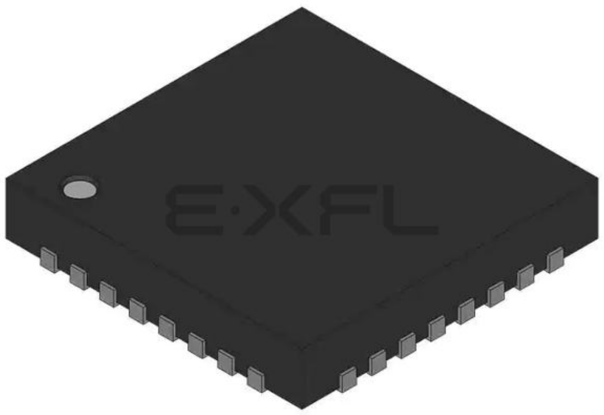


Welcome to E-XFL.COM

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	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	CANbus, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT
Number of I/O	-
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 11x10b; D/A 1x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VQFN Exposed Pad
Supplier Device Package	32-QFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/atmel/atmega16m1-15mz

6.2 Idle Mode

When the SM2..0 bits are written to 000, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing SPI, UART, analog comparator, ADC, Timer/Counters, watchdog, and the interrupt system to continue operating. This sleep mode basically halt clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the timer overflow and UART transmit complete interrupts. If wake-up from the analog comparator interrupt is not required, the analog comparator can be powered down by setting the ACD bit in the analog comparator control and status register – ACSR. This will reduce power consumption in Idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

6.3 ADC noise reduction Mode

When the SM2..0 bits are written to 001, the SLEEP instruction makes the MCU enter ADC noise reduction mode, stopping the CPU but allowing the ADC, the External Interrupts, Timer/Counter (if their clock source is external - T0 or T1) and the watchdog to continue operating (if enabled). This sleep mode basically halts $clk_{I/O}$, clk_{CPU}, and clk_{FLASH}, while allowing the other clocks to run.

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart from the ADC conversion complete interrupt, only an external reset, a watchdog reset, a brown-out reset, a Timer/Counter interrupt, an SPM/EEPROM ready interrupt, an external level interrupt on INT3:0 can wake up the MCU from ADC noise reduction mode.

6.4 Power-down Mode

When the SM2..0 bits are written to 010, the SLEEP instruction makes the MCU enter power-down mode. In this mode, the external oscillator is stopped, while the external interrupts and the watchdog continue operating (if enabled). Only an external reset, a watchdog reset, a brown-out reset, a PSC interrupt, an external level interrupt on INT3:0 can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from power-down mode, the changed level must be held for some time to wake up the MCU. Refer to Section 10. “External Interrupts” on page 70 for details.

When waking up from power-down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL fuses that define the reset time-out period, as described in Section 5.2 “Clock Sources” on page 26.

6.5 Standby Mode

When the SM2..0 bits are 110 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

Table 6-2. Active Clock Domains and Wake-up Sources in the Different Sleep Modes

	Active Clock Domains					Oscillators	Wake-up Sources					
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{PLL}	Main Clock Source Enabled	INT3..0	PSC	SPM/EEPROM Ready	ADC	WDT	OtherI/O
Idle			X	X	X	X	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X[2]	X	X	X	X	
Power-down							X[2]				X	
Standby[1]						X	X[2]				X	

Notes: 1. Only recommended with external crystal or resonator selected as clock source.

2. Only level interrupt.

The following code example shows one assembly and one C function for turning off the Watchdog Timer. The example assumes that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during the execution of these functions.

Assembly Code Example(1)

```
WDT_off:
      ; Turn off global interrupt
      cli
      ; Reset Watchdog Timer
      wdr
      ; Clear WDRF in MCUSR
      in    r16, MCUSR
      andi r16, (0xff & (0<<WDRF))
      out   MCUSR, r16
      ; Write logical one to WDCE and WDE
      ; Keep old prescaler setting to prevent unintentional time-out
      lds r16, WDTCSR
      ori   r16, (1<<WDCE) | (1<<WDE)
      sts WDTCSR, r16
      ; Turn off WDT
      ldi   r16, (0<<WDE)
      sts WDTCSR, r16
      ; Turn on global interrupt
      sei
      ret
```

C Code Example(1)

```
void WDT_off(void)
{
      __disable_interrupt();
      __watchdog_reset();
      /* Clear WDRF in MCUSR */
      MCUSR &= ~(1<<WDRF);
      /* Write logical one to WDCE and WDE */
      /* Keep old prescaler setting to prevent unintentional time-out */
      WDTCSR |= (1<<WDCE) | (1<<WDE);
      /* Turn off WDT */
      WDTCSR = 0x00;
      __enable_interrupt();
}
```

Notes: 1. The example code assumes that the part specific header file is included.

2. If the watchdog is accidentally enabled, for example by a runaway pointer or brown-out condition, the device will be reset and the watchdog timer will stay enabled. If the code is not set up to handle the watchdog, this might lead to an eternal loop of time-out resets. To avoid this situation, the application software should always clear the watchdog system reset flag (WDRF) and the WDE control bit in the initialization routine, even if the watchdog is not in use.

9.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 9-2, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU sleep controller in power-down mode, power-save mode, and standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in Section 9.3 “Alternate Port Functions” on page 55.

If a logic high level (“one”) is present on an asynchronous external interrupt pin configured as “Interrupt on Rising Edge, Falling Edge, or Any Logic Change on Pin” while the external interrupt is not enabled, the corresponding external interrupt flag will be set when resuming from the above mentioned sleep modes, as the clamping in these sleep modes produces the requested logic change.

9.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 9-5 shows how the port pin control signals from the simplified Figure 9-2 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR® microcontroller family.

9.3.3 Alternate Functions of Port C

The Port C pins with alternate functions are shown in Table 9-6.

Table 9-6. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC7	D2A (DAC output) AMP2+ (Analog Differential Amplifier 2 Positive Input) PCINT15 (Pin Change Interrupt 15)
PC6	ADC10 (Analog Input Channel 10) ACMP1 (analog comparator 1 Positive Input) PCINT14 (Pin Change Interrupt 14)
PC5	ADC9 (Analog Input Channel 9) AMP1+ (Analog Differential Amplifier 1 Input Channel) ACMP3 (Analog Comparator 3 Positive Input) PCINT13 (Pin Change Interrupt 13)
PC4	ADC8 (Analog Input Channel 8) AMP1- (Analog Differential Amplifier 1 Input Channel) ACMPN3 (Analog Comparator 3 Negative Input) PCINT12 (Pin Change Interrupt 12)
PC3	T1 (Timer 1 clock input) RXCAN (CAN Rx Data) ICP1B (Timer 1 Input Capture Alternate Input) PCINT11 (Pin Change Interrupt 11)
PC2	T0 (Timer 0 clock input) TXCAN (CAN Tx Data) PCINT10 (Pin Change Interrupt 10)
PC1	PSCIN1 (PSC 1 Digital Input) OC1B (Timer 1 Output Compare B) SS_A (Alternate SPI Slave Select) PCINT9 (Pin Change Interrupt 9)
PC0	PSCOUT1A (PSC output 2A) INT3 (External Interrupt 3) PCINT8 (Pin Change Interrupt 8)

Note: On the engineering samples (Parts marked AT90PWM324), the ACMPN3 alternate function is not located on PC4. It is located on PE2.

The alternate pin configuration is as follows:

- **D2A/AMP2+/PCINT15 – Bit 7**

D2A, digital to analog output

AMP2+, analog differential amplifier 2 positive input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the amplifier.

PCINT15, pin change interrupt 15.

10. External Interrupts

The external interrupts are triggered by the INT3:0 pins or any of the PCINT23..0 pins. Observe that, if enabled, the interrupts will trigger even if the INT3:0 or PCINT23..0 pins are configured as outputs. This feature provides a way of generating a software interrupt. The pin change interrupt PCI2 will trigger if any enabled PCINT23..16 pin toggles. The pin change interrupt PCI1 will trigger if any enabled PCINT14..8 pin toggles. The pin change interrupt PCI0 will trigger if any enabled PCINT7..0 pin toggles. The PCMSK3, PCMSK2, PCMSK1 and PCMSK0 registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT26..0 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

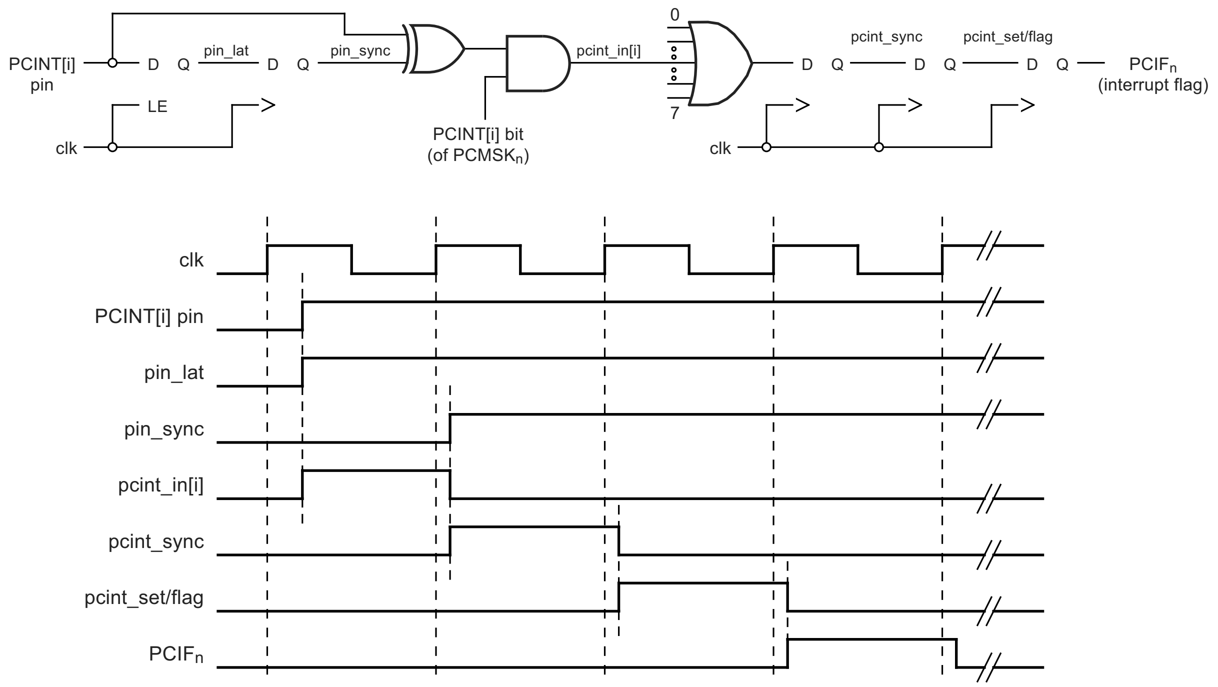
The INT3:0 interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the external interrupt control register A – EICRA. When the INT3:0 interrupts are enabled and are configured as level triggered, the interrupts will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT3:0 requires the presence of an I/O clock, described in Section 5.1 “Clock Systems and their Distribution” on page 25. Low level interrupt on INT3:0 is detected asynchronously. This implies that this interrupt can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note that if a level triggered interrupt is used for wake-up from power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL Fuses as described in Section 5.1 “Clock Systems and their Distribution” on page 25.

10.1 Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in Figure 10-1

Figure 10-1. Timing of a Pin Change Interrupts



On-time 0 = 2 × POCRnSAH/L × 1/Fclkpsc

On-time 1 = 2 × (POCRnRBH/L – POCRnSBH/L + 1) × 1/Fclkpsc

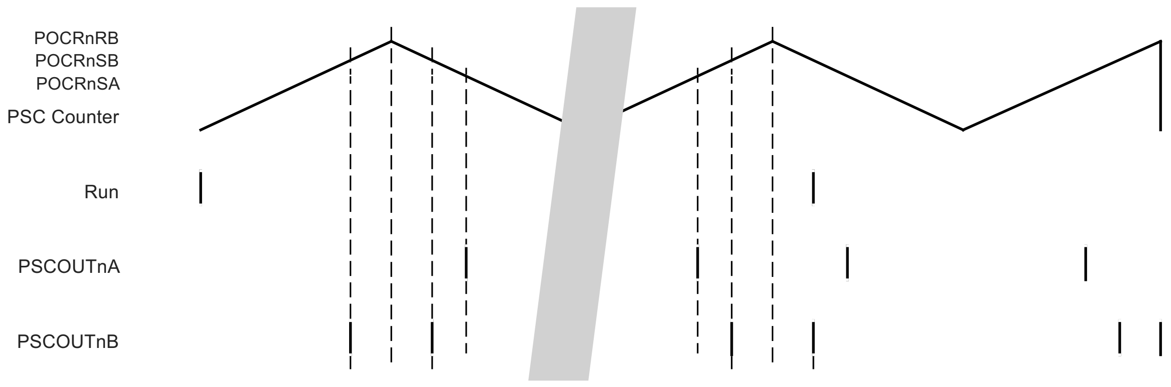
Dead-time = (POCRnSBH/L – POCRnSAH/L) × 1/Fclkpsc

PSC cycle = 2 × (POCRnRBH/L + 1) × 1/Fclkpsc

Minimal value for PSC cycle = 2 × 1/Fclkpsc

Note that in center aligned mode, POCRnRAH/L is not required (as it is in one-ramp mode) to control PSC Output waveform timing. This allows POCRnRAH/L to be freely used to adjust ADC synchronization (See Section 14.12 “Analog Synchronization” on page 126).

Figure 14-7. Controlled Start and Stop Mechanism in Centered Mode

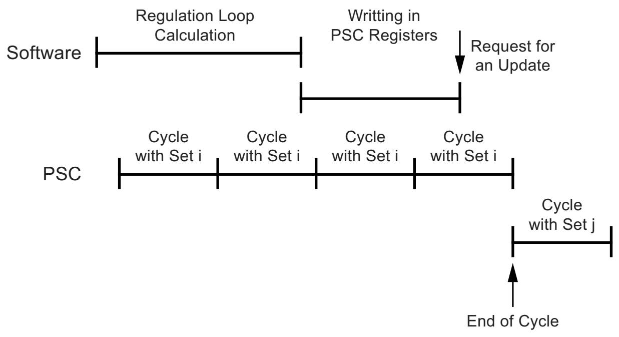


Note: See Section 14.16.8 “PSC Control Register – PCTL” on page 130 (PCCYC = 1)

14.6 Update of Values

To avoid unasynchronous and incoherent values in a cycle, if an update of one of several values is necessary, all values are updated at the same time at the end of the cycle by the PSC. The new set of values is calculated by software and the update is initiated by software.

Figure 14-8. Update at the End of Complete PSC Cycle



The software can stop the cycle before the end to update the values and restart a new PSC cycle.

14.9.1.2 Signal Polarity

One can select the active edge (edge modes) or the active level (level modes). See PELEVnx bit description in Section 14.16.9 “PSC Module n Input Control Register – PMICn” on page 131.

If PELEVnx bit set, the significant edge of PSCn Input A or B is rising (edge modes) or the active level is high (level modes) and vice versa for unset/falling/low

- In 2- or 4-ramp mode, PSCn Input A is taken into account only during Dead-Time0 and On-Time0 period (respectively Dead-Time1 and On-Time1 for PSCn input B).
- In 1-ramp-mode PSC Input A or PSC Input B act on the whole ramp.

14.9.1.3 Input Mode Operation

Thanks to 4 configuration bits (PRFM3:0), it’s possible to define the mode of the PSC inputs.

Table 14-5. PSC Input Mode Operation

PRFMn2:0	Description
000b	No action, PSC input is ignored
001b	Disactivate module n outputs A
010b	Disactivate module n output B
011b	Disactivate module n output A and B
10x	Disactivate all PSC output
11xb	Halt PSC and wait for software action

Note: All following examples are given with rising edge or high level active inputs.

14.10 PSC Input Modes 001b to 10xb: Deactivate Outputs without Changing Timing

Figure 14-12. PSC Behavior versus PSCn Input in Mode 001b to 10xb

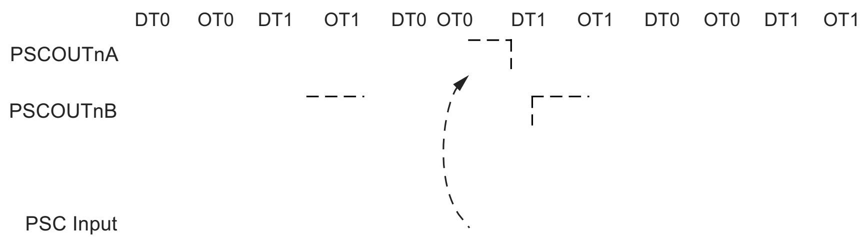
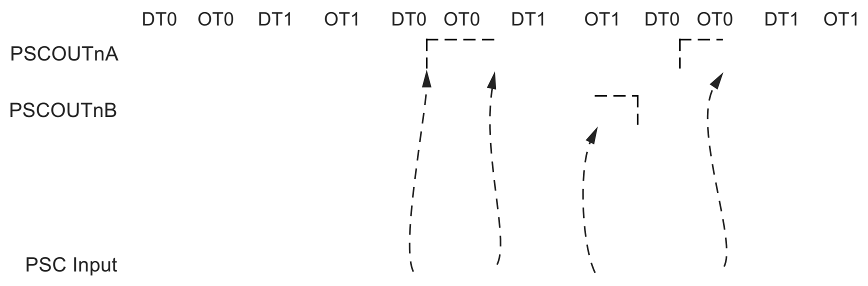


Figure 14-13. PSC Behavior versus PSCn Input A or Input B in Fault Mode 4



PSCn Input acts indifferently on On-Time0/Dead-Time0 or on On-Time1/Dead-Time1.

Table 14-9. Synchronization Source Description in Centered Mode

PSYNCn1	PSYNCn0	Description
0	0	Send signal on match with OCRnRA (during counting down of PSC). The min value of OCRnRA must be 1.
0	1	Send signal on match with OCRnRA (during counting up of PSC). The min value of OCRnRA must be 1.
1	0	no synchronization signal
1	1	no synchronization signal

14.16.3 PSC Output Compare SA Register – POCRnSAH and POCRnSAL

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	POCRnSA[11:8]				POCRnSAH
	POCRnSA[7:0]								POCRnSAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.16.4 PSC Output Compare RA Register – POCRnRAH and POCRnRAL

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	POCRnRA[11:8]				POCRnRAH
	POCRnRA[7:0]								POCRnRAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.16.5 PSCOutput Compare SB Register – POCRnSBH and POCRnSBL

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	POCRnSB[11:8]				POCRnSBH
	POCRnSB[7:0]								OCRnSBL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.16.6 PSC Output Compare RB Register – POCR_RBH and POCR_RBL

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	POCRnRB[11:8]				POCR_RBH
	POCRnRB[7:0]								POCR_RBL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Note: n = 0 to 2 according to module number.

The output compare registers RA, RB, SA and SB contain a 12-bit value that is continuously compared with the PSC counter value. A match can be used to generate an output compare interrupt, or to generate a waveform output on the associated pin.

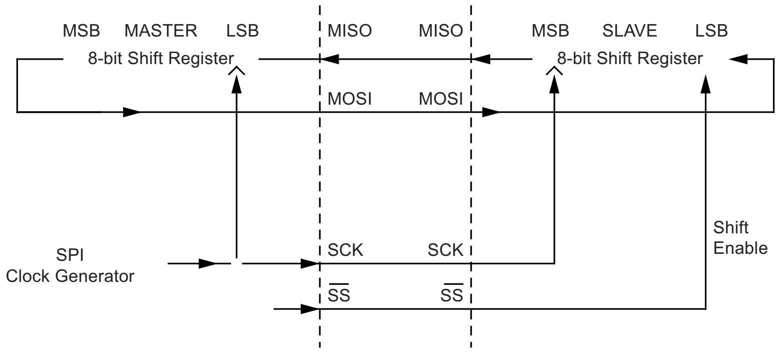
The output compare registers are 16bit and 12-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers.

The interconnection between master and slave CPUs with SPI is shown in Figure 15-2. The system consists of two shift registers, and a master clock generator. The SPI master initiates the communication cycle when pulling low the slave select $\overline{SS}$ pin of the desired slave. Master and slave prepare the data to be sent in their respective shift registers, and the master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from master to slave on the master out – slave in, MOSI, line, and from slave to master on the master in – slave out, MISO, line. After each data packet, the master will synchronize the slave by pulling high the slave select, $\overline{SS}$, line.

When configured as a master, the SPI interface has no automatic control of the $\overline{SS}$ line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI data register starts the SPI clock generator, and the hardware shifts the eight bits into the slave. After shifting one byte, the SPI clock generator stops, setting the end of transmission flag (SPIF). If the SPI interrupt enable bit (SPIE) in the SPCR register is set, an interrupt is requested. The master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the slave select, $\overline{SS}$ line. The last incoming byte will be kept in the buffer register for later use.

When configured as a slave, the SPI interface will remain sleeping with MISO tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the SPI data register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. As one byte has been completely shifted, the end of transmission flag, SPIF is set. If the SPI interrupt enable bit, SPIE, in the SPCR register is set, an interrupt is requested. The slave may continue to place new data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the buffer register for later use.

Figure 15-2. SPI Master-slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI data register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI data register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the frequency of the SPI clock should never exceed $f_{clkio}/4$.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 15-1. For more details on automatic port overrides, refer to Section 9.3 “Alternate Port Functions” on page 55.

Table 15-1. SPI Pin Overrides(1)

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User defined	Input
MISO	Input	User defined
SCK	User defined	Input
$\overline{SS}$	User defined	Input

Note: 1. See Section 9.3.2 “Alternate Functions of Port B” on page 58 for a detailed description of how to define the direction of the user defined SPI pins.

16.4 CAN Channel

16.4.1 Configuration

The CAN channel can be in:

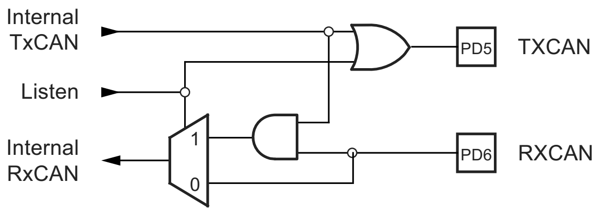
- Enabled mode

  In this mode:
  - the CAN channel (internal TxCAN and RxCAN) is enabled,
  - the input clock is enabled.
- Standby mode

  In standby mode:
  - the transmitter constantly provides a recessive level (on internal TxCAN) and the receiver is disabled,
  - input clock is enabled,
  - the registers and pages remain accessible.
- Listening mode

  This mode is transparent for the CAN channel:
  - enables a hardware loop back, internal TxCAN on internal RxCAN
  - provides a recessive level on TXCAN output pin
  - does not disable RXCAN input pin
  - freezes TEC and REC error counters

Figure 16-6. Listening Mode



16.4.2 Bit Timing

FSM's (finite state machine) of the CAN channel need to be synchronous to the time quantum. So, the input clock for bit timing is the clock used into CAN channel FSM's.

Field and segment abbreviations:

- BRP: Baud rate prescaler.
- TQ: Time quantum (output of baud rate prescaler).
- SYNS: Synchronization segment is 1 TQ long.
- PRS: Propagation time segment is programmable to be 1, 2, ..., 8 TQ long.
- PHS1: Phase segment 1 is programmable to be 1, 2, ..., 8 TQ long.
- PHS2: Phase segment 2 is programmable to be ≤ PHS1 and ≥ INFORMATION PROCESSING TIME.
- INFORMATION PROCESSING TIME is 2 TQ.
- SJW: (Re) Synchronization jump width is programmable between 1 and min(4, PHS1).

The total number of TQ in a bit time has to be programmed at least from 8 to 25.

16.5.4 MOb Page

Every MOb is mapped into a page to save place. The page number is the MOb number. This page number is set in CANPAGE register. The other numbers are reserved for factory tests.

CANHPMOB register gives the MOb having the highest priority in CANSIT registers. It is formatted to provide a direct entry for CANPAGE register. Because CANHPMOB codes CANSIT registers, it will be only updated if the corresponding enable bits (ENRX, ENTX, ENERR) are enabled (c.f. Figure 16-14 on page 155).

16.5.5 CAN Data Buffers

To preserve register allocation, the CAN data buffer is seen such as a FIFO (with address pointer accessible) into a MOb selection.This also allows to reduce the risks of un-controlled accesses.

There is one FIFO per MOb. This FIFO is accessed into a MOb page thanks to the CAN message register.

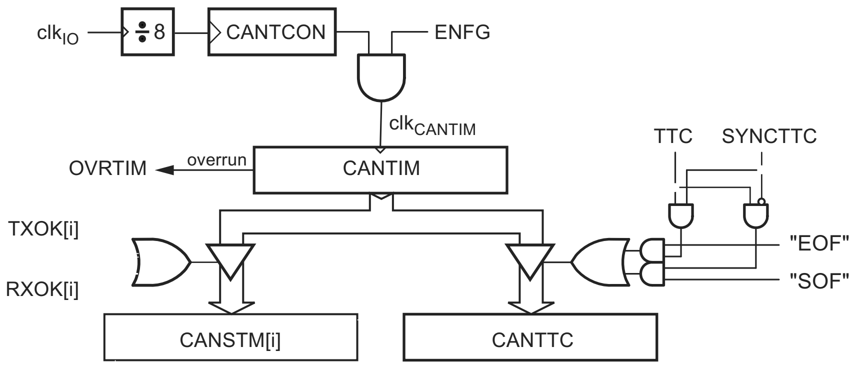
The data index (INDX) is the address pointer to the required data byte. The data byte can be read or write. The data index is automatically incremented after every access if the AINC* bit is reset. A roll-over is implemented, after data index=7 it is data index=0.

The first byte of a CAN frame is stored at the data index=0, the second one at the data index=1, ...

16.6 CAN Timer

A programmable 16-bit timer is used for message stamping and time trigger communication (TTC).

Figure 16-11. CAN Timer Block Diagram



16.6.1 Prescaler

An 8-bit prescaler is initialized by CANTCON register. It receives the clk_{IO} frequency divided by 8. It provides clk_{CANTIM} frequency to the CAN timer if the CAN controller is enabled.

$T\text{clk}_{CANTIM} = T\text{clk}_{IO} \times 8 \times (\text{CANTCON}[7:0] + 1)$

16.6.2 16-bit Timer

This timer starts counting from 0x0000 when the CAN controller is enabled (ENFG bit). When the timer rolls over from 0xFFFF to 0x0000, an interrupt is generated (OVRTIM).

16.6.3 Time Triggering

Two synchronization modes are implemented for TTC (TTC bit):

- synchronization on start of frame (SYNCTTC=0),
- synchronization on end of frame (SYNCTTC=1).

In TTC mode, **a frame is sent once, even if an error occurs**.

16.11 MOb Registers

The MOb registers has **no** initial (default) value after RESET.

16.11.1 CAN MOb Status Register - CANSTMOB

Bit	7	6	5	4	3	2	1	0	
	DLCW	**TXOK**	**RXOK**	**BERR**	**SERR**	**CERR**	**FERR**	**AERR**	**CANSTMOB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	-	-	-	-	-	-	-	

- **Bit 7 – DLCW: Data Length Code Warning**

The incoming message does not have the DLC expected. Whatever the frame type, the DLC field of the CANCDMOB register is updated by the received DLC.

- **Bit 6 – TXOK: Transmit OK**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The communication enabled by transmission is completed. TxOK rises at the end of EOF field. When the controller is ready to send a frame, if two or more message objects are enabled as producers, the lower MOb index (0 to 14) is supplied first.

- **Bit 5 – RXOK: Receive OK**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The communication enabled by reception is completed. RxOK rises at the end of the 6th bit of EOF field. In case of two or more message object reception hits, the lower MOb index (0 to 14) is updated first.

- **Bit 4 – BERR: Bit Error (Only in Transmission)**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The bit value monitored is different from the bit value sent.

Exceptions: the monitored recessive bit sent as a dominant bit during the arbitration field and the acknowledge slot detecting a dominant bit during the sending of an error frame.

- **Bit 3 – SERR: Stuff Error**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

Detection of more than five consecutive bits with the same polarity. This flag can generate an interrupt.

- **Bit 2 – CERR: CRC Error**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The receiver performs a CRC check on every de-stuffed received message from the start of frame up to the data field. If this checking does not match with the de-stuffed CRC field, a CRC error is set.

- **Bit 1 – FERR: Form Error**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The form error results from one or more violations of the fixed form in the following bit fields:

- CRC delimiter.
- Acknowledgment delimiter.
- EOF

17.6.7 LIN Data Length Register - LINDLR

Bit	7	6	5	4	3	2	1	0	
	LTXDL3	**LTXDL2**	**LTXDL1**	**LTXDL0**	**LRXDL3**	**LRXDL2**	**LRXDL1**	**LRXDL0**	**LINDLR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:4 - LTXDL[3:0]: LIN Transmit Data Length**
  In LIN mode, this field gives the number of bytes to be transmitted (clamped to 8 Max).

  In UART mode this field is unused.

- **Bits 3:0 - LRXDL[3:0]: LIN Receive Data Length**
  In LIN mode, this field gives the number of bytes to be received (clamped to 8 Max).

  In UART mode this field is unused.

17.6.8 LIN Identifier Register - LINIDR

Bit	7	6	5	4	3	2	1	0	
	LP1	**LP0**	**LID5 / LDL1**	**LID4 / LDL0**	**LID3**	**LID2**	**LID1**	**LID0**	**LINIDR**
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 - LP[1:0]: Parity**
  In LIN mode:

```
LP0 = LID4 ^ LID2 ^ LID1 ^ LID0
LP1 = ! ( LID1 ^ LID3 ^ LID4 ^ LID5 )
```

  In UART mode this field is unused.

- **Bits 5:4 - LDL[1:0]: LIN 1.3 Data Length**
  In LIN 1.3 mode:
  - 00 = 2-byte response,
  - 01 = 2-byte response,
  - 10 = 4-byte response,
  - 11 = 8-byte response.

  In UART mode this field is unused.

- **Bits 3:0 - LID[3:0]: LIN 1.3 Identifier**
  In LIN 1.3 mode: 4-bit identifier.

  In UART mode this field is unused.

- **Bits 5:0 - LID[5:0]: LIN 2.1 Identifier**
  In LIN 2.1 mode: 6-bit identifier (no length transported).

  In UART mode this field is unused.

18. Analog to Digital Converter - ADC

18.1 Features

- 10-bit resolution
- 0.8 LSB integral non-linearity (at 2Mhz)
- ±3.2 LSB absolute accuracy
- 8 to 250µs conversion time
- Up to 125kSPS at maximum resolution
- 11 multiplexed single ended input channels
- 3 differential input channels with programmable gain 5, 10, 20 and 40
- Optional left adjustment for ADC result readout
- 0 to V_{CC} ADC input voltage range
- Selectable 2.56 V ADC reference voltage
- Free running or single conversion mode
- ADC start conversion by auto triggering on interrupt sources
- Interrupt on ADC conversion complete
- Sleep mode noise canceler
- Temperature sensor
- LIN address sense (ISRC voltage measurement)
- V_{CC} voltage measurement

The ATmega16/32/64/M1/C1 features a 10-bit successive approximation ADC. The ADC is connected to an 15-channel analog multiplexer which allows eleven single-ended input. The single-ended voltage inputs refer to 0V (GND).

The device also supports 3 differential voltage input amplifiers which are equipped with a programmable gain stage, providing amplification steps of 14dB (5x), 20dB (10x), 26dB (20x), or 32dB (40x) on the differential input voltage before the A/D conversion. On the amplified channels, 8-bit resolution can be expected.

The ADC contains a sample and hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 18-1 on page 198.

The ADC has a separate analog supply voltage pin, AV_{CC}. AV_{CC} must not differ more than ±0.3V from V_{CC}. See Section 18.6 "ADC Noise Canceler" on page 203 on how to connect this pin.

Internal reference voltages of nominally 2.56V or AV_{CC} are provided on-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor (e.g., 10nF) for better noise performance. In any case this capacitor shout not be greater than 10% of the AVCC smoothing capacitor.

24.7.10 Reading the Signature Row from Software

To read the signature row from software, load the Z-pointer with the signature byte address given in Table 24-5 on page 249 and set the SIGRD and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the SIGRD and SPMEN bits are set in SPMCSR, the signature byte value will be loaded in the destination register. The SIGRD and SPMEN bits will auto-clear upon completion of reading the signature row lock bits or if no LPM instruction is executed within three CPU cycles. When SIGRD and SPMEN are cleared, LPM will work as described in the instruction set manual.

Note: Before attempting to set SPMEN it is important to test this bit is cleared showing that the hardware is ready for a new operation.

Table 24-5. Signature Row Addressing

Signature Byte	Z-Pointer Address
Device signature byte 1	0x0000
Device signature byte 2	0x0002
Device signature byte 3	0x0004
RC oscillator calibration byte	0x0001
TSOFFSET temp sensor offset	0x0005
TSGAIN temp sensor gain	0x0007

Note: All other addresses are reserved for future use.

24.7.11 Preventing Flash Corruption

During periods of low V_{CC}, the flash program can be corrupted because the supply voltage is too low for the CPU and the flash to operate properly. These issues are the same as for board level systems using the flash, and the same design solutions should be applied.

A flash program corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the flash requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

1. If there is no need for a boot loader update in the system, program the boot loader lock bits to prevent any boot loader software updates.
2. Keep the AVR® RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal brown-out detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
3. Keep the AVR core in power-down sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR register and thus the flash from unintentional writes.

24.7.12 Programming Time for Flash when Using SPM

The calibrated RC oscillator is used to time flash accesses. Table 24-6 shows the typical programming time for flash accesses from the CPU.

Table 24-6. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash write (page erase, page write, and write lock bits by SPM)	3.7ms	4.5ms

25.8.9 Programming the Fuse High Bits

The algorithm for programming the fuse high bits is as follows (refer to Section 25.8.4 “Programming the Flash” on page 262 for details on command and data loading):

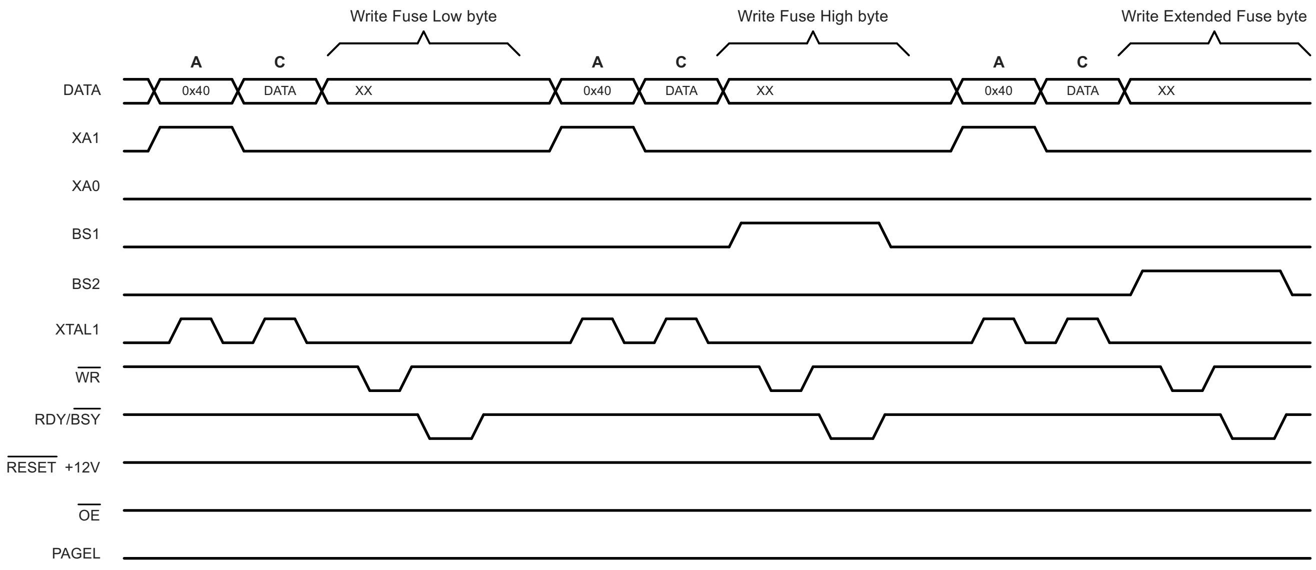
1. A: Load command “0100 0000”.
2. C: Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.
5. Set BS1 to “0”. This selects low data byte.

25.8.10 Programming the Extended Fuse Bits

The algorithm for programming the extended fuse bits is as follows (refer to Section 25.8.4 “Programming the Flash” on page 262 for details on command and data loading):

1. A: Load command “0100 0000”.
2. C: Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit.
3. Set BS1 to “0” and BS2 to “1”. This selects extended data byte.
4. Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.
5. Set BS2 to “0”. This selects low data byte.

Figure 25-5. Programming the FUSES Waveforms



25.8.11 Programming the Lock Bits

The algorithm for programming the lock bits is as follows (refer to Section 25.8.4 “Programming the Flash” on page 262 for details on command and data loading):

1. A: Load command “0010 0000”.
2. C: Load data low byte. Bit n = “0” programs the Lock bit. If LB mode 3 is programmed (LB1 and LB2 is programmed), it is not possible to program the boot lock bits by any external programming mode.
3. Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.

The lock bits can only be cleared by executing chip erase.

Table 25-17. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable serial programming after $\overline{\text{RESET}}$ goes low.
Chip erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip erase EEPROM and flash.
Read program memory	0010 **H**000	000**a aaaa**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from program memory at word address **a**:**b**.
Load program memory page	0100 **H**000	000x xxxx	**bbbb bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to program memory page at word address **b**. Data low byte must be loaded before Data high byte is applied within the same address.
Write program memory page	0100 1100	**aaaa aaaa**	**bb**xx xxxx	xxxx xxxx	Write program memory page at address **a**:**b**.
Read EEPROM memory	1010 0000	000x xx**aa**	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **a**:**b**.
Write EEPROM memory	1100 0000	000x xx**aa**	**bbbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **a**:**b**.
Load EEPROM memory page (page access)	1100 0001	0000 0000	0000 00**bb**	**iiii iiii**	Load data **i** to EEPROM memory page buffer. After data is loaded, program EEPROM page.
Write EEPROM memory page (page access)	1100 0010	00xx xx**aa**	**bbbb bb00**	xxxx xxxx	Write EEPROM page at address **a**:**b**.
Read lock bits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Read lock bits. "0" = programmed, "1" = unprogrammed. See Table 25-1 on page 255 for details.
Write lock bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Write lock bits. Set bits = "0" to program lock bits. See Table 25-1 on page 255 for details.
Read signature byte	0011 0000	000x xxxx	xxxx xx**bb**	**oooo oooo**	Read signature byte **o** at address **b**.
Write fuse bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Set bits = "0" to program, "1" to unprogram.
Write fuse high bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = "0" to program, "1" to unprogram. See Table 25-6 on page 257 for details.
Write extended fuse bits	1010 1100	1010 0100	xxxx xxxx	**xxii iiii**	Set bits = "0" to program, "1" to unprogram. See Table 25-4 on page 256 for details.
Read fuse bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read Fuse bits. "0" = programmed, "1" = unprogrammed.
Read fuse high bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read fuse high bits. "0" = programmed, "1" = unprogrammed. See Table 25-6 on page 257 for details.
Read extended fuse bits	0101 0000	0000 1000	xxxx xxxx	**oooo oooo**	Read extended fuse bits. "0" = programmed, "1" = unprogrammed. See Table 25-4 on page 256 for details.
Read calibration byte	0011 1000	000x xxxx	0000 0000	**oooo oooo**	Read calibration byte
Poll RDY/$\overline{\text{BSY}}$	1111 0000	0000 0000	xxxx xxxx	xxxx xxx**o**	If **o** = "1", a programming operation is still busy. Wait until this bit returns to "0" before applying another command.
Note: **a** = address high bits, **b** = address low bits, **H** = 0 - Low byte, 1 - High Byte, **o** = data out, **i** = data in, x = don't care					

25.9.4 SPI Serial Programming Characteristics

For characteristics of the SPI module see Section 25.9.4 "SPI Serial Programming Characteristics" on page 272.

Figure 27-21. I/O Pin Input Hysteresis Voltage versus V_{CC}

Figure 27-22. Reset Input Threshold Voltage versus V_{CC} (VIH, Reset Pin Read As '1')

Figure 27-23. Reset Input Threshold Voltage versus V_{CC} (VIL, Reset Pin Read As '0')

29. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x76)	AMP1CSR	AMP1EN	AMP1IS	AMP1G1	AMP1G0	AMPCMP1	AMP1TS2	AMP1TS1	AMP1TS0	219
(0x75)	AMP0CSR	AMP0EN	AMP0IS	AMP0G1	AMP0G0	AMPCMP0	AMP0TS2	AMP0TS1	AMP0TS0	218
(0x74)	Reserved	–	–	–	–	–	–	–	–	
(0x73)	Reserved	–	–	–	–	–	–	–	–	
(0x72)	Reserved	–	–	–	–	–	–	–	–	
(0x71)	Reserved	–	–	–	–	–	–	–	–	
(0x70)	Reserved	–	–	–	–	–	–	–	–	
(0x6F)	TIMSK1	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	114
(0x6E)	TIMSK0	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	90
(0x6D)	PCMSK3	–	–	–	–	–	PCINT26	PCINT25	PCINT24	73
(0x6C)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	73
(0x6B)	PCMSK1	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	74
(0x6A)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	74
(0x69)	EICRA	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	71
(0x68)	PCICR	–	–	–	–	PCIE3	PCIE2	PCIE1	PCIE0	72
(0x67)	Reserved	–	–	–	–	–	–	–	–	
(0x66)	OSCCAL	–	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	29
(0x65)	Reserved	–	–	–	–	–	–	–	–	
(0x64)	PRR	–	PRCAN	PRPSC	PRTIM1	PRTIM0	PRSPI	PRLIN	PRADC	36
(0x63)	Reserved	–	–	–	–	–	–	–	–	
(0x62)	Reserved	–	–	–	–	–	–	–	–	
(0x61)	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	33
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	45
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	12
0x3E (0x5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	15
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	15
0x3C (0x5C)	Reserved	–	–	–	–	–	–	–	–	
0x3B (0x5B)	Reserved	–	–	–	–	–	–	–	–	
0x3A (0x5A)	Reserved	–	–	–	–	–	–	–	–	
0x39 (0x59)	Reserved	–	–	–	–	–	–	–	–	
0x38 (0x58)	Reserved	–	–	–	–	–	–	–	–	
0x37 (0x57)	SPMCSR	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	244
0x36 (0x56)	Reserved	–	–	–	–	–	–	–	–	
0x35 (0x55)	MCUCR	SPIPS	–	–	PUD	–	–	IVSEL	IVCE	50, 57
0x34 (0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	42

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega16/32/64/M1/C1 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.
5. These registers are only available on ATmega32/64M1. For other products described in this datasheet, these locations are reserved.

Enabling Unlimited Possibilities®