Welcome to E-XFL.COM

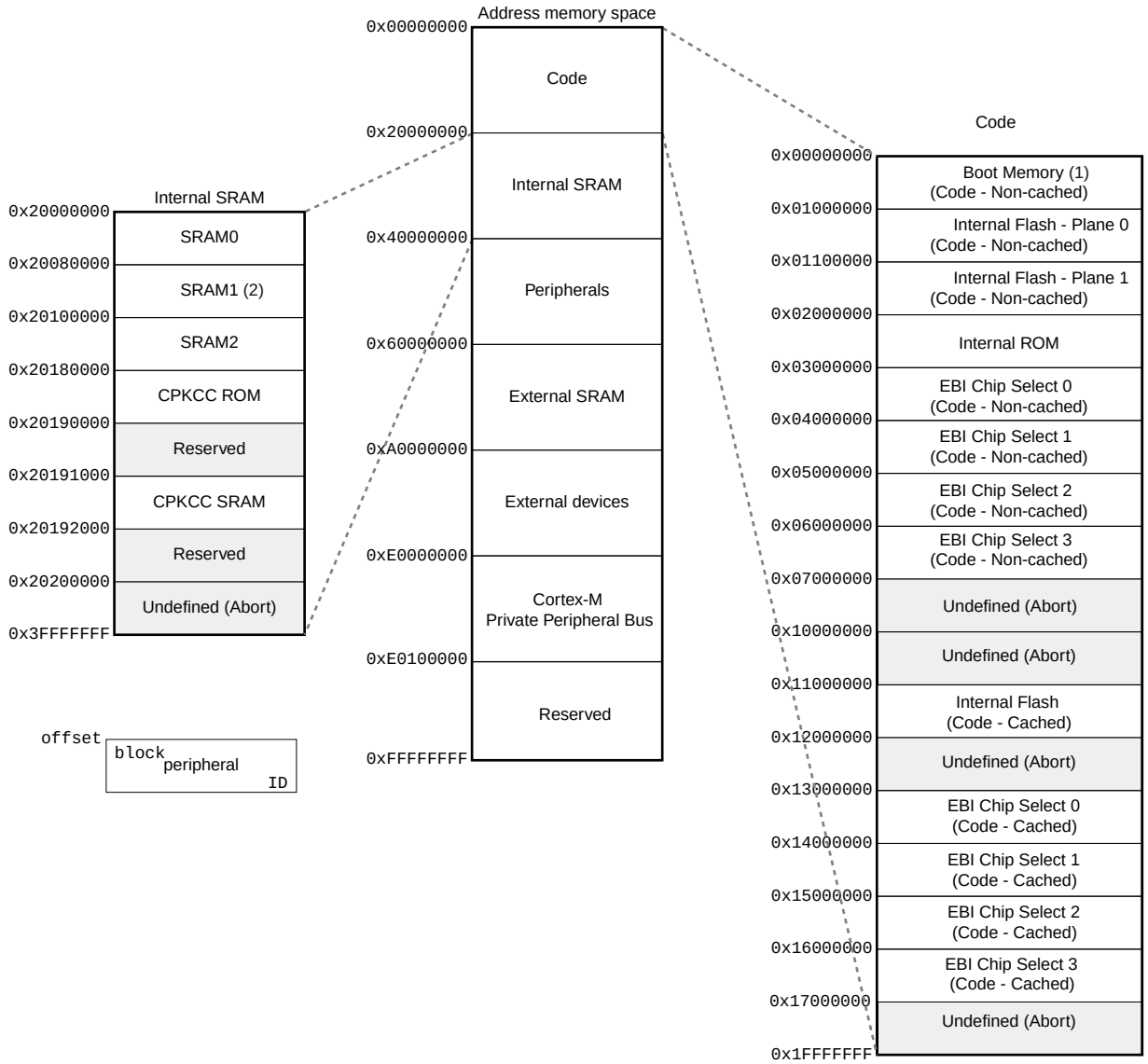
What is "Embedded - Microcontrollers"?

"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	74
Program Memory Size	256KB (256K 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 8x10b
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/atsam4c4cb-au

Figure 7-3. SAM4C32 Memory Mapping of CODE and SRAM Area



Notes: 1. Boot Memory for Core 0.
2. Boot Memory for Core 1 at 0x00000000.

12.4.1.10 Interrupt Program Status Register

Name: IPSR

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
–							ISR_NUMBER

7	6	5	4	3	2	1	0
ISR_NUMBER							

The IPSR contains the exception type number of the current *Interrupt Service Routine* (ISR).

- **ISR_NUMBER: Number of the Current Exception**

0 = Thread mode

1 = Reserved

2 = NMI

3 = Hard fault

4 = Memory management fault

5 = Bus fault

6 = Usage fault

7–10 = Reserved

11 = SVCall

12 = Reserved for Debug

13 = Reserved

14 = PendSV

15 = SysTick

16 = IRQ0

56 = IRQ40

See “Exception Types” for more information.

12.6.3.8 Instruction Width Selection

There are many instructions that can generate either a 16-bit encoding or a 32-bit encoding depending on the operands and destination register specified. For some of these instructions, the user can force a specific instruction size by using an instruction width suffix. The .W suffix forces a 32-bit instruction encoding. The .N suffix forces a 16-bit instruction encoding.

If the user specifies an instruction width suffix and the assembler cannot generate an instruction encoding of the requested width, it generates an error.

Note: In some cases, it might be necessary to specify the .W suffix, for example if the operand is the label of an instruction or literal data, as in the case of branch instructions. This is because the assembler might not automatically generate the right size encoding.

To use an instruction width suffix, place it immediately after the instruction mnemonic and condition code, if any. The example below shows instructions with the instruction width suffix.

```
BCS.W  label      ; creates a 32-bit instruction even for a short
                  ; branch
ADDS.W R0, R0, R1 ; creates a 32-bit instruction even though the same
                  ; operation can be done by a 16-bit instruction
```

12.6.6.4 SMLAD

Signed Multiply Accumulate Long Dual

Syntax

```
op{X}{cond} Rd, Rn, Rm, Ra  ;
```

where:

op is one of:

SMLAD Signed Multiply Accumulate Dual.

SMLADX Signed Multiply Accumulate Dual Reverse.

X specifies which halfword of the source register *Rn* is used as the multiply operand.
If *X* is omitted, the multiplications are bottom × bottom and top × top.
If *X* is present, the multiplications are bottom × top and top × bottom.

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

Rd is the destination register.

Rn is the first operand register holding the values to be multiplied.

Rm the second operand register.

Ra is the accumulate value.

Operation

The SMLAD and SMLADX instructions regard the two operands as four halfword 16-bit values. The SMLAD and SMLADX instructions:

- If *X* is not present, multiply the top signed halfword value in *Rn* with the top signed halfword of *Rm* and the bottom signed halfword values in *Rn* with the bottom signed halfword of *Rm*.
- Or if *X* is present, multiply the top signed halfword value in *Rn* with the bottom signed halfword of *Rm* and the bottom signed halfword values in *Rn* with the top signed halfword of *Rm*.
- Add both multiplication results to the signed 32-bit value in *Ra*.
- Writes the 32-bit signed result of the multiplication and addition to *Rd*.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not change the flags.

Examples

```
SMLAD   R10, R2, R1, R5 ; Multiplies two halfword values in R2 with
                        ; corresponding halfwords in R1, adds R5 and
                        ; writes to R10
SMLALDX R0, R2, R4, R6  ; Multiplies top halfword of R2 with bottom
                        ; halfword of R4, multiplies bottom halfword of R2
                        ; with top halfword of R4, adds R6 and writes to
                        ; R0.
```

12.8.3.2 Interrupt Clear-enable Registers

Name: NVIC_ICERx [x=0..7]

Access: Read/Write

Reset: 0x000000000

31	30	29	28	27	26	25	24
CLRENA							

23	22	21	20	19	18	17	16
CLRENA							

15	14	13	12	11	10	9	8
CLRENA							

7	6	5	4	3	2	1	0
CLRENA							

These registers disable interrupts, and show which interrupts are enabled.

- **CLRENA: Interrupt Clear-enable**

Write:

0: No effect.

1: Disables the interrupt.

Read:

0: Interrupt disabled.

1: Interrupt enabled.

15.5.3 Reset Controller Mode Register

Name: RSTC_MR

Address: 0x400E1408

Access: Read/Write

31	30	29	28	27	26	25	24
KEY							

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

15	14	13	12	11	10	9	8
–	–	–	–	ERSTL			

7	6	5	4	3	2	1	0
–	–	–	URSTIEN	–	–	–	URSTEN

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_WPMR).

- **URSTEN: User Reset Enable**

0: The detection of a low level on the NRST pin does not generate a user reset.

1: The detection of a low level on the NRST pin triggers a user reset.

- **URSTIEN: User Reset Interrupt Enable**

0: USRTS bit in RSTC_SR at 1 has no effect on rstc_irq.

1: USRTS bit in RSTC_SR at 1 asserts rstc_irq if URSTEN = 0.

- **ERSTL: External Reset Length**

This field defines the external reset length. The external reset is asserted during a time of $2^{(ERSTL+1)}$ slow clock cycles. This allows assertion duration to be programmed between 60 µs and 2 seconds. Note that synchronization cycles must also be considered when calculating the actual reset length as previously described.

- **KEY: Write Access Password**

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation. Always reads as 0.

- **LPDBCCLR: Low-power Debouncer Clear**

0 (NOT_ENABLE): A low-power debounce event does not create an immediate clear on the first half of GPBR registers.

1 (ENABLE): A low-power debounce event on WKUP0/TMP0 or WKUP10/14/15/TMP1/2/3 (if DISTMPCLR1/2/3 is cleared) generates an immediate clear on the first half of GPBR registers.

- **FWUPDBC: Force Wakeup Debouncer Period**

Value	Name	Description
0	IMMEDIATE	Immediate, no debouncing, detected active at least on one Slow Clock edge.
1	3_SLCK	FWUP shall be low for at least 3 SLCK periods
2	32_SLCK	FWUP shall be low for at least 32 SLCK periods
3	512_SLCK	FWUP shall be low for at least 512 SLCK periods
4	4096_SLCK	FWUP shall be low for at least 4,096 SLCK periods
5	32768_SLCK	FWUP shall be low for at least 32,768 SLCK periods

- **WKUPDBC: Wakeup Inputs Debouncer Period**

Value	Name	Description
0	IMMEDIATE	Immediate, no debouncing, detected active at least on one Slow Clock edge.
1	3_SLCK	WKUPx shall be in its active state for at least 3 SLCK periods
2	32_SLCK	WKUPx shall be in its active state for at least 32 SLCK periods
3	512_SLCK	WKUPx shall be in its active state for at least 512 SLCK periods
4	4096_SLCK	WKUPx shall be in its active state for at least 4,096 SLCK periods
5	32768_SLCK	WKUPx shall be in its active state for at least 32,768 SLCK periods

- **LPDBC: Low Power Debouncer Period**

Value	Name	Description
0	DISABLE	Disable the low-power debouncers.
1	2_RTCOUT0	WKUP0/10/14/15/TMP0/1/2/3 in active state for at least 2 RTCOUT0 periods
2	3_RTCOUT0	WKUP0/10/14/15/TMP0/1/2/3 in active state for at least 3 RTCOUT0 periods
3	4_RTCOUT0	WKUP0/10/14/15/TMP0/1/2/3 in active state for at least 4 RTCOUT0 periods
4	5_RTCOUT0	WKUP0/10/14/15/TMP0/1/2/3 in active state for at least 5 RTCOUT0 periods
5	6_RTCOUT0	WKUP0/10/14/15/TMP0/1/2/3 in active state for at least 6 RTCOUT0 periods
6	7_RTCOUT0	WKUP0/10/14/15/TMP0/1/2/3 in active state for at least 7 RTCOUT0 periods
7	8_RTCOUT0	WKUP0/10/14/15/TMP0/1/2/3 in active state for at least 8 RTCOUT0 periods

- **LPDBCEN2: Low Power Debouncer Enable WKUP14/TMP2**

0 (NOT_ENABLE): The WKUP14/TMP2 input pin is not connected to the low-power debouncer.

1 (ENABLE): The WKUP14/TMP2 input pin is connected to the low-power debouncer and can force a system wakeup.

- **LPDBCEN3: Low Power Debouncer Enable WKUP15/TMP3**

0 (NOT_ENABLE): The WKUP15/TMP3 input pin is not connected to the low-power debouncer.

1 (ENABLE): The WKUP15/TMP3 input pin is connected to the low-power debouncer and can force a system wakeup.

22.5.4 EEFC Flash Result Register

Name: EEFC_FRR

Address: 0x400E0A0C (0), 0x400E0C0C (1)

Access: Read-only

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

- **FVALUE: Flash Result Value**

The result of a Flash command is returned in this register. If the size of the result is greater than 32 bits, the next resulting value is accessible at the next register read.

8. 128-beat bursts: predetermined end of burst is generated at the end of each 128-beat boundary during INCR transfer.

The use of undefined length 8-beat bursts or less is discouraged since this may decrease the overall bus bandwidth due to arbitration and slave latencies at each first access of a burst.

However, if the usual length of undefined length bursts is known for a master, it is recommended to configure the ULBT according to this length.

This selection can be done through the ULBT field of the Master Configuration registers (MATRIX_MCFG).

26.7.1.2 Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break long accesses, such as very long bursts on a very slow slave (e.g., an external low speed memory). At each arbitration time, a counter is loaded with the value previously written in the SLOT_CYCLE field of the related Slave Configuration register (MATRIX_SCFG) and decreased at each clock cycle. When the counter elapses, the arbiter has the ability to re-arbitrate at the end of the current AHB bus access cycle.

Unless a master has a very tight access latency constraint, which could lead to data overflow or underflow due to a badly undersized internal FIFO with respect to its throughput, the Slot Cycle Limit should be disabled (SLOT_CYCLE = 0) or set to its default maximum value in order not to inefficiently break long bursts performed by some Atmel masters.

In most cases, this feature is not needed and should be disabled for power saving.

Warning: This feature cannot prevent any slave from locking its access indefinitely.

26.7.2 Arbitration Priority Scheme

The Bus Matrix arbitration scheme is organized in priority pools. The corresponding access criticality class is assigned to each priority pool as shown in the “Latency Quality of Service” column in Table 26-7. Latency Quality of Service is determined through the Bus Matrix user interface. See Section 26.9.3 “Bus Matrix Priority Registers A For Slaves” for details.

Table 26-7. Arbitration Priority Pools

Priority Pool	Latency Quality of Service
3	Latency Critical
2	Latency Sensitive
1	Bandwidth Sensitive
0	Background Transfers

Round-robin priority is used in the highest and lowest priority pools 3 and 0, whereas fixed level priority is used between priority pools and in the intermediate priority pools 2 and 1. See Section 26.7.2.2 “Round-robin Arbitration”.

For each slave, each master is assigned to one of the slave priority pools through the Latency Quality of Service inputs or through the priority registers for slaves (MxPR fields of MATRIX_PRAS and MATRIX_PRBS). When evaluating master requests, this priority pool level always takes precedence.

After reset, most of the masters belong to the lowest priority pool (MxPR = 0, Background Transfer) and, therefore, are granted bus access in a true round-robin order.

The highest priority pool must be specifically reserved for masters requiring very low access latency. If more than one master belongs to this pool, they will be granted bus access in a biased round-robin manner which allows tight and deterministic maximum access latency from AHB bus requests. In the worst case, any currently occurring

27. Static Memory Controller (SMC)

27.1 Description

The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the ARM-based microcontroller. The Static Memory Controller (SMC) is part of the EBI.

This SMC can handle several types of external memory and peripheral devices, such as SRAM, PSRAM, PROM, EPROM, EEPROM, LCD Module, NOR Flash and NAND Flash.

The SMC generates the signals that control the access to the external memory devices or peripheral devices. It has 4 Chip Selects, a 24-bit address bus, and a configurable 8 or 16-bit data bus. Separate read and write control signals allow for direct memory and peripheral interfacing. Read and write signal waveforms are fully adjustable.

The SMC can manage wait requests from external devices to extend the current access. The SMC is provided with an automatic Slow clock mode. In Slow clock mode, it switches from user-programmed waveforms to slow-rate specific waveforms on read and write signals. The SMC supports asynchronous burst read in Page mode access for page sizes up to 32 bytes.

The External Data Bus can be scrambled/unscrambled by means of user keys.

27.2 Embedded Characteristics

- Four Chip Selects Available
- 16-Mbyte Address Space per Chip Select
- 8-bit or 16-bit Data Bus
- Zero Wait State Scrambling/Unscrambling Function with User Key
- Word, Halfword, Byte Transfers
- Byte Write or Byte Select Lines
- Programmable Setup, Pulse And Hold Time for Read Signals per Chip Select
- Programmable Setup, Pulse And Hold Time for Write Signals per Chip Select
- Programmable Data Float Time per Chip Select
- External Wait Request
- Automatic Switch to Slow Clock Mode
- Asynchronous Read in Page Mode Supported: Page Size Ranges from 4 to 32 Bytes
- Register Write Protection

30.19.17PMC Interrupt Mask Register

Name: PMC_IMR

Address: 0x400E046C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	XT32KERR	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCXTS

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

0: No effect.

1: Enables the corresponding interrupt.

- **MOSCXTS: 3 to 20 MHz Crystal Oscillator Status Interrupt Mask**
- **LOCKA: PLLA Lock Interrupt Mask**
- **LOCKB: PLLB Lock Interrupt Mask**
- **MCKRDY: Master Clock Ready Interrupt Mask**
- **PCKRDYx: Programmable Clock Ready x Interrupt Mask**
- **MOSCSELS: Main Clock Source Oscillator Selection Status Interrupt Mask**
- **MOSCRCS: 4/8/12 MHz RC Oscillator Status Interrupt Mask**
- **CFDEV: Clock Failure Detector Event Interrupt Mask**
- **XT32KERR: 32.768 kHz Oscillator Error Interrupt Mask**

32.6.37 PIO Additional Interrupt Modes Disable Register

Name: PIO_AIMDR

Address: 0x400E0EB4 (PIOA), 0x400E10B4 (PIOB), 0x4800C0B4 (PIOC), 0x400E12B4 (PIOD)

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

- **P0–P31: Additional Interrupt Modes Disable**

0: No effect.

1: The interrupt mode is set to the default interrupt mode (both-edge detection).

33.8.10 SPI Write Protection Mode Register

Name: SPI_WPMR

Address: 0x400080E4 (0), 0x480000E4 (1)

Access: Read/Write.

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

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x535049 (“SPI” in ASCII)

1: Enables the write protection if WPKEY corresponds to 0x535049 (“SPI” in ASCII)

See Section 33.7.5 “Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x535049	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

35.6.4 UART Interrupt Disable Register

Name: UART_IDR

Address: 0x400E060C (0), 0x4800400C (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
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

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

0: No effect.

1: Disables the corresponding interrupt.

- **RXRDY: Disable RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Disable End of Receive Transfer Interrupt**
- **ENDTX: Disable End of Transmit Interrupt**
- **OVRE: Disable Overrun Error Interrupt**
- **FRAME: Disable Framing Error Interrupt**
- **PARE: Disable Parity Error Interrupt**
- **TXEMPTY: Disable TXEMPTY Interrupt**
- **TXBUFE: Disable Buffer Empty Interrupt**
- **RXBUFF: Disable Buffer Full Interrupt**

36.7.21 USART Manchester Configuration Register

Name: US_MAN

Address: 0x40024050 (0), 0x40028050 (1), 0x4002C050 (2), 0x40030050 (3), 0x40034050 (4)

Access: Read/Write

31	30	29	28	27	26	25	24
–	DRIFT	ONE	RX_MPOL	–	–	RX_PP	

23	22	21	20	19	18	17	16
–	–	–	–	RX_PL			

15	14	13	12	11	10	9	8
–	–	–	TX_MPOL	–	–	TX_PP	

7	6	5	4	3	2	1	0
–	–	–	–	TX_PL			

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

- **TX_PL: Transmitter Preamble Length**

0: The transmitter preamble pattern generation is disabled

1–15: The preamble length is TX_PL × Bit Period

- **TX_PP: Transmitter Preamble Pattern**

The following values assume that TX_MPOL field is not set:

Value	Name	Description
0	ALL_ONE	The preamble is composed of ‘1’s
1	ALL_ZERO	The preamble is composed of ‘0’s
2	ZERO_ONE	The preamble is composed of ‘01’s
3	ONE_ZERO	The preamble is composed of ‘10’s

- **TX_MPOL: Transmitter Manchester Polarity**

0: Logic zero is coded as a zero-to-one transition, Logic one is coded as a one-to-zero transition.

1: Logic zero is coded as a one-to-zero transition, Logic one is coded as a zero-to-one transition.

- **RX_PL: Receiver Preamble Length**

0: The receiver preamble pattern detection is disabled

1–15: The detected preamble length is RX_PL × Bit Period

- **RX_PP: Receiver Preamble Pattern detected**

The following values assume that RX_MPOL field is not set:

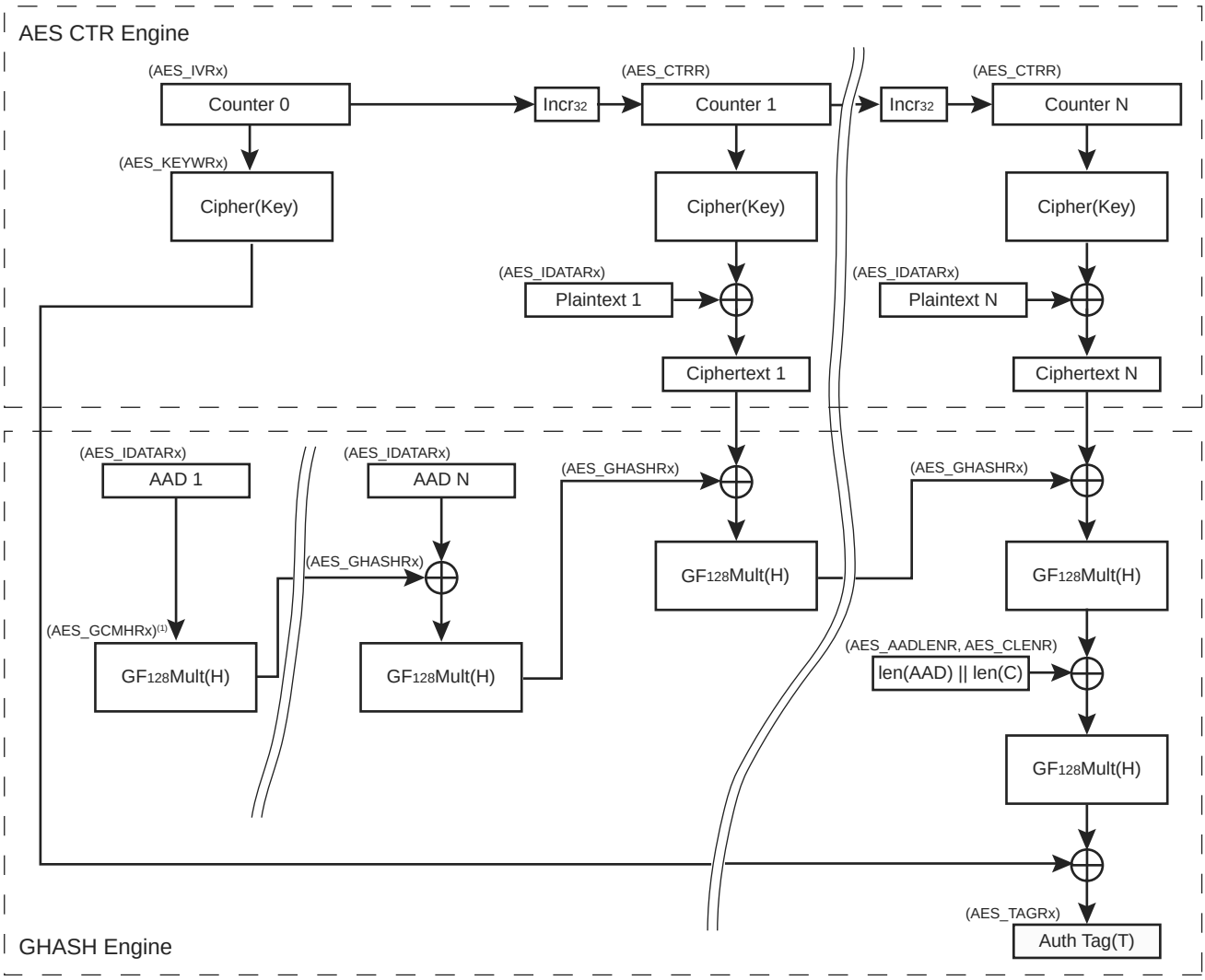
Value	Name	Description
00	ALL_ONE	The preamble is composed of ‘1’s
01	ALL_ZERO	The preamble is composed of ‘0’s
10	ZERO_ONE	The preamble is composed of ‘01’s
11	ONE_ZERO	The preamble is composed of ‘10’s

- **CPD: Channel Update Period**

0 = Writing to the PWM_CUPDx will modify the duty cycle at the next period start event.

1 = Writing to the PWM_CUPDx will modify the period at the next period start event.

Figure 41-5. GCM Block Diagram



Notes: 1. Optional

41.4.6.2 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

41.5.10 AES Initialization Vector Register x

Name: AES_IVRx [x=0..3]

Address: 0x40000060

Access: Write-only

31	30	29	28	27	26	25	24
IV							

23	22	21	20	19	18	17	16
IV							

15	14	13	12	11	10	9	8
IV							

7	6	5	4	3	2	1	0
IV							

- **IV: Initialization Vector**

The four 32-bit Initialization Vector Registers set the 128-bit Initialization Vector data block that is used by some modes of operation as an additional initial input.

AES_IVR0 corresponds to the first word of the Initialization Vector, AES_IVR3 to the last one.

These registers are write-only to prevent the Initialization Vector from being read by another application.

For CBC, OFB and CFB modes, the IV input value corresponds to the initialization vector.

For CTR mode, the IV input value corresponds to the initial counter value.

Note: These registers are not used in ECB mode and must not be written.

The bits RHIEN or ECIEN must be written to 1 in the region descriptor structure member ICM_RCTRL. The flag RHC[*i*], *i* being the region index, is set (if RHIEN is set) when the hash result is available at address defined in ICM_HASH. The flag REC[*i*], *i* being the region index, is set (if ECIEN is set) when the hash result is available at the address defined in ICM_HASH.

An interrupt is generated if the bit RHC[*i*] is written to 1 in the ICM_IER (if RHC[*i*] is set in ICM_RCTRL of region *i*) or if the bit REC[*i*] is written to 1 in the ICM_IER (if REC[*i*] is set in ICM_RCTRL of region *i*).

42.5.4.2 Processing Period

The SHA engine processing period can be configured.

The short processing period allows to allocate bandwidth to the SHA module whereas the long processing period allocates more bandwidth on the system bus to other applications.

In SHA mode, the shortest processing period is 85 clock cycles + 2 clock cycles for start command synchronization. The longest period is 209 clock cycles + 2 clock cycles.

In SHA256 and SHA224 modes, the shortest processing period is 72 clock cycles + 2 clock cycles for start command synchronization. The longest period is 194 clock cycles + 2 clock cycles.

42.5.5 ICM Automatic Monitoring Mode

The ASCD bit of the ICM_CFG register is used to activate the ICM Automatic Mode. When ICM_CFG.ASCD is set, the ICM performs the following actions:

- The ICM controller passes through the Main List once with CDWBN bit in the context register at 0 (i.e., Write Back activated) and EOM bit in context register at 0.
- When WRAP = 1 in ICM_RCFG, the ICM controller enters active monitoring with CDWBN bit in context register now set and EOM bit in context register cleared. Bits CDWBN and EOM in ICM_RCFG have no effect.

42.5.6 Programming the ICM for Multiple Regions

Table 42-8. Region Attributes

Transfer Type		Main List	ICM_RCFG			ICM_RNEXT	Comments
			CDWBN	WRAP	EOM	NEXT	
Single Region	Contiguous list of blocks Digest written to memory Monitoring disabled	1 item	0	0	1	0	The Main List contains only one descriptor. The Secondary List is empty for that descriptor. The digest is computed and saved to memory.
	Non-contiguous list of blocks Digest written to memory Monitoring disabled	1 item	0	0	1	Secondary List address of the current region identifier	The Main List contains only one descriptor. The Secondary List describes the layout of the non-contiguous region.
	Contiguous list of blocks Digest comparison enabled Monitoring enabled	1 item	1	1	0	0	When the hash computation is terminated, the digest is compared with the one saved in memory.

- **EPDISHDMA: Endpoint Interrupts Disable HDMA Request**

This bit is set when USBFS_DEVEPTIERx.EPDISHDMAS = 1. This pauses the on-going DMA channel x transfer on any Endpoint x interrupt (PEP_x), whatever the state of the Endpoint x Interrupt Enable bit (PEP_x).

The user then has to acknowledge or to disable the interrupt source (e.g. USBFS_DEVEPTISRx.RXOUTI) or to clear the EPDISHDMA bit (by writing a one to the USBFS_DEVEPTIDRx.EPDISHDMAC bit) in order to complete the DMA transfer.

In Ping-pong mode, if the interrupt is associated to a new system-bank packet (e.g. Bank1) and the current DMA transfer is running on the previous packet (Bank0), then the previous-packet DMA transfer completes normally, but the new-packet DMA transfer does not start (not requested).

If the interrupt is not associated to a new system-bank packet (USBFS_DEVEPTISRx.NAKINI, NAKOUTI, etc.), then the request cancellation may occur at any time and may immediately pause the current DMA transfer.

This may be used for example to identify erroneous packets, to prevent them from being transferred into a buffer, to complete a DMA transfer by software after reception of a short packet, etc.

- **RSTDT: Reset Data Toggle**

This bit is set when USBFS_DEVEPTIERx.RSTDTS = 1. This clears the data toggle sequence, i.e., sets to Data0 the data toggle sequence of the next sent (IN endpoints) or received (OUT endpoints) packet.

This bit is cleared instantaneously.

The user does not have to wait for this bit to be cleared.