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	CANbus, Ethernet, IrDA, MMC/SD, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	79
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 16x12b; D/A 2x12b
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/atsam4e16ca-au

10.2.2 PIO Controller B Multiplexing

Table 10-3. Multiplexing on PIO Controller B (PIOB)

I/O Line	Peripheral A	Peripheral B	Peripheral C	Extra Function	System Function
PB0	PWMH0		RXD0	AFE0_AD4/RTCOUT0(1)	
PB1	PWMH1		TXD0	AFE0_AD5/RTCOUT1(1)	
PB2	CANTX0	NPCS2	CTS0	AFE1_AD0/WKUP12(2)	
PB3	CANRX0	PCK2	RTS0	AFE1_AD1(3)	
PB4	TWD1	PWMH2			TDI(5)
PB5	TWCK1	PWML0		WKUP13(4)	TDO/TRACESWO(5)
PB6					TMS/SWDIO(5)
PB7					TCK/SWCLK(5)
PB8					XOUT(5)
PB9					XIN(5)
PB10					DDM
PB11					DDP
PB12	PWML1				ERASE(5)
PB13	PWML2	PCK0	SCK0	DAC0(6)	
PB14	NPCS1	PWMH3		DAC1(6)	

Notes:
1. Analog input has priority over RTCOUTx pin. See Section 15.5.8 “Waveform Generation”.
2. Analog input has priority over WKUPx pin.
3. To select this extra function, refer to Section 43.5.1 “I/O Lines”.
4. WKUPx can be used if PIO controller defines the I/O line as "input".
5. Refer to Section 6.2 “System I/O Lines”.
6. DAC0 is selected when DACC_CHER.CH0 is set. DAC1 is selected when DACC_CHER.CH1 is set. See Section 44.7.3 “DACC Channel Enable Register”.

11.4.1.8 Program Status Register

Name: PSR

Access: Read/Write

Reset: 0x000000000

31	30	29	28	27	26	25	24
N	Z	C	V	Q	ICI/IT		T

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
ICI/IT						–	ISR_NUMBER

7	6	5	4	3	2	1	0
ISR_NUMBER							

The *Program Status Register* (PSR) combines:

- *Application Program Status Register* (APSR)
- *Interrupt Program Status Register* (IPSR)
- *Execution Program Status Register* (EPSR).

These registers are mutually exclusive bitfields in the 32-bit PSR.

The PSR accesses these registers individually or as a combination of any two or all three registers, using the register name as an argument to the MSR or MRS instructions. For example:

- Read of all the registers using PSR with the MRS instruction
- Write to the APSR N, Z, C, V and Q bits using APSR_nzcvq with the MSR instruction.

The PSR combinations and attributes are:

Name	Access	Combination
PSR	Read/Write(1)(2)	APSR, EPSR, and IPSR
IEPSR	Read-only	EPSR and IPSR
IAPSR	Read/Write(1)	APSR and IPSR
EAPSR	Read/Write(2)	APSR and EPSR

Notes: 1. The processor ignores writes to the IPSR bits.
2. Reads of the EPSR bits return zero, and the processor ignores writes to these bits.

See the instruction descriptions “MRS” and “MSR” for more information about how to access the program status registers.

19.3.1 General Purpose Backup Register x

Name: SYS_GPBRx

Address: 0x400E1890

Access: Read/Write

31	30	29	28	27	26	25	24
GPBR_VALUE							

23	22	21	20	19	18	17	16
GPBR_VALUE							

15	14	13	12	11	10	9	8
GPBR_VALUE							

7	6	5	4	3	2	1	0
GPBR_VALUE							

These registers are reset at first power-up and on each loss of VDDIO.

- **GPBR_VALUE: Value of GPBR x**

If a Tamper event has been detected, it is not possible to write GPBR_VALUE as long as the LPDBCS0 or LPDBCS1 flag has not been cleared in the Supply Controller Status Register (SUPC_SR).

22.4 Functional Description

22.4.1 Cache Operation

On reset, the cache controller data entries are all invalidated and the cache is disabled. The cache is transparent to processor operations. The cache controller is activated with its configuration registers. The configuration interface is memory-mapped in the private peripheral bus.

Use the following sequence to enable the cache controller:

1. Verify that the cache controller is disabled by reading the value of the CSTS (Cache Controller Status) bit of the Status register (CMCC_SR).
2. Enable the cache controller by writing a one to the CEN (Cache Enable) bit of the Control register (CMCC_CTRL).

22.4.2 Cache Maintenance

If the contents seen by the cache have changed, the user must invalidate the cache entries. This can be done line-by-line or for all cache entries.

22.4.2.1 Cache Invalidate-by-Line Operation

When an invalidate-by-line command is issued, the cache controller resets the valid bit information of the decoded cache line. As the line is no longer valid, the replacement counter points to that line.

Use the following sequence to invalidate one line of cache:

1. Disable the cache controller by clearing the CEN bit of CMCC_CTRL.
2. Check the CSTS bit of CMCC_SR to verify that the cache is successfully disabled.
3. Perform an invalidate-by-line by configuring the bits INDEX and WAY in the Maintenance Register 1 (CMCC_MAINT1).
4. Enable the cache controller by writing a one the CEN bit of the CMCC_CTRL.

22.4.2.2 Cache Invalidate All Operation

To invalidate all cache entries, write a one to the INVALL bit of the Maintenance Register 0 (CMCC_MAINT0).

22.4.3 Cache Performance Monitoring

The Cortex-M cache controller includes a programmable 32-bit monitor counter. The monitor can be configured to count the number of clock cycles, the number of data hits or the number of instruction hits.

Use the following sequence to activate the counter:

1. Configure the monitor counter by writing to the MODE field of the Monitor Configuration register (CMCC_MCFG).
2. Enable the counter by writing a one to the MENABLE bit of the Monitor Enable register (CMCC_MEN).
3. If required, clear the counter by writing a one to the SWRST bit of the Monitor Control register (CMCC_MCTRL).
4. Check the value of the monitor counter by reading the EVENT_CNT field of the CMCC_MSR.

24. Bus Matrix (MATRIX)

24.1 Description

The Bus Matrix (MATRIX) implements a multi-layer AHB, based on the AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system, thus increasing the overall bandwidth. The Bus Matrix interconnects up to 7 AHB masters to up to 6 AHB slaves. The normal latency to connect a master to a slave is one cycle except for the default master of the accessed slave which is connected directly (zero cycle latency). The Bus Matrix user interface is compliant with ARM Advanced Peripheral Bus.

24.2 Embedded Characteristics

- Configurable Number of Masters (up to 7)
- Configurable Number of Slaves (up to 6)
- One Decoder for Each Master
- Several Possible Boot Memories for Each Master before Remap
- One Remap Function for Each Master
- Support for Long Bursts of 32, 64, 128 and up to the 256-beat Word Burst AHB Limit
- Enhanced Programmable Mixed Arbitration for Each Slave
  - Round-Robin
  - Fixed Priority
- Programmable Default Master for Each Slave
  - No Default Master
  - Last Accessed Default Master
  - Fixed Default Master
- Deterministic Maximum Access Latency for Masters
- Zero or One Cycle Arbitration Latency for the First Access of a Burst
- Bus Lock Forwarding to Slaves
- Master Number Forwarding to Slaves
- One Special Function Register for Each Slave (not dedicated)
- Write Protection of User Interface Registers

24.2.1 Matrix Masters

The Bus Matrix manages 7 masters, which means that each master can perform an access concurrently with others, to an available slave.

Each master has its own decoder, which is defined specifically for each master. In order to simplify the addressing, all the masters have the same decodings.

24.2.2 Matrix Slaves

The Bus Matrix manages 6 slaves. Each slave has its own arbiter, allowing a different arbitration per slave.

Table 24-1. List of Bus Matrix Slaves

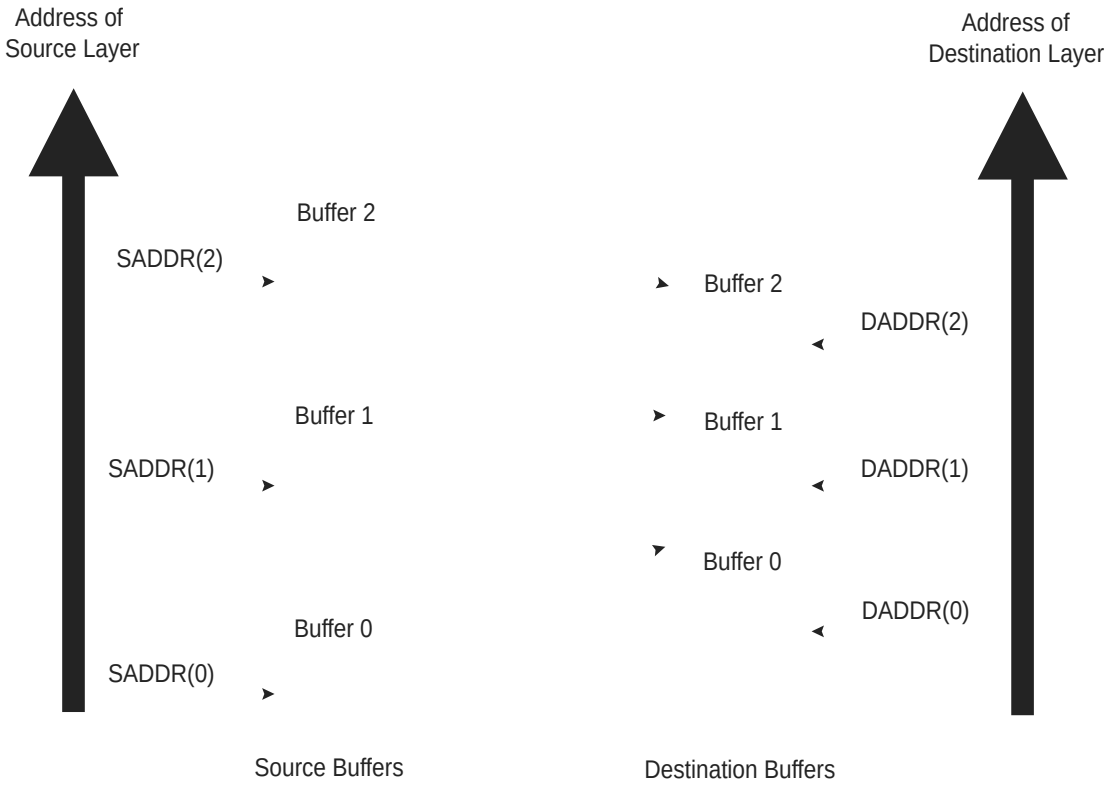
Slave 0	Internal SRAM
Slave 1	Internal ROM
Slave 2	Internal Flash

Note: Do not poll the DMAC_CTRLAx.DONE bit in the DMAC memory map. Instead, poll the LLI.DMAC_CTRLAx.DONE bit in the LLI for that buffer. If the poll LLI.DMAC_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC_CTRLAx.DONE bit was cleared at the start of the transfer.

16. The DMAC does not wait for the buffer interrupt to be cleared, but continues and fetches the next LLI from the memory location pointed to by the current DMAC_DSCRx register, then automatically reprograms the DMAC_SADDRx, DMAC_CTRLAx, DMAC_CTRLBx and DMAC_DSCRx channel registers. DMAC_DADDRx is left unchanged. The DMAC transfer continues until the DMAC samples the DMAC_CTRLAx, DMAC_CTRLBx and DMAC_DSCRx registers at the end of a buffer transfer match that described in Row 1 of Table 25-3 on page 473. The DMAC then knows that the previous buffer transferred was the last buffer in the DMAC transfer.

The DMAC transfer might look like that shown in Figure 25-8. Note that the destination address is decrementing.

Figure 25-8. DMAC Transfer with Linked List Source Address and Contiguous Destination Address



The DMAC transfer flow is shown in Figure 25-9 on page 481.

25.8.6 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Enable Register

Name: DMAC_EBCIER

Address: 0x400C0018

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	ERR3	ERR2	ERR1	ERR0
15	14	13	12	11	10	9	8
–	–	–	–	CBTC3	CBTC2	CBTC1	CBTC0
7	6	5	4	3	2	1	0
–	–	–	–	BTC3	BTC2	BTC1	BTC0

- **BTCx: Buffer Transfer Completed [3:0]**

Buffer Transfer Completed Interrupt Enable Register. Set the relevant bit in the BTC field to enable the interrupt for channel i.

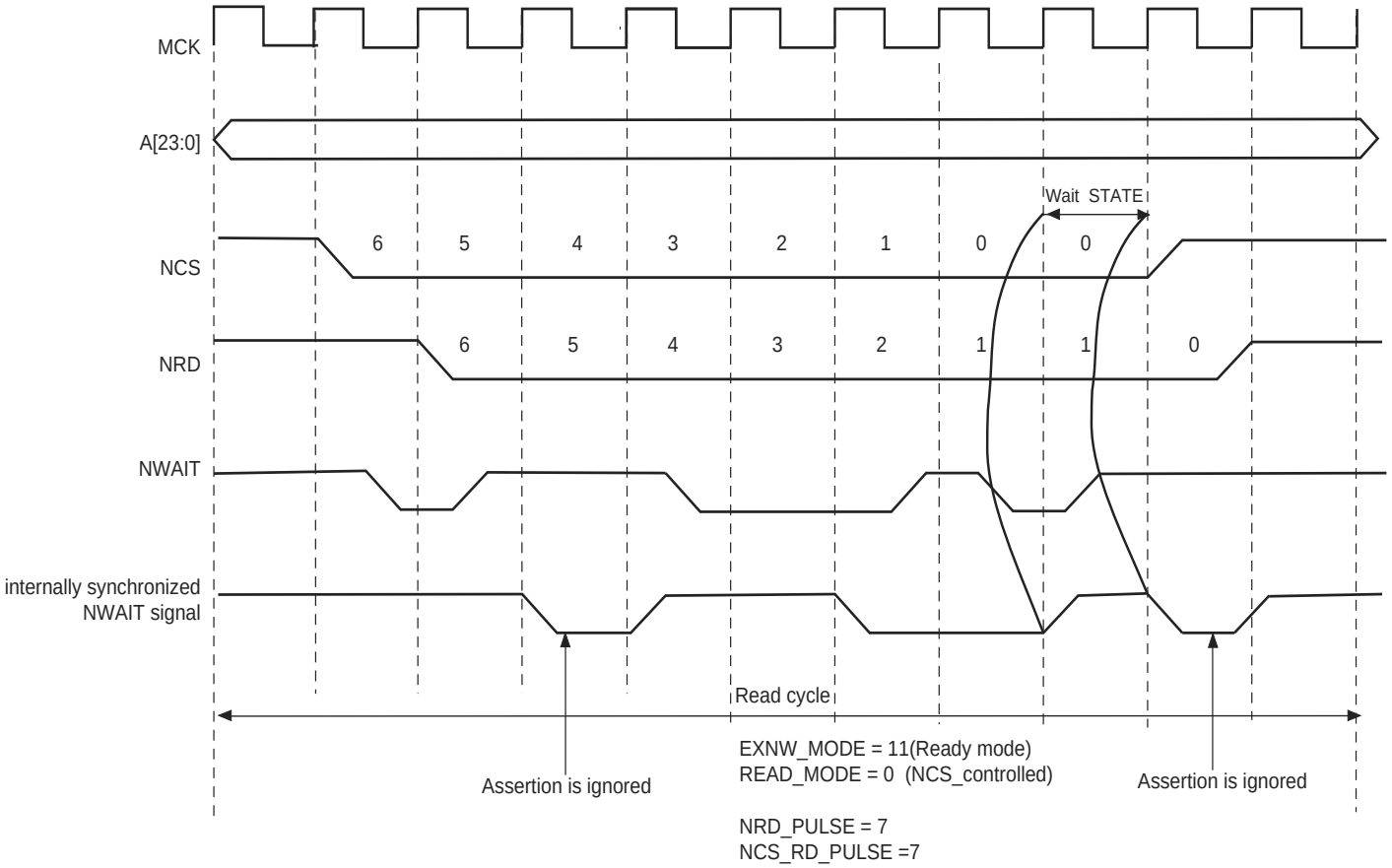
- **CBTCx: Chained Buffer Transfer Completed [3:0]**

Chained Buffer Transfer Completed Interrupt Enable Register. Set the relevant bit in the CBTC field to enable the interrupt for channel i.

- **ERRx: Access Error [3:0]**

Access Error Interrupt Enable Register. Set the relevant bit in the ERR field to enable the interrupt for channel i.

Figure 27-26. NWAIT Assertion in Read Access: Ready Mode (SMC_MODE.EXNW_MODE = 11)



29.18.3 PMC System Clock Status Register

Name: PMC_SCSR

Address: 0x400E0408

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
–	–	–	–	–	PCK2	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	–	–	–	–	–	–	–

- **UDP: USB Device Port Clock Status**

0: The 48 MHz clock (UDPCK) of the USB Device Port is disabled.

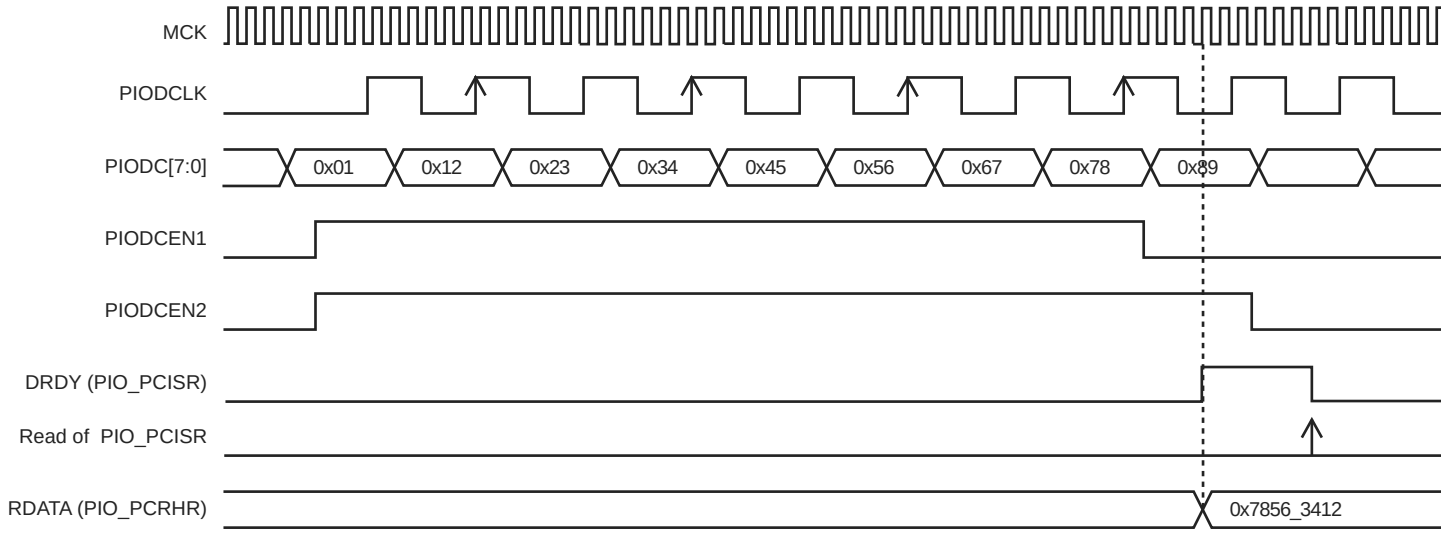
1: The 48 MHz clock (UDPCK) of the USB Device Port is enabled.

- **PCKx: Programmable Clock x Output Status**

0: The corresponding Programmable Clock output is disabled.

1: The corresponding Programmable Clock output is enabled.

Figure 33-13. Parallel Capture Mode Waveforms (DSIZE = 2, ALWYS = 0, HALFS = 1, FRSTS = 1)



33.5.14.3 Restrictions

- Configuration fields DSIZE, ALWYS, HALFS and FRSTS in PIO_PCMR can be changed **ONLY** if the parallel capture mode is disabled at this time (PCEN = 0 in PIO_PCMR).
- The frequency of peripheral clock must be strictly superior to two times the frequency of the clock of the device which generates the parallel data.

33.5.14.4 Programming Sequence

Without PDC

1. Write PIO_PCIDR and PIO_PCIER in order to configure the parallel capture mode interrupt mask.
2. Write PIO_PCMR to set the fields DSIZE, ALWYS, HALFS and FRSTS in order to configure the parallel capture mode **WITHOUT** enabling the parallel capture mode.
3. Write PIO_PCMR to set the PCEN bit to one in order to enable the parallel capture mode **WITHOUT** changing the previous configuration.
4. Wait for a data ready by polling the DRDY flag in PIO_PCISR or by waiting for the corresponding interrupt.
5. Check OVRE flag in PIO_PCISR.
6. Read the data in PIO_PCRHR.
7. If new data are expected, go to step 4.
8. Write PIO_PCMR to set the PCEN bit to zero in order to disable the parallel capture mode **WITHOUT** changing the previous configuration.

With PDC

33.6 Parallel Input/Output Controller (PIO) User Interface

Each I/O line controlled by the PIO Controller is associated with a bit in each of the PIO Controller User Interface registers. Each register is 32-bit wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero. If the I/O line is not multiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO_PSR returns one systematically.

Table 33-5. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	PIO Enable Register	PIO_PER	Write-only	–
0x0004	PIO Disable Register	PIO_PDR	Write-only	–
0x0008	PIO Status Register	PIO_PSR	Read-only	(1)
0x000C	Reserved	–	–	–
0x0010	Output Enable Register	PIO_OER	Write-only	–
0x0014	Output Disable Register	PIO_ODR	Write-only	–
0x0018	Output Status Register	PIO_OSR	Read-only	0x00000000
0x001C	Reserved	–	–	–
0x0020	Glitch Input Filter Enable Register	PIO_IFER	Write-only	–
0x0024	Glitch Input Filter Disable Register	PIO_IFDR	Write-only	–
0x0028	Glitch Input Filter Status Register	PIO_IFSR	Read-only	0x00000000
0x002C	Reserved	–	–	–
0x0030	Set Output Data Register	PIO_SODR	Write-only	–
0x0034	Clear Output Data Register	PIO_CODR	Write-only	
0x0038	Output Data Status Register	PIO_ODSR	Read-only or(2) Read/Write	–
0x003C	Pin Data Status Register	PIO_PDSR	Read-only	(3)
0x0040	Interrupt Enable Register	PIO_IER	Write-only	–
0x0044	Interrupt Disable Register	PIO_IDR	Write-only	–
0x0048	Interrupt Mask Register	PIO_IMR	Read-only	0x00000000
0x004C	Interrupt Status Register(4)	PIO_ISR	Read-only	0x00000000
0x0050	Multi-driver Enable Register	PIO_MDER	Write-only	–
0x0054	Multi-driver Disable Register	PIO_MDDR	Write-only	–
0x0058	Multi-driver Status Register	PIO_MDSR	Read-only	0x00000000
0x005C	Reserved	–	–	–
0x0060	Pull-up Disable Register	PIO_PUDR	Write-only	–
0x0064	Pull-up Enable Register	PIO_PUER	Write-only	–
0x0068	Pad Pull-up Status Register	PIO_PUSR	Read-only	(1)
0x006C	Reserved	–	–	–

34.5 Signal Description

Table 34-1. Signal Description

Pin Name	Pin Description	Type	
		Master	Slave
MISO	Master In Slave Out	Input	Output
MOSI	Master Out Slave In	Output	Input
SPCK	Serial Clock	Output	Input
NPCS1–NPCS3	Peripheral Chip Selects	Output	Unused
NPCS0/NSS	Peripheral Chip Select/Slave Select	Output	Input

34.6 Product Dependencies

34.6.1 I/O Lines

The pins used for interfacing the compliant external devices can be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions.

Table 34-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
SPI	MISO	PA12	A
SPI	MOSI	PA13	A
SPI	NPCS0	PA11	A
SPI	NPCS1	PA9	B
SPI	NPCS1	PA31	A
SPI	NPCS1	PB14	A
SPI	NPCS1	PC4	B
SPI	NPCS2	PA10	B
SPI	NPCS2	PA30	B
SPI	NPCS2	PB2	B
SPI	NPCS3	PA3	B
SPI	NPCS3	PA5	B
SPI	NPCS3	PA22	B
SPI	SPCK	PA14	A

34.6.2 Power Management

The SPI can be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SPI clock.

35. Two-wire Interface (TWI)

35.1 Description

The Atmel Two-wire Interface (TWI) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 Kbits per second, based on a byte-oriented transfer format. It can be used with any Atmel Two-wire Interface bus Serial EEPROM and I²C compatible device such as a Real Time Clock (RTC), Dot Matrix/Graphic LCD Controllers and temperature sensor. The TWI is programmable as a master or a slave with sequential or single-byte access. Multiple master capability is supported.

A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.

Table 35-1 lists the compatibility level of the Atmel Two-wire Interface in Master mode and a full I^2C compatible device.

Table 35-1. Atmel TWI Compatibility with I^2C Standard

I^2C Standard	Atmel TWI
Standard Mode Speed (100 kHz)	Supported
Fast Mode Speed (400 kHz)	Supported
7- or 10-bit Slave Addressing	Supported
START byte(1)	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NACK Management	Supported
Slope Control and Input Filtering (Fast mode)	Not Supported
Clock Stretching/Synchronization	Supported
Multi Master Capability	Supported

Note: 1. START + b000000001 + Ack + Sr

35.2 Embedded Characteristics

- Compatible with Atmel Two-wire Interface Serial Memory and I^2C Compatible Devices(1)
- One, Two or Three Bytes for Slave Address
- Sequential Read/Write Operations
- Master, Multi-master and Slave Mode Operation
- Bit Rate: Up to 400 Kbit/s
- General Call Supported in Slave Mode
- SMBus Quick Command Supported in Master Mode
- Connection to Peripheral DMA Controller (PDC) Channel Capabilities Optimizes Data Transfers
  - One Channel for the Receiver, One Channel for the Transmitter
- Register Write Protection

Note: 1. See Table 35-1 for details on compatibility with I²C Standard.

Figure 37-37. SPI Transfer Format (CPHA = 1, 8 bits per transfer)

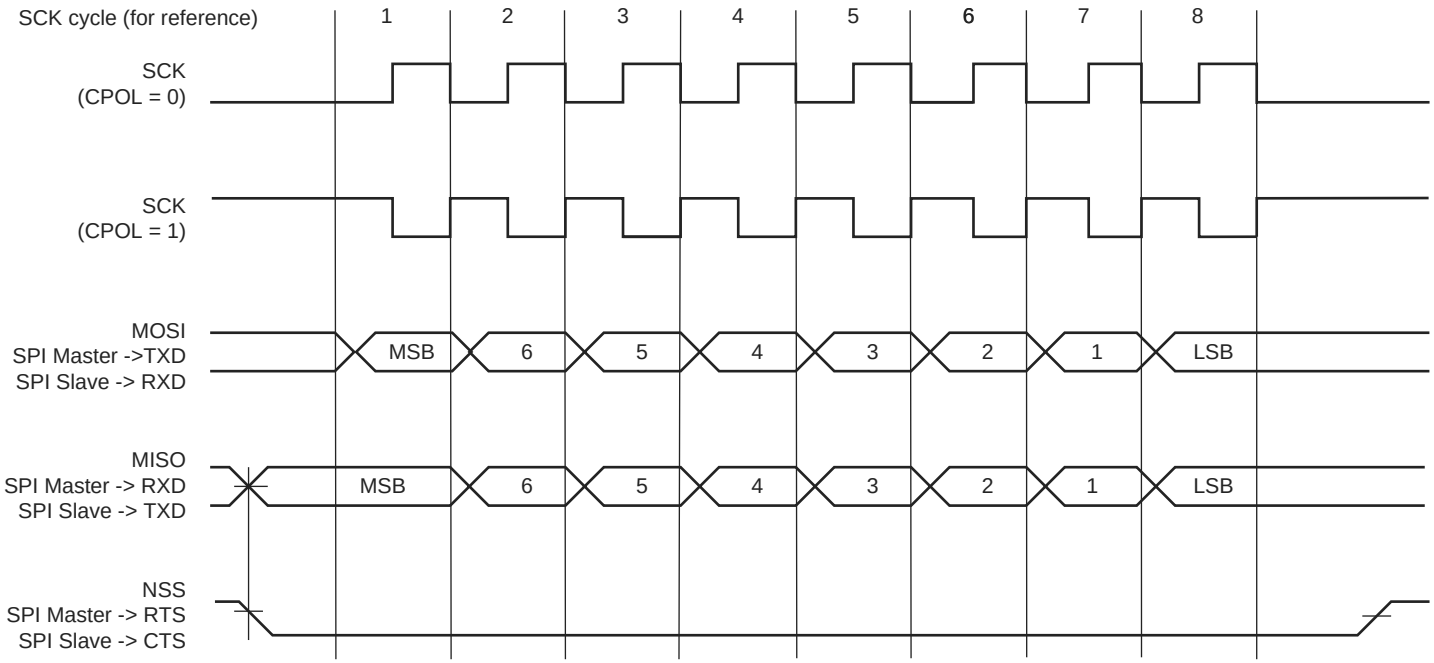
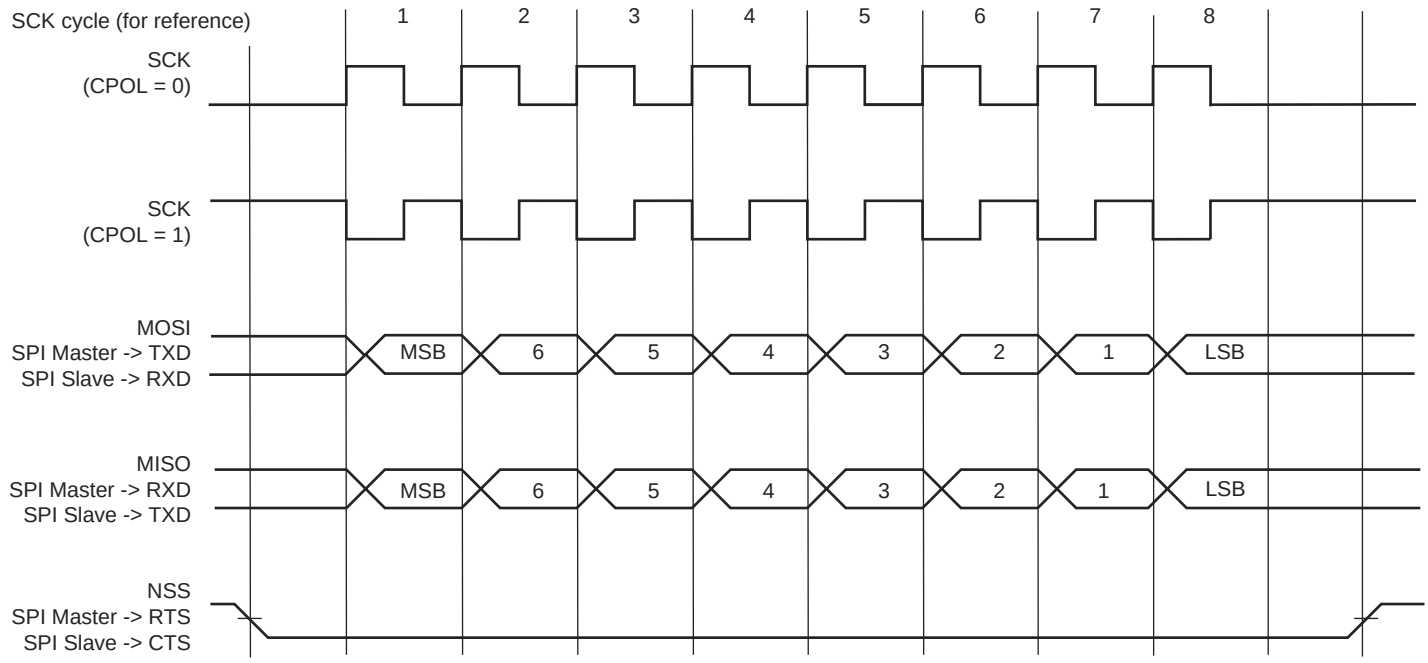


Figure 37-38. SPI Transfer Format (CPHA = 0, 8 bits per transfer)



37.6.8.4 Receiver and Transmitter Control

See Section 37.6.2 ”Receiver and Transmitter Control”

37.6.8.5 Character Transmission

The characters are sent by writing in the Transmit Holding register (US_THR). An additional condition for transmitting a character can be added when the USART is configured in SPI Master mode. In the USART Mode Register (SPI_MODE) (USART_MR), the value configured on the bit WRDBT can prevent any character

38.7.4 TC Stepper Motor Mode Register

Name: TC_SMMRx [x=0..2]

Address: 0x40090008 (0)[0], 0x40090048 (0)[1], 0x40090088 (0)[2], 0x40094008 (1)[0], 0x40094048 (1)[1], 0x40094088 (1)[2], 0x40098008 (2)[0], 0x40098048 (2)[1], 0x40098088 (2)[2]

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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	DOWN	GCEN

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

- **GCEN: Gray Count Enable**

0: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by internal counter of channel x.

1: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by a 2-bit Gray counter.

- **DOWN: Down Count**

0: Up counter.

1: Down counter.

39.7.7 PWM Interrupt Mask Register 1

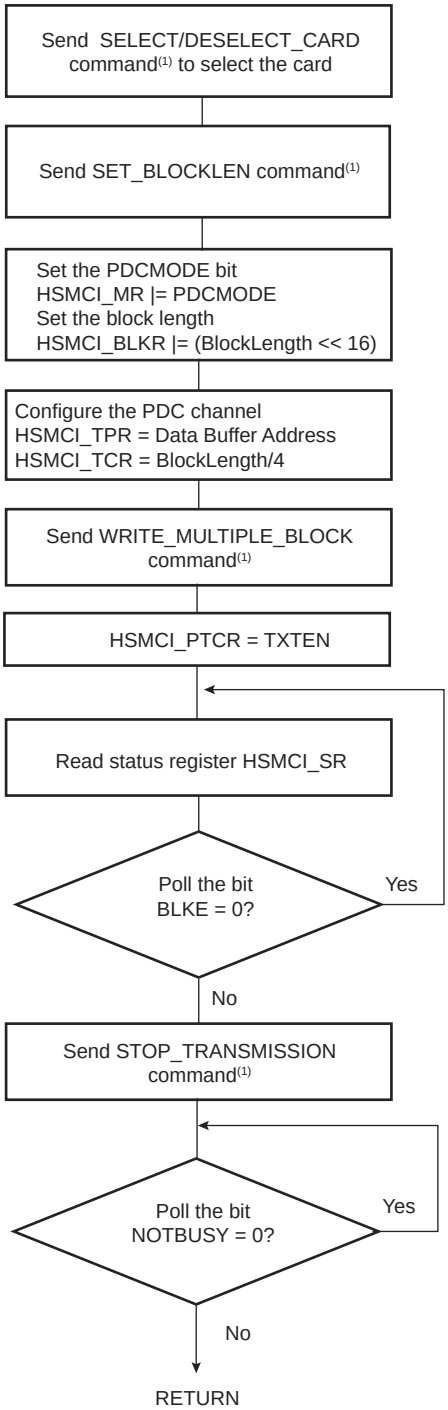
Name: PWM_IMR1

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Counter Event on Channel x Interrupt Mask**
- **FCHIDx: Fault Protection Trigger on Channel x Interrupt Mask**

Figure 40-10. Multiple Write Functional Flow Diagram



Note: 1. It is assumed that this command has been correctly sent (see Figure 40-7).

40.9 SD/SDIO Card Operation

The High Speed MultiMedia Card Interface allows processing of SD Memory (Secure Digital Memory Card) and SDIO (SD Input Output) Card commands.

SD/SDIO cards are based on the MultiMedia Card (MMC) format, but are physically slightly thicker and feature higher data transfer rates, a lock switch on the side to prevent accidental overwriting and security features. The

42.8.7 GMAC Receive Buffer Queue Base Address Register

Name: GMAC_RBQB

Address: 0x40034018

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						–	–

This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the Network Control Register. Once reception is enabled, any write to the Receive Buffer Queue Base Address Register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the “used” bits.

In terms of AMBA AHB operation, the descriptors are read from memory using a single 32-bit AHB access. The descriptors should be aligned at 32-bit boundaries and the descriptors are written to using two individual non sequential accesses.

- **ADDR: Receive Buffer Queue Base Address**

Written with the address of the start of the receive queue.

42.8.12 GMAC Interrupt Disable Register

Name: GMAC_IDR

Address: 0x4003402C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	WOL	–	SRI	PDRSFT	PDRQFT

23	22	21	20	19	18	17	16
PDRSFR	PDRQFR	SFT	DRQFT	SFR	DRQFR	–	–

15	14	13	12	11	10	9	8
EXINT	PFTR	PTZ	PFNZ	HRESP	ROVR	–	–

7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	TUR	TXUBR	RXUBR	RCOMP	MFS

This register is write-only and when read will return zero.

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

0: No effect.

1: Disables the corresponding interrupt.

- **MFS: Management Frame Sent**
- **RCOMP: Receive Complete**
- **RXUBR: RX Used Bit Read**
- **TXUBR: TX Used Bit Read**
- **TUR: Transmit Underrun**
- **RLEX: Retry Limit Exceeded or Late Collision**
- **TFC: Transmit Frame Corruption Due to AHB Error**
- **TCOMP: Transmit Complete**
- **ROVR: Receive Overrun**
- **HRESP: HRESP Not OK**
- **PFNZ: Pause Frame with Non-zero Pause Quantum Received**
- **PTZ: Pause Time Zero**
- **PFTR: Pause Frame Transmitted**
- **EXINT: External Interrupt**
- **DRQFR: PTP Delay Request Frame Received**

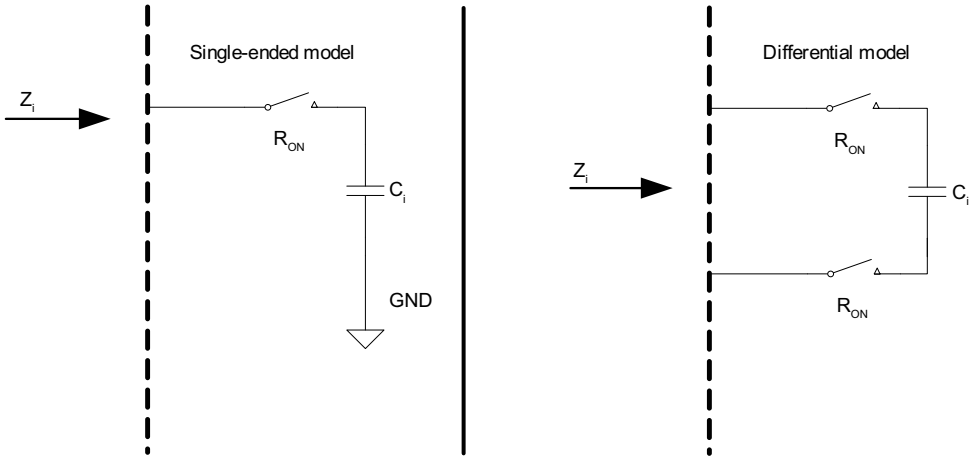
Low Voltage Supply

The ADC operates in 10-bit mode or 12-bit mode. Working at low voltage (V_{DDIN} or/and V_{ADVREF}) between 2 and 2.4V is subject to the following restrictions:

- The field IBCTL must be 00 to reduce the biasing of the ADC under low voltage. See Section 46.7.1.1 “ADC Bias Current”.
- In 10-bit mode, the ADC clock should not exceed 5 MHz (max signal bandwidth is 250 kHz).
- In 12-bit mode, the ADC clock should not exceed 2 MHz (max signal bandwidth is 100 kHz).

46.7.5.3 ADC Channel Input Impedance

Figure 46-18. Input Channel Model



where:

- Z_i is input impedance in single-ended or differential mode
- C_i = 1 to 8 pF ±20% depending on the gain value and mode (SE or DIFF); temperature dependency is negligible
- R_{ON} is typical 2 kΩ and 8 kΩ max (worst case process and high temperature)
- R_{ON} is negligible regarding the value of Z_i

The following formula is used to calculate input impedance:

$$Z_i = \frac{1}{f_S \times C_i}$$

where:

- f_S is the sampling frequency of the ADC channel
- Typ values are used to compute ADC input impedance Z_i

Table 46-43. Input Capacitance (C_{IN}) Values

Gain Selection	Single-ended	Differential
0.5	–	2 pF
1	2 pF	4 pF
2	2 pF	8 pF
4	4 pF	–