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	57
Program Memory Size	1MB (1M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4cms16cb-aur

- Debug
  - Star Topology AHB-AP Debug Access Port Implementation with Common SW-DP / SWJ-DP Providing Higher Performance than Daisy-chain Topology
  - Debug Synchronization between both Cores (cross triggering to/from each core for Halt and Run Mode)
- I/O
  - Up to 57 I/O lines with External Interrupt Capability (edge or level sensitivity), Schmitt Trigger, Internal Pull-up/pull-down, Debouncing, Glitch Filtering and On-die Series Resistor Termination
- Package
  - 100-lead LQFP, 14 x 14 mm, pitch 0.5 mm

Note: 1. 120 MHz: -40°C/+85°C, VDDCORE = 1.2V

Examples

```
SMUAD    R0, R4, R5  ; Multiplies bottom halfword of R4 with the bottom
                     ; halfword of R5, adds multiplication of top halfword
                     ; of R4 with top halfword of R5, writes to R0
SMUADX   R3, R7, R4  ; Multiplies bottom halfword of R7 with top halfword
                     ; of R4, adds multiplication of top halfword of R7
                     ; with bottom halfword of R4, writes to R3
SMUSD    R3, R6, R2  ; Multiplies bottom halfword of R4 with bottom halfword
                     ; of R6, subtracts multiplication of top halfword of R6
                     ; with top halfword of R3, writes to R3
SMUSDX   R4, R5, R3  ; Multiplies bottom halfword of R5 with top halfword of
                     ; R3, subtracts multiplication of top halfword of R5
                     ; with bottom halfword of R3, writes to R4.
```

12.6.6.10 SMUL and SMULW

Signed Multiply (halfwords) and Signed Multiply (word by halfword)

Syntax

op{*XY*}{*cond*} *Rd*,*Rn*, *Rm*
op{*Y*}{*cond*} *Rd*. *Rn*, *Rm*

For *SMULXY* only:

op is one of:

SMUL{*XY*} Signed Multiply (halfwords).

X and *Y* specify which halfword of the source registers *Rn* and *Rm* is used as the first and second multiply operand.
If *X* is B, then the bottom halfword, bits [15:0] of *Rn* is used.
If *X* is T, then the top halfword, bits [31:16] of *Rn* is used.If *Y* is B, then the bot tom halfword, bits [15:0], of *Rm* is used.
If *Y* is T, then the top halfword, bits [31:16], of *Rm* is used.

SMULW{Y} Signed Multiply (word by halfword).

Y specifies which halfword of the source register *Rm* is used as the second multiply operand.
If *Y* is B, then the bottom halfword (bits [15:0]) of *Rm* is used.
If *Y* is T, then the top halfword (bits [31:16]) of *Rm* is used.

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 SMULBB, SMULTB, SMULBT and SMULTT instructions interprets the values from *Rn* and *Rm* as four signed 16-bit integers. These instructions:

- Multiplies the specified signed halfword, Top or Bottom, values from *Rn* and *Rm*.
- Writes the 32-bit result of the multiplication in *Rd.*

The SMULWT and SMULWB instructions interprets the values from *Rn* as a 32-bit signed integer and *Rm* as two halfword 16-bit signed integers. These instructions:

- Multiplies the first operand and the top, T suffix, or the bottom, B suffix, halfword of the second operand.
- Writes the signed most significant 32 bits of the 48-bit result in the destination register.

12.10.1.4 SysTick Calibration Value Register

Name: SYST_CALIB

Access: Read/Write

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

The SysTick SYST_CSR indicates the SysTick calibration properties.

- **NOREF: No Reference Clock**

It indicates whether the device provides a reference clock to the processor:

0: Reference clock provided.

1: No reference clock provided.

If your device does not provide a reference clock, the SYST_CSR.CLKSOURCE bit reads-as-one and ignores writes.

- **SKEW: TENMS Value Verification**

It indicates whether the TENMS value is exact:

0: TENMS value is exact.

1: TENMS value is inexact, or not given.

An inexact TENMS value can affect the suitability of SysTick as a software real time clock.

- **TENMS: Ten Milliseconds**

The reload value for 10 ms (100 Hz) timing is subject to system clock skew errors. If the value reads as zero, the calibration value is not known.

The TENMS field default value is 0x000030D4 (12500 decimal).

In order to achieve a 1 ms timebase on SystTick, the TENMS field must be programmed to a value corresponding to the processor clock frequency (in kHz) divided by 8.

For example, for devices running the processor clock at 48 MHz, the TENMS field value must be 0x0001770 (48000 kHz/8).

12.11.2.4 MPU Region Base Address Register

Name: MPU_RBAR

Access: Read/Write

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

The MPU_RBAR defines the base address of the MPU region selected by the MPU_RNR, and can update the value of the MPU_RNR.

Write MPU_RBAR with the VALID bit set to 1 to change the current region number and update the MPU_RNR.

- **ADDR: Region Base Address**

Software must ensure that the value written to the ADDR field aligns with the size of the selected region (SIZE field in the MPU_RASR).

If the region size is configured to 4 GB, in the MPU_RASR, there is no valid ADDR field. In this case, the region occupies the complete memory map, and the base address is 0x00000000.

The base address is aligned to the size of the region. For example, a 64 KB region must be aligned on a multiple of 64 KB, for example, at 0x00010000 or 0x00020000.

- **VALID: MPU Region Number Valid**

Write:

0: MPU_RNR not changed, and the processor updates the base address for the region specified in the MPU_RNR, and ignores the value of the REGION field.

1: The processor updates the value of the MPU_RNR to the value of the REGION field, and updates the base address for the region specified in the REGION field.

Always reads as zero.

- **REGION: MPU Region**

For the behavior on writes, see the description of the VALID field.

On reads, returns the current region number, as specified by the MPU_RNR.

- **SIZE: Size of the MPU Protection Region**

The minimum permitted value is 3 (b00010).

The SIZE field defines the size of the MPU memory region specified by the MPU_RNR. as follows:

(Region size in bytes) = $2^{(SIZE+1)}$

The smallest permitted region size is 32B, corresponding to a SIZE value of 4. The table below gives an example of SIZE values, with the corresponding region size and value of N in the MPU_RBAR.

SIZE Value	Region Size	Value of N[1]	Note
b00100 (4)	32 B	5	Minimum permitted size
b01001 (9)	1 KB	10	–
b10011 (19)	1 MB	20	–
b11101 (29)	1 GB	30	–
b11111 (31)	4 GB	b01100	Maximum possible size

Note: 1. In the MPU_RBAR; see “MPU Region Base Address Register”

- **ENABLE: Region Enable**

Note: For information about access permission, see “MPU Access Permission Attributes”.

19.5.3 Reinforced Safety Watchdog Timer Status Register

Name: RSWDT_SR

Address: 0x400E1508

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
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDERR	WDUNF

- **WDUNF: Watchdog Underflow**

0: No watchdog underflow occurred since the last read of RSWDT_SR.

1: At least one watchdog underflow occurred since the last read of RSWDT_SR.

- **WDERR: Watchdog Error**

0: No watchdog error occurred since the last read of RSWDT_SR.

1: At least one watchdog error occurred since the last read of RSWDT_SR.

Only one page can be programmed at a time. It is possible to program all the bits of a page (full page programming) or only some of the bits of the page (partial page programming).

Depending on the number of bits to be programmed within the page, the EEFC adapts the write operations required to program the Flash.

When a ‘Write Page’ (WP) command is issued, the EEFC starts the programming sequence and all the bits written at 0 in the latch buffer are cleared in the Flash memory array.

During programming, i.e., until EEFC_FSR.FDRY rises, access to the Flash is not allowed.

Full Page Programming

To program a full page, all the bits of the page must be erased before writing the latch buffer and issuing the WP command. The latch buffer must be written in ascending order, starting from the first address of the page. See Figure 22-8 "Full Page Programming".

Partial Page Programming

To program only part of a page using the WP command, the following constraints must be respected:

- Data to be programmed must be contained in integer multiples of 64-bit address-aligned words.
- 64-bit words can be programmed only if all the corresponding bits in the Flash array are erased (at logical value 1).

See Figure 22-9 "Partial Page Programming".

Programming Bytes

Individual bytes can be programmed using the Partial page programming mode.
In this case, an area of 64 bits must be reserved for each byte.

Refer to Figure 22-10 "Programming Bytes in the Flash".

26.9.8 Write Protection Status Register

Name: MATRIX_WPSR

Address: 0x400E03E8 (0), 0x480101E8 (1)

Access: Read-only

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

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

For more information on Write Protection registers, refer to Section 26.8 “Register Write Protection”.

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the MATRIX_WPMR register.

1: A write protection violation has occurred since the last write of the MATRIX_WPMR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

32.6.8 PIO Input Filter Disable Register

Name: PIO_IFDR

Address: 0x400E0E24 (PIOA), 0x400E1024 (PIOB), 0x4800C024 (PIOC)

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

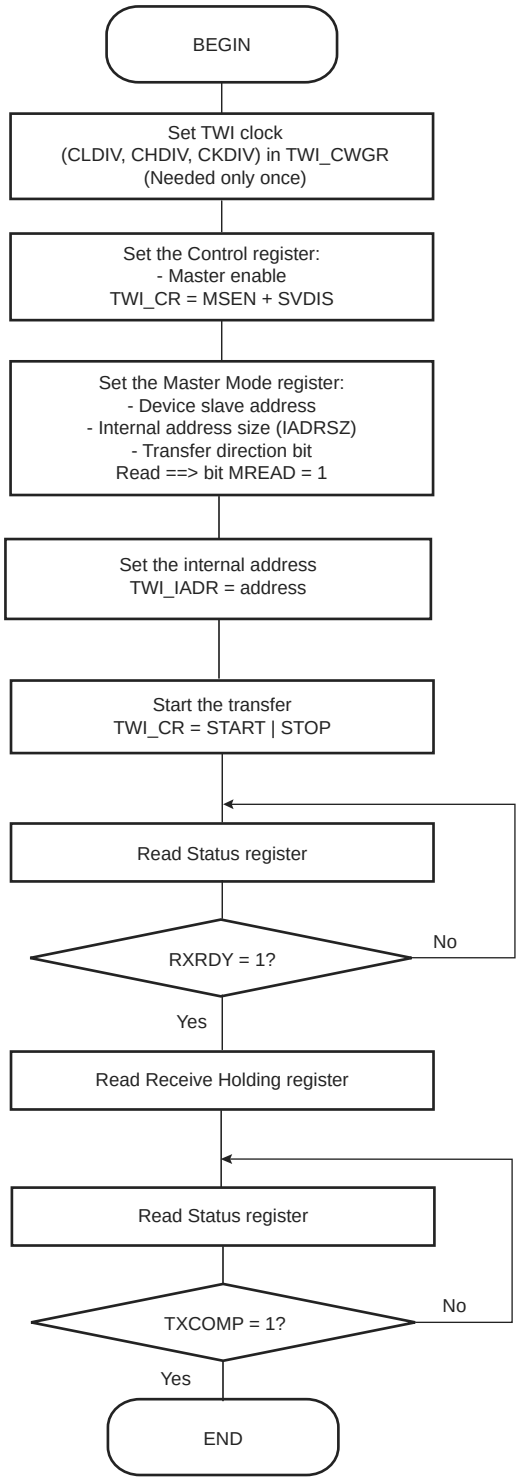
This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Input Filter Disable**

0: No effect.

1: Disables the input glitch filter on the I/O line.

Figure 34-19. TWI Read Operation with Single Data Byte and Internal Address



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.

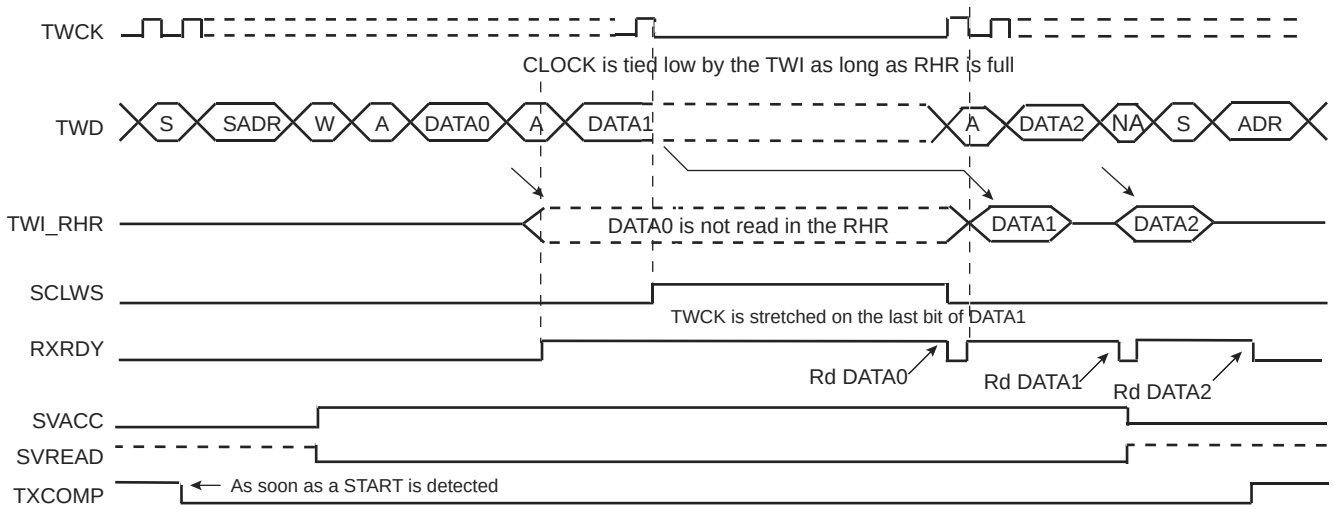
Figure 34-25 describes the write operation.

Clock Synchronization in Write Mode

The clock is tied low outside of the acknowledge phase if the internal shifter and the TWI_RHR is full. If a STOP or REPEATED_START condition was not detected, it is tied low until TWI_RHR is read.

Figure 34-29 describes the clock synchronization in Write mode.

Figure 34-29. Clock Synchronization in Write Mode



Notes:
1. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
2. SCLWS is automatically set when the clock synchronization mechanism is started and automatically reset when the mechanism is finished.

36.5 Product Dependencies

36.5.1 I/O Lines

The pins used for interfacing the USART may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

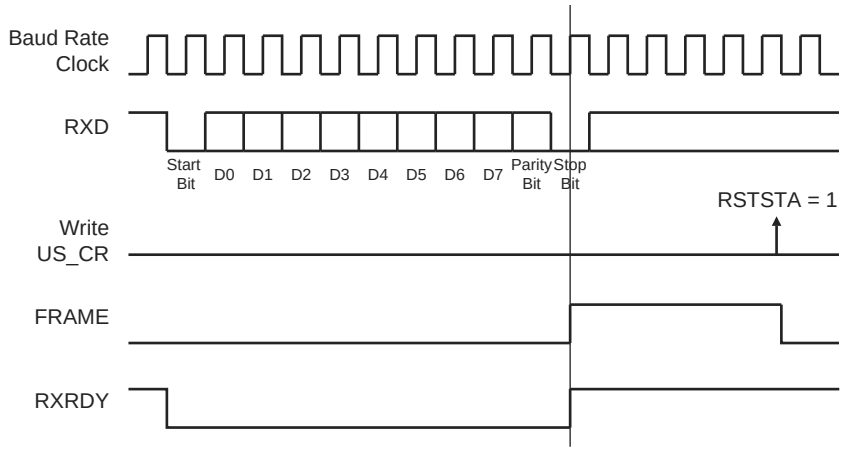
Table 36-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
USART0	CTS0	PA20	A
USART0	RTS0	PA19	A
USART0	RXD0	PB16	A
USART0	SCK0	PB18	A
USART0	TXD0	PB17	A
USART1	CTS1	PA18	A
USART1	RTS1	PA17	A
USART1	RXD1	PA11	A
USART1	SCK1	PA16	A
USART1	TXD1	PA12	A
USART2	CTS2	PA15	A
USART2	RTS2	PA14	A
USART2	RXD2	PA9	A
USART2	SCK2	PA13	A
USART2	TXD2	PA10	A
USART3	CTS3	PA1	A
USART3	RTS3	PA0	A
USART3	RXD3	PA3	A
USART3	SCK3	PA2	A
USART3	TXD3	PA4	A
USART4	CTS4	PA26	A
USART4	RTS4	PB22	A
USART4	RXD4	PB19	A
USART4	SCK4	PB21	A
USART4	TXD4	PB20	A

36.5.2 Power Management

The USART is not continuously clocked. The programmer must first enable the USART clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off.

Figure 36-24. Framing Error Status



36.6.3.13 Transmit Break

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing a 1 to the STTBRK bit in the US_CR. This can be performed at any time, either while the transmitter is empty (no character in either the Shift register or in US_THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBRK command is requested further STTBRK commands are ignored until the end of the break is completed.

The break condition is removed by writing a 1 to the STPBRK bit in the US_CR. If the STPBRK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e., the STTBRK and STPBRK commands are processed only if the TXRDY bit in US_CSR is to 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

Writing US_CR with both STTBRK and STPBRK bits to 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding register while a break is pending, but not started, is ignored.

After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 bit times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 36-25 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

36.7.8 USART Interrupt Disable Register (SPI_MODE)

Name: US_IDR (SPI_MODE)

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

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
–	–	–	RXBUFF	TXBUFE	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
–	–	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

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**
- **ENDRX: End of Receive Buffer Transfer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **UNRE: SPI Underrun Error Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**

37.7.1 TC Channel Control Register

Name: TC_CCRx [x=0..2]

Address: 0x40010000 (0)[0], 0x40010040 (0)[1], 0x40010080 (0)[2], 0x40014000 (1)[0], 0x40014040 (1)[1], 0x40014080 (1)[2]

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
–	–	–	–	–	SWTRG	CLKDIS	CLKEN

- **CLKEN: Counter Clock Enable Command**

0: No effect.

1: Enables the clock if CLKDIS is not 1.

- **CLKDIS: Counter Clock Disable Command**

0: No effect.

1: Disables the clock.

- **SWTRG: Software Trigger Command**

0: No effect.

1: A software trigger is performed: the counter is reset and the clock is started.

39.8.1 SLCDC Control Register

Name: SLCDC_CR

Address: 0x4003C000

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	–	LCDDIS	LCDEN

- **LCDEN: Enable the LCDC**

0: No effect.

1: The SLCDC is enabled.

- **LCDDIS: Disable LCDC**

0: No effect.

1: The SLCDC is disabled.

Note: LCDDIS is processed at the beginning of the next frame.

- **SWRST: Software Reset**

0: No effect.

1: Equivalent to a power-up reset. When this command is performed, the SLCDC immediately ties all segments end commons lines to values corresponding to a “ground voltage”.

- **BIAS: LCD Display Configuration**

(For safety reasons, can be configured when SLCDC is disabled)

Value	Name	Description
0x0	STATIC	Static
0x1	BIAS_1_2	Bias 1/2
0x2	BIAS_1_3	Bias 1/3

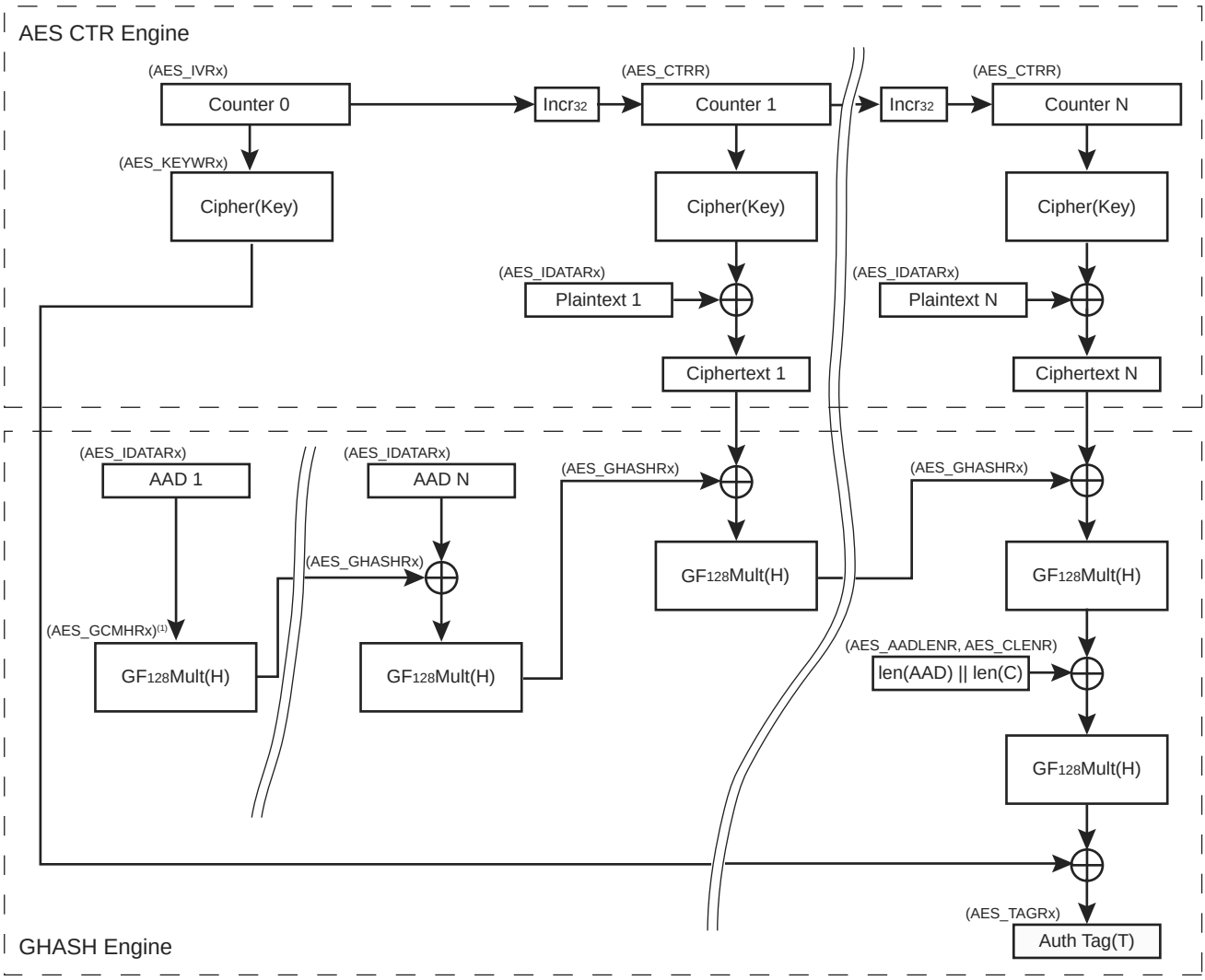
- **LPMODE: Low Power Mode**

(Processed at beginning of next frame)

0: Normal mode.

1: Low Power Waveform is enabled.

Figure 42-5. GCM Block Diagram



Notes: 1. Optional

42.4.6.2 Key Writing and Automatic Hash Subkey Calculation

Whenever a new key (AES_KEYWRx) is written to the hardware, two automatic actions are processed:

- GCM Hash Subkey *H* generation—The GCM hash subkey (*H*) is automatically generated. The GCM hash subkey generation must be complete before doing any other action. The DATRDY bit of the AES_ISR indicates when the subkey generation is complete (with interrupt if configured). The GCM hash subkey calculation is processed with the formula *H* = CIPHER(Key, <128 bits to zero>. The generated GCM *H* value is then available in the AES_GCMHRx. If the application software requires a specific hash subkey, the automatically generated H value can be overwritten in the AES_GCMHRx.
  The AES_GCMHRx can be written after the end of the hash subkey generation (see AES_ISR.DATRDY) and prior to starting the input data feed.
- AES_GHASHRx Clear—The AES_GHASHRx are automatically cleared. If a hash initial value is needed for the GHASH, it must be written to the AES_GHASHRx
  - after a write to AES_KEYWRx, if any
  - before starting the input data feed

Problem Fix/Workaround

Do not use the Windowing mode of the RSWDT and set WDD to 4095 in RSWDT_MR.

50.6 Enhanced Embedded Flash Controller (EEFC)

50.6.1 EEFC: Erase Sector (ES) Command Cannot be Performed if a Subsector is Locked (ONLY in Flash sector 0)

If one of the subsectors

- small sector 0
- small sector 1
- larger sector

is locked within the Flash sector 0, the erase sector (ES) command cannot be processed on non-locked subsectors. Refer to Section 8.1.4.1 “Flash Overview”.

Problem Fix/Workaround

All the lock bits of the sector 0 must be cleared prior to issuing the ES command. After the ES command has been issued, the lock bits must be reverted to the state before clearing them.

50.7 Wait For Interrupt (WFI)

50.7.1 Unpredictable Software Behavior When Entering Sleep Mode

When entering Sleep mode, if an interrupt occurs during WFI or WFE instruction (with PMC_FSMR.LPM=0), the ARM core may read a wrong data, thus leading to unpredictable behavior of the software. This issue is not present in Wait mode.

Problem Fix/Workaround

The slave interface for the Flash must be set to no default master in the Bus Matrix Controller.

This is done by setting the field DEFMSTR_TYPE in the register MATRIX_SCFG to NO_DEFAULT.

```
MATRIX_SCFG[2] = MATRIX_SCFG.SLOT_CYCLE(0x1FF) | MATRIX_SCFG.DEFMSTR_TYPE(0x0);
```

This operation must be done once in the software or the instruction before WFI or WFE.

50.8 Power Supply and Power Control / Clock System

50.8.1 CORE 1 Systick Counter Erratic Behavior

If the CORE 0 processor clock (HCLK) frequency is higher than four times the frequency of the CORE 1 processor clock (CPHCLK), the systick counter behavior is erratic.

Problem Fix/Workaround

Always ensure that $f_{HCLK} < 4 \times f_{CPHCLK}$.