Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

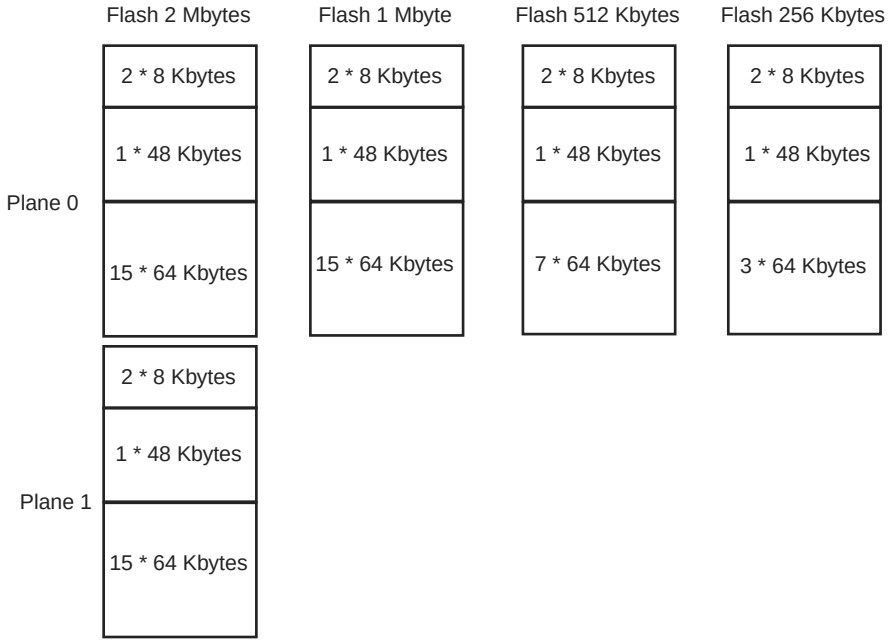
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	57
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/atsam4cms8cc-aur

Figure 8-4 illustrates the organization of the Flash depending on its size.

Figure 8-4. Flash Size



The following erase commands can be used depending on the sector size:

- 8-Kbyte small sector
  - Erase and write page (EWP)
  - Erase and write page and lock (EWPL)
  - Erase sector (ES) with FARG set to a page number in the sector to erase
  - Erase pages (EPA) with FARG [1:0] = 0 to erase four pages or FARG [1:0] = 1 to erase eight pages. FARG [1:0] = 2 and FARG [1:0] = 3 must not be used.
- 48-Kbyte and 64-Kbyte sectors
  - One block of 8 pages inside any sector, with the command Erase pages (EPA) with FARG[1:0] = 1
  - One block of 16 pages inside any sector, with the command Erase pages (EPA) and FARG[1:0] = 2
  - One block of 32 pages inside any sector, with the command Erase pages (EPA) and FARG[1:0] = 3
  - One sector with the command Erase sector (ES) and FARG set to a page number in the sector to erase
- Entire memory plane
  - The entire Flash, with the command Erase all (EA)

8.1.4.2 Enhanced Embedded Flash Controller

The Enhanced Embedded Flash Controller manages accesses performed by masters of the system. It enables reading the Flash and writing the write buffer. It also contains a User Interface, mapped on the APB.

The Enhanced Embedded Flash Controller ensures the interface of the Flash block. It manages the programming, erasing, locking and unlocking sequences of the Flash using the full set of commands.

One of the commands returns the embedded Flash descriptor definition that informs the system about the Flash organization, thus making the software generic.

Restrictions

In these instructions:

- *Rn* must not be PC
- *Rm* must not be SP and must not be PC
- *Rt* can be SP only for word loads and word stores
- *Rt* can be PC only for word loads.

When *Rt* is PC in a word load instruction:

- Bit[0] of the loaded value must be 1 for correct execution, and a branch occurs to this halfword-aligned address
- If the instruction is conditional, it must be the last instruction in the IT block.

Condition Flags

These instructions do not change the flags.

Examples

```
STR    R0, [R5, R1]         ; Store value of R0 into an address equal to
                            ; sum of R5 and R1
LDRSB  R0, [R5, R1, LSL #1] ; Read byte value from an address equal to
                            ; sum of R5 and two times R1, sign extended it
                            ; to a word value and put it in R0
STR    R0, [R1, R2, LSL #2] ; Stores R0 to an address equal to sum of R1
                            ; and four times R2
```

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.

12.6.12.1 BKPT

Breakpoint.

Syntax

```
BKPT #imm
```

where:

imm is an expression evaluating to an integer in the range 0–255 (8-bit value).

Operation

The BKPT instruction causes the processor to enter Debug state. Debug tools can use this to investigate system state when the instruction at a particular address is reached.

imm is ignored by the processor. If required, a debugger can use it to store additional information about the breakpoint.

The BKPT instruction can be placed inside an IT block, but it executes unconditionally, unaffected by the condition specified by the IT instruction.

Condition Flags

This instruction does not change the flags.

Examples

```
BKPT 0xAB   ; Breakpoint with immediate value set to 0xAB (debugger can
            ; extract the immediate value by locating it using the PC)
```

Note: ARM does not recommend the use of the BKPT instruction with an immediate value set to 0xAB for any purpose other than Semi-hosting.

12.6.12.2 CPS

Change Processor State.

Syntax

```
CPSeffect iflags
```

where:

effect is one of:

- IE Clears the special purpose register.
- ID Sets the special purpose register.

iflags is a sequence of one or more flags:

- i Set or clear PRIMASK.
- f Set or clear FAULTMASK.

Operation

CPS changes the PRIMASK and FAULTMASK special register values. See “Exception Mask Registers” for more information about these registers.

Restrictions

The restrictions are:

- Use CPS only from privileged software, it has no effect if used in unprivileged software
- CPS cannot be conditional and so must not be used inside an IT block.

12.11.2.10 MPU Region Base Address Register Alias 3

Name: MPU_RBAR_A3

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.

The value of N depends on the region size. The ADDR field is bits[31:N] of the MPU_RBAR. The region size, as specified by the SIZE field in the MPU_RASR, defines the value of N:

N = Log2(Region size in bytes),

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.

17.6.15 RTC TimeStamp Date Register

Name: RTC_TSDRx

Address: 0x400E1514 [0], 0x400E1520 [1]

Access: Read-only

31	30	29	28	27	26	25	24
–	–	DATE					

23	22	21	20	19	18	17	16
DAY			MONTH				

15	14	13	12	11	10	9	8
YEAR							

7	6	5	4	3	2	1	0
–	CENT						

RTC_TSDR0 reports the timestamp of the first tamper event after reading RTC_TSSR0, and RTC_TSDR1 reports the timestamp of the last tamper event.

This register is cleared by reading RTC_TSSR.

- **CENT: Century of the Tamper**
- **YEAR: Year of the Tamper**
- **MONTH: Month of the Tamper**
- **DAY: Day of the Tamper**
- **DATE: Date of the Tamper**

The fields contain the date and the source of a tamper occurrence if the TEVCNT is not null.

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".

The status of GPNVM bits can be returned by the EEFC. The sequence is the following:

1. Execute the ‘Get GPNVM Bit’ command by writing EEFC_FCR.FCMD with the GGPB command. Field EEFC_FCR.FARG is meaningless.
2. GPNVM bits can be read by the software application in EEFC_FRR. The first word read corresponds to the 32 first GPNVM bits, following reads provide the next 32 GPNVM bits as long as it is meaningful. Extra reads to EEFC_FRR return 0.

For example, if the third bit of the first word read in EEFC_FRR is set, the third GPNVM bit is active.

One error can be detected in EEFC_FSR after a programming sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.

Note: Access to the Flash in read is permitted when a ‘Set GPNVM Bit’, ‘Clear GPNVM Bit’ or ‘Get GPNVM Bit’ command is executed.

22.4.3.6 Calibration Bit

Calibration bits do not interfere with the embedded Flash memory plane.

The calibration bits cannot be modified.

The status of calibration bits are returned by the EEFC. The sequence is the following:

1. Execute the ‘Get CALIB Bit’ command by writing EEFC_FCR.FCMD with the GCALB command. Field EEFC_FCR.FARG is meaningless.
2. Calibration bits can be read by the software application in EEFC_FRR. The first word read corresponds to the first 32 calibration bits. The following reads provide the next 32 calibration bits as long as it is meaningful. Extra reads to EEFC_FRR return 0.

The 4/8/12 MHz fast RC oscillator is calibrated in production. This calibration can be read through the GCALB command. The following table shows the bit implementation for each frequency.

Table 22-5. Calibration Bit Indexes

RC Calibration Frequency	EEFC_FRR Bits
8 MHz output	[28–22]
12 MHz output	[38–32]

The RC calibration for the 4 MHz is set to ‘1000000’.

22.4.3.7 Security Bit Protection

When the security bit is enabled, access to the Flash through the SWD interface or through the Fast Flash Programming interface is forbidden. This ensures the confidentiality of the code programmed in the Flash.

The security bit is GPNVM0.

Disabling the security bit can only be achieved by asserting the ERASE pin at ‘1’, and after a full Flash erase is performed. When the security bit is deactivated, all accesses to the Flash are permitted.

22.4.3.8 Unique Identifier Area

Each device is programmed with a 2x512-bytes unique identifier area. For dual-plane devices, the unique identifier area is accessible on both memory planes.

See Figure 22-1 "Flash Memory Areas".

The sequence to read the unique identifier area is the following:

1. Execute the ‘Start Read Unique Identifier’ command by writing EEFC_FCR.FCMD with the STUI command. Field EEFC_FCR.FARG is meaningless.
2. Wait until the bit EEFC_FSR.FRDY falls to read the unique identifier area. The unique identifier field is located in the first 128 bits of the Flash memory mapping. The ‘Start Read Unique Identifier’ command

22.4.3.10 ECC Errors and Corrections

The Flash embeds an ECC module able to correct one unique error and able to detect two errors. The errors are detected while a read access is performed into memory array and stored in EEFC_FSR (see Section 22.5.3 ”EEFC Flash Status Register”). The error report is kept until EEFC_FSR is read.

There is one flag for a unique error on lower half part of the Flash word (64 LSB) and one flag for the upper half part (MSB). The multiple errors are reported in the same way.

Due to the anticipation technique to improve bandwidth throughput on instruction fetch, a reported error can be located in the next sequential Flash word compared to the location of the instruction being executed, which is located in the previously fetched Flash word.

If a software routine processes the error detection independently from the main software routine, the entire Flash located software must be rewritten because there is no storage of the error location.

If only a software routine is running to program and check pages by reading EEFC_FSR, the situation differs from the previous case. Performing a check for ECC unique errors just after page programming completion involves a read of the newly programmed page. This read sequence is viewed as data accesses and is not optimized by the Flash controller. Thus, in case of unique error, only the current page must be reprogrammed.

23. Fast Flash Programming Interface (FFPI)

23.1 Description

The Fast Flash Programming Interface (FFPI) provides parallel high-volume programming using a standard gang programmer. The parallel interface is fully handshaked and the device is considered to be a standard EEPROM. Additionally, the parallel protocol offers an optimized access to all the embedded Flash functionalities.

Although the Fast Flash Programming mode is a dedicated mode for high volume programming, this mode is not designed for in-situ programming.

23.2 Embedded Characteristics

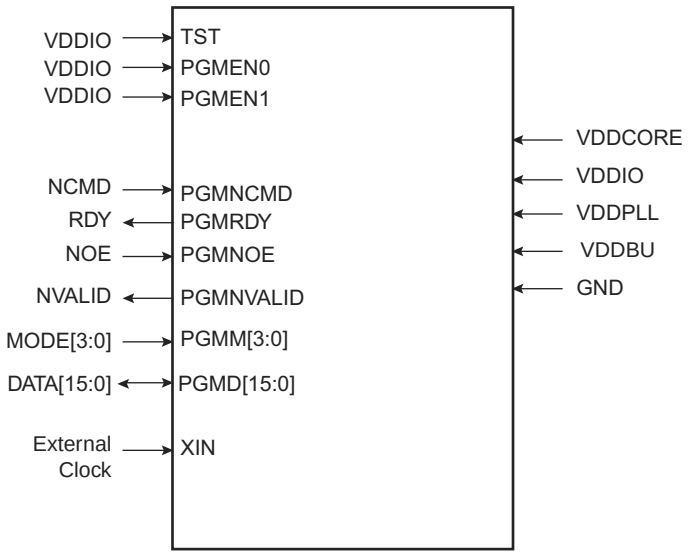
- Programming Mode for High-volume Flash Programming Using Gang Programmer
  - Offers Read and Write Access to the Flash Memory Plane
  - Enables Control of Lock Bits and General-purpose NVM Bits
  - Enables Security Bit Activation
  - Disabled Once Security Bit is Set
- Parallel Fast Flash Programming Interface
  - Provides an 16-bit Parallel Interface to Program the Embedded Flash
  - Full Handshake Protocol

23.3 Parallel Fast Flash Programming

23.3.1 Device Configuration

In Fast Flash Programming mode, the device is in a specific test mode. Only a certain set of pins is significant. The rest of the PIOs are used as inputs with a pull-up. The crystal oscillator is in bypass mode. Other pins must be left unconnected.

Figure 23-1. 16-bit Parallel Programming Interface



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

30.18.16PMC 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**

Value	Name	Description
8	160K	160 Kbytes
9	256K	256 Kbytes
10	512K	512 Kbytes
11	–	Reserved
12	1024K	1024 Kbytes
13	–	Reserved
14	2048K	2048 Kbytes
15	–	Reserved

- **NVPSIZ2: Second Nonvolatile Program Memory Size**

Value	Name	Description
0	NONE	None
1	8K	8 Kbytes
2	16K	16 Kbytes
3	32K	32 Kbytes
4	–	Reserved
5	64K	64 Kbytes
6	–	Reserved
7	128K	128 Kbytes
8	–	Reserved
9	256K	256 Kbytes
10	512K	512 Kbytes
11	–	Reserved
12	1024K	1024 Kbytes
13	–	Reserved
14	2048K	2048 Kbytes
15	–	Reserved

- **SRAMSIZ: Internal SRAM Size**

Value	Name	Description
0	48K	48 Kbytes
1	192K	192 Kbytes
2	384K	384 Kbytes
3	6K	6 Kbytes
4	24K	24 Kbytes
5	4K	4 Kbytes
6	80K	80 Kbytes
7	160K	160 Kbytes
8	8K	8 Kbytes

33.7.3.10 Mode Fault Detection

The SPI has the capability to operate in multi-master environment. Consequently, the NPCS0/NSS line must be monitored. If one of the masters on the SPI bus is currently transmitting, the NPCS0/NSS line is low and the SPI must not transmit any data. A mode fault is detected when the SPI is programmed in Master mode and a low level is driven by an external master on the NPCS0/NSS signal. In multi-master environment, NPCS0, MOSI, MISO and SPCK pins must be configured in open drain (through the PIO controller). When a mode fault is detected, the SPI_SR.MODF bit is set until SPI_SR is read and the SPI is automatically disabled until it is re-enabled by writing a 1 to the SPI_CR.SPIEN bit.

By default, the mode fault detection is enabled. The user can disable it by setting the SPI_MR.MODFDIS bit.

33.7.4 SPI Slave Mode

When operating in Slave mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits until NSS goes active before receiving the serial clock from an external master. When NSS falls, the clock is validated and the data is loaded in the SPI_RDR depending on the BITS field configured in the SPI_CSR0. These bits are processed following a phase and a polarity defined respectively by the NCPHA and CPOL bits in the SPI_CSR0. Note that BITS, CPOL and NCPHA of the other Chip Select registers have no effect when the SPI is programmed in Slave mode.

The bits are shifted out on the MISO line and sampled on the MOSI line.

Note: For more information on the BITS field, see also the note below the SPI_CSRx register bitmap (Section 33.8.9 “SPI Chip Select Register”).

When all bits are processed, the received data is transferred in the SPI_RDR and the RDRF bit rises. If the SPI_RDR has not been read before new data is received, the Overrun Error Status (OVRES) bit in the SPI_SR is set. As long as this flag is set, data is loaded in the SPI_RDR. The user must read SPI_SR to clear the OVRES bit.

When a transfer starts, the data shifted out is the data present in the Shift register. If no data has been written in the SPI_TDR, the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift register resets to 0.

When a first data is written in the SPI_TDR, it is transferred immediately in the Shift register and the TDRE flag rises. If new data is written, it remains in the SPI_TDR until a transfer occurs, i.e., NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in the SPI_TDR is transferred in the Shift register and the TDRE flag rises. This enables frequent updates of critical variables with single transfers.

Then, new data is loaded in the Shift register from the SPI_TDR. If no character is ready to be transmitted, i.e., no character has been written in the SPI_TDR since the last load from the SPI_TDR to the Shift register, the SPI_TDR is retransmitted. In this case the Underrun Error Status Flag (UNDES) is set in the SPI_SR.

Figure 33-13 shows a block diagram of the SPI when operating in Slave mode.

Figure 34-2. START and STOP Conditions

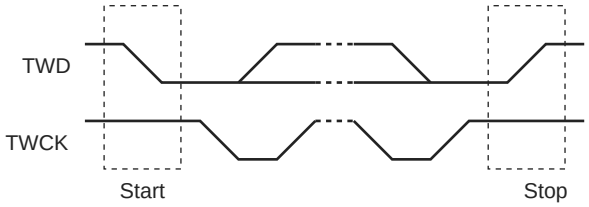
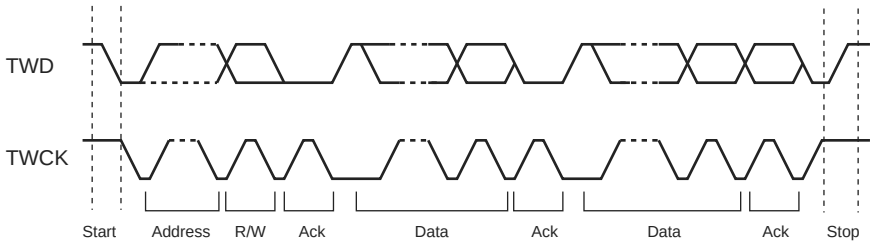


Figure 34-3. Transfer Format



34.7.2 Modes of Operation

The TWI has different modes of operations:

- Master transmitter mode
- Master receiver mode
- Multi-master transmitter mode
- Multi-master receiver mode
- Slave transmitter mode
- Slave receiver mode

These modes are described in the following sections.

34.7.3 Master Mode

34.7.3.1 Definition

The master is the device that starts a transfer, generates a clock and stops it.

37.7.11 TC Interrupt Disable Register

Name: TC_IDRx [x=0..2]

Address: 0x40010028 (0)[0], 0x40010068 (0)[1], 0x400100A8 (0)[2], 0x40014028 (1)[0], 0x40014068 (1)[1], 0x400140A8 (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
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**

0: No effect.

1: Disables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0: No effect.

1: Disables the Load Overrun Interrupt (if TC_CMRx.WAVE = 0).

- **CPAS: RA Compare**

0: No effect.

1: Disables the RA Compare Interrupt (if TC_CMRx.WAVE = 1).

- **CPBS: RB Compare**

0: No effect.

1: Disables the RB Compare Interrupt (if TC_CMRx.WAVE = 1).

- **CPCS: RC Compare**

0: No effect.

1: Disables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0: No effect.

1: Disables the RA Load Interrupt (if TC_CMRx.WAVE = 0).

- **LDRBS: RB Loading**

0: No effect.

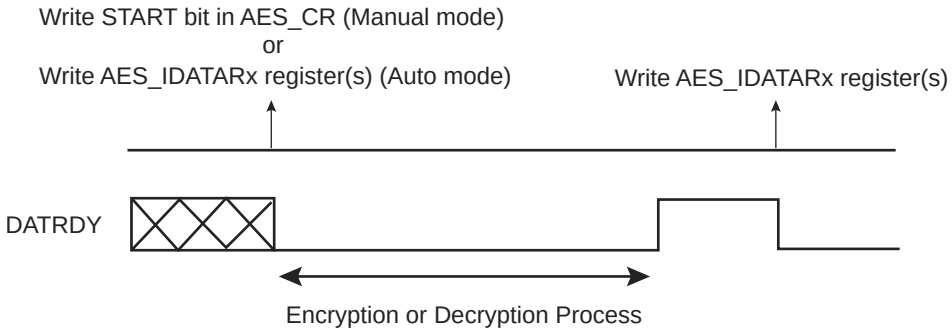
1: Disables the RB Load Interrupt (if TC_CMRx.WAVE = 0).

If AES_MR.LOD = 1

This mode is optimized to process AES CPC-MAC operating mode.

The DATRDY flag is cleared when at least one AES_IDATAR is written (see Figure 42-2). No more AES_ODATAR reads are necessary between consecutive encryptions/decryptions.

Figure 42-2. Manual and Auto Modes with AES_MR.LOD = 1



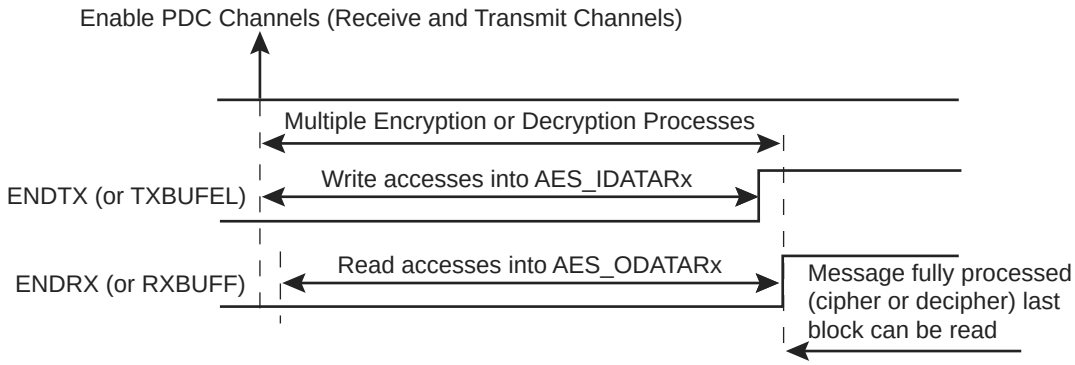
42.4.5.2 PDC Mode

If AES_MR.LOD = 0

This mode may be used for all AES operating modes except CBC-MAC where AES_MR.LOD = 1 mode is recommended.

The end of the encryption/decryption is indicated when the ENDRX (or RXBUFF) flag is raised (see Figure 42-3).

Figure 42-3. PDC Transfer with AES_MR.LOD = 0



If AES_MR.LOD = 1

This mode is optimized to process AES CBC-MAC operating mode.

The user must first wait for the ENDTX (or TXBUFE) flag to be raised, then for DATRDY to ensure that the encryption/decryption is completed (see Figure 42-4).

In this case, no receive buffers are required.

The output data are only available on the AES_ODATARx.

Figure 46-6. SPI Slave Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

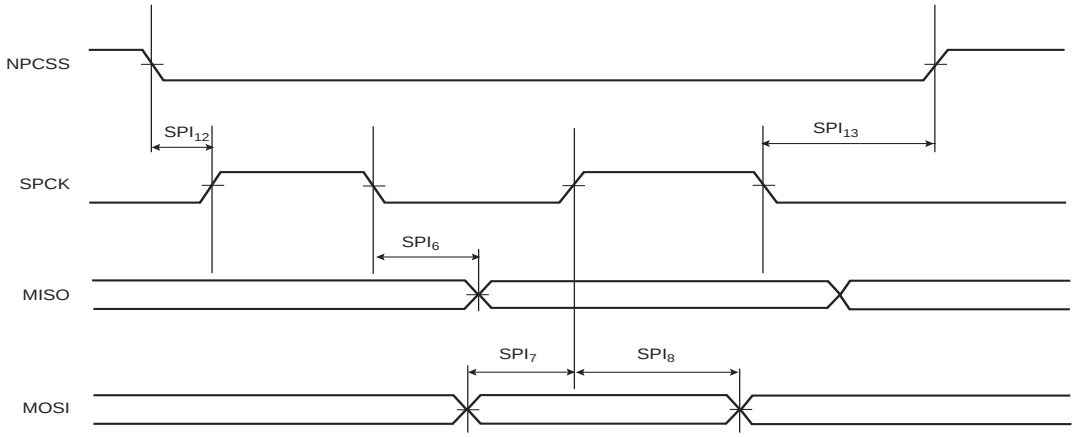
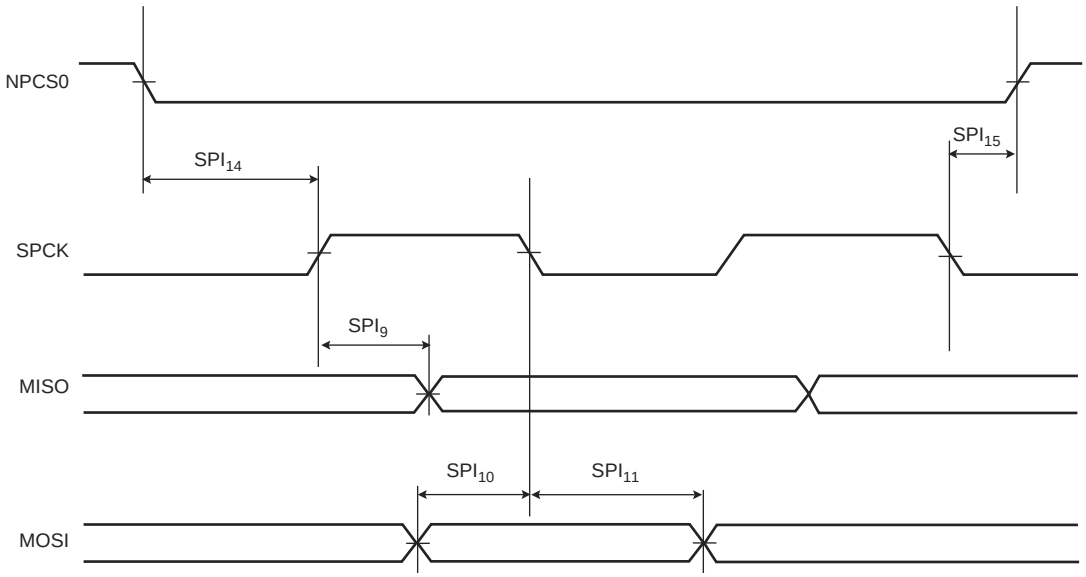


Figure 46-7. SPI Slave Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)



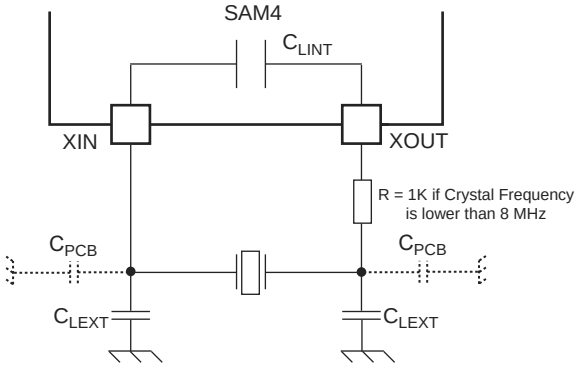
46.4.3.1 Maximum SPI Frequency

The formulas that follow give the maximum SPI frequency in Master Write and Read modes, and in Slave Read and Write modes.

Master Write Mode

The SPI is only sending data to a slave device such as an LCD, for example. The limit is given by SPI_2 (or SPI_5) timing. Since it gives a maximum frequency above the maximum pad speed (refer to Section 46.4.2 “I/O AC Characteristics”), the max SPI frequency is the one from the pad.

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

Table 55-3. SAM4CM Datasheet Rev. 11203C Revision History (Continued)

Doc. Rev. 11203C	Changes
06-Oct-14	Section 46. “Electrical Characteristics”
	Table 46-2 “Recommended Operating Conditions on Power Supply Inputs”: changed min value of VDDLCD. modified column “Conditions” for all parameters. Added notes at end of table.
	Updated R_{JA} and P_D in Table 46-4 “Recommended Thermal Operating Conditions”.
	Table 46-7 “Input Characteristics”: added mention of voltage reference to VDDIO in paragraph preceding table.
	Updated Table 46-8 “SPI Timings”.
	Updated Table 46-9 “SMC Read Signals - NRD Controlled (READ_MODE = 1)”, Table 46-10 “SMC Read Signals - NCS Controlled (READ_MODE= 0)”, Table 46-11 “SMC Write Signals - NWE Controlled (WRITE_MODE = 1)” and Table 46-12 “SMC Write NCS Controlled (WRITE_MODE = 0)”.
	Updated Table 46-13 “USART SPI Timings”.
	In Table 46-18 “LCD Buffers Characteristics”, changed max value for Z_{OUT} ‘Buffer output impedance’. Changed convergence value and max value for t_r / t_f ‘Rising or falling time’.
	Table 46-21 “VDDIO Supply Monitor”: added Note (2). Removed figure “VDDIO Supply Monitor”.
	Improved definition of parameters and modified equations and figures in Section 46.5.11 “32.768 kHz Crystal Oscillator” and Section 46.5.12 “3 to 20 MHz Crystal Oscillator”.
	Table 46-27 “32.768 kHz Crystal Oscillator Characteristics”: modified min/typ/max values for CLEXT.
	Table 46-33 “Temperature Sensor Characteristics”: modified condition of parameter V_T settling time.
	Table 46-35 “ADC Power Supply Characteristics”: modified typ for supply voltage range (VDDIN).
	Table 46-36 “ADC Voltage Reference Input Characteristics (ADVREF pin)”: modified min value of V_{ADVREF}
	Table 46-41 “Programmable Voltage Reference Characteristics”: modified condition for V_{ADVREF}
	Table 46-45 “Current or Voltage Measurement Channel Electrical Characteristics”: modified max value I_{DDON} when OFF
	Added Section 46.6.2.2 “SAM4CM32 Flash Wait States and Operating Frequency”.
	Added Table 46-53 “SAM4CM32 Typical Current Consumption Values for Backup Mode Configurations A and B” and Figure 46-17 “Typical Current Consumption in Backup Mode for Configurations A and B”.
	Modified Table 46-54 “SAM4CM8/16 Typical Current Consumption Values for Backup Mode Configurations C and D”.
	Added Table 46-55 “SAM4CM32 Typical Current Consumption Values for Backup Mode Configurations C and D” and Figure 46-19 “Typical Current Consumption in Backup Mode for Configurations C and D”.
	Updated Section 46.7.2.1 “Wait Mode Configuration”.
	Updated Table 46-56 “SAM4CM8/16 Typical Current Consumption in Wait Mode”.
	Added Table 46-57 “SAM4CM32 Typical Current Consumption in Wait Mode”.
	Section 46.7.3 “Sleep Mode Current Consumption”: modified information on sub-system frequencies in bullets.
	Updated Table 46-58 “SAM4CM8/16 Typical Sleep Mode Current Consumption Versus Frequency”.
	Added Table 46-59 “SAM4CM32 Typical Sleep Mode Current Consumption Versus Frequency”. Added Figure 46-22 “Typical Current Consumption in Sleep Mode”.
	Section 46.7.4 “Active Mode Power Consumption”: modified information on sub-system frequencies in bullets.
	Updated Table 46-60 “SAM4CM8/16 Test Setup 1 Current Consumption”. Added Table 46-61 “SAM4CM32 Test Setup 1 Current Consumption” and Figure 46-24 “Typical Current Consumption in Active Mode (Test Setup 1)”.
	Updated Table 46-62 “SAM4CM8/16 Test Setup 2 Current Consumption”. Added Table 46-63 “SAM4CM32 Test Setup 2 Current Consumption” and Figure 46-25 “Typical Current Consumption in Active Mode (Test Setup 2)”.
	Updated Table 46-64 “SAM4CM8/16 Test Setup 3 Current Consumption”. Added Table 46-65 “SAM4CM32 Test Setup 3 Current Consumption” and Figure 46-26 “Typical Current Consumption in Active Mode (Test Setup 3)”.
	Updated Table 46-66 “SAM4CM8/16 Test Setup 4 Current Consumption”. Added Table 46-67 “SAM4CM32 Test Setup 4 Current Consumption” and Figure 46-27 “Typical Current Consumption in Active Mode (Test Setup 4)”.
	Updated Table 46-68 “SAM4CM8/16 Test Setup 5 Current Consumption”. Added Table 46-69 “SAM4CM32Test Setup 5 Current Consumption” and Figure 46-28 “Typical Current Consumption in Active Mode (Test Setup 5)”.