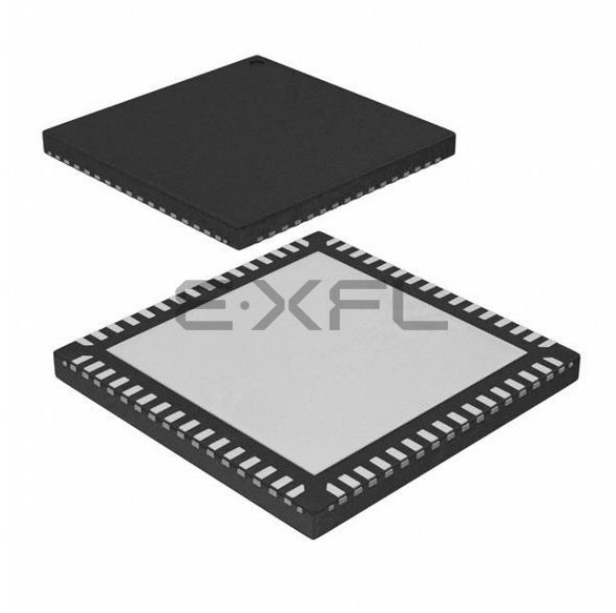


Welcome to E-XFL.COM

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
Core Size	32-Bit Single-Core
Speed	120MHz
Connectivity	I²C, IrDA, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	47
Program Memory Size	1MB (1M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 11x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-VFQFN Exposed Pad
Supplier Device Package	64-QFN (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4s16ba-mu

Safety Features Highlight

- Flash
  - Built-in ECC (hamming), single error correction
  - Security bit and lock bits

12.4.3.5 Exception Priorities

As Table 12-9 shows, all exceptions have an associated priority, with:

- A lower priority value indicating a higher priority
- Configurable priorities for all exceptions except Reset, Hard fault and NMI.

If the software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities see “System Handler Priority Registers” , and “Interrupt Priority Registers” .

Note: Configurable priority values are in the range 0– . This means that the Reset, Hard fault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

For example, assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

12.4.3.6 Interrupt Priority Grouping

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This divides each interrupt priority register entry into two fields:

- An upper field that defines the *group priority*
- A lower field that defines a *subpriority* within the group.

Only the group priority determines preemption of interrupt exceptions. When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled does not preempt the handler.

If multiple pending interrupts have the same group priority, the subpriority field determines the order in which they are processed. If multiple pending interrupts have the same group priority and subpriority, the interrupt with the lowest IRQ number is processed first.

For information about splitting the interrupt priority fields into group priority and subpriority, see “Application Interrupt and Reset Control Register” .

12.4.3.7 Exception Entry and Return

Descriptions of exception handling use the following terms:

Preemption

When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See “Interrupt Priority Grouping” for more information about preemption by an interrupt.

When one exception preempts another, the exceptions are called nested exceptions. See “Exception Entry” more information.

Return

This occurs when the exception handler is completed, and:

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

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

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the condition code flags.

Examples

```
UQADD16  R7, R4, R2   ; Adds halfwords in R4 to corresponding halfword in R2,
                      ; saturates to 16 bits, writes to corresponding halfword of R7
UQADD8   R4, R2, R5   ; Adds bytes of R2 to corresponding byte of R5, saturates
                      ; to 8 bits, writes to corresponding bytes of R4
UQSUB16  R6, R3, R0   ; Subtracts halfwords in R0 from corresponding halfword
                      ; in R3, saturates to 16 bits, writes to corresponding
                      ; halfword in R6
UQSUB8   R1, R5, R6   ; Subtracts bytes in R6 from corresponding byte of R5,
                      ; saturates to 8 bits, writes to corresponding byte of R1.
```

12.6.8 Packing and Unpacking Instructions

The table below shows the instructions that operate on packing and unpacking data.

Table 12-23. Packing and Unpacking Instructions

Mnemonic	Description
PKH	Pack Halfword
SXTAB	Extend 8 bits to 32 and add
SXTAB16	Dual extend 8 bits to 16 and add
SXTAH	Extend 16 bits to 32 and add
SXTB	Sign extend a byte
SXTB16	Dual extend 8 bits to 16 and add
SXTH	Sign extend a halfword
UXTAB	Extend 8 bits to 32 and add
UXTAB16	Dual extend 8 bits to 16 and add
UXTAH	Extend 16 bits to 32 and add
UXTB	Zero extend a byte
UXTB16	Dual zero extend 8 bits to 16 and add
UXTH	Zero extend a halfword

12.6.9.1 BFC and BFI

Bit Field Clear and Bit Field Insert.

Syntax

```
BFC{cond} Rd, #lsb, #width
BFI{cond} Rd, Rn, #lsb, #width
```

where:

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

Rd is the destination register.

Rn is the source register.

lsb is the position of the least significant bit of the bitfield. *lsb* must be in the range 0 to 31.

width is the width of the bitfield and must be in the range 1 to 32-*lsb*.

Operation

BFC clears a bitfield in a register. It clears *width* bits in *Rd*, starting at the low bit position *lsb*. Other bits in *Rd* are unchanged.

BFI copies a bitfield into one register from another register. It replaces *width* bits in *Rd* starting at the low bit position *lsb*, with *width* bits from *Rn* starting at bit[0]. Other bits in *Rd* are unchanged.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the flags.

Examples

```
BFC   R4, #8, #12       ; Clear bit 8 to bit 19 (12 bits) of R4 to 0
BFI   R9, R2, #8, #12   ; Replace bit 8 to bit 19 (12 bits) of R9 with
                        ; bit 0 to bit 11 from R2.
```

12.6.10.3 IT

If-Then condition instruction.

Syntax

```
IT{x{y{z}}} cond
```

where:

x specifies the condition switch for the second instruction in the IT block.

y specifies the condition switch for the third instruction in the IT block.

z specifies the condition switch for the fourth instruction in the IT block.

cond specifies the condition for the first instruction in the IT block.

The condition switch for the second, third and fourth instruction in the IT block can be either:

T Then. Applies the condition *cond* to the instruction.

E Else. Applies the inverse condition of *cond* to the instruction.

It is possible to use AL (the *always* condition) for *cond* in an IT instruction. If this is done, all of the instructions in the IT block must be unconditional, and each of *x*, *y*, and *z* must be T or omitted but not E.

Operation

The IT instruction makes up to four following instructions conditional. The conditions can be all the same, or some of them can be the logical inverse of the others. The conditional instructions following the IT instruction form the *IT block*.

The instructions in the IT block, including any branches, must specify the condition in the {*cond*} part of their syntax.

The assembler might be able to generate the required IT instructions for conditional instructions automatically, so that the user does not have to write them. See the assembler documentation for details.

A BKPT instruction in an IT block is always executed, even if its condition fails.

Exceptions can be taken between an IT instruction and the corresponding IT block, or within an IT block. Such an exception results in entry to the appropriate exception handler, with suitable return information in LR and stacked PSR.

Instructions designed for use for exception returns can be used as normal to return from the exception, and execution of the IT block resumes correctly. This is the only way that a PC-modifying instruction is permitted to branch to an instruction in an IT block.

Restrictions

The following instructions are not permitted in an IT block:

- IT
- CBZ and CBNZ
- CPSID and CPSIE.

Other restrictions when using an IT block are:

- A branch or any instruction that modifies the PC must either be outside an IT block or must be the last instruction inside the IT block. These are:
  - ADD PC, PC, Rm
  - MOV PC, Rm
  - B, BL, BX, BLX
  - Any LDM, LDR, or POP instruction that writes to the PC
  - TBB and TBH
- Do not branch to any instruction inside an IT block, except when returning from an exception handler

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.

Exception	An event that interrupts program execution. When an exception occurs, the processor suspends the normal program flow and starts execution at the address indicated by the corresponding exception vector. The indicated address contains the first instruction of the handler for the exception. An exception can be an interrupt request, a fault, or a software-generated system exception. Faults include attempting an invalid memory access, attempting to execute an instruction in an invalid processor state, and attempting to execute an undefined instruction.
Exception service routine	*See* “Interrupt handler” .
Exception vector	*See* “Interrupt vector” .
Flat address mapping	A system of organizing memory in which each physical address in the memory space is the same as the corresponding virtual address.
Halfword	A 16-bit data item.
Illegal instruction	An instruction that is architecturally Undefined.
Implementation-defined	The behavior is not architecturally defined, but is defined and documented by individual implementations.
Implementation-specific	The behavior is not architecturally defined, and does not have to be documented by individual implementations. Used when there are a number of implementation options available and the option chosen does not affect software compatibility.
Index register	In some load and store instruction descriptions, the value of this register is used as an offset to be added to or subtracted from the base register value to form the address that is sent to memory. Some addressing modes optionally enable the index register value to be shifted prior to the addition or subtraction. *See also* “Base register” .
Instruction cycle count	The number of cycles that an instruction occupies the Execute stage of the pipeline.
Interrupt handler	A program that control of the processor is passed to when an interrupt occurs.
Interrupt vector	One of a number of fixed addresses in low memory, or in high memory if high vectors are configured, that contains the first instruction of the corresponding interrupt handler.

22.5.5 Cache Controller Maintenance Register 0

Name: CMCC_MAINT0

Address: 0x4007C020

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
–	–	–	–	–	–	–	INVALL

- **INVALL: Cache Controller Invalidate All**

0: No effect.

1: All cache entries are invalidated.

27. Peripheral DMA Controller (PDC)

27.1 Description

The Peripheral DMA Controller (PDC) transfers data between on-chip serial peripherals and the target memories. The link between the PDC and a serial peripheral is operated by the AHB to APB bridge.

The user interface of each PDC channel is integrated into the user interface of the peripheral it serves. The user interface of mono-directional channels (receive-only or transmit-only) contains two 32-bit memory pointers and two 16-bit counters, one set (pointer, counter) for the current transfer and one set (pointer, counter) for the next transfer. The bidirectional channel user interface contains four 32-bit memory pointers and four 16-bit counters. Each set (pointer, counter) is used by the current transmit, next transmit, current receive and next receive.

Using the PDC decreases processor overhead by reducing its intervention during the transfer. This lowers significantly the number of clock cycles required for a data transfer, improving microcontroller performance.

To launch a transfer, the peripheral triggers its associated PDC channels by using transmit and receive signals. When the programmed data is transferred, an end of transfer interrupt is generated by the peripheral itself.

27.2 Embedded Characteristics

- Performs Transfers to/from APB Communication Serial Peripherals
- Supports Half-duplex and Full-duplex Peripherals

31.6.9 PIO Input Filter Status Register

Name: PIO_IFSR

Address: 0x400E0E28 (PIOA), 0x400E1028 (PIOB), 0x400E1228 (PIOC)

Access: Read-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: Input Filter Status**

0: The input glitch filter is disabled on the I/O line.

1: The input glitch filter is enabled on the I/O line.

31.6.49 PIO Parallel Capture Mode Register

Name: PIO_PCMR

Address: 0x400E0F50 (PIOA), 0x400E1150 (PIOB), 0x400E1350 (PIOC)

Access: Read/Write

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

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

- **PCEN: Parallel Capture Mode Enable**

0: The parallel capture mode is disabled.

1: The parallel capture mode is enabled.

- **DSIZE: Parallel Capture Mode Data Size**

Value	Name	Description
0	BYTE	The reception data in the PIO_PCRHR is a byte (8-bit)
1	HALF-WORD	The reception data in the PIO_PCRHR is a half-word (16-bit)
2	WORD	The reception data in the PIO_PCRHR is a word (32-bit)
3	–	Reserved

- **ALWYS: Parallel Capture Mode Always Sampling**

0: The parallel capture mode samples the data when both data enables are active.

1: The parallel capture mode samples the data whatever the data enables are.

- **HALFS: Parallel Capture Mode Half Sampling**

Independently from the ALWYS bit:

0: The parallel capture mode samples all the data.

1: The parallel capture mode samples the data only every other time.

- **FRSTS: Parallel Capture Mode First Sample**

This bit is useful only if the HALFS bit is set to 1. If data are numbered in the order that they are received with an index from 0 to n:

0: Only data with an even index are sampled.

1: Only data with an odd index are sampled.

32.9 Synchronous Serial Controller (SSC) User Interface

Table 32-5. Register Mapping

Offset	Register	Name	Access	Reset
0x0	Control Register	SSC_CR	Write-only	–
0x4	Clock Mode Register	SSC_CMR	Read/Write	0x0
0x8–0xC	Reserved	–	–	–
0x10	Receive Clock Mode Register	SSC_RCMR	Read/Write	0x0
0x14	Receive Frame Mode Register	SSC_RFMR	Read/Write	0x0
0x18	Transmit Clock Mode Register	SSC_TCMR	Read/Write	0x0
0x1C	Transmit Frame Mode Register	SSC_TFMR	Read/Write	0x0
0x20	Receive Holding Register	SSC_RHR	Read-only	0x0
0x24	Transmit Holding Register	SSC_THR	Write-only	–
0x28–0x2C	Reserved	–	–	–
0x30	Receive Sync. Holding Register	SSC_RSHR	Read-only	0x0
0x34	Transmit Sync. Holding Register	SSC_TSHR	Read/Write	0x0
0x38	Receive Compare 0 Register	SSC_RC0R	Read/Write	0x0
0x3C	Receive Compare 1 Register	SSC_RC1R	Read/Write	0x0
0x40	Status Register	SSC_SR	Read-only	0x000000CC
0x44	Interrupt Enable Register	SSC_IER	Write-only	–
0x48	Interrupt Disable Register	SSC_IDR	Write-only	–
0x4C	Interrupt Mask Register	SSC_IMR	Read-only	0x0
0x50–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	SSC_WPMR	Read/Write	0x0
0xE8	Write Protection Status Register	SSC_WPSR	Read-only	0x0
0xEC–0xFC	Reserved	–	–	–
0x100–0x128	Reserved for PDC registers	–	–	–

36.7.4 USART Mode Register (SPI_MODE)

Name: US_MR (SPI_MODE)

Address: 0x40024004 (0), 0x40028004 (1)

Access: Read/Write

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

23	22	21	20	19	18	17	16
–	–	–	WRDBT	–	CLKO	–	CPOL

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

7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **USART_MODE: USART Mode of Operation**

Value	Name	Description
0xE	SPI_MASTER	SPI master
0xF	SPI_SLAVE	SPI Slave

- **USCLKS: Clock Selection**

Value	Name	Description
0	MCK	Peripheral clock is selected
1	DIV	Peripheral clock divided (DIV=8) is selected
3	SCK	Serial Clock SLK is selected

- **CHRL: Character Length**

Value	Name	Description
3	8_BIT	Character length is 8 bits

- **CPHA: SPI Clock Phase**
  – Applicable if USART operates in SPI mode (USART_MODE = 0xE or 0xF):

0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

CPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **CPOL: SPI Clock Polarity**

Applicable if USART operates in SPI mode (slave or master, USART_MODE = 0xE or 0xF):

0: The inactive state value of SPCK is logic level zero.

1: The inactive state value of SPCK is logic level one.

Depending on the quadrature signals, the direction is decoded and allows to count up or down in timer/counter channels 0 and 1. The direction status is reported on TC_QISR.

37.6.14.5 Speed Measurement

When SPEEDEN is set in the TC_BMR, the speed measure is enabled on channel 0.

A time base must be defined on channel 2 by writing the TC_RC2 period register. Channel 2 must be configured in Waveform mode (WAVE bit set) in TC_CMR2. The WAVSEL field must be defined with 0x10 to clear the counter by comparison and matching with TC_RC value. Field ACPC must be defined at 0x11 to toggle TIOA output.

This time base is automatically fed back to TIOA of channel 0 when QDEN and SPEEDEN are set.

Channel 0 must be configured in Capture mode (WAVE = 0 in TC_CMR0). The ABETRG bit of TC_CMR0 must be configured at 1 to select TIOA as a trigger for this channel.

EDGTRG must be set to 0x01, to clear the counter on a rising edge of the TIOA signal and field LDRA must be set accordingly to 0x01, to load TC_RA0 at the same time as the counter is cleared (LDRB must be set to 0x01). As a consequence, at the end of each time base period the differentiation required for the speed calculation is performed.

The process must be started by configuring bits CLKEN and SWTRG in the TC_CCR.

The speed can be read on field RA in TC_RA0.

Channel 1 can still be used to count the number of revolutions of the motor.

37.6.15 2-bit Gray Up/Down Counter for Stepper Motor

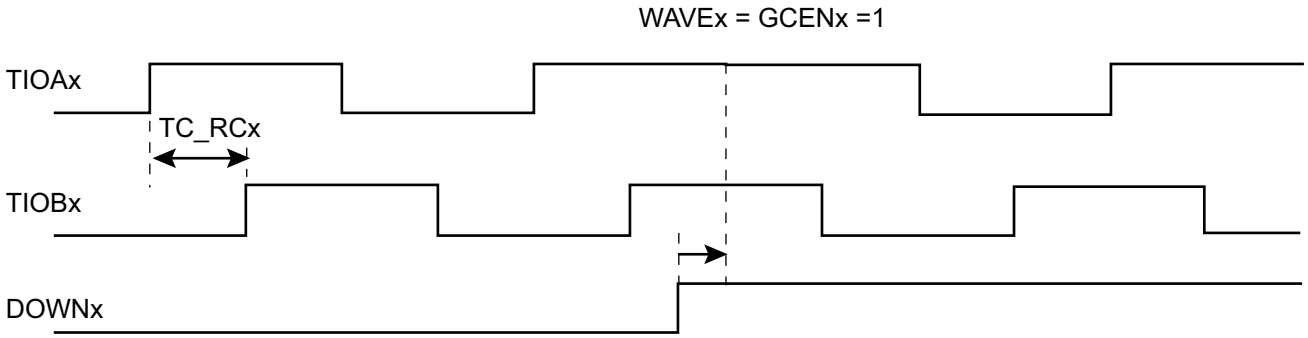
Each channel can be independently configured to generate a 2-bit gray count waveform on corresponding TIOA, TIOB outputs by means of the GCEN bit in TC_SMMRx.

Up or Down count can be defined by writing bit DOWN in TC_SMMRx.

It is mandatory to configure the channel in Waveform mode in the TC_CMR.

The period of the counters can be programmed in TC_RCx.

Figure 37-20. 2-bit Gray Up/Down Counter



37.6.16 Fault Mode

At any time, the TC_RCx registers can be used to perform a comparison on the respective current channel counter value (TC_CVx) with the value of TC_RCx register.

The CPCSx flags can be set accordingly and an interrupt can be generated.

This interrupt is processed but requires an unpredictable amount of time to be achieve the required action.

It is possible to trigger the FAULT output of the TIMER1 with CPCS from TC_SR0 and/or CPCS from TC_SR1. Each source can be independently enabled/disabled in the TC_FMR.

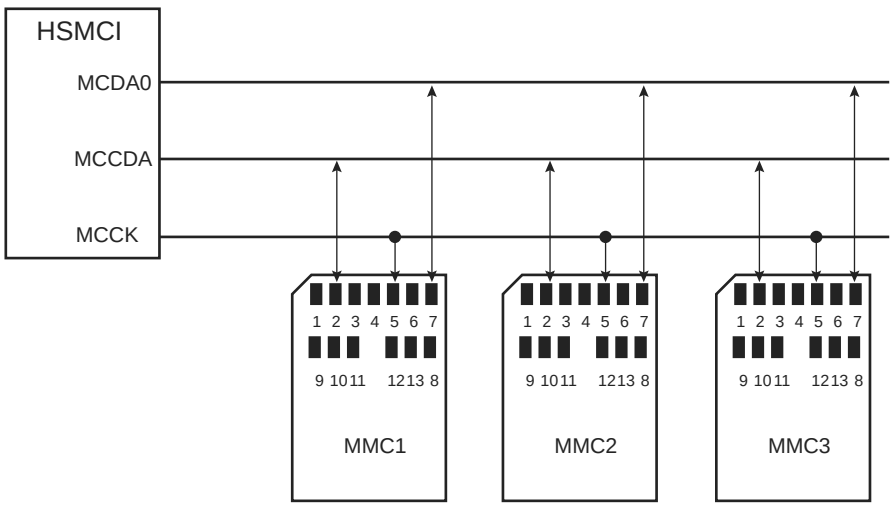
The High Speed MultiMedia Card communication is based on a 13-pin serial bus interface. It has three communication lines and four supply lines.

Table 38-4. Bus Topology

Pin Number	Name	Type[1]	Description	HSMCI Pin Name[2] (Slot z)
1	DAT[3]	I/O/PP	Data	MCDz3
2	CMD	I/O/PP/OD	Command/response	MCCDz
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	MCCK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data 0	MCDz0
8	DAT[1]	I/O/PP	Data 1	MCDz1
9	DAT[2]	I/O/PP	Data 2	MCDz2
10	DAT[4]	I/O/PP	Data 4	MCDz4
11	DAT[5]	I/O/PP	Data 5	MCDz5
12	DAT[6]	I/O/PP	Data 6	MCDz6
13	DAT[7]	I/O/PP	Data 7	MCDz7

Notes: 1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.
2. When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

Figure 38-4. MMC Bus Connections (One Slot)



Note: When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA MCDAy to HSMCIx_DAy.

Table 39-2. I/O Lines

PWM	PWML3	PA15	C
PWM	PWML3	PC3	B
PWM	PWML3	PC22	B

39.5.2 Power Management

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

39.5.3 Interrupt Sources

The PWM interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the PWM interrupt requires the Interrupt Controller to be programmed first.

Table 39-3. Peripheral IDs

Instance	ID
PWM	31

39.5.4 Fault Inputs

The PWM has the fault inputs connected to the different modules. Please refer to the implementation of these module within the product for detailed information about the fault generation procedure. The PWM receives faults from PIO inputs, the PMC, the ADC controller, the Analog Comparator Controller and Timer/Counters.

Table 39-4. Fault Inputs

Fault Generator	External PWM Fault Input Number	Polarity Level(1)	Fault Input ID
PA9	PWMFI0	User-defined	0
Main OSC (PMC)	–	To be configured to 1	1
ADC	–	To be configured to 1	2
Analog Comparator	–	To be configured to 1	3
Timer0	–	To be configured to 1	4
Timer1	–	To be configured to 1	5

Note: 1. FPOL field in PWMC_FMR.

Figure 39-16. Event Line Block Diagram

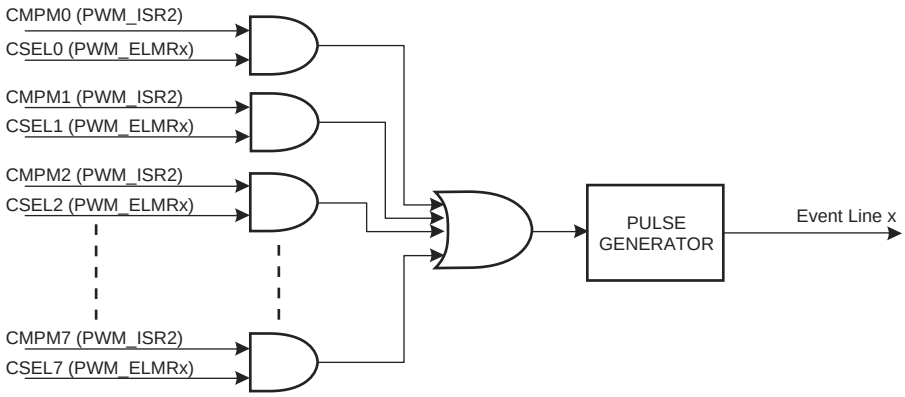
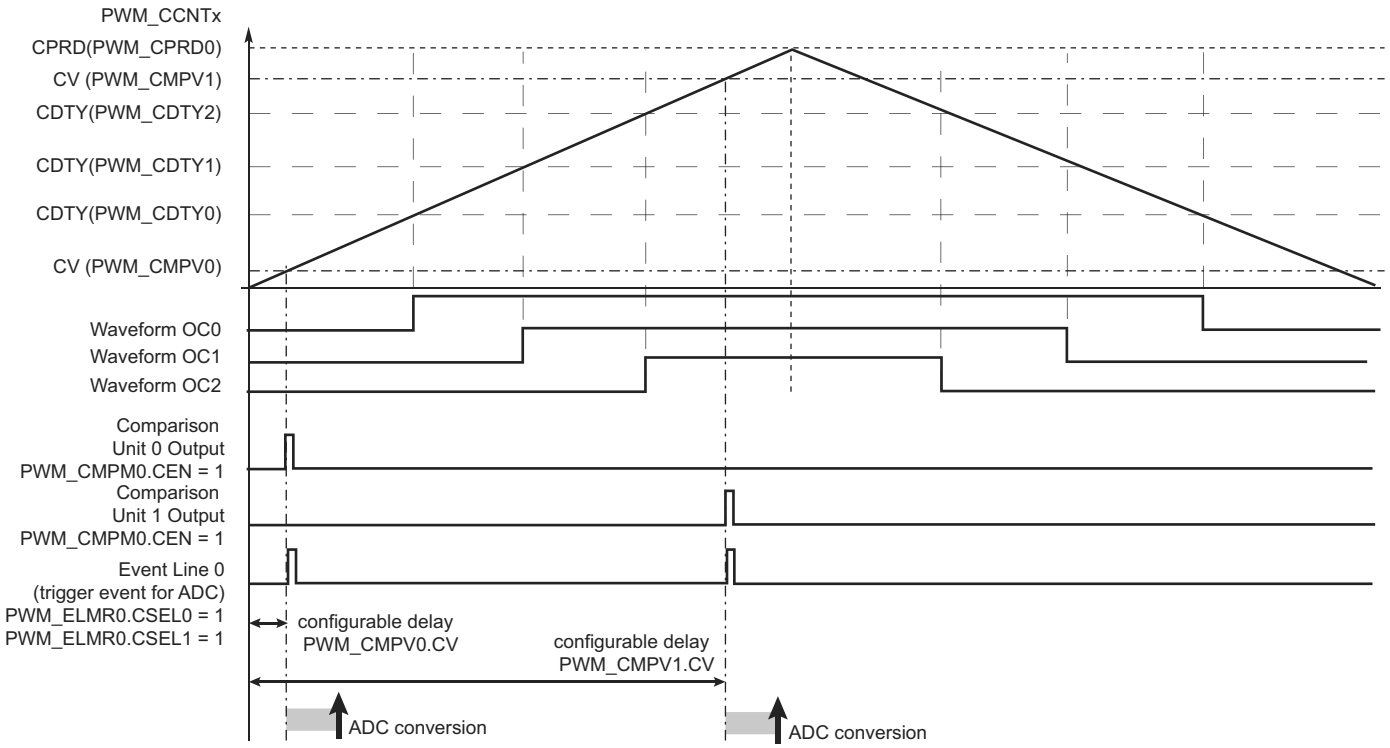


Figure 39-17. Event Line Generation Waveform (Example)



39.6.5 PWM Controller Operations

39.6.5.1 Initialization

Before enabling the channels, they must be configured by the software application as described below:

- Unlock User Interface by writing the WPCMD field in the PWM_WPCR.
- Configuration of the clock generator (DIVA, PREA, DIVB, PREB in the PWM_CLK register if required).
- Selection of the clock for each channel (CPRE field in PWM_CMRx)
- Configuration of the waveform alignment for each channel (CALG field in PWM_CMRx)
- Selection of the counter event selection (if CALG = 1) for each channel (CES field in PWM_CMRx)
- Configuration of the output waveform polarity for each channel (CPOL bit in PWM_CMRx)

- **TAG: Tag Selection Mode**

Value	Name	Description
0	DIS	Tag selection mode disabled. Using USER_SEL to select the channel for the conversion.
1	EN	Tag selection mode enabled

- **MAXS: Maximum Speed Mode**

Value	Name	Description
0	NORMAL	Normal mode
1	MAXIMUM	Maximum speed mode enabled

- **STARTUP: Startup Time Selection**

Value	Name	Description
0	0	0 periods of peripheral clock
1	8	8 periods of peripheral clock
2	16	16 periods of peripheral clock
3	24	24 periods of peripheral clock
4	64	64 periods of peripheral clock
5	80	80 periods of peripheral clock
6	96	96 periods of peripheral clock
7	112	112 periods of peripheral clock
8	512	512 periods of peripheral clock
9	576	576 periods of peripheral clock
10	640	640 periods of peripheral clock
11	704	704 periods of peripheral clock
12	768	768 periods of peripheral clock
13	832	832 periods of peripheral clock
14	896	896 periods of peripheral clock
15	960	960 periods of peripheral clock
16	1024	1024 periods of peripheral clock
17	1088	1088 periods of peripheral clock
18	1152	1152 periods of peripheral clock
19	1216	1216 periods of peripheral clock
20	1280	1280 periods of peripheral clock
21	1344	1344 periods of peripheral clock
22	1408	1408 periods of peripheral clock
23	1472	1472 periods of peripheral clock
24	1536	1536 periods of peripheral clock
25	1600	1600 periods of peripheral clock
26	1664	1664 periods of peripheral clock
27	1728	1728 periods of peripheral clock

Table 49-4. SAM4S Datasheet Rev. 11100H Revision History

Doc. Date	Changes
08-Jan-15	Section 30. “Chip Identifier (CHIPID)”
	Table 30-1 ”SAM4S Chip ID Registers”: updated with Rev B values
	Section 30.3.1 “Chip ID Register”:
	- field ”EPROC: added new row with value 0, name SAMx7, description Cortex-M7
	- field ”NVPSIZ: changed name and description for value 8
	- field ”SRAMSIZ: changed name and description for value 2
	Section 30.3.2 “Chip ID Extension Register”: in EXID field description, replaced "bit" with "field"
	Section 44. “Electrical Characteristics”:
	Updated and harmonized parameter symbols throughout
	Table 44-3 ”DC Characteristics”: updated footnotes
	Table 44-4 ”1.2V Voltage Regulator Characteristics”: replaced two footnotes with single footnote for V_{DDIN} conditions; deleted “Cf. External Capacitor Requirements” from CD_{IN} and CD_{OUT} conditions
	Table 44-5 ”Core Power Supply Brownout Detector Characteristics”: added parameter “Reset Period”
	Table 44-8 ”Zero-Power-on Reset Characteristics”: modified parameter name “Reset Time-out Period” to “Reset Period”
	Section 44.4.2.1 “Sleep Mode”: deleted sentence “Table 44-14 shows the current consumption in typical conditions”; corrected cross-reference link in title of Figure 44-6
	Figure 44-9 ”Measurement Setup for Wait Mode”: replaced “3.3V” with “3.6V”
	Table 44-21 ”SAM4S4/2 Active Power Consumption with VDDCORE @ 1.2V Running from Flash Memory or SRAM”: added footnote “Flash Wait State (FWS) in EEFC_FMR adjusted versus core frequency”
	Table 44-22 ”SAM4S16/S8 Active Power Consumption with VDDCORE @ 1.2V Running from Flash Memory or SRAM”: added “AMP2” caption to SRAM column
	Table 44-27 ”32.768 kHz Crystal Oscillator Characteristics”: added “Allowed Crystal Capacitance Load” parameter
	Figure 44-13 ”32.768 kHz Crystal Oscillator Schematic”: added label “$C_{crystal}$”
	Added Section 44.5.5 “32.768 kHz XIN32 Clock Input Characteristics in Bypass Mode”
	Table 44-30 ”3 to 20 MHz Crystal Oscillator Characteristics”: added “Allowed Crystal Capacitance Load”
	parameter
	Table 44-31 ”3 to 20 MHz Crystal Characteristics”: corrected ESR unit ‘W’ to ‘Ω’
	Table 44-32 ”XIN Clock Electrical Characteristics (In Bypass Mode)”: for $R_{PARASTANDBY}$, replaced “Impedance” with “Resistance” in parameter description and corrected unit ‘W’ to ‘Ω’
	Updated Figure 44-16 ”XIN Clock Timing”
	Figure 44-21 ”Simplified Acquisition Path”: added caption “ADC Input”; replaced caption “12-bit ADC Core” with “12-bit ADC”