the destination register and the operand register.

fbits is the number of fraction bits in the fixed-point number:

If *Td* is S16 or U16, *fbits* must be in the range 0–16.
If *Td* is S32 or U32, *fbits* must be in the range 1–32.

Operation

These instructions:

1. Either
   - Converts a value in a register from floating-point to fixed-point.
   - Converts a value in a register from fixed-point to floating-point.
2. Places the result in a second register.

The floating-point values are single-precision.

The fixed-point value can be 16-bit or 32-bit. Conversions from fixed-point values take their operand from the low-order bits of the source register and ignore any remaining bits.

Signed conversions to fixed-point values sign-extend the result value to the destination register width.

Unsigned conversions to fixed-point values zero-extend the result value to the destination register width.

The floating-point to fixed-point operation uses the *Round towards Zero* rounding mode. The fixed-point to floating-point operation uses the *Round to Nearest* rounding mode.

Restrictions

There are no restrictions.

Condition Flags

These instructions do not change the flags.

17.4 Product Dependencies

17.4.1 Power Management

The Real-time Clock is continuously clocked at 32.768 kHz. The Power Management Controller has no effect on RTC behavior.

17.4.2 Interrupt

RTC interrupt line is connected on one of the internal sources of the interrupt controller. RTC interrupt requires the interrupt controller to be programmed first.

Table 17-1. Peripheral IDs

Instance	ID
RTC	2

17.5 Functional Description

The RTC provides a full binary-coded decimal (BCD) clock that includes century (19/20), year (with leap years), month, date, day, hours, minutes and seconds reported in RTC Time Register (RTC_TIMR) and RTC Calendar Register (RTC_CALR).

The valid year range is up to 2099 in Gregorian mode (or 1300 to 1499 in Persian mode).

The RTC can operate in 24-hour mode or in 12-hour mode with an AM/PM indicator.

Corrections for leap years are included (all years divisible by 4 being leap years except 1900). This is correct up to the year 2099.

The RTC can generate configurable waveforms on RTCOUT0 output.

17.5.1 Reference Clock

The reference clock is the Slow Clock (SLCK). It can be driven internally or by an external 32.768 kHz crystal.

During low power modes of the processor, the oscillator runs and power consumption is critical. The crystal selection has to take into account the current consumption for power saving and the frequency drift due to temperature effect on the circuit for time accuracy.

17.5.2 Timing

The RTC is updated in real time at one-second intervals in Normal mode for the counters of seconds, at one-minute intervals for the counter of minutes and so on.

Due to the asynchronous operation of the RTC with respect to the rest of the chip, to be certain that the value read in the RTC registers (century, year, month, date, day, hours, minutes, seconds) are valid and stable, it is necessary to read these registers twice. If the data is the same both times, then it is valid. Therefore, a minimum of two and a maximum of three accesses are required.

17.5.3 Alarm

The RTC has five programmable fields: month, date, hours, minutes and seconds.

Each of these fields can be enabled or disabled to match the alarm condition:

- If all the fields are enabled, an alarm flag is generated (the corresponding flag is asserted and an interrupt generated if enabled) at a given month, date, hour/minute/second.
- If only the “seconds” field is enabled, then an alarm is generated every minute.

25.5.5 IPC Interrupt Disable Command Register

Name: IPC_IDCR

Address: 0x4004C010 (0), 0x48014010 (1)

Access: Write-only

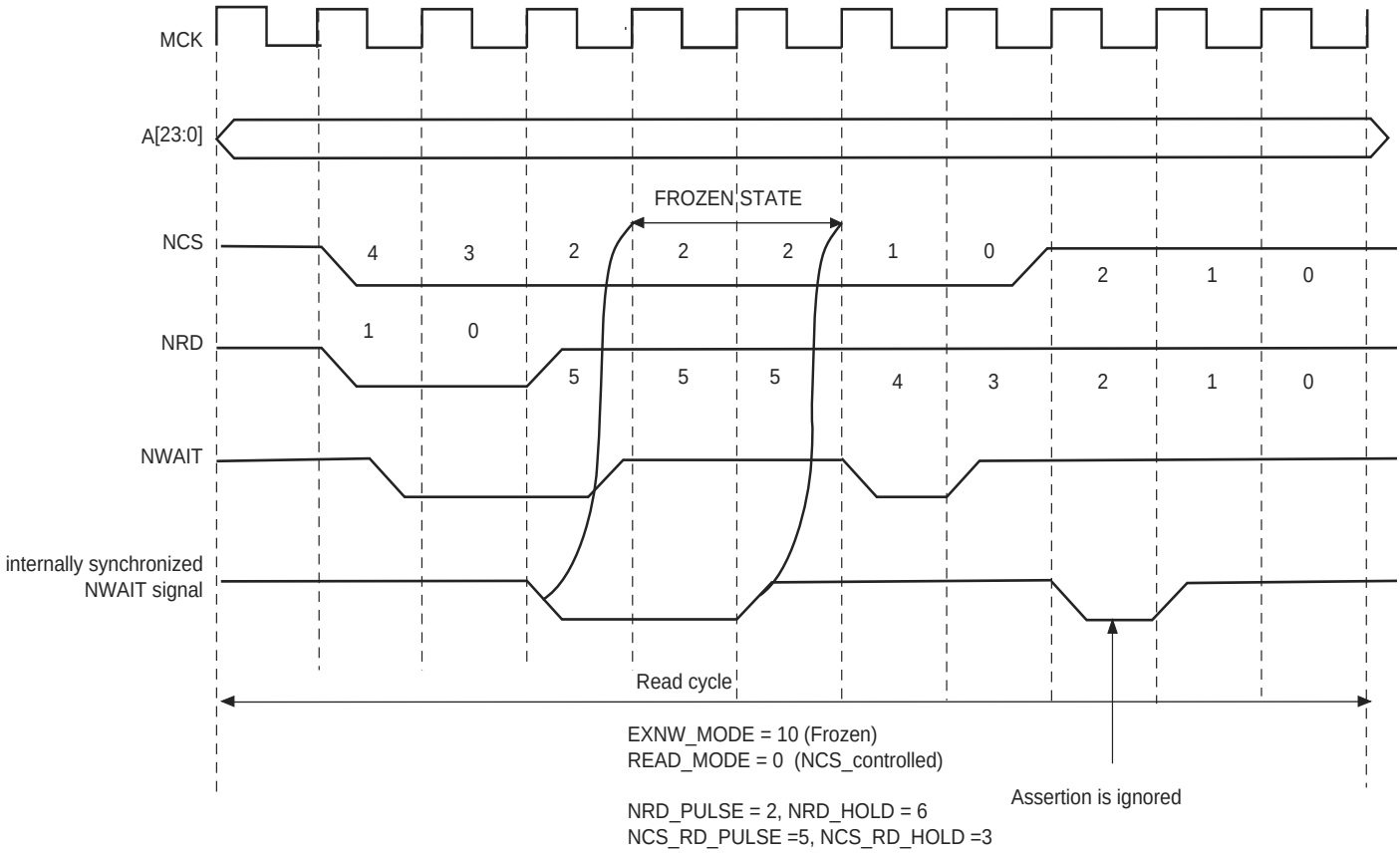
31	30	29	28	27	26	25	24
IRQ31	IRQ30	IRQ29	IRQ28	IRQ27	IRQ26	IRQ25	IRQ24
23	22	21	20	19	18	17	16
IRQ23	IRQ22	IRQ21	IRQ20	IRQ19	IRQ18	IRQ17	IRQ16
15	14	13	12	11	10	9	8
IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8
7	6	5	4	3	2	1	0
IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0

- **IRQ0-IRQ31: Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

Figure 27-26. Read Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)



28.5.10 Transfer Status Register

Name: PERIPH_PTSR

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
–	–	–	–	–	–	–	TXTEN
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RXTEN

- **RXTEN: Receiver Transfer Enable**

0: PDC receiver channel requests are disabled.

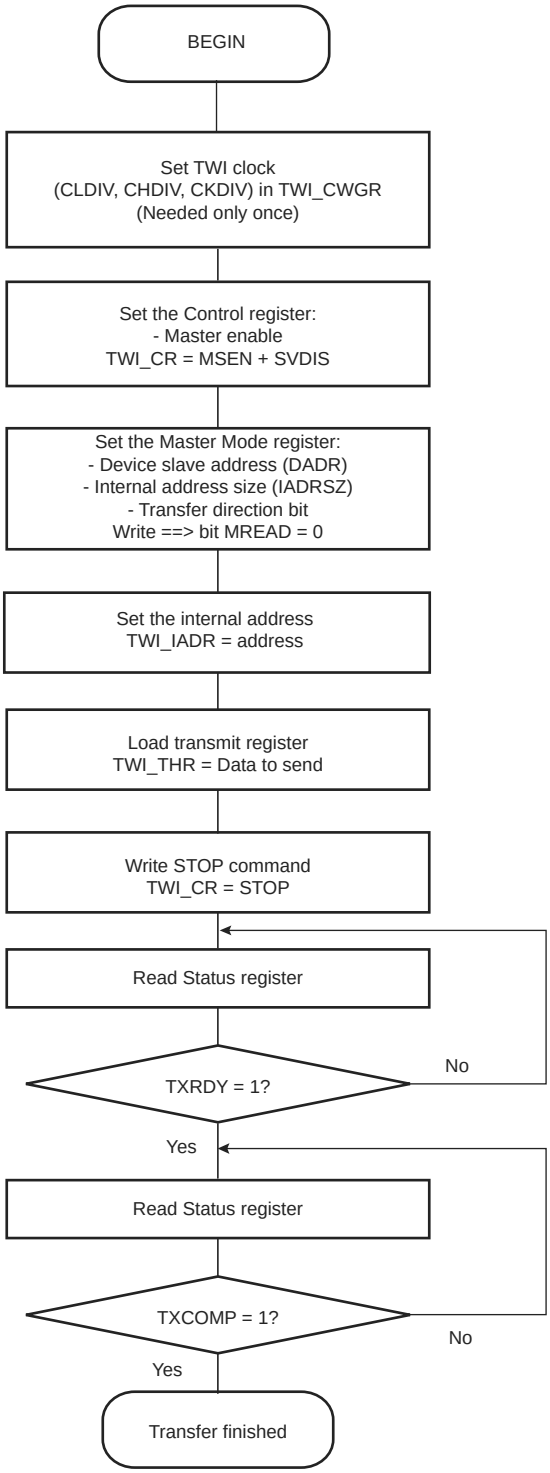
1: PDC receiver channel requests are enabled.

- **TXTEN: Transmitter Transfer Enable**

0: PDC transmitter channel requests are disabled.

1: PDC transmitter channel requests are enabled.

Figure 34-16. TWI Write Operation with Single Data Byte and Internal Address



- **LDBDIS: Counter Clock Disable with RB Loading**

0: Counter clock is not disabled when RB loading occurs.

1: Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

Value	Name	Description
0	NONE	The clock is not gated by an external signal.
1	RISING	Rising edge
2	FALLING	Falling edge
3	EDGE	Each edge

- **ABETRG: TIOA or TIOB External Trigger Selection**

0: TIOB is used as an external trigger.

1: TIOA is used as an external trigger.

- **CPCTRG: RC Compare Trigger Enable**

0: RC Compare has no effect on the counter and its clock.

1: RC Compare resets the counter and starts the counter clock.

- **WAVE: Waveform Mode**

0: Capture mode is enabled.

1: Capture mode is disabled (Waveform mode is enabled).

- **LDRA: RA Loading Edge Selection**

Value	Name	Description
0	NONE	None
1	RISING	Rising edge of TIOA
2	FALLING	Falling edge of TIOA
3	EDGE	Each edge of TIOA

- **LDRB: RB Loading Edge Selection**

Value	Name	Description
0	NONE	None
1	RISING	Rising edge of TIOA
2	FALLING	Falling edge of TIOA
3	EDGE	Each edge of TIOA

37.7.13 TC Block Control Register

Name: TC_BCR

Address: 0x400100C0 (0), 0x400140C0 (1)

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
–	–	–	–	–	–	–	SYNC

- **SYNC: Synchro Command**

0: No effect.

1: Asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.

40.7.4 ADC Channel Enable Register

Name: ADC_CHER

Address: 0x40038010

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
CH7	CH6	–	–	CH3	CH2	CH1	CH0

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

- **CHx: Channel x Enable**

0: No effect.

1: Enables the corresponding channel.

Note: If USEQ = 1 in ADC_MR, CHx corresponds to the xth channel of the sequence described in ADC_SEQR1.

42.4.6.3 GCM Processing

GCM processing comprises three phases:

1. Processing the Additional Authenticated Data (AAD), hash computation only.
2. Processing the Ciphertext (*C*), hash computation + ciphering/deciphering.
3. Generating the Tag using length of *AAD*, length of *C* and J_0 (see NIST documentation for details).

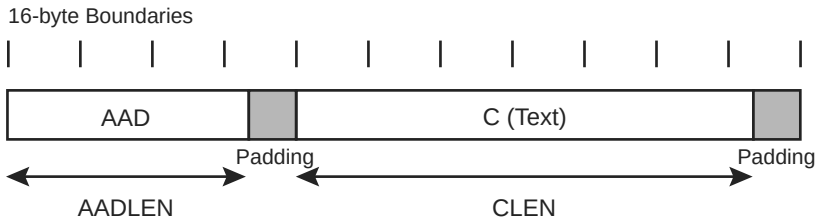
The Tag generation can be done either automatically, after the end of *AAD*/*C* processing if TAG_EN bit is set in the AES_MR or done manually, using the GHASH field in AES_GHASHRx (see below “Processing a Complete Message with Tag Generation” and “Manual GCM Tag Generation” for details).

Processing a Complete Message with Tag Generation

Use this procedure only if J_0 four LSB bytes ≠ 0xFFFFFFFF.

NOTE: In the case where J_0 four LSB bytes = 0xFFFFFFFF or if the value is unknown, use the procedure described in “Processing a Complete Message without Tag Generation” followed by the procedure in “Manual GCM Tag Generation”.

Figure 42-6. Full Message Alignment



To process a complete message with Tag generation, the sequence is as follows:

1. In AES_MR set OPMOD to GCM and GTAGEN to ‘1’ (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. See Section 42.4.6.2 “Key Writing and Automatic Hash Subkey Calculation” for details.
3. Calculate the J_0 value as described in NIST documentation $J_0 = IV \,||\, 0^{31} \,||\, 1$ when len(*IV*) = 96 and $J_0 = \text{GHASH}_H(IV \,||\, 0^{s+64} \,||\, [\text{len}(IV)]_{64})$ if len(*IV*) ≠ 96. See “Processing a Message with only AAD (GHASHH)” for J_0 generation.
4. Set IV in AES_IVRx with inc32(J_0) (J_0 + 1 on 32 bits).
5. Set AADLEN field in AES_AADLENR and CLEN field in AES_CLENR.
6. Fill the IDATA field of AES_IDATARx with the message to process according to the SMOD configuration used. If Manual Mode or Auto Mode is used, the DATRDY bit indicates when the data have been processed (however, no output data are generated when processing *AAD*).
7. Wait for TAGRDY to be set (use interrupt if needed), then read the TAG field of AES_TAGRx to obtain the authentication tag of the message.

Processing a Complete Message without Tag Generation

Processing a message without generating the Tag can be used to customize the Tag generation, or to process a fragmented message. To manually generate the GCM Tag, refer to “Manual GCM Tag Generation”.

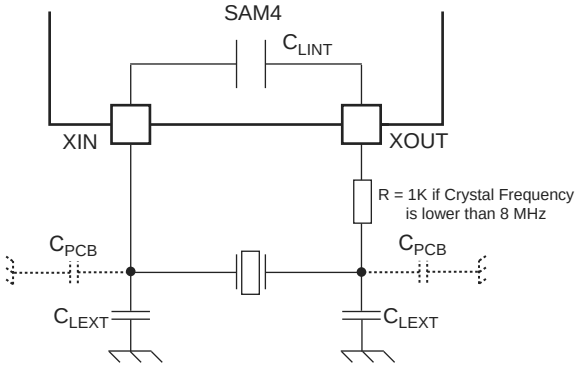
To process a complete message without Tag generation, the sequence is as follows:

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

Table 42-5. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x90	GCM Authentication Tag Word Register 2	AES_TAGR2	Read-only	_
0x94	GCM Authentication Tag Word Register 3	AES_TAGR3	Read-only	_
0x98	GCM Encryption Counter Value Register	AES_CTRR	Read-only	_
0x9C	GCM H Word Register 0	AES_GCMHR0	Read/Write	_
0xA0	GCM H Word Register 1	AES_GCMHR1	Read/Write	_
0xA4	GCM H Word Register 2	AES_GCMHR2	Read/Write	_
0xA8	GCM H Word Register 3	AES_GCMHR3	Read/Write	_
0xAC	Reserved	–	_	_
0xB0–0xFC	Reserved	–	_	_
0x100–0x124	Reserved for the PDC	–	_	_

Figure 46-17. 3- to 20-MHz Crystal Oscillator Schematic



$C_{LEXT} = 2 \times (C_{CRYSTAL} - C_{LINT} - C_{PCB} / 2)$.

where C_{PCB} is the ground referenced parasitic capacitance of the printed circuit board (PCB) on XIN and XOUT tracks. As an example, if the crystal is specified for an 18-pF load, with C_{PCB} = 1 pF (on XIN and on XOUT), C_{LEXT} = 2 x (18 - 9.5 - 0.5) = 16 pF.

Table 46-32 summarizes recommendations to be followed when choosing a crystal.

Table 46-32. Recommended Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor (R_S)	Fundamental @ 3 MHz	–	–	200	Ω
		Fundamental @ 8 MHz			100	
		Fundamental @ 12 MHz			80	
		Fundamental @ 16 MHz			80	
		Fundamental @ 20 MHz			50	
C_M	Motional capacitance	–	–	–	8	fF
C_{SHUNT}	Shunt capacitance	–	–	–	7	pF

46.5.13 Crystal Oscillator Design Considerations

When choosing a crystal for the 32768-Hz slow clock oscillator or for the 3- to 20-MHz oscillator, several parameters must be taken into account. Important parameters are as follows:

- Crystal Load Capacitance.
  The total capacitance loading the crystal, including the oscillator's internal parasitics and the PCB parasitics, must match the load capacitance for which the crystal's frequency is specified. Any mismatch in the load capacitance with respect to the crystal's specification will lead to inaccurate oscillation frequency.
- Crystal Drive Level.
  Use only crystals with the specified drive levels greater than the specified MCU oscillator drive level. Applications that do not respect this criterion may damage the crystal.
- Crystal Equivalent Series Resistor (ESR).
  Use only crystals with the specified ESR lower than the specified MCU oscillator ESR. In applications where this criterion is not respected, the crystal oscillator may not start.
- Crystal Shunt Capacitance.
  Use only crystal with the specified shunt capacitance lower than the specified MCU oscillator shunt capacitance. In applications where this criterion is not respected, the crystal oscillator may not start.
- PCB Layout Considerations.
  To minimize inductive and capacitive parasitics associated with XIN, XOUT, XIN32, XOUT32 nets, it is recommended to route them as short as possible. It is also of prime importance to keep those nets away

from noisy switching signals (clock, data, PWM, etc.). A good practice is to shield them with a quiet ground net to avoid coupling to neighboring signals.

46.5.14 PLLA, PLLB Characteristics

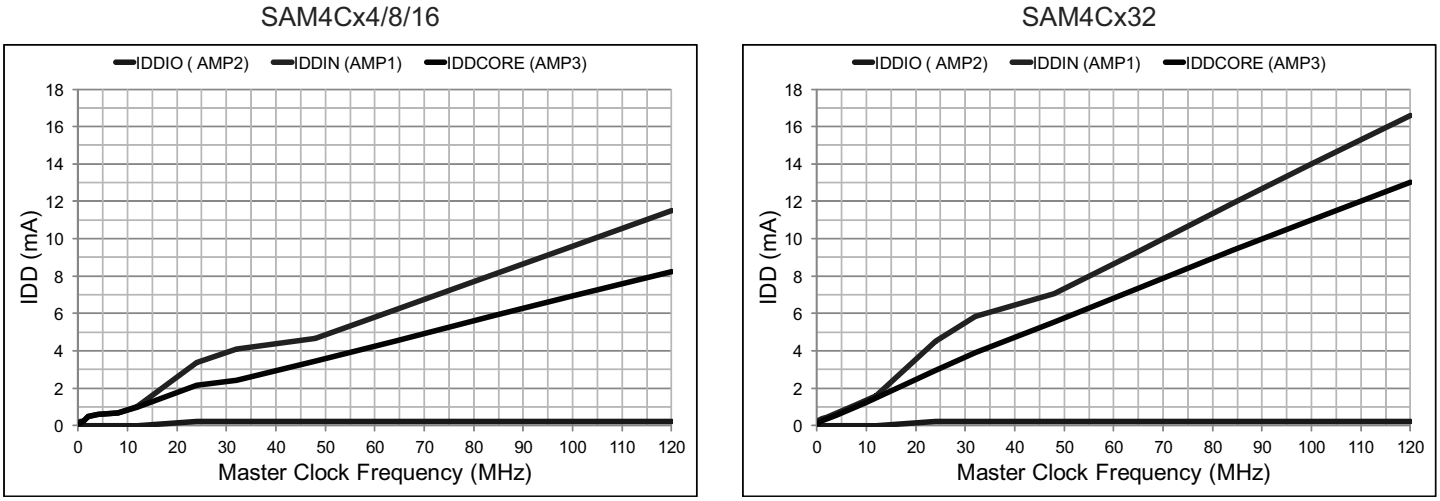
Table 46-33. PLLA Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDPLL}	Supply voltage range (VDDPLL)	–	1.08	1.2	1.32	V
f_{IN}	Input frequency range	–	30	32.768	34	kHz
f_{OUT}	Output frequency range	–	7.5	8.192	8.5	MHz
N_{RATIO}	Frequency multiplying ratio (MULA +1)	–	–	250	–	–
J_P	Period jitter	Peak value	–	4	–	ns
t_{ON}	Start-up time	From OFF to output oscillations (Output frequency within 10% of target frequency)	–	–	250	µs
t_{LOCK}	Lock time	From OFF to PLL locked	–	–	2.5	ms
I_{PLLON}	Active mode current consumption (VDDPLL)	f_{OUT} = 8.192 MHz	–	50	–	µA
I_{PLLOFF}	OFF mode current consumption (VDDPLL)	@25°C Over the temperature range	–	0.05 0.05	0.30 5	µA

Table 46-34. PLLB Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{VDDPLL}	Supply voltage range (VDDPLL)	–	1.08	1.2	1.32	V
f_{IN}	Input frequency range	–	3	–	32	MHz
f_{OUT}	Output frequency range	–	80	–	240	MHz
N_{RATIO}	Frequency multiplying ratio (MULB +1)	–	3	–	62	–
Q_{RATIO}	Frequency dividing ratio (DIVB)	–	2	–	24	–
t_{ON}	Start-up time	–	–	60	150	µs
IDD_{PLL}	Current consumption on VDDPLL	Active mode @ 80 MHz @1.2V Active mode @ 96 MHz @1.2V Active mode @ 160 MHz @1.2V Active mode @ 240 MHz @1.2V	–	0.94 1.2 2.1 3.34	1.2 1.5 2.5 4	mA

Figure 46-25. Typical Current Consumption in Sleep Mode



46.7.4 Active Mode Power Consumption

The current consumption configuration for Active mode, i.e., Core executing codes, is as follows:

- VDDIO = VDDIN = 3.3V
- VDDCORE = 1.2V (internal voltage regulator used)
- T_A = 25°C
- Sub-system 0 Master Clock (MCK), Sub-system 1 Master Clock (CPBMCK) running at various frequencies (PLLB used for frequencies above 12 MHz, fast RC oscillator at 12 MHz for the 12 MHz point, and fast RC oscillator at 8 MHz divided by 1/2/4/8/16/32 for lower frequencies)
- All peripheral clocks deactivated
- No activity on IO lines
- Flash Wait State (FWS) in EEFC_FMR adjusted versus core frequency
- Current measurement as per Figure 46-26

Figure 46-26. Measurement Setup for Active Mode