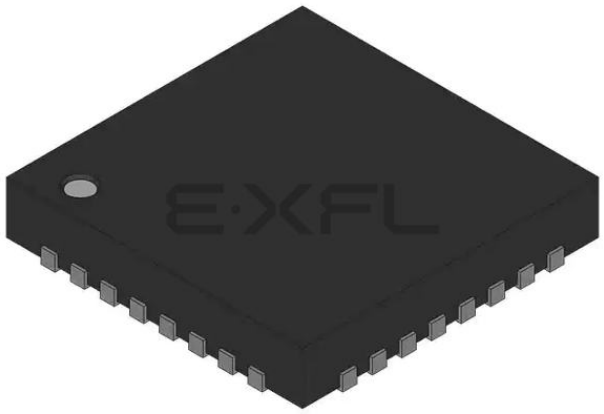


Welcome to E-XFL.COM

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	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K 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/atmega32m1-15mz

9.2.2 Toggling the Pin

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn. Note that the SBI instruction can be used to toggle one single bit in a port.

9.2.3 Switching Between Input and Output

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled {DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the MCUCR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b11) as an intermediate step.

Table 9-1 summarizes the control signals for the pin value.

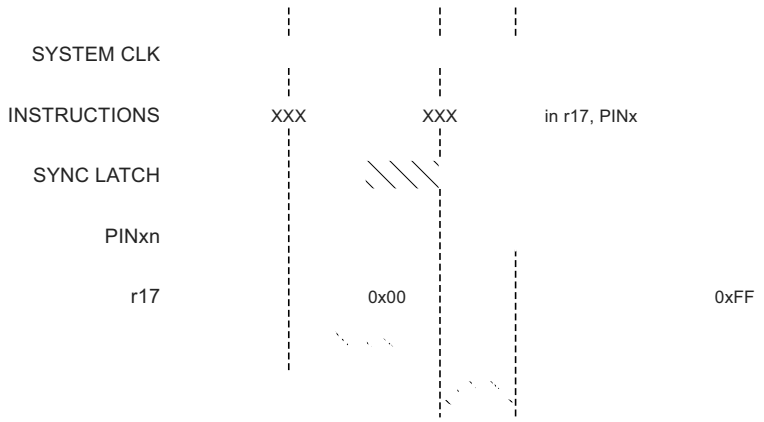
Table 9-1. Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Default configuration after reset. Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output low (sink)
1	1	X	Output	No	Output high (source)

9.2.4 Reading the Pin Value

Independent of the setting of data direction bit DDxn, the port pin can be read through the PINxn register bit. As shown in Figure 9-2, the PINxn register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 9-3 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

Figure 9-3. Synchronization when Reading an Externally Applied Pin Value



Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn register at the succeeding positive clock edge. As indicated by the two arrows $t_{pd,max}$ and $t_{pd,min}$, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

9.3.2 Alternate Functions of Port B

The Port B pins with alternate functions are shown in Table 9-3.

Table 9-3. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	PSCOUT0B (PSC output 0B) ADC4 (Analog Input Channel 4) SCK (SPI Bus Serial Clock) PCINT7 (Pin Change Interrupt 7)
PB6	ADC7 (Analog Input Channel 7) PSCOUT1B (PSC output 1B) PCINT6 (Pin Change Interrupt 6)
PB5	ADC6 (Analog Input Channel 6) INT2 (External Interrupt 2) ACMPN1 (analog comparator 1 Negative Input) AMP2- (Analog Differential Amplicator 2 Negative Input) PCINT5 (Pin Change Interrupt 5)
PB4	AMP0+ (Analog Differential Amplifier 0 Positive Input) PCINT4 (Pin Change Interrupt 4)
PB3	AMP0- (Analog Differential Amplifier 0 Negative Input) PCINT3 (Pin Change Interrupt 3)
PB2	ADC5 (Analog Input Channel5) INT1 (External Interrupt 1) ACMPN0 (analog comparator 0 Negative Input) PCINT2 (Pin Change Interrupt 2)
PB1	MOSI (SPI Master Out Slave In) PSCOUT2B (PSC output 2B) PCINT1 (Pin Change Interrupt 1)
PB0	MISO (SPI Master In Slave Out) PSCOUT2A (PSC output 2A) PCINT0 (Pin Change Interrupt 0)

The alternate pin configuration is as follows:

- **ADC4/PSCOUT0B/SCK/PCINT7 – Bit 7**

PSCOUT0B, output 0B of PSC.

ADC4, analog to digital converter, input channel 4.

SCK, master clock output, slave clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB7. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB7 bit.

PCINT7, pin change interrupt 7.

- **ADC7/PSCOUT1B/PCINT6 – Bit 6**

ADC7, analog to digital converter, input channel 7.

PSCOUT1B, output 1B of PSC.

PCINT6, pin change interrupt 6.

Table 9-13 relates the alternate functions of Port E to the overriding signals shown in Figure 9-5 on page 56.

Table 9-13. Overriding Signals for Alternate Functions in PE2..PE0

Signal Name	PE2/ADC0/XTAL2/ PCINT26	PE1/XTAL1/OC0B/ PCINT25	PE0/RESET/ OCD/PCINT24
PUOE	0	0	0
PUOV	0	0	0
DDOE	0	0	0
DDOV	0	0	0
PVOE	0	OC0Ben	0
PVOV	0	OC0B	0
DIEOE	ADC0D	0	0
DIEOV	0	0	0
DI			
AIO	Osc Output ADC0	Osc / Clock input	

9.4 Register Description for I/O-Ports

9.4.1 Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

9.4.2 Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

9.4.3 Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

9.4.4 Port C Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
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	

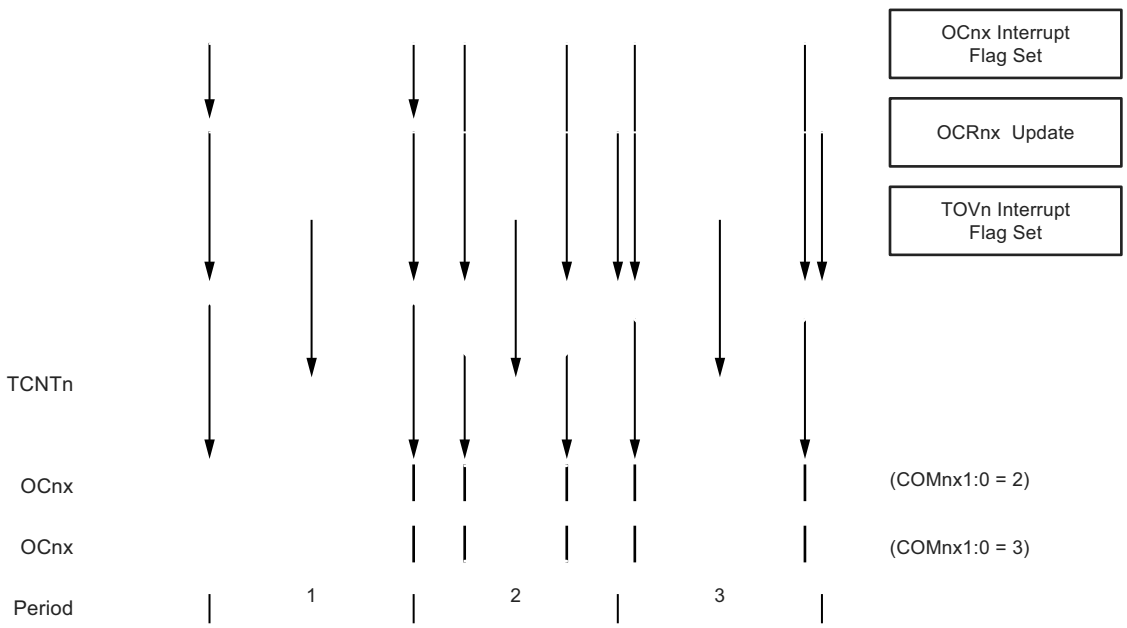
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0x to toggle its logical level on each compare match (COM0x1:0 = 1). The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is set to zero. This feature is similar to the OC0A toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

12.6.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM02:0 = 1 or 5) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. TOP is defined as 0xFF when WGM2:0 = 1, and OCR0A when WGM2:0 = 5. In non-inverting Compare Output mode, the Output Compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x while upcounting, and set on the compare match while downcounting. In inverting output compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT0 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 12-7. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 12-7. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV0) is set each time the counter reaches BOTTOM. The interrupt flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0x1:0 to three: Setting the COM0A0 bits to one allows the OC0A pin to toggle on compare matches if the WGM02 bit is set. This option is not available for the OC0B pin (see Table 12-7 on page 88). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0x register at the compare match between OCR0x and TCNT0 when the counter increments, and setting (or clearing) the OC0x register at compare match between OCR0x and TCNT0 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

13.10 16-bit Timer/Counter Register Description

13.10.1 Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COMnA1:0: Compare Output Mode for Channel A**
- **Bit 5:4 – COMnB1:0: Compare Output Mode for Channel B**

The COMnA1:0 and COMnB1:0 control the output compare pins (OCnA and OCnB respectively) behavior. If one or both of the COMnA1:0 bits are written to one, the OCnA output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COMnB1:0 bit are written to one, the OCnB output overrides the normal port functionality of the I/O pin it is connected to. However, note that the *Data Direction Register* (DDR) bit corresponding to the OCnA or OCnB pin must be set in order to enable the output driver.

When the OCnA or OCnB is connected to the pin, the function of the COMnx1:0 bits is dependent of the WGMn3:0 bits setting. Table 13-1 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to a Normal or a CTC mode (non-PWM).

Table 13-1. Compare Output Mode, non-PWM

COMnA1/COMnB1	COMnA0/COMnB0	Description
0	0	Normal port operation, OCnA/OCnB disconnected.
0	1	Toggle OCnA/OCnB on compare match.
1	0	Clear OCnA/OCnB on compare match (set output to low level).
1	1	Set OCnA/OCnB on compare match (set output to high level).

Table 13-2 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to the fast PWM mode.

Table 13-2. Compare Output Mode, Fast PWM(1)

COMnA1/COMnB1	COMnA0/COMnB0	Description
0	0	Normal port operation, OCnA/OCnB disconnected.
0	1	WGMn3:0 = 14 or 15: Toggle OC1A on compare match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OCnA/OCnB on compare match, set OCnA/OCnB at TOP
1	1	Set OCnA/OCnB on compare match, clear OCnA/OCnB at TOP

Note: 1. A special case occurs when OCRnA/OCRnB equals TOP and COMnA1/COMnB1 is set. In this case the compare match is ignored, but the set or clear is done at TOP. See Section 13.8.3 “Fast PWM Mode” on page 103 for more details.

- **Bit 4:3:2 – SWAPn: SWAP Funtion Select (not implemented in ATmega32M1 up to revision C)**

When this bit is set; the channels PSCOUTnA and PSCOUTnB are exchanged. This allows to invert the waveforms of both channels at one time.

- **Bit 1 – PCCYC: PSC Complete Cycle**

When this bit is set, the PSC completes the entire waveform cycle before halt operation requested by clearing PRUN.

- **Bit 0 – PRUN: PSC Run**

Writing this bit to one starts the PSC.

14.16.9 PSC Module n Input Control Register – PMICn

Bit	7	6	5	4	3	2	1	0	
	POVENn	**PISELn**	**PELEVn**	**PFLTEn**	**PAOCn**	**PRFMn2**	**PRFMn1**	**PRFMn0**	**PMICn**
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	

The input control registers are used to configure the 2 PSC's Retrigger/Fault block A and B. The 2 blocks are identical, so they are configured on the same way.

- **Bit 7 – POVENn: PSC Module n Overlap Enable**

Set this bit to disactivate the overlap protection. See Section 14.7 "Overlap Protection" on page 122.

- **Bit 6 – PISELn: PSC Module n Input Select**

Clear this bit to select PSCINn as module n input.

Set this bit to select comparator n output as module n input.

- **Bit 5 –PELEVn: PSC Module n Input Level Selector**

When this bit is clear, the low level of selected input generates the significative event for fault function.

When this bit is set, the high level of selected input generates the significative event for fault function.

- **Bit 4 – PFLTEn: PSC Module n Input Filter Enable**

Setting this bit (to one) activates the input noise canceler. When the noise canceler is activated, the input from the input pin is filtered. The filter function requires four successive equal valued samples of the input pin for changing its output. The input is therefore delayed by four oscillator cycles when the noise canceler is enabled.

- **Bit 3 – PAOCn: PSC Module n 0 Asynchronous Output Control**

When this bit is clear, fault input can act directly to PSC module n outputs A and B. See Section 14.9.1 "PSC Input Configuration" on page 124.

- **Bit 2:0 – PRFMn2:0: PSC Module n Input Mode**

These three bits define the mode of operation of the PSC inputs.

Table 14-12. 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

15.2 $\overline{SS}$ Pin Functionality

15.2.1 Slave Mode

When the SPI is configured as a slave, the slave select ($\overline{SS}$) pin is always input. When $\overline{SS}$ is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When $\overline{SS}$ is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the $\overline{SS}$ pin is driven high.

The $\overline{SS}$ pin is useful for packet/byte synchronization to keep the slave bit counter synchronous with the master clock generator. When the $\overline{SS}$ pin is driven high, the SPI slave will immediately reset the send and receive logic, and drop any partially received data in the shift register.

15.2.2 Master Mode

When the SPI is configured as a master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin.

If $\overline{SS}$ is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the $\overline{SS}$ pin of the SPI slave.

If $\overline{SS}$ is configured as an input, it must be held high to ensure master SPI operation. If the $\overline{SS}$ pin is driven low by peripheral circuitry when the SPI is configured as a master with the $\overline{SS}$ pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a slave. As a result of the SPI becoming a slave, the MOSI and SCK pins become inputs.
2. The SPIF flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in master mode, and there exists a possibility that $\overline{SS}$ is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI master mode.

15.2.3 MCU Control Register – MCUCR

Bit	7	6	5	4	3	2	1	0	
	SPIPS	**–**	**–**	**PUD**	**–**	**–**	**IVSEL**	**IVCE**	**MCUCR**
Read/Write	R/W	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7– SPIPS: SPI Pin Redirection**
  - Thanks to SPIPS (SPI pin select) in MCUCR Sfr, SPI pins can be redirected.
  - When the SPIPS bit is written to zero, the SPI signals are directed on pins MISO,MOSI, SCK and SS.
  - When the SPIPS bit is written to one, the SPI signals are directed on alternate SPI pins, MISO_A, MOSI_A, SCK_A and SS_A.

Note that programming port are always located on alternate SPI port.

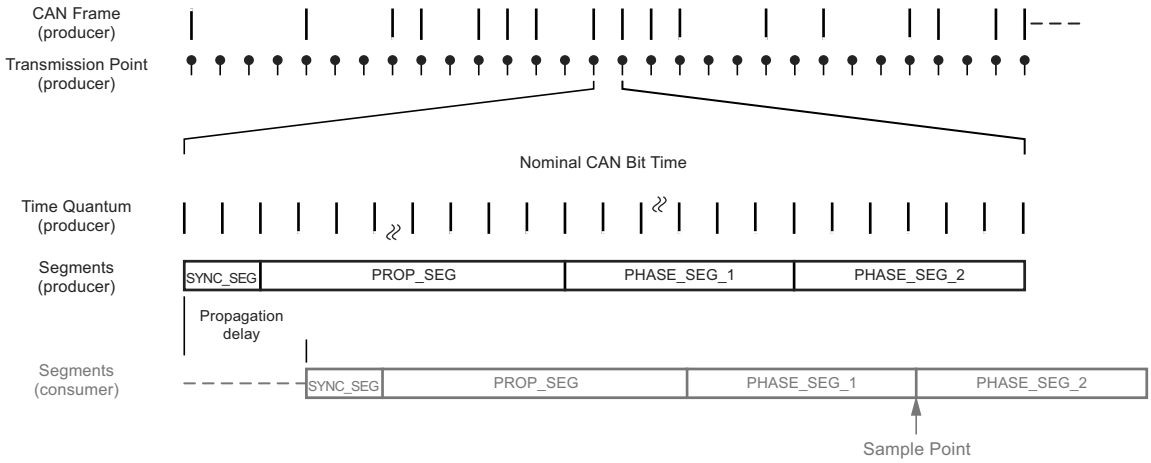
16.2.3 CAN Bit Timing

To ensure correct sampling up to the last bit, a CAN node needs to re-synchronize throughout the entire frame. This is done at the beginning of each message with the falling edge SOF and on each recessive to dominant edge.

16.2.3.1 Bit Construction

One CAN bit time is specified as four non-overlapping time segments. Each segment is constructed from an integer multiple of the time quantum. The time quantum or TQ is the smallest discrete timing resolution used by a CAN node.

Figure 16-3. CAN Bit Construction



16.2.3.2 Synchronization Segment

The first segment is used to synchronize the various bus nodes.

On transmission, at the start of this segment, the current bit level is output. If there is a bit state change between the previous bit and the current bit, then the bus state change is expected to occur within this segment by the receiving nodes.

16.2.3.3 Propagation Time Segment

This segment is used to compensate for signal delays across the network.

This is necessary to compensate for signal propagation delays on the bus line and through the transceivers of the bus nodes.

16.2.3.4 Phase Segment 1

Phase Segment 1 is used to compensate for edge phase errors.

This segment may be lengthened during re-synchronization.

16.2.3.5 Sample Point

The sample point is the point of time at which the bus level is read and interpreted as the value of the respective bit. Its location is at the end of phase segment 1 (between the two phase segments).

16.2.3.6 Phase Segment 2

This segment is also used to compensate for edge phase errors.

This segment may be shortened during re-synchronization, but the length has to be at least as long as the information processing time (IPT) and may not be more than the length of phase segment 1.

16.2.3.7 Information Processing Time

It is the time required for the logic to determine the bit level of a sampled bit.

The IPT begins at the sample point, is measured in TQ and is fixed at 2TQ for the Atmel CAN. Since phase segment 2 also begins at the sample point and is the last segment in the bit time, PS2 minimum shall not be less than the IPT.

16.10.7 CAN Status Interrupt MOb Registers - CANSIT2 and CANSIT1

Bit	7	6	5	4	3	2	1	0	
	-	-	SIT5	SIT4	SIT3	SIT2	SIT1	SIT0	CANSIT2
	-	-	-	-	-	-	-	-	CANSIT1
Bit	15	14	13	12	11	10	9	8	
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 5:0 - SIT5:0: Status of Interrupt by MOb**
  - 0 - no interrupt.
  - 1- MOb interrupt.

Note: Example: CANSIT2 = 0010 0001_b: MOb 0 and 5 interrupts.

- **Bit 15:6 – Reserved Bits**

These bits are reserved for future use.

16.10.8 CAN Bit Timing Register 1 - CANBT1

Bit	7	6	5	4	3	2	1	0	
	-	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	-	CANBT1
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	-	
Initial Value	-	0	0	0	0	0	0	-	

- **Bit 7– Reserved Bit**

This bit is reserved for future use. For compatibility with future devices, it must be written to zero when CANBT1 is written.

- **Bit 6:1 – BRP5:0: Baud Rate Prescaler**

The period of the CAN controller system clock Tscl is programmable and determines the individual bit timing.

$$\text{Tscl} = \frac{\text{BRP[5:0]} + 1}{\text{clk}_{\text{IO}}\text{frequency}}$$

If ‘BRP[5..0]=0’, see Section 16.4.3 “Baud Rate” on page 148 and Section • “Bit 0 – SMP: Sample Point(s)” on page 164.

- **Bit 0 – Reserved Bit**

This bit is reserved for future use. For compatibility with future devices, it must be written to zero when CANBT1 is written.

16.10.9 CAN Bit Timing Register 2 - CANBT2

Bit	7	6	5	4	3	2	1	0	
	-	SJW1	SJW0	-	PRS2	PRS1	PRS0	-	CANBT2
Read/Write	-	R/W	R/W	-	R/W	R/W	R/W	-	
Initial Value	-	0	0	-	0	0	0	-	

- **Bit 7– Reserved Bit**

This bit is reserved for future use. For compatibility with future devices, it must be written to zero when CANBT2 is written.

17.5.4 Configuration

Depending on the mode (LIN or UART), LCONF[1..0] bits of the LINCR register set the controller in the following configuration (Table 17-3):

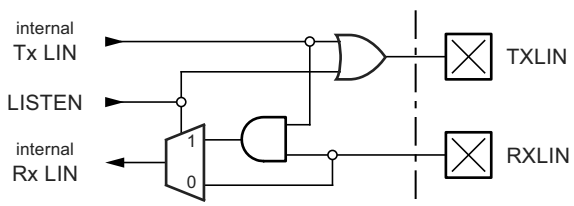
Table 17-3. Configuration Table versus Mode

Mode	LCONF[1..0]	Configuration
LIN	00_b	LIN standard configuration (default)
	01_b	No CRC field detection or transmission
	10_b	Frame_Time_Out disable
	11_b	Listening mode
UART	00_b	8-bit data, no parity and 1 stop-bit
	01_b	8-bit data, even parity and 1 stop-bit
	10_b	8-bit data, odd parity and 1 stop-bit
	11_b	Listening mode, 8-bit data, no parity and 1 stop-bit

The LIN configuration is independent of the programmed LIN protocol.

The listening mode connects the internal Tx LIN and the internal Rx LIN together. In this mode, the TXLIN output pin is disabled and the RXLIN input pin is always enabled. The same scheme is available in UART mode.

Figure 17-6. Listening Mode

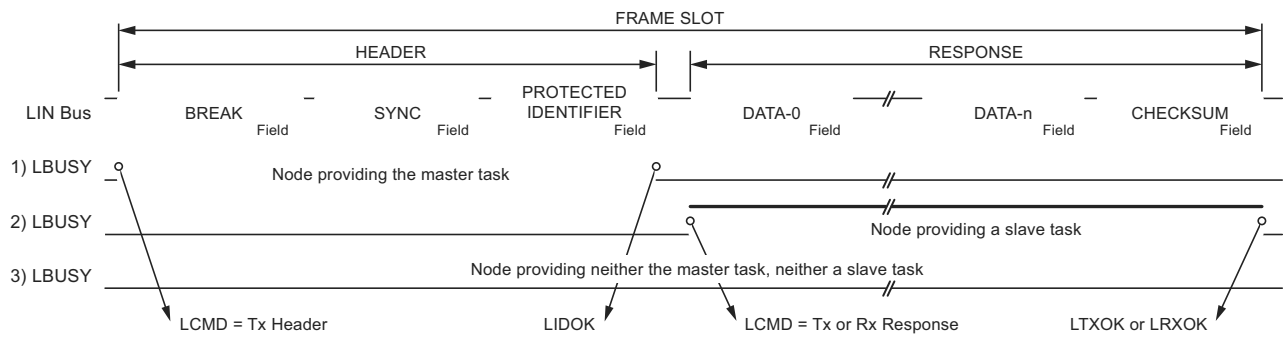


17.5.5 Busy Signal

LBUSY bit flag in LINSIR register is the image of the BUSY signal. It is set and cleared by hardware. It signals that the controller is busy with LIN or UART communication.

17.5.5.1 Busy Signal in LIN Mode

Figure 17-7. Busy Signal in LIN Mode



When the busy signal is set, some registers are locked, user writing is not allowed:

- “LIN Control Register” - LINCR - except LCMD[2..0], LENA and LSWRES,
- “LIN Baud Rate Registers” - LINBRRL and LINBRRH,
- “LIN Data Length Register” - LINDLR,
- “LIN Identifier Register” - LINIDR,
- “LIN Data Register” - LINDAT.

If the busy signal is set, the only available commands are:

- LCMD[1..0] = 00_b, the abort command is taken into account at the end of the byte,
- LENA = 0 and/or LCMD[2] = 0, the kill command is taken into account immediately,
- LSWRES = 1, the reset command is taken into account immediately.

Note that, if another command is entered during busy signal, the new command is not validated and the LOVRERR bit flag of the LINERR register is set. The on-going transfer is not interrupted.

17.5.5.2 Busy Signal in UART Mode

During the byte transmission, the busy signal is set. This locks some registers from being written:

- “LIN Control Register” - LINCR - except LCMD[2..0], LENA and LSWRES,
- “LIN Data Register” - LINDAT.

The busy signal is not generated during a byte reception.

17.5.6 Bit Timing

17.5.6.1 Baud rate Generator

The baud rate is defined to be the transfer rate in bits per second (bps):

- BAUD: Baud rate (in bps),
- $fclk_{i/o}$: System I/O clock frequency,
- LDIV[11..0]: Contents of LINBRRH & LINBRRL registers - (0-4095), the pre-scaler receives $clk_{i/o}$ as input clock.
- LBT[5..0]: Least significant bits of - LINBTR register- (0-63) is the number of samplings in a LIN or UART bit (default value 32).

Equation for calculating baud rate:

$$BAUD = fclk_{i/o} / LBT[5..0] \times (LDIV[11..0] + 1)$$

Equation for setting LINDIV value:

$$LDIV[11..0] = (fclk_{i/o} / LBT[5..0] \times BAUD) - 1$$

Note that in reception a majority vote on three samplings is made.

17.5.6.2 Re-synchronization in LIN Mode

When waiting for Rx Header, LBT[5..0] = 32 in LINBTR register. The re-synchronization begins when the BREAK is detected. If the BREAK size is not in the range (11 bits min., 28 bits max. — 13 bits nominal), the BREAK is refused. The re-synchronization is done by adjusting LBT[5..0] value to the SYNCH field of the received header (0x55). Then the PROTECTED IDENTIFIER is sampled using the new value of LBT[5..0]. The re-synchronization implemented in the controller tolerates a clock deviation of ±20% and adjusts the baud rate in a ±2% range.

The new LBT[5..0] value will be used up to the end of the response. Then, the LBT[5..0] will be reset to 32 for the next header.

The LINBTR register can be used to re-calibrate the clock oscillator.

The re-synchronization is not performed if the LIN node is enabled as a master.

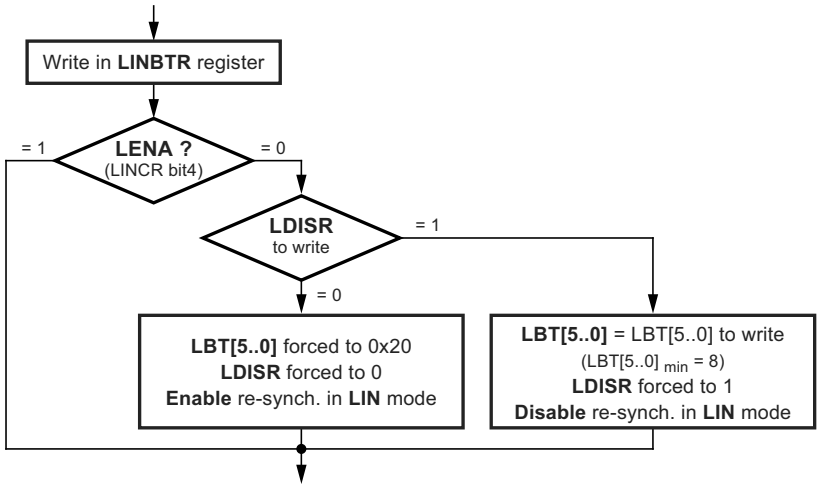
17.5.6.3 Handling LBT[5..0]

LDISR bit of LINBTR register is used to:

- To enable the setting of LBT[5..0] (to manually adjust the baud rate especially in the case of UART mode). A minimum of 8 is required for LBT[5..0] due to the sampling operation.
- Disable the re-synchronization in LIN Slave Mode for test purposes.

Note that the LENA bit of LINCR register is important for this handling (see Figure 17-8).

Figure 17-8. Handling LBT[5..0]



17.5.7 Data Length

Section 17.4.6 “LIN Commands” on page 179 describes how to set or how are automatically set the LRXDL[3..0] or LTXDL[3..0] fields of LINDLR register before receiving or transmitting a response.

In the case of Tx Response the LRXDL[3..0] will be used by the hardware to count the number of bytes already successfully sent.

In the case of Rx Response the LTXDL[3..0] will be used by the hardware to count the number of bytes already successfully received.

If an error occurs, this information is useful to the programmer to recover the LIN messages.

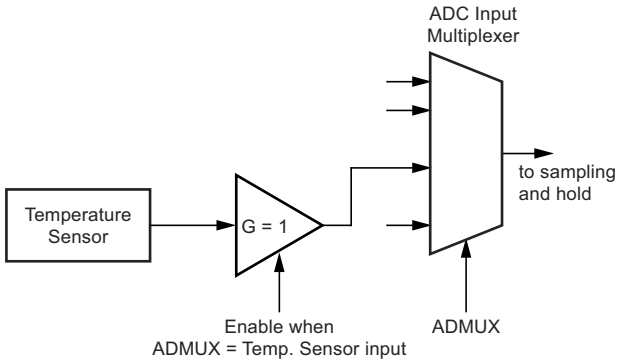
17.5.7.1 Data Length in LIN 2.1

- If LTXDL[3..0]=0 only the CHECKSUM will be sent,
- If LRXDL[3..0]=0 the first byte received will be interpreted as the CHECKSUM,
- If LTXDL[3..0] or LRXDL[3..0] >8, values will be forced to 8 after the command setting and before sending or receiving of the first byte.

17.5.7.2 Data Length in LIN 1.3

- LRXDL and LTXDL fields are both hardware updated before setting LIDOK by decoding the data length code contained in the received PROTECTED IDENTIFIER (LRXDL = LTXDL).
- Via the above mechanism, a length of 0 or >8 is not possible.

Figure 18-14. Temperature Sensor Block Diagram



The measured voltage has a linear relationship to the temperature as described in Table 18-3. The voltage sensitivity is approximately 2.5mV/°C and the accuracy of the temperature measurement is ±10°C after bandgap calibration.

Table 18-3. Temperature versus Sensor Output Voltage (Typical Case)

Temperature/°C	–40°C	+25°C	+125°C
Voltage/mV	600mV	762mv	1012mV

The values described in Table 18-3 on page 209 are typical values. However, due to the process variation the temperature sensor output voltage varies from one chip to another. To be capable of achieving more accurate results, the temperature measurement can be calibrated in the application software.

18.8.1 User Calibration

The software calibration requires that a calibration value is measured and stored in a register or EEPROM for each chip. The software calibration can be done utilizing the formula:

$T = \{[(ADCH << 8) \mid ADC] - T_{OS}\} / k$

where ADCH and ADCL are the ADC data registers, k is a fixed coefficient and T_{OS} is the temperature sensor offset value determined and stored into EEPROM.

18.8.2 Manufacturing Calibration

One can also use the calibration values available in the signature row (see Section 24.7.10 “Reading the Signature Row from Software” on page 249).

The calibration values are determined from values measured during test at room temperature which is approximately +25°C and during test at hot temperature which is approximately +125°C. Calibration measures are done at V_{CC} = 3V and with ADC in internal Vref (2.56V) mode.

There are two algorithms for determining the Centigrade Temperature

formula 1 for ATmega32 up to rev B

formula 2 for ATmega16/64 and ATmega32 rev C.

formula 1: Temp_C = (((ADC_ts – 273) × TS_Gain) / 128) + TS_Offset [Applicable to devices with 0xFF or 0x42 ('B') in the signature memory at address 0x003F]

formula 2: Temp_C = ((((ADC_ts – (298 – TS_Offset)) × TS_Gain) / 128) + 25 [Applicable to devices with 0x43 ('C') in the signature memory at address 0x003F]

Where:

Temp_C is the result temperature in degrees centigrade.

ADC_ts is the 10 bit result the ADC returns from reading the temperature sensor.

TS_Gain is the unsigned fixed point 8-bit temperature sensor gain factor in 1/128th units stored as previously in the signature row at address 0x0007.

19.2.4 Threshold Reference for Internal analog comparator

An external resistor used in conjunction with the Current Source can be used as threshold reference for internal analog comparator (see Section 20. “Analog Comparator” on page 225). This can be connected to AIN0 (negative analog compare input pin) as well as AIN1 (positive analog compare input pin). Using a resistor in series with a lower tolerance than the current source accuracy (≤ 2%) is recommended. Table 19-2 gives an example of threshold references using standard values of resistors.

19.3 Control Register

19.3.1 ADC control and status register B– ADCSRB

Bit	7	6	5	4	3	2	1	0	
	ADHSM	**ISRCEN**	**AREFEN**	**-**	**ADTS3**	**ADTS2**	**ADTS1**	**ADTS0**	**ADCSRB**
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – ISRCEN: Current Source Enable**

Set this bit to source a 100µA current to the AREF pin.
Clear this bit to disconnect.

- **Bit 5 – AREFEN: Analog Reference pin Enable**

Set this bit to connect the internal AREF circuit to the AREF pin.
Clear this bit to disconnect the internal AREF circuit from the AREF pin.

24.6 Addressing the Flash during Self-Programming

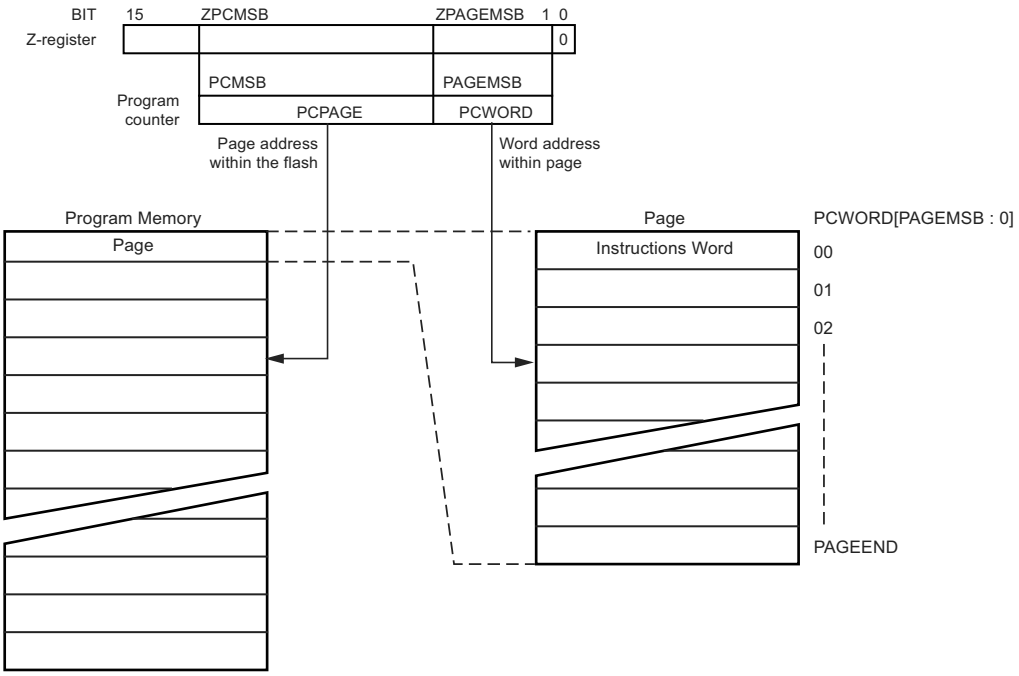
The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
ZL (R30)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
	7	6	5	4	3	2	1	0

Since the flash is organized in pages (see Table 25-12 on page 260), the program counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is1 shown in Figure 24-3. Note that the page erase and page write operations are addressed independently. Therefore it is of major importance that the boot loader software addresses the same page in both the page erase and page write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

The only SPM operation that does not use the Z-pointer is setting the boot loader lock bits. The content of the Z-pointer is ignored and will have no effect on the operation. The LPM instruction does also use the Z-pointer to store the address. Since this instruction addresses the flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

Figure 24-3. Addressing the Flash during SPM(1)



Note: 1. The different variables used in Figure 24-3 are listed in Table 24-9 on page 252.

24.7 Self-programming the Flash

The program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the page erase command or between a page erase and a page write operation:

Alternative 1, fill the buffer before a page erase

- Fill temporary page buffer
- Perform a page erase
- Perform a page write

Alternative 2, fill the buffer after page erase

- Perform a page erase
- Fill temporary page buffer
- Perform a page write

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be rewritten. When using alternative 1, the boot loader provides an effective read-modify-write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the page erase and page write operation is addressing the same page. See Section 24.7.13 “Simple Assembly Code Example for a Boot Loader” on page 250 for an assembly code example.

24.7.1 Performing Page Erase by SPM

To execute page erase, set up the address in the Z-pointer, write “X0000011” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer will be ignored during this operation.

- Page erase to the RWW section: The NRWW section can be read during the page erase.
- Page erase to the NRWW section: The CPU is halted during the operation.

24.7.2 Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write “00000001” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will auto-erase after a page write operation or by writing the RWWSRE bit in SPMCSR. It is also erased after a system reset. Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

If the EEPROM is written in the middle of an SPM page load operation, all data loaded will be lost.

24.7.3 Performing a Page Write

To execute page write, set up the address in the Z-pointer, write “X0000101” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

- Page write to the RWW section: The NRWW section can be read during the page write.
- Page write to the NRWW section: The CPU is halted during the operation.

24.7.4 Using the SPM Interrupt

If the SPM interrupt is enabled, the SPM interrupt will generate a constant interrupt when the SPMEN bit in SPMCSR is cleared. This means that the interrupt can be used instead of polling the SPMCSR register in software. When using the SPM interrupt, the interrupt vectors should be moved to the BLS section to avoid that an interrupt is accessing the RWW section when it is blocked for reading.

24.7.5 Consideration While Updating BLS

Special care must be taken if the user allows the boot loader section to be updated by leaving boot lock bit11 unprogrammed. An accidental write to the boot loader itself can corrupt the entire boot loader, and further software updates might be impossible. If it is not necessary to change the boot loader software itself, it is recommended to program the boot lock bit11 to protect the boot loader software from any internal software changes.

24.7.6 Prevent Reading the RWW Section during Self-programming

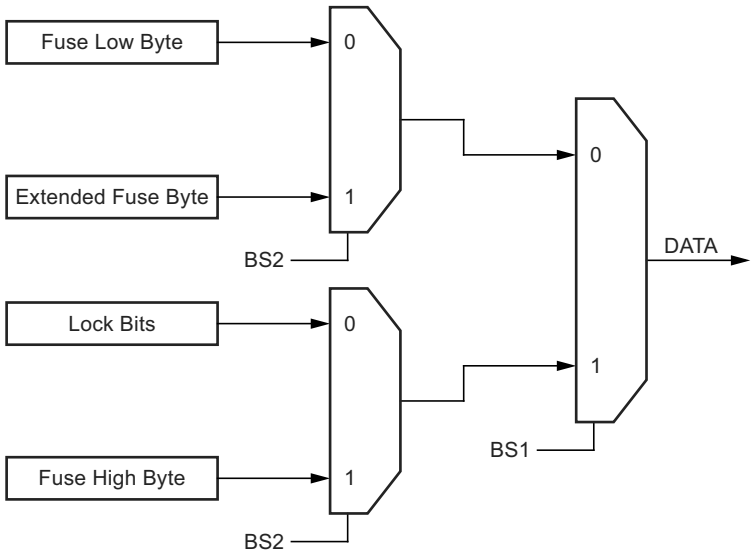
During self-programming (either page erase or page write), the RWW section is always blocked for reading. The user software itself must prevent that this section is addressed during the self programming operation. The RWWSB in the SPMCSR will be set as long as the RWW section is busy. During self-programming the Interrupt vector table should be moved to the BLS or the interrupts must be disabled. Before addressing the RWW section after the programming is completed, the user software must clear the RWWSB by writing the RWWSRE. See Section 24.7.13 “Simple Assembly Code Example for a Boot Loader” on page 250 for an example.

25.8.12 Reading the Fuse and Lock Bits

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

1. A: Load command “0000 0100”.
2. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “0”. The status of the fuse low bits can now be read at DATA (“0” means programmed).
3. Set $\overline{OE}$ to “0”, BS2 to “1” and BS1 to “1”. The status of the fuse high bits can now be read at DATA (“0” means programmed).
4. Set OE to “0”, BS2 to “1”, and BS1 to “0”. The status of the extended fuse bits can now be read at DATA (“0” means programmed).
5. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “1”. The status of the Lock bits can now be read at DATA (“0” means programmed).
6. Set $\overline{OE}$ to “1”.

Figure 25-6. Mapping Between BS1, BS2 and the Fuse and Lock Bits during Read



25.8.13 Reading the Signature Bytes

The algorithm for reading the signature bytes is as follows (refer to Section 25.8.4 “Programming the Flash” on page 262 for details on command and address loading):

1. A: Load command “0000 1000”.
2. B: Load address low byte (0x00 - 0x02).
3. Set $\overline{OE}$ to “0”, and BS1 to “0”. The selected signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

25.8.14 Reading the Calibration Byte

The algorithm for reading the calibration byte is as follows (refer to Section 25.8.4 “Programming the Flash” on page 262 for details on command and address loading):

1. A: Load command “0000 1000”.
2. B: Load address low byte, 0x00.
3. Set $\overline{OE}$ to “0”, and BS1 to “1”. The calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

4. The flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 6 LSB of the address and data together with the load program memory page instruction. To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address. The program memory page is stored by loading the write program memory page instruction with the 8 MSB of the address. If polling is not used, the user must wait at least t_{WD_FLASH} before issuing the next page. (See Table 25-16.) Accessing the serial programming interface before the flash write operation completes can result in incorrect programming.
5. The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte. (See Table 25-16.) In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed): Set $\overline{\text{RESET}}$ to “1”. Turn V_{CC} power off.

25.9.2 Data Polling Flash

When a page is being programmed into the flash, reading an address location within the page being programmed will give the value 0xFF. At the time the device is ready for a new page, the programmed value will read correctly. This is used to determine when the next page can be written. Note that the entire page is written simultaneously and any address within the page can be used for polling. Data polling of the flash will not work for the value 0xFF, so when programming this value, the user will have to wait for at least t_{WD_FLASH} before programming the next page. As a chip-erased device contains 0xFF in all locations, programming of addresses that are meant to contain 0xFF, can be skipped. See Table 25-16 for t_{WD_FLASH} value.

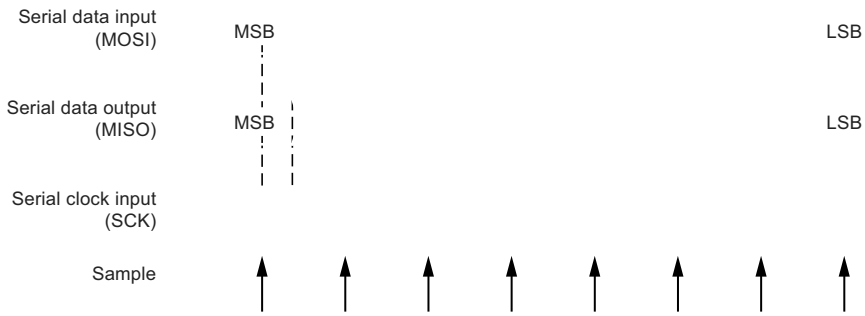
25.9.3 Data Polling EEPROM

When a new byte has been written and is being programmed into EEPROM, reading the address location being programmed will give the value 0xFF. At the time the device is ready for a new byte, the programmed value will read correctly. This is used to determine when the next byte can be written. This will not work for the value 0xFF, but the user should have the following in mind: As a chip-erased device contains 0xFF in all locations, programming of addresses that are meant to contain 0xFF, can be skipped. This does not apply if the EEPROM is re-programmed without chip erasing the device. In this case, data polling cannot be used for the value 0xFF, and the user will have to wait at least t_{WD_EEPROM} before programming the next byte. See
Table 25-16 for t_{WD_EEPROM} value.

Table 25-16. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	4.5ms
t_{WD_EEPROM}	3.6ms
t_{WD_ERASE}	9.0ms

Figure 25-11. Serial Programming Waveforms



29. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x33 (0x53)	SMCR	–	–	–	–	SM2	SM1	SM0	SE	34
0x32 (0x52)	MSMCR	Monitor Stop Mode Control Register								Reserved
0x31 (0x51)	MONDR	Monitor Data Register								Reserved
0x30 (0x50)	ACSR	AC3IF	AC2IF	AC1IF	AC0IF	AC3O	AC2O	AC1O	AC0O	231
0x2F (0x4F)	Reserved	–	–	–	–	–	–	–	–	
0x2E (0x4E)	SPDR	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0	139
0x2D (0x4D)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	139
0x2C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	138
0x2B (0x4B)	Reserved	–	–	–	–	–	–	–	–	
0x2A (0x4A)	Reserved	–	–	–	–	–	–	–	–	
0x29 (0x49)	PLLCSR	–	–	–	–	–	PLLF	PLLE	PLOCK	31
0x28 (0x48)	OCR0B	OCR0B7	OCR0B6	OCR0B5	OCR0B4	OCR0B3	OCR0B2	OCR0B1	OCR0B0	90
0x27 (0x47)	OCR0A	OCR0A7	OCR0A6	OCR0A5	OCR0A4	OCR0A3	OCR0A2	OCR0A1	OCR0A0	90
0x26 (0x46)	TCNT0	TCNT07	TCNT06	TCNT05	TCNT04	TCNT03	TCNT02	TCNT01	TCNT00	90
0x25 (0x45)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	89
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	86
0x23 (0x43)	GTCCR	TSM	ICPSEL1	–	–	–	–	–	PSRSYNC	76
0x22 (0x42)	EEARH	–	–	–	–	–	–	EEAR9	EEAR8	20
0x21 (0x41)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	20
0x20 (0x40)	EEDR	EEDR7	EEDR6	EEDR5	EEDR4	EEDR3	EEDR2	EEDR1	EEDR0	20
0x1F (0x3F)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	21
0x1E (0x3E)	GPIOR0	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	24
0x1D (0x3D)	EIMSK	–	–	–	–	INT3	INT2	INT1	INT0	71
0x1C (0x3C)	EIFR	–	–	–	–	INTF3	INTF2	INTF1	INTF0	72
0x1B (0x3B)	PCIFR	–	–	–	–	PCIF3	PCIF2	PCIF1	PCIF0	73
0x1A (0x3A)	GPIOR2	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	24
0x19 (0x39)	GPIOR1	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	24
0x18 (0x38)	Reserved	–	–	–	–	–	–	–	–	
0x17 (0x37)	Reserved	–	–	–	–	–	–	–	–	
0x16 (0x36)	TIFR1	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	115
0x15 (0x35)	TIFR0	–	–	–	–	–	OCF0B	OCF0A	TOV0	91
0x14 (0x34)	Reserved	–	–	–	–	–	–	–	–	
0x13 (0x33)	Reserved	–	–	–	–	–	–	–	–	
0x12 (0x32)	Reserved	–	–	–	–	–	–	–	–	
0x11 (0x31)	Reserved	–	–	–	–	–	–	–	–	

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.

30. Errata

30.1 Errata Summary

30.1.1 ATmega16M1/16C1/32M1/32C1 Rev. C (Mask Revision)

- LIN break delimiter
- ADC with PSC2-synchronized
- ADC amplifier measurement is unstable

30.1.2 ATmega16M1/16C1/32M1/32C1 Rev. B (Mask Revision)

- The AMPCMPx bits return 0
- No comparison when amplifier is used as comparator input and ADC input
- CRC calculation of diagnostic frames in LIN 2.x.
- Wrong TSOFFSET manufacturing calibration value
- PD0-PD3 set to outputs and PD4 pulled down following power-on with external reset active.
- LIN break delimiter
- ADC with PSC2-synchronized
- ADC amplifier measurement is unstable
- PSC emulation
- PSC OCRxx register update according to PLOCK2 usage
- Read/Write instructions of MUXn and REFS1:0

30.1.3 ATmega16M1/16C1/32M1/32C1 Rev. A (Mask Revision)

- Inopportune reset of the CANIDM registers
- The AMPCMPx bits return 0
- No comparison when amplifier is used as comparator input and ADC input
- CRC calculation of diagnostic frames in LIN 2.x
- PD0-PD3 set to outputs and PD4 pulled down following power-on with external reset active
- LIN break delimiter
- ADC with PSC2-synchronized
- ADC amplifier measurement is unstable
- PSC emulation
- Read/Write instructions of MUXn and REFS1:0

30.1.4 Errata Description

1. **Inopportune reset of the CANIDM registers**
   After the reception of a CAN frame in a MOb, the ID mask registers are reset.
   Problem fix / workaround
   Before enabling a MOb in reception, re-initialize the ID mask registers - **CANIDM[4..1]**.
2. **The AMPCMPx bits return 0**
   When they are read the AMPCMPx bits in AMPxCSR registers return 0.
   Problem fix / workaround
   If the reading of the AMPCMPx bits is required, store the AMPCMPx value in a variable in memory before writing in the AMPxCSR register and read the variable when necessary.
3. **No comparison when amplifier is used as comparator input and ADC input**
   When it is selected as ADC input, an amplifier receives no clock signal when the ADC is stopped. In that case, if the amplifier is also used as comparator input, no analog signal is propagated and no comparison is done.
   Problem fix / workaround
   Select another ADC channel rather than the working amplified channel.