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/atsam4cmp8ca-au

12.6.4.7 PUSH and POP

Push registers onto, and pop registers off a full-descending stack.

Syntax

```
PUSH{cond} reglist
POP{cond} reglist
```

where:

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

reglist — is a non-empty list of registers, enclosed in braces. It can contain register ranges. It must be comma separated if it contains more than one register or register range.

PUSH and POP are synonyms for STMDB and LDM (or LDMIA) with the memory addresses for the access based on SP, and with the final address for the access written back to the SP. PUSH and POP are the preferred mnemonics in these cases.

Operation

PUSH stores registers on the stack in order of decreasing the register numbers, with the highest numbered register using the highest memory address and the lowest numbered register using the lowest memory address.

POP loads registers from the stack in order of increasing register numbers, with the lowest numbered register using the lowest memory address and the highest numbered register using the highest memory address.

See “LDM and STM” for more information.

Restrictions

In these instructions:

- *reglist* must not contain SP
- For the PUSH instruction, *reglist* must not contain PC
- For the POP instruction, *reglist* must not contain PC if it contains LR.

When PC is in *reglist* in a POP instruction:

- Bit[0] of the value loaded to the PC 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

```
PUSH    {R0,R4-R7}
PUSH    {R2,LR}
POP     {R0,R10,PC}
```

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.

12.6.7.2 SSAT16 and USAT16

Signed Saturate and Unsigned Saturate to any bit position for two halfwords.

Syntax

```
op{cond} Rd, #n, Rm
```

where:

op	is one of:
	SSAT16 Saturates a signed halfword value to a signed range.
	USAT16 Saturates a signed halfword value to an unsigned range.
cond	is an optional condition code, see “Conditional Execution”.
Rd	is the destination register.
n	specifies the bit position to saturate to:
n ranges from 1	n ranges from 0 to 15 for USAT.
to 16 for SSAT	
Rm	is the register containing the value to saturate.

Operation

The SSAT16 instruction:

Saturates two signed 16-bit halfword values of the register with the value to saturate from selected by the bit position in *n*.

Writes the results as two signed 16-bit halfwords to the destination register.

The USAT16 instruction:

Saturates two unsigned 16-bit halfword values of the register with the value to saturate from selected by the bit position in *n*.

Writes the results as two unsigned halfwords in the destination register.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the condition code flags.

If saturation occurs, these instructions set the Q flag to 1.

Examples

```
SSAT16    R7, #9, R2   ; Saturates the top and bottom highwords of R2
                       ; as 9-bit values, writes to corresponding halfword
                       ; of R7
USAT16NE  R0, #13, R5  ; Conditionally saturates the top and bottom
                       ; halfwords of R5 as 13-bit values, writes to
                       ; corresponding halfword of R0.
```

12.6.11 Floating-point Instructions

The table below shows the floating-point instructions.

These instructions are only available if the FPU is included, and enabled, in the system. See “Enabling the FPU” for information about enabling the floating-point unit.

Table 12-27. Floating-point Instructions

Mnemonic	Description
VABS	Floating-point Absolute
VADD	Floating-point Add
VCMP	Compare two floating-point registers, or one floating-point register and zero
VCMPE	Compare two floating-point registers, or one floating-point register and zero with Invalid Operation check
VCVT	Convert between floating-point and integer
VCVT	Convert between floating-point and fixed point
VCVTR	Convert between floating-point and integer with rounding
VCVTB	Converts half-precision value to single-precision
VCVTT	Converts single-precision register to half-precision
VDIV	Floating-point Divide
VFMA	Floating-point Fused Multiply Accumulate
VFNMA	Floating-point Fused Negate Multiply Accumulate
VFMS	Floating-point Fused Multiply Subtract
VFNMS	Floating-point Fused Negate Multiply Subtract
VLDM	Load Multiple extension registers
VLDR	Loads an extension register from memory
VLMA	Floating-point Multiply Accumulate
VLMS	Floating-point Multiply Subtract
VMOV	Floating-point Move Immediate
VMOV	Floating-point Move Register
VMOV	Copy ARM core register to single precision
VMOV	Copy 2 ARM core registers to 2 single precision
VMOV	Copies between ARM core register to scalar
VMOV	Copies between Scalar to ARM core register
VMRS	Move to ARM core register from floating-point System Register
VMSR	Move to floating-point System Register from ARM Core register
VMUL	Multiply floating-point
VNEG	Floating-point negate
VNMLA	Floating-point multiply and add
VNMLS	Floating-point multiply and subtract
VNMUL	Floating-point multiply
VPOP	Pop extension registers

12.6.11.9 VFNMA, VFNMS

Floating-point Fused Negate Multiply Accumulate and Subtract.

Syntax

```
VFNMA{cond}.F32 {Sd,} Sn, Sm
VFNMS{cond}.F32 {Sd,} Sn, Sm
```

where:

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

Sd is the destination register.

Sn, Sm are the operand registers.

Operation

The VFNMA instruction:

1. Negates the first floating-point operand register.
2. Multiplies the first floating-point operand with second floating-point operand.
3. Adds the negation of the floating -point destination register to the product
4. Places the result into the destination register.

The result of the multiply is not rounded before the addition.

The VFNMS instruction:

1. Multiplies the first floating-point operand with second floating-point operand.
2. Adds the negation of the floating-point value in the destination register to the product.
3. Places the result in the destination register.

The result of the multiply is not rounded before the addition.

Restrictions

There are no restrictions.

Condition Flags

These instructions do not change the flags.

12.6.11.11 VLDR

Loads a single extension register from memory

Syntax

```
VLDR{cond}{.64} Dd, [Rn{#imm}]
VLDR{cond}{.64} Dd, label
VLDR{cond}{.64} Dd, [PC, #imm}]
VLDR{cond}{.32} Sd, [Rn {, #imm}]
VLDR{cond}{.32} Sd, label
VLDR{cond}{.32} Sd, [PC, #imm]
```

where:

cond	is an optional condition code, see “Conditional Execution”.
64, 32	are the optional data size specifiers.
Dd	is the destination register for a doubleword load.
Sd	is the destination register for a singleword load.
Rn	is the base register. The SP can be used.
imm	is the + or - immediate offset used to form the address. Permitted address values are multiples of 4 in the range 0 to 1020.
label	is the label of the literal data item to be loaded.

Operation

This instruction:

- Loads a single extension register from memory, using a base address from an ARM core register, with an optional offset.

Restrictions

There are no restrictions.

Condition Flags

These instructions do not change the flags.

12.6.11.14 VMOV Register

Copies the contents of one register to another.

Syntax

```
VMOV{cond}.F64 Dd, Dm
VMOV{cond}.F32 Sd, Sm
```

where:

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

Dd is the destination register, for a doubleword operation.

Dm is the source register, for a doubleword operation.

Sd is the destination register, for a singleword operation.

Sm is the source register, for a singleword operation.

Operation

This instruction copies the contents of one floating-point register to another.

Restrictions

There are no restrictions

Condition Flags

These instructions do not change the flags.

- **CALEVSEL: Calendar Event Selection**

The event that generates the flag CALEV in RTC_SR depends on the value of CALEVSEL

Value	Name	Description
0	WEEK	Week change (every Monday at time 00:00:00)
1	MONTH	Month change (every 01 of each month at time 00:00:00)
2	YEAR	Year change (every January 1 at time 00:00:00)
3	–	Reserved

17.6.3 RTC Time Register

Name: RTC_TIMR

Address: 0x400E1468

Access: Read/Write

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

23	22	21	20	19	18	17	16
–	AMPM	HOUR					

15	14	13	12	11	10	9	8
–	MIN						

7	6	5	4	3	2	1	0
–	SEC						

- **SEC: Current Second**

The range that can be set is 0–59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **MIN: Current Minute**

The range that can be set is 0–59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **HOUR: Current Hour**

The range that can be set is 1–12 (BCD) in 12-hour mode or 0–23 (BCD) in 24-hour mode.

- **AMPM: Ante Meridiem Post Meridiem Indicator**

This bit is the AM/PM indicator in 12-hour mode.

0: AM.

1: PM.

18.5.3 Watchdog Timer Status Register

Name: WDT_SR

Address: 0x400E1458

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 (cleared on read)**

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

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

- **WDERR: Watchdog Error (cleared on read)**

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

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

26.2.1.2 Matrix 0 Slaves

The Bus Matrix manages the slaves listed in Table 26-2. Each slave has its own arbiter providing a dedicated arbitration per slave.

Table 26-2. List of Bus Matrix Slaves

Slave 0	Internal SRAM0
Slave 1	Internal ROM
Slave 2	Internal Flash
Slave 3	External Bus Interface
Slave 4	Peripheral Bridge 0
Slave 5	CPKCC RAM and ROM
Slave 6	Matrix1
Slave 7	CMCC0

26.2.1.3 Master to Slave Access (Matrix 0)

Table 26-3 gives valid paths for master to slave access on Matrix 0. The paths shown as "-" are forbidden or not wired, e.g. access from the Cortex-M4 S Bus to the Internal ROM.

Table 26-3. Matrix 0 Master to Slave Access

Slaves	Masters	0	1	2	3	4	5	6	Reserved
		Cortex-M4 I/D Bus	Cortex-M4 S Bus	PDC0	ICM	Matrix1	EBI Matrix 1	CMCC0	
0	Internal SRAM0	-	X	X	X	X	-	-	-
1	Internal ROM	X	-	X	X	-	-	-	-
2	Internal Flash	X	-	-	X	X	-	X	-
3	External Bus Interface	X	X	X	X	-	X	X	-
4	Peripheral Bridge 0	-	X	X	-	X	-	-	-
5	CPKCC SRAM, ROM	-	X	-	X	-	-	-	-
6	Matrix1	-	X	-	X	-	-	-	-
7	CMCC0	X	-	-	-	-	-	-	-

26.2.1.4 Accesses through Matrix 0

- CM4P0 I/D Bus access to:
  - Flash, ROM
  - EBI (0x03000000 to 0x06FFFFFF)
  - Flash and EBI through Cache Controller CMCC0 (respectively through 0x11000000 to 0x11FFFFFF and 0x13000000 to 0x16FFFFFF)
- CMP4P0 S Bus access to:
  - SRAM0, SRAM1 through Matrix1, SRAM2 through Matrix1
  - PKCC
  - EBI (through 0x60000000 to 0x63FFFFFF, 0xA0000000 to A3FFFFFF)
- PDC0 access to:
  - SRAM0, ROM

high-priority master request will be granted after the current bus master access has ended and other high priority pool master requests, if any, have been granted once each.

The lowest priority pool shares the remaining bus bandwidth between AHB Masters.

Intermediate priority pools allow fine priority tuning. Typically, a latency-sensitive master or a bandwidth-sensitive master will use such a priority level. The higher the priority level (MxPR value), the higher the master priority.

To ensure a good level of CPU performance, it is recommended to configure the CPU priority with the default reset value 2 (Latency Sensitive).

All combinations of MxPR values are allowed for all masters and slaves. For example, some masters might be assigned the highest priority pool (round-robin), and remaining masters the lowest priority pool (round-robin), with no master for intermediate fix priority levels.

26.7.2.1 Fixed Priority Arbitration

Fixed priority arbitration algorithm is the first and only arbitration algorithm applied between masters from distinct priority pools. It is also used in priority pools other than the highest and lowest priority pools (intermediate priority pools).

Fixed priority arbitration allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave by using the fixed priority defined by the user in the MxPR field for each master in the Priority registers, MATRIX_PRAS and MATRIX_PRBS. If two or more master requests are active at the same time, the master with the highest priority MxPR number is serviced first.

In intermediate priority pools, if two or more master requests with the same priority are active at the same time, the master with the highest number is serviced first.

26.7.2.2 Round-robin Arbitration

This algorithm is only used in the highest and lowest priority pools. It allows the Bus Matrix arbiters to properly dispatch requests from different masters to the same slave. If two or more master requests are active at the same time in the priority pool, they are serviced in a round-robin increasing master number order.

26.8 Register Write Protection

To prevent any single software error from corrupting the Bus Matrix behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the “Write Protection Mode Register” (MATRIX_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the “Write Protection Status Register” (MATRIX_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS flag is reset by writing the Bus Matrix Write Protect Mode Register (MATRIX_WPMR) with the appropriate access key WPKEY.

The following registers can be write-protected:

- “Bus Matrix Master Configuration Registers”
- “Bus Matrix Slave Configuration Registers”
- “Bus Matrix Priority Registers A For Slaves”
- “System I/O Configuration Register”

27.16.8 SMC Write Protection Mode Register

Name: SMC_WPMR

Address: 0x400E00E4 (0), 0x4801C0E4 (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 Protect Enable**

0: Disables the write protection if WPKEY corresponds to 0x534D43 (“SMC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x534D43 (“SMC” in ASCII).

See Section 27.9.5 ”Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x534D43	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Figure 33-8 shows the behavior of Transmit Data Register Empty (TDRE), Receive Data Register (RDRF) and Transmission Register Empty (TXEMPTY) status flags within the SPI_SR during an 8-bit data transfer in Fixed mode without the PDC involved.

Figure 33-8. Status Register Flags Behavior

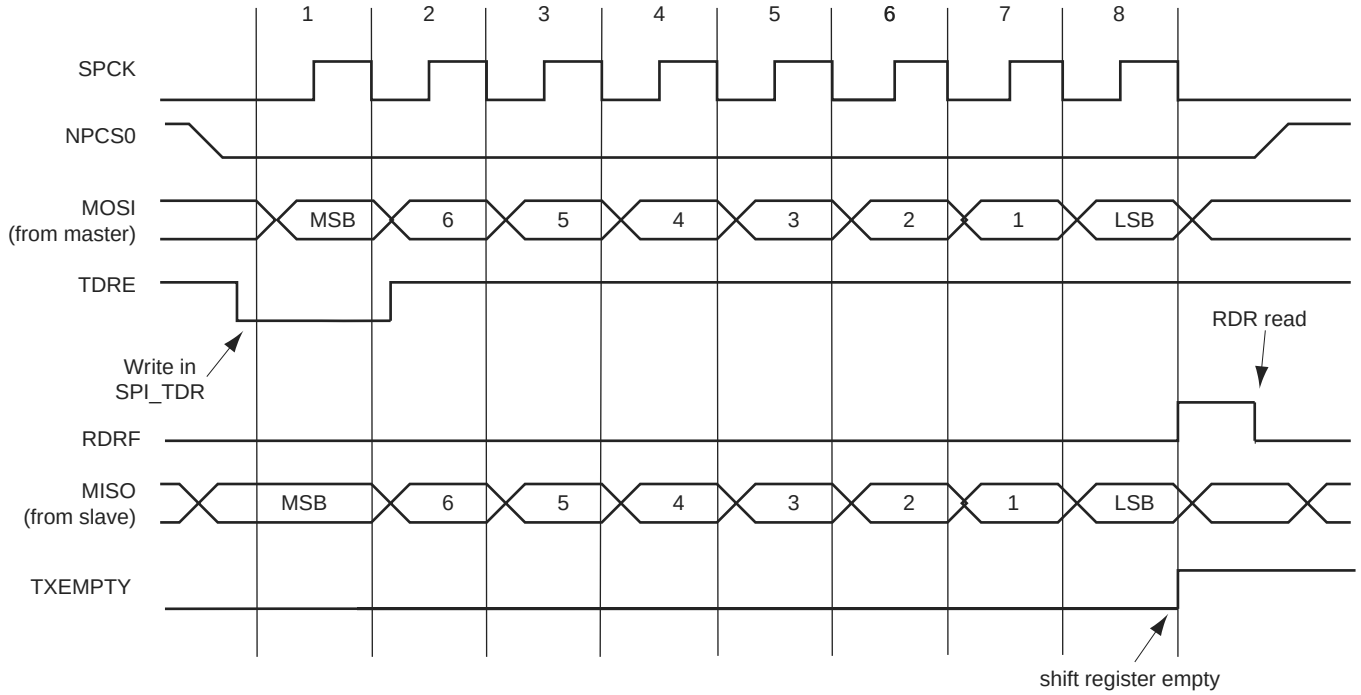
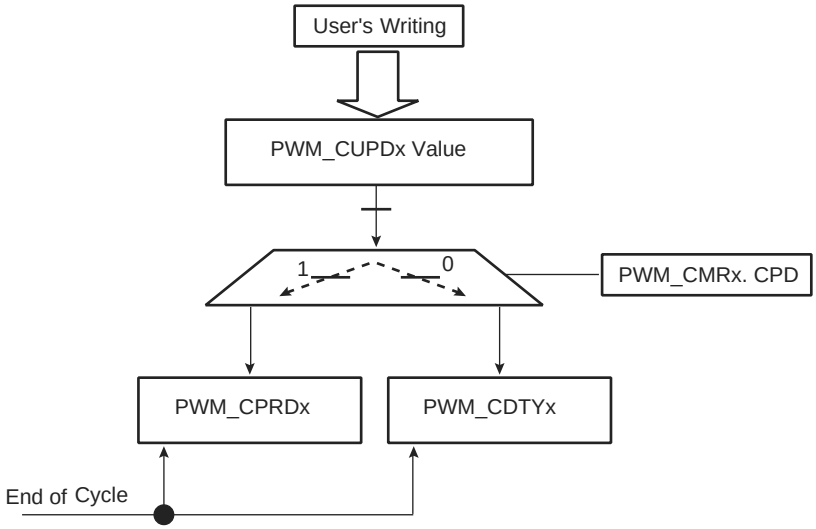


Figure 38-6. Synchronized Period or Duty Cycle Update



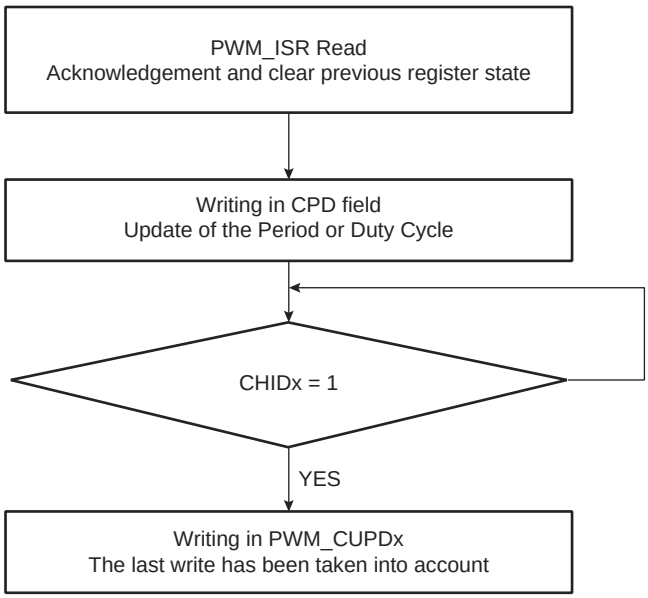
To prevent overwriting the PWM_CUPDx by software, the user can use status events in order to synchronize his software. Two methods are possible. In both, the user must enable the dedicated interrupt in PWM_IER at PWM Controller level.

The first method (polling method) consists of reading the relevant status bit in PWM_ISR Register according to the enabled channel(s). See Figure 38-7.

The second method uses an Interrupt Service Routine associated with the PWM channel.

Note: Reading the PWM_ISR register automatically clears CHIDx flags.

Figure 38-7. Polling Method



Note: Polarity and alignment can be modified only when the channel is disabled.

42.5.1 AES Control Register

Name: AES_CR

Address: 0x40000000

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
–	–	–	–	–	–	–	SWRST
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	START

- **START: Start Processing**

0: No effect.

1: Starts manual encryption/decryption process.

- **SWRST: Software Reset**

0: No effect.

1: Resets the AES. A software-triggered hardware reset of the AES interface is performed.

46.4.4 SMC Timings

Timings are given in the following domains:

- 1.8V domain: VDDIO from 1.65V to 1.95V, maximum external capacitor = 10 pF
- 3.3V domain: VDDIO from 2.85V to 3.6V, maximum external capacitor = 10 pF

Timings are given assuming a capacitance load on data, control and address pads.

In the tables that follow, t_{CPMCK} is the MCK period.

46.4.4.1 Read Timings

Table 46-11. SMC Read Signals - NRD Controlled (READ_MODE = 1)

Symbol	Parameter	Min		Max		Unit
	VDDIO Supply	1.8V(1)	3.3V(2)	–	–	
NO HOLD SETTINGS (nrd hold = 0)						
SMC_1	Data setup before NRD high	18	18	–	–	ns
SMC_2	Data hold after NRD high	-6.7	-6.7	–	–	ns
HOLD SETTINGS (nrd hold ≠ 0)						
SMC_3	Data setup before NRD high	11.7	11.7	–	–	ns
SMC_4	Data hold after NRD high	-6.5	-6.5	–	–	ns
HOLD or NO HOLD SETTINGS (nrd hold ≠ 0, nrd hold = 0)						
SMC_5	NBS0/A0, NBS1, A1 - A23 Valid before NRD high	(nrd setup + nrd pulse) * t_{CPMCK} - 5.2	(nrd setup + nrd pulse) * t_{CPMCK} - 5.2	–	–	ns
SMC_6	NCS low before NRD high	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 1.1	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 1.1	–	–	ns
SMC_7	NRD pulse width	nrd pulse * t_{CPMCK} - 3.2	nrd pulse * t_{CPMCK} - 3.2	–	–	ns

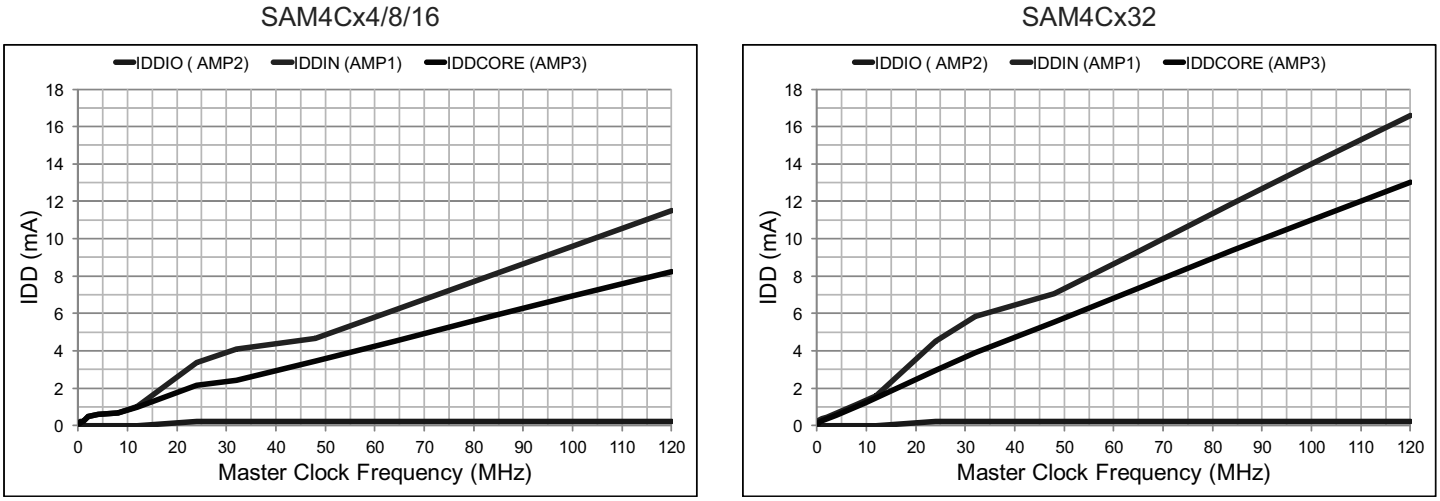
Notes: 1. 1.8V domain: VDDIO from 1.65V to 1.95V, maximum external capacitor = 10 pF.
2. 3.3V domain: VDDIO from 3.0V to 3.6V, maximum external capacitor = 10 pF.

Table 46-12. SMC Read Signals - NCS Controlled (READ_MODE= 0)

Symbol	Parameter	Min		Max		Unit
	VDDIO supply	1.8V(1)	3.3V(2)	–	–	
NO HOLD SETTINGS (ncs rd hold = 0)						
SMC_8	Data setup before NCS high	31.3	31.3	–	–	ns
SMC_9	Data hold after NCS high	-6.9	-6.9	–	–	ns
HOLD SETTINGS (ncs rd hold ≠ 0)						
SMC_{10}	Data setup before NCS high	23	23	–	–	ns
SMC_{11}	Data hold after NCS high	-6.6	-6.6	–	–	ns
HOLD or NO HOLD SETTINGS (ncs rd hold ≠ 0, ncs rd hold = 0)						
SMC_{12}	NBS0/A0, NBS1, A1–A23 valid before NCS high	(ncs rd setup + ncs rd pulse)* t_{CPMCK} - 4.9	(ncs rd setup + ncs rd pulse)* t_{CPMCK} - 4.9	–	–	ns
SMC_{13}	NRD low before NCS high	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} - 1.5	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} - 1.5	–	–	ns
SMC_{14}	NCS pulse width	ncs rd pulse length * t_{CPMCK} - 5.4	ncs rd pulse length * t_{CPMCK} - 5.4	–	–	ns

Notes: 1. 1.8V domain: VDDIO from 1.65V to 1.95V, maximum external capacitor = 10 pF.
2. 3.3V domain: VDDIO from 3.0V to 3.6V, maximum external capacitor = 10 pF.

Figure 46-25. Typical Current Consumption in Sleep Mode

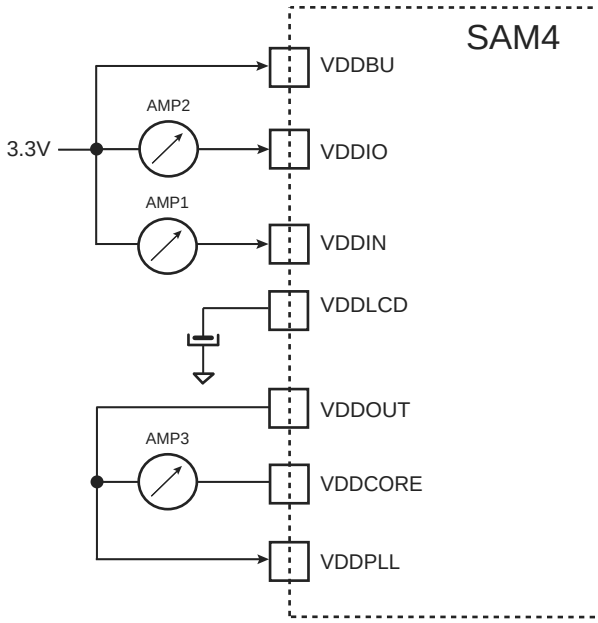


46.7.4 Active Mode Power Consumption

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



52. SAM4CM16/8/4 Errata Revision C (MRL C) Parts

52.1 Device Identification

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

Table 52-1. Device List

Device Marking	Chip ID
ATSAM4CMP16CC-AU	0xA64C_0CE2
ATSAM4CMP16CC-AUR	0xA64C_0CE2
ATSAM4CMP8CC-AU	0xA64C_0AE2
ATSAM4CMP8CC-AUR	0xA64C_0AE2
ATSAM4CMS16CC-AU	0xA64C_0CE2
ATSAM4CMS16CC-AUR	0xA64C_0CE2
ATSAM4CMS8CC-AU	0xA64C_0AE2
ATSAM4CMS8CC-AUR	0xA64C_0AE2
ATSAM4CMS4CC-AU	0xA64C_0CE6
ATSAM4CMS4CC-AUR	0xA64C_0CE6

52.2 Supply Controller (SUPC)

52.2.1 SUPC: Supply Monitor (SM) on VDDIO

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

Problem Fix/Workaround

Use the Supply Monitor in Continuous mode only.

52.2.2 SUPC: Core Voltage Regulator Standby Mode Control

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

Problem Fix/Workaround

None. Do not use the ONREG Bit.

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

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

Problem Fix/Workaround

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

50.5 Watchdog (WDT) / Reinforced Safety Watchdog (RSWDT) 1135
50.6 Enhanced Embedded Flash Controller (EEFC) 1136
50.7 Wait For Interrupt (WFI) 1136
50.8 Power Supply and Power Control / Clock System 1136
50.9 Power Management Controller (PMC) 1137
50.10 EMAFE 1137

51. SAM4CM16/8/4 Errata Revision B (MRL B) Parts 1138
51.1 Device Identification 1138
51.2 Supply Controller (SUPC) 1138
51.3 Parallel Input Output (PIO) Controller 1139
51.4 Watchdog (WDT) / Reinforced Safety Watchdog (RSWDT) 1139
51.5 Enhanced Embedded Flash Controller (EEFC) 1139
51.6 Wait For Interrupt (WFI) 1140
51.7 Power Supply and Power Control / Clock System 1140
51.8 Power Management Controller (PMC) 1140
51.9 EMAFE 1140

52. SAM4CM16/8/4 Errata Revision C (MRL C) Parts 1141
52.1 Device Identification 1141
52.2 Supply Controller (SUPC) 1141
52.3 Parallel Input Output (PIO) Controller 1142
52.4 Watchdog (WDT) / Reinforced Safety Watchdog (RSWDT) 1142
52.5 Enhanced Embedded Flash Controller (EEFC) 1142
52.6 Wait For Interrupt (WFI) 1143
52.7 Power Supply and Power Control / Clock System 1143
52.8 Power Management Controller (PMC) 1143

53. SAM4CM32 Errata Revision A (MRL A) Parts 1144
53.1 Device Identification 1144
53.2 Supply Controller (SUPC) 1144
53.3 Parallel Input Output (PIO) Controller 1144
53.4 Reinforced Safety Watchdog (RSWDT) 1145
53.5 Enhanced Embedded Flash Controller (EEFC) 1145
53.6 Wait For Interrupt (WFI) 1145
53.7 Power Management Controller (PMC) 1146
53.8 EMAFE 1146

54. SAM4CM32 Errata Revision B (MRL B) Parts 1147
54.1 Device Identification 1147
54.2 Supply Controller (SUPC) 1147
54.3 Parallel Input Output (PIO) Controller 1147
54.4 Reinforced Safety Watchdog (RSWDT) 1148
54.5 Enhanced Embedded Flash Controller (EEFC) 1148
54.6 Wait For Interrupt (WFI) 1148
54.7 Power Management Controller (PMC) 1149

55. Revision History 1150