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	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 11x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	64-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4s8ba-an

12.4.1.17 Exceptions and Interrupts

The Cortex-M4 processor supports interrupts and system exceptions. The processor and the *Nested Vectored Interrupt Controller* (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses the Handler mode to handle all exceptions except for reset. See "Exception Entry" and "Exception Return" for more information.

The NVIC registers control interrupt handling. See "Nested Vectored Interrupt Controller (NVIC)" for more information.

12.4.1.18 Data Types

The processor supports the following data types:

- 32-bit words
- 16-bit halfwords
- 8-bit bytes
- The processor manages all data memory accesses as little-endian. Instruction memory and *Private Peripheral Bus* (PPB) accesses are always little-endian. See "Memory Regions, Types and Attributes" for more information.

12.4.1.19 Cortex Microcontroller Software Interface Standard (CMSIS)

For a Cortex-M4 microcontroller system, the *Cortex Microcontroller Software Interface Standard* (CMSIS) defines:

- A common way to:
  – Access peripheral registers
  – Define exception vectors
- The names of:
  – The registers of the core peripherals
  – The core exception vectors
- A device-independent interface for RTOS kernels, including a debug channel.

The CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M4 processor.

The CMSIS simplifies the software development by enabling the reuse of template code and the combination of CMSIS-compliant software components from various middleware vendors. Software vendors can expand the CMSIS to include their peripheral definitions and access functions for those peripherals.

This document includes the register names defined by the CMSIS, and gives short descriptions of the CMSIS functions that address the processor core and the core peripherals.

Note: This document uses the register short names defined by the CMSIS. In a few cases, these differ from the architectural short names that might be used in other documents.

The following sections give more information about the CMSIS:

- Section 12.5.3 "Power Management Programming Hints"
- Section 12.6.2 "CMSIS Functions"
- Section 12.8.2.1 "NVIC Programming Hints".

Atmel

Examples

```
LDR     R8, [R10]                ; Loads R8 from the address in R10.
LDRNE   R2, [R5, #960]!          ; Loads (conditionally) R2 from a word
                                 ; 960 bytes above the address in R5, and
                                 ; increments R5 by 960.
STR     R2, [R9,#const-struc]    ; const-struc is an expression evaluating
                                 ; to a constant in the range 0-4095.
STRH    R3, [R4], #4             ; Store R3 as halfword data into address in
                                 ; R4, then increment R4 by 4
LDRD    R8, R9, [R3, #0x20]      ; Load R8 from a word 32 bytes above the
                                 ; address in R3, and load R9 from a word 36
                                 ; bytes above the address in R3
STRD    R0, R1, [R8], #-16       ; Store R0 to address in R8, and store R1 to
                                 ; a word 4 bytes above the address in R8,
                                 ; and then decrement R8 by 16.
```

12.6.4.3   LDR and STR, Register Offset

Load and Store with register offset.

Syntax

   op{type}{cond} Rt, [Rn, Rm {, LSL #n}]

where:

op   is one of:

   LDR   Load Register.

   STR   Store Register.

type   is one of:

   B   unsigned byte, zero extend to 32 bits on loads.

   SB   signed byte, sign extend to 32 bits (LDR only).

   H   unsigned halfword, zero extend to 32 bits on loads.

   SH   signed halfword, sign extend to 32 bits (LDR only).

   -       omit, for word.

cond   is an optional condition code, see "Conditional Execution" .

Rt   is the register to load or store.

Rn   is the register on which the memory address is based.

Rm   is a register containing a value to be used as the offset.

LSL #*n*   is an optional shift, with *n* in the range 0 to 3.

Operation

LDR instructions load a register with a value from memory.

STR instructions store a register value into memory.

The memory address to load from or store to is at an offset from the register *Rn*. The offset is specified by the register *Rm* and can be shifted left by up to 3 bits using LSL.

The value to load or store can be a byte, halfword, or word. For load instructions, bytes and halfwords can either be signed or unsigned. See "Address Alignment" .

Restrictions

In these instructions:

● *Rn* must not be PC

Atmel

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}
```

Atmel

12.7 Cortex-M4 Core Peripherals

12.7.1 Peripherals

- Nested Vectored Interrupt Controller (NVIC)
  The Nested Vectored Interrupt Controller (NVIC) is an embedded interrupt controller that supports low latency interrupt processing. See Section 12.8 "Nested Vectored Interrupt Controller (NVIC)".
- System Control Block (SCB)
  The System Control Block (SCB) is the programmers model interface to the processor. It provides system implementation information and system control, including configuration, control, and reporting of system exceptions. See Section 12.9 "System Control Block (SCB)".
- System Timer (SysTick)
  The System Timer, SysTick, is a 24-bit count-down timer. Use this as a Real Time Operating System (RTOS) tick timer or as a simple counter. See Section 12.10 "System Timer (SysTick)".
- Memory Protection Unit (MPU)
  The Memory Protection Unit (MPU) improves system reliability by defining the memory attributes for different memory regions. It provides up to eight different regions, and an optional predefined background region. See Section 12.11 "Memory Protection Unit (MPU)".

12.7.2 Address Map

The address map of the *Private peripheral bus* (PPB) is given in the following table.

Table 12-28.   Core Peripheral Register Regions

Address	Core Peripheral
0xE000E008–0xE000E00F	System Control Block
0xE000E010–0xE000E01F	System Timer
0xE000E100–0xE000E4EF	Nested Vectored Interrupt Controller
0xE000ED00–0xE000ED3F	System control block
0xE000ED90–0xE000EDB8	Memory Protection Unit
0xE000EF00–0xE000EF03	Nested Vectored Interrupt Controller

In register descriptions:

- The *required privilege* gives the privilege level required to access the register, as follows:
    - Privileged: Only privileged software can access the register.
    - Unprivileged: Both unprivileged and privileged software can access the register.

12.9.1.5 Application Interrupt and Reset Control Register

Name: SCB_AIRCR

Access: Read/Write

31	30	29	28	27	26	25	24
			VECTKEYSTAT/VECTKEY				

23	22	21	20	19	18	17	16
			VECTKEYSTAT/VECTKEY				

15	14	13	12	11	10	9	8
ENDIANNESS	–	–	–	–		PRIGROUP	

7	6	5	4	3	2	1	0
–	–	–	–	–	SYSRESETREQ	VECTCLRACTIVE	VECTRESET

The SCB_AIRCR provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system. To write to this register, write 0x5FA to the VECTKEY field, otherwise the processor ignores the write.

* **VECTKEYSTAT: Register Key (Read)**

Reads as 0xFA05.

* **VECTKEY: Register Key (Write)**

Writes 0x5FA to VECTKEY, otherwise the write is ignored.

* **ENDIANNESS: Data Endianness**

0: Little-endian.

1: Big-endian.

* **PRIGROUP: Interrupt Priority Grouping**

This field determines the split of group priority from subpriority. It shows the position of the binary point that splits the PRI_*n* fields in the Interrupt Priority Registers into separate *group priority* and *subpriority* fields. The table below shows how the PRIGROUP value controls this split.

	Interrupt Priority Level Value, PRI_*N*[7:0]			Number of	
PRIGROUP	**Binary Point**[1]	**Group Priority Bits**	**Subpriority Bits**	**Group Priorities**	**Subpriorities**
0b000	bxxxxxxx.y	[7:1]	None	128	2
0b001	bxxxxxx.yy	[7:2]	[4:0]	64	4
0b010	bxxxxx.yyy	[7:3]	[4:0]	32	8
0b011	bxxxx.yyyy	[7:4]	[4:0]	16	16
0b100	bxxx.yyyyy	[7:5]	[4:0]	8	32
0b101	bxx.yyyyyy	[7:6]	[5:0]	4	64
0b110	bx.yyyyyyy	[7]	[6:0]	2	128
0b111	b.yyyyyyy	None	[7:0]	1	256

Note: 1. PRI_n[7:0] field showing the binary point. x denotes a group priority field bit, and y denotes a subpriority field bit.

Determining preemption of an exception uses only the group priority field.

Atmel

```
        STM R0, {R1-R2}        ; Region base address, region number and VALID bit,
                               ; and Region Attribute, Size and Enable
```

12.11.1.5   Subregions

Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the SRD field of the MPU_RASR field to disable a subregion. See "MPU Region Attribute and Size Register" . The least significant bit of SRD controls the first subregion, and the most significant bit controls the last subregion. Disabling a subregion means another region overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion, the MPU issues a fault.

Regions of 32, 64, and 128 bytes do not support subregions. With regions of these sizes, the SRD field must be set to 0x00, otherwise the MPU behavior is unpredictable.

12.11.1.6   Example of SRD Use

Two regions with the same base address overlap. Region 1 is 128 KB, and region 2 is 512 KB. To ensure the attributes from region 1 apply to the first 128 KB region, set the SRD field for region 2 to b00000011 to disable the first two subregions, as in Figure 12-13 below:

Figure 12-13.   SRD Use
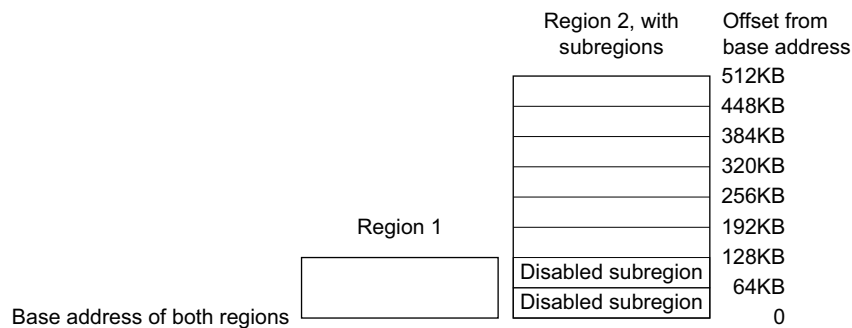
12.11.1.7   MPU Design Hints And Tips

To avoid unexpected behavior, disable the interrupts before updating the attributes of a region that the interrupt handlers might access.

Ensure the software uses aligned accesses of the correct size to access MPU registers:

- Except for the MPU_RASR, it must use aligned word accesses
- For the MPU_RASR, it can use byte or aligned halfword or word accesses.

The processor does not support unaligned accesses to MPU registers.

When setting up the MPU, and if the MPU has previously been programmed, disable unused regions to prevent any previous region settings from affecting the new MPU setup.

MPU Configuration for a Microcontroller

Usually, a microcontroller system has only a single processor and no caches. In such a system, program the MPU as follows:

Table 12-39.   Memory Region Attributes for a Microcontroller

Memory Region	TEX	C	B	S	Memory Type and Attributes
Flash memory	b000	1	0	0	Normal memory, non-shareable, write-through
Internal SRAM	b000	1	0	1	Normal memory, shareable, write-through
External SRAM	b000	1	1	1	Normal memory, shareable, write-back, write-allocate
Peripherals	b000	0	1	1	Device memory, shareable

3. To stop reading the unique identifier area, execute the 'Stop Read Unique Identifier' command by writing EEFC_FCR.FCMD with the SPUI command. Field EEFC_FCR.FARG is meaningless.

4. When the SPUI command has been executed, the bit EEFC_FSR.FRDY rises. If an interrupt was enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.

Note that during the sequence, the software cannot be fetched from the Flash.

20.4.3.9   User Signature Area

Each product contains a user signature area of 512-bytes. It can be used for storage. Read, write and erase of this area is allowed.

See Figure 20-1 "Flash Memory Areas".

The sequence to read the user signature area is the following:

1. Execute the 'Start Read User Signature' command by writing EEFC_FCR.FCMD with the STUS command. Field EEFC_FCR.FARG is meaningless.

2. Wait until the bit EEFC_FSR.FRDY falls to read the user signature area. The user signature area is located in the first 512 bytes of the Flash memory mapping. The 'Start Read User Signature' command reuses some addresses of the memory plane but the user signature area is physically different from the memory plane

3. To stop reading the user signature area, execute the 'Stop Read User Signature' command by writing EEFC_FCR.FCMD with the SPUS command. Field EEFC_FCR.FARG is meaningless.

4. When the SPUI command has been executed, the bit EEFC_FSR.FRDY rises. If an interrupt was enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.

Note that during the sequence, the software cannot be fetched from the Flash or from the second plane in case of dual plane.

One error can be detected in EEFC_FSR after this sequence:

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

The sequence to write the user signature area is the following:

1. Write the full page, at any page address, within the internal memory area address space.

2. Execute the 'Write User Signature' command by writing EEFC_FCR.FCMD with the WUS command. Field EEFC_FCR.FARG is meaningless.

3. When programming is completed, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the corresponding interrupt line of the interrupt controller is activated.

Two errors can be detected in EEFC_FSR after this sequence:

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

● Flash Error: At the end of the programming, the WriteVerify test of the Flash memory has failed.

The sequence to erase the user signature area is the following:

1. Execute the 'Erase User Signature' command by writing EEFC_FCR.FCMD with the EUS command. Field EEFC_FCR.FARG is meaningless.

2. When programming is completed, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the corresponding interrupt line of the interrupt controller is activated.

Two errors can be detected in EEFC_FSR after this sequence:

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

● Flash Error: At the end of the programming, the EraseVerify test of the Flash memory has failed.

23.7.10 CRCCU Mode Register

Name: CRCCU_MR

Address: 0x40044038

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
DIVIDER				PTYPE		COMPARE	ENABLE

- **ENABLE: CRC Enable**

Always write a 1 to this bit.

- **COMPARE: CRC Compare**

If set to one, this bit indicates that the CRCCU DMA will compare the CRC computed on the data stream with the value stored in the TR_CRC reference register. If a mismatch occurs, the ERRISR bit in the CRCCU_ISR is set.

- **PTYPE: Primitive Polynomial**

| Value	Name	Description
0	CCITT8023	Polynom 0x04C11DB7
1	CASTAGNOLI	Polynom 0x1EDC6F41
2	CCITT16	Polynom 0x1021

- **DIVIDER: Request Divider**

CRCCU DMA performs successive transfers. It is possible to reduce the bandwidth drained by the CRCCU DMA by programming the DIVIDER field. The transfer request frequency is divided by $2^{(DIVIDER+1)}$.
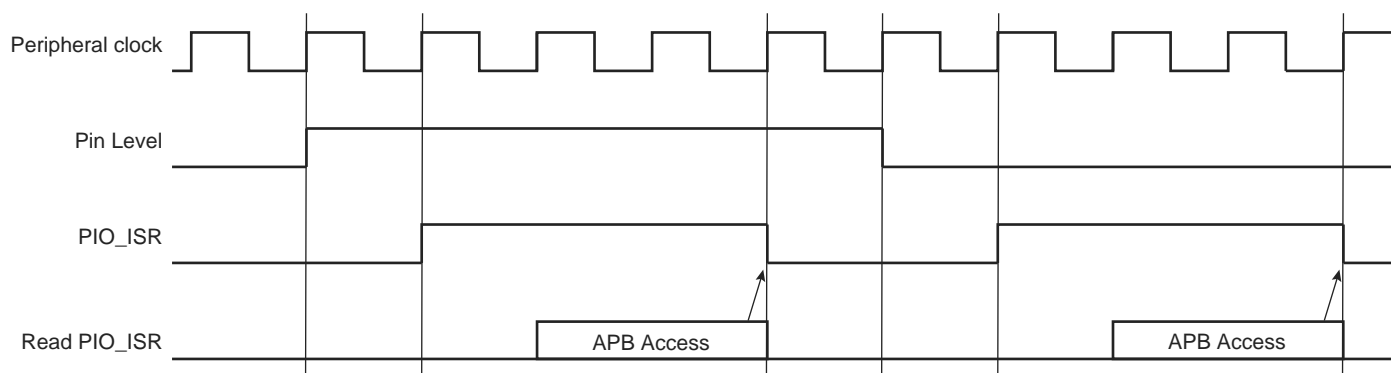
Atmel

29.15.2 Clock Switching Waveforms

Figure 29-6. Switch Master Clock from Slow Clock to PLLx Clock
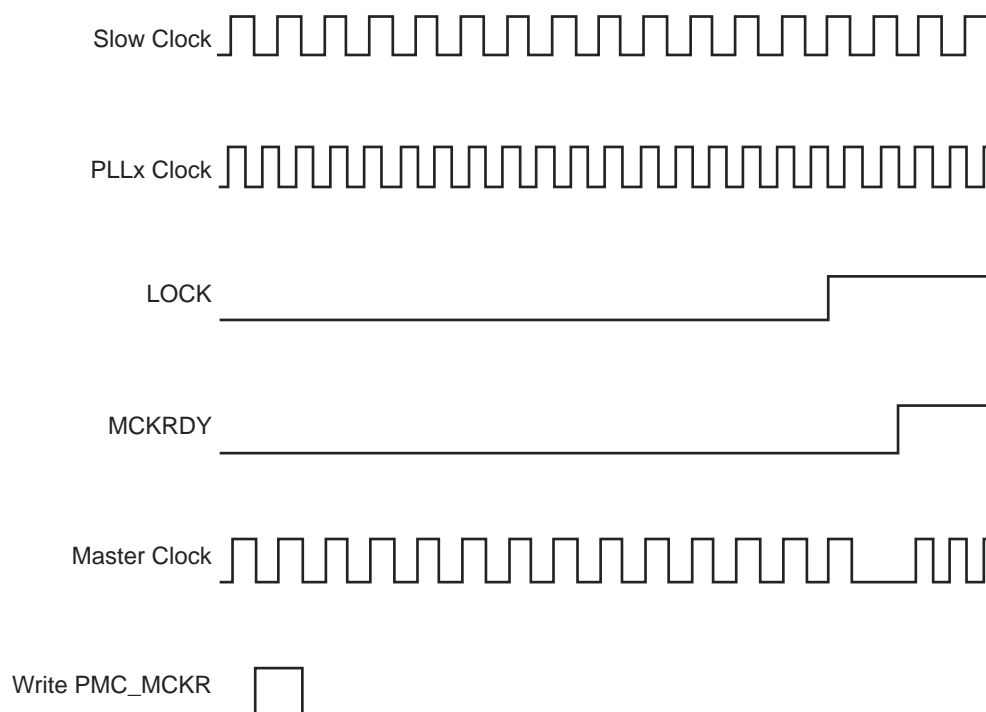
Figure 29-7. Switch Master Clock from Main Clock to Slow Clock

Atmel
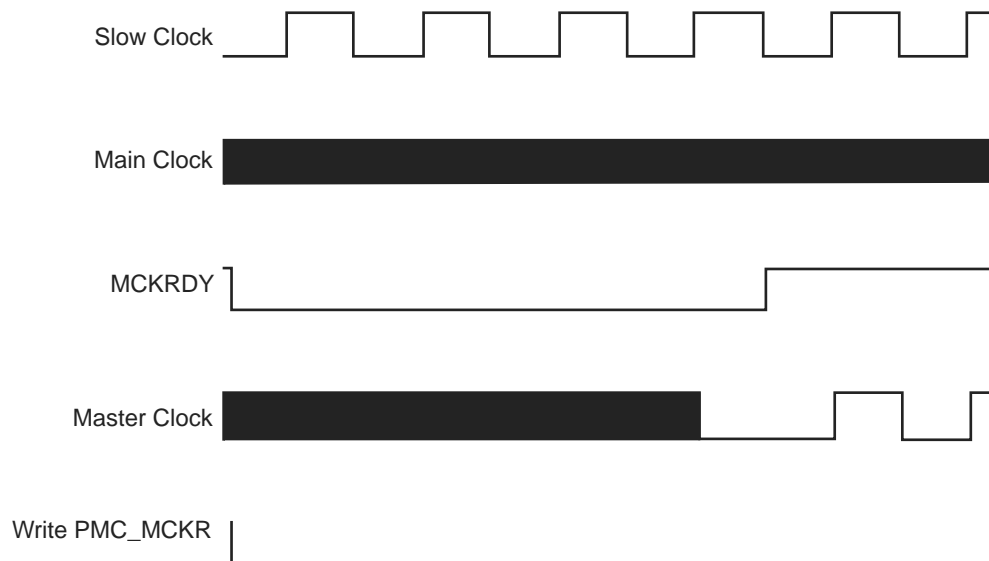
Figure 31-7. Input Change Interrupt Timings When No Additional Interrupt Modes

31.5.11 I/O Lines Lock

When an I/O line is controlled by a peripheral (particularly the Pulse Width Modulation Controller PWM), it can become locked by the action of this peripheral via an input of the PIO Controller. When an I/O line is locked, the write of the corresponding bit in PIO_PER, PIO_PDR, PIO_MDER, PIO_MDDR, PIO_PUDR, PIO_PUER, PIO_ABCDSR1 and PIO_ABCDSR2 is discarded in order to lock its configuration. The user can know at anytime which I/O line is locked by reading the PIO Lock Status Register (PIO_LOCKSR). Once an I/O line is locked, the only way to unlock it is to apply a hardware reset to the PIO Controller.

31.5.12 Programmable Schmitt Trigger

It is possible to configure each input for the Schmitt trigger. By default the Schmitt trigger is active. Disabling the Schmitt trigger is requested when using the QTouch® Library.

### 31.5.13 Parallel Capture Mode

31.5.13.1 Overview

The PIO Controller integrates an interface able to read data from a CMOS digital image sensor, a high-speed parallel ADC, a DSP synchronous port in synchronous mode, etc. For better understanding and to ease reading, the following description uses an example with a CMOS digital image sensor.

31.5.13.2 Functional Description

The CMOS digital image sensor provides a sensor clock, an 8-bit data synchronous with the sensor clock and two data enables which are also synchronous with the sensor clock.

Figure 31-8. PIO Controller Connection with CMOS Digital Image Sensor

Atmel
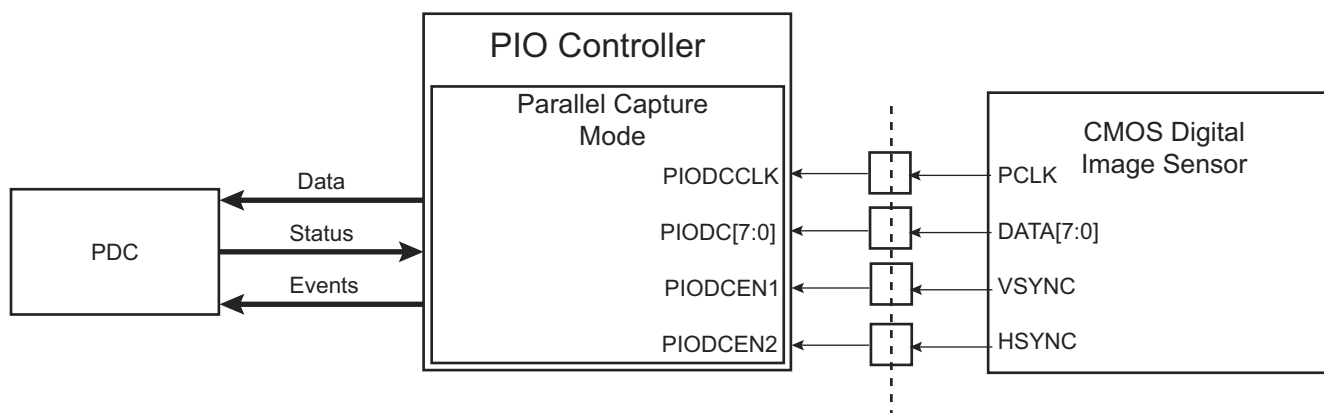
31.6.11 PIO Clear Output Data Register

Name: PIO_CODR

Address: 0x400E0E34 (PIOA), 0x400E1034 (PIOB), 0x400E1234 (PIOC)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Clear Output Data**

0: No effect.

1: Clears the data to be driven on the I/O line.

Atmel

31.6.46 PIO Write Protection Mode Register

Name: PIO_WPMR

Address: 0x400E0EE4 (PIOA), 0x400E10E4 (PIOB), 0x400E12E4 (PIOC)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x50494F ("PIO" in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x50494F ("PIO" in ASCII).

See Section 31.5.15 "Register Write Protection" for the list of registers that can be protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x50494F	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Atmel

31.6.48 PIO Schmitt Trigger Register

Name: PIO_SCHMITT

Address: 0x400E0F00 (PIOA), 0x400E1100 (PIOB), 0x400E1300 (PIOC)

Access: Read/Write

31	30	29	28	27	26	25	24
SCHMITT31	SCHMITT30	SCHMITT29	SCHMITT28	SCHMITT27	SCHMITT26	SCHMITT25	SCHMITT24

23	22	21	20	19	18	17	16
SCHMITT23	SCHMITT22	SCHMITT21	SCHMITT20	SCHMITT19	SCHMITT18	SCHMITT17	SCHMITT16

15	14	13	12	11	10	9	8
SCHMITT15	SCHMITT14	SCHMITT13	SCHMITT12	SCHMITT11	SCHMITT10	SCHMITT9	SCHMITT8

7	6	5	4	3	2	1	0
SCHMITT7	SCHMITT6	SCHMITT5	SCHMITT4	SCHMITT3	SCHMITT2	SCHMITT1	SCHMITT0

• **SCHMITTx [x=0..31]: Schmitt Trigger Control**

0: Schmitt trigger is enabled.

1: Schmitt trigger is disabled.

- **RXBUFF: Receive Buffer Full Interrupt Enable**

0: No effect.

1: Enables the Receive Buffer Full Interrupt.

- **CP0: Compare 0 Interrupt Enable**

0: No effect.

1: Enables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Enable**

0: No effect.

1: Enables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0: No effect.

1: Enables the Tx Sync Interrupt.

- **RXSYN: Rx Sync Interrupt Enable**

0: No effect.

1: Enables the Rx Sync Interrupt.

Atmel

Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 34-29 describes the repeated start + reversal from Read to Write mode.

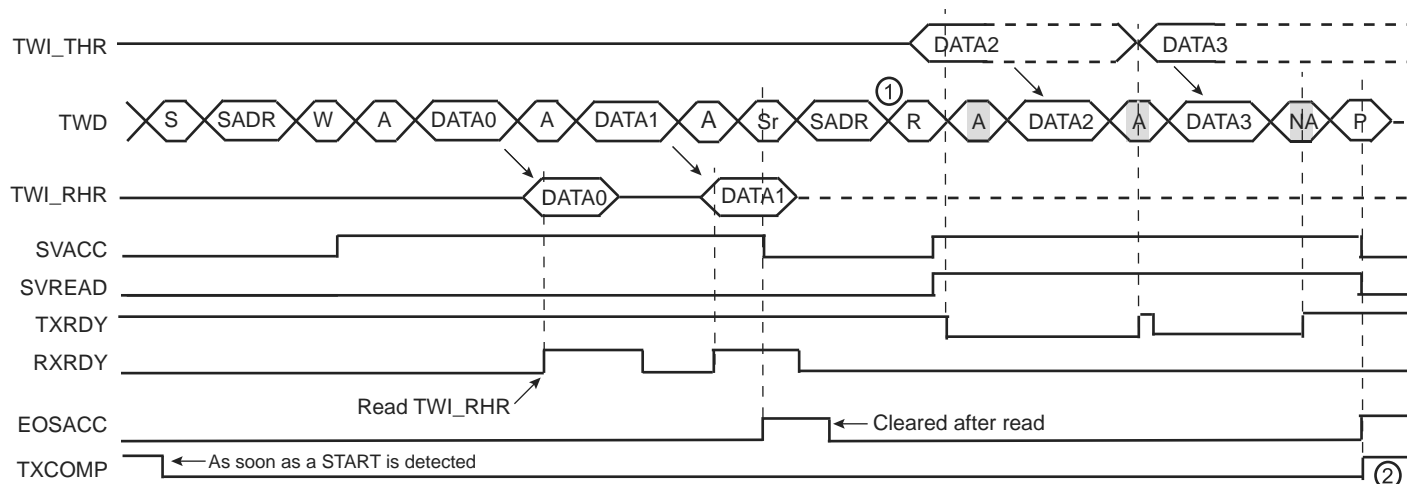**Figure 34-29. Repeated Start + Reversal from Read to Write Mode**
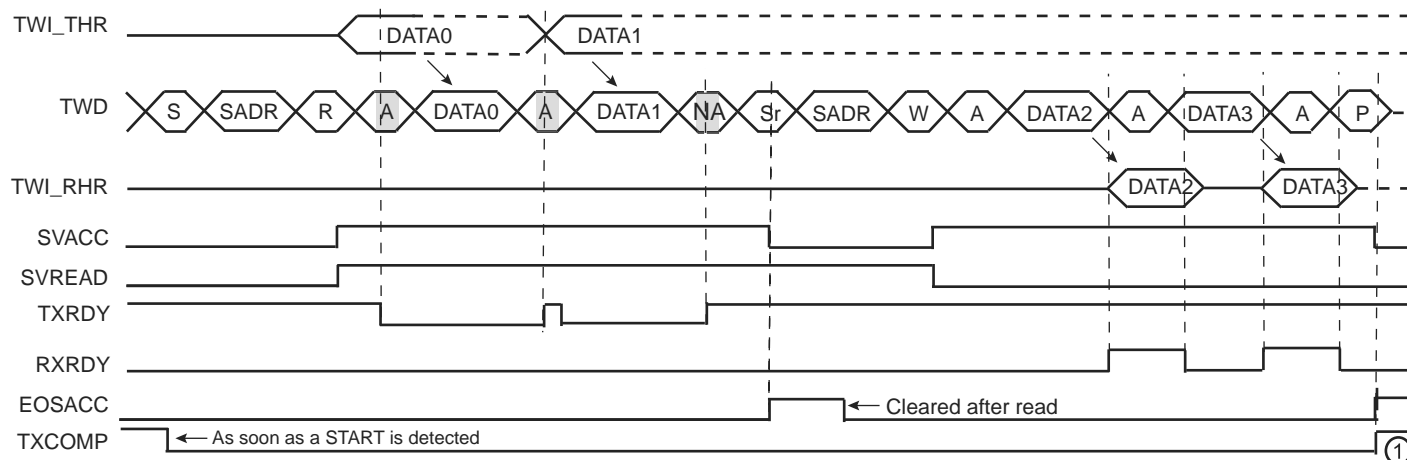
Note: 1. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command.

Figure 34-30 describes the repeated start + reversal from Write to Read mode.

Figure 34-30. Repeated Start + Reversal from Write to Read Mode

Notes: 1. In this case, if TWI_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.
   2. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

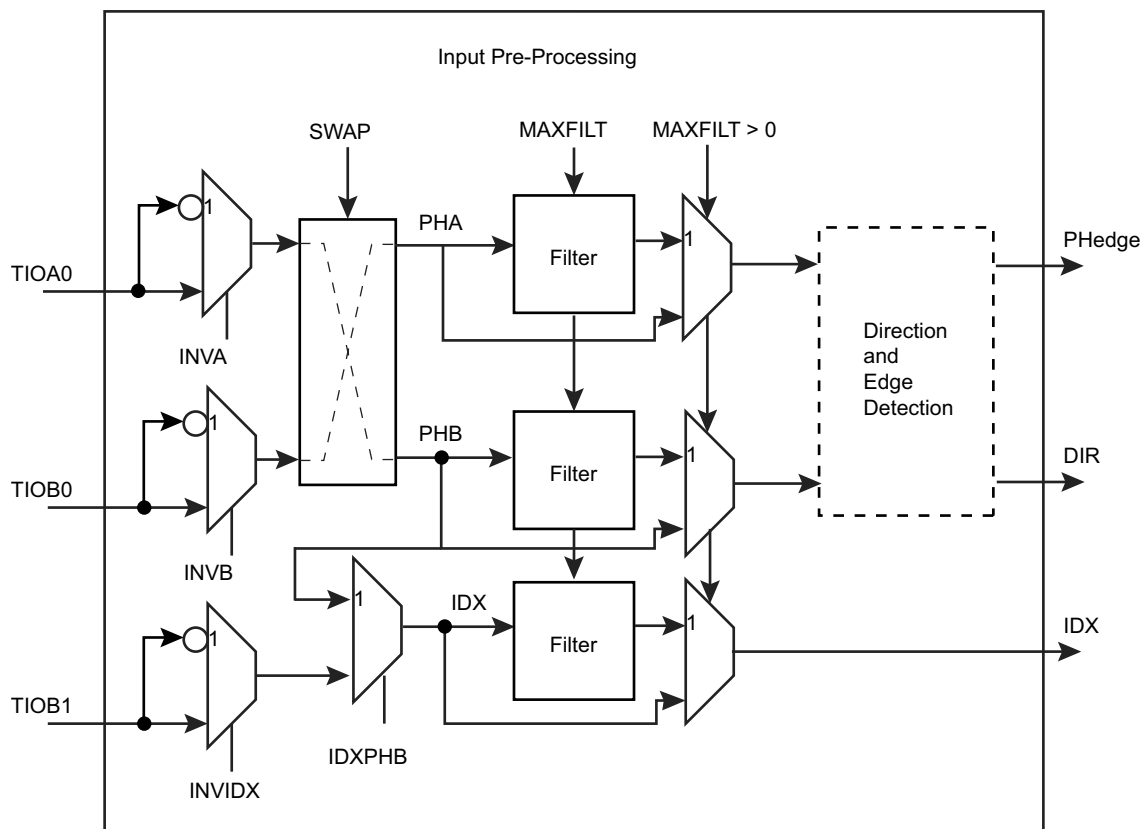**34.7.5.6   Using the Peripheral DMA Controller (PDC) in Slave Mode**

The use of the PDC significantly reduces the CPU load.

Data Transmit with the PDC in Slave Mode

The following procedure shows an example of data transmission with PDC.

- Offers Buffer Transfer without Processor Intervention
- Register Write Protection

Figure 37-16. Input Stage

Input filtering can efficiently remove spurious pulses that might be generated by the presence of particulate contamination on the optical or magnetic disk of the rotary sensor.

Spurious pulses can also occur in environments with high levels of electro-magnetic interference. Or, simply if vibration occurs even when rotation is fully stopped and the shaft of the motor is in such a position that the beginning of one of the reflective or magnetic bars on the rotary sensor disk is aligned with the light or magnetic (Hall) receiver cell of the rotary sensor. Any vibration can make the PHA, PHB signals toggle for a short duration.

Atmel

Figure 38-10. Multiple Write Functional Flow Diagram
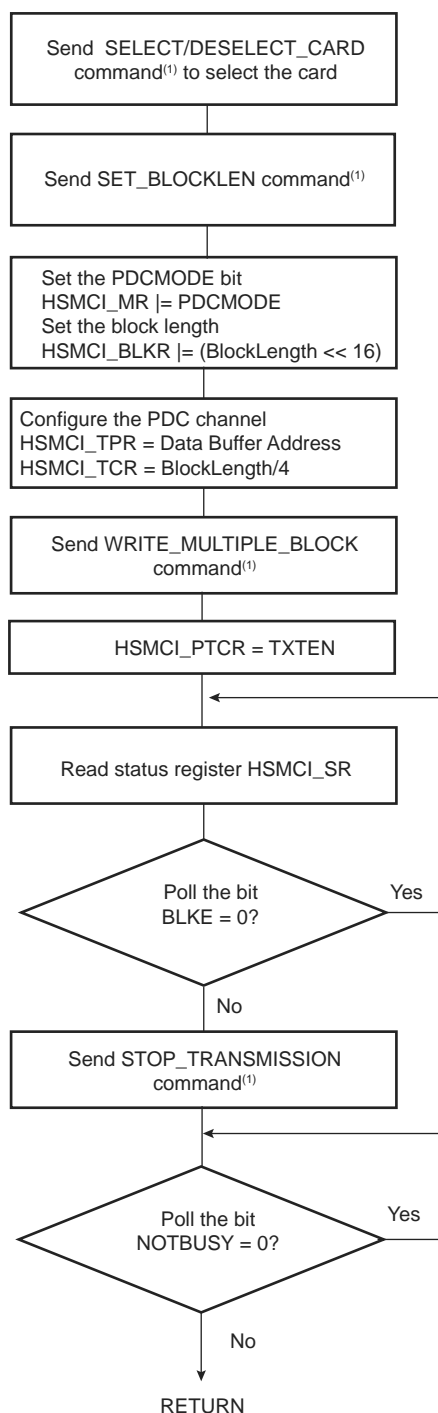
Note: 1.   It is assumed that this command has been correctly sent (see Figure 38-7).

38.9   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

Atmel

42.7.14 ADC Extended Mode Register

Name: ADC_EMR

Address: 0x40038040

Access: Read/Write

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

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

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

7	6	5	4	3	2	1	0
CMPSEL				–	–	CMPMODE	

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

- **CMPMODE: Comparison Mode**

Value	Name	Description
0	LOW	Generates an event when the converted data is lower than the low threshold of the window.
1	HIGH	Generates an event when the converted data is higher than the high threshold of the window.
2	IN	Generates an event when the converted data is in the comparison window.
3	OUT	Generates an event when the converted data is out of the comparison window.

- **CMPSEL: Comparison Selected Channel**

If CMPALL = 0: CMPSEL indicates which channel has to be compared.

If CMPALL = 1: No effect.

- **CMPALL: Compare All Channels**

0: Only channel indicated in CMPSEL field is compared.

1: All channels are compared.

- **TAG: Tag of the ADC_LCDR**

0: Sets CHNB field to zero in ADC_LCDR.

1: Appends the channel number to the conversion result in ADC_LCDR.

48.3.3 Brownout Detector

Issue: **Unpredictable Behavior if BOD is Disabled, VDDCORE is Lost and VDDIO is Connected**

In active mode or in wait mode, if the Brownout Detector is disabled (SUPC_MR.BODDIS = 1) and power is lost on VDDCORE while VDDIO is powered, the device might not be properly reset and may behave unpredictably.

Workaround: When the Brownout Detector is disabled in active or in wait mode, VDDCORE always needs to be powered.

48.3.4 Low-power Mode

Issue: **Unpredictable Behavior When Entering Sleep Mode**

When entering Sleep mode, if an interrupt occurs during WFI or WFE (PMC_FSMR.LPM = 0) instruction processing, the ARM core may read an incorrect data, thus leading to unpredictable behavior of the software. This issue is not present in Wait mode.

Workaround: The following conditions must be met:
1. The interrupt vector table must be located in Flash.
2. The Matrix slave interface for the Flash must be set to 'No default master'. This is done by setting the field DEFMSTR_TYPE to 0 in the register MATRIX_SCFG. The code example below can be used to program the NO_DEFAULT_MASTER state:

```
MATRIX_SCFG[2] = MATRIX_SCFG_SLOT_CYCLE(0xFF) | MATRIX_SCFG_DEFMSTR_TYPE(0x0);
```

This must be done once in the software before entering Sleep mode.

48.3.5 PIO

Issue: **PB4 Input Low-level Voltage Range**

The undershoot is limited to -0.1V.

In normal operating conditions, the V_{IL} minimum value on PB4 is limited to 0V.

Workaround: The voltage on PB4 with respect to ground must be in the range
-0.1V to + VDDIO + 0.4V instead of -0.3V to + VDDIO + 0.4V for all other input pins, as shown in Table 44.1 "Absolute Maximum Ratings".

The minimum V_{IL} on PB4 must be 0V instead of -0.3V for all other input pins, as shown in Table 44.3 "DC Characteristics".

Atmel