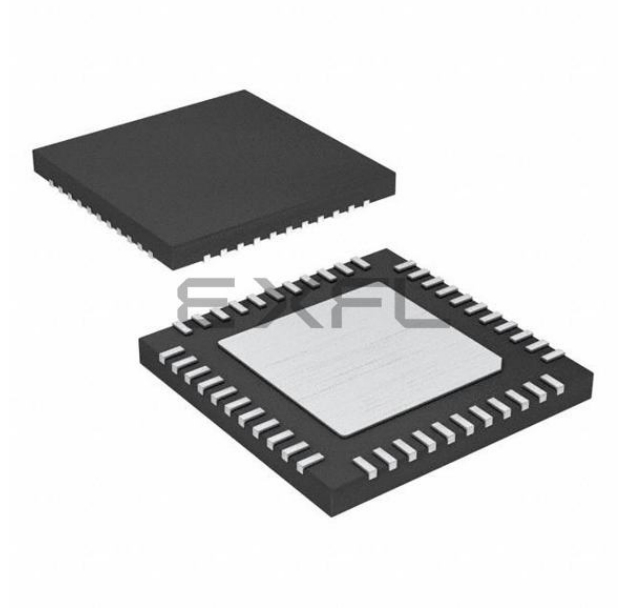


Welcome to E-XFL.COM

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	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	32
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 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VFQFN Exposed Pad
Supplier Device Package	44-VQFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega164p-b15mz

The CKDIV8 fuse determines the initial value of the CLKPS bits. If CKDIV8 is unprogrammed, the CLKPS bits will be reset to “0000”. If CKDIV8 is programmed, CLKPS bits are reset to “0011”, giving a division factor of 8 at start up. This feature should be used if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. Note that any value can be written to the CLKPS bits regardless of the CKDIV8 fuse setting. The Application software must ensure that a sufficient division factor is chosen if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. The device is shipped with the CKDIV8 fuse programmed.

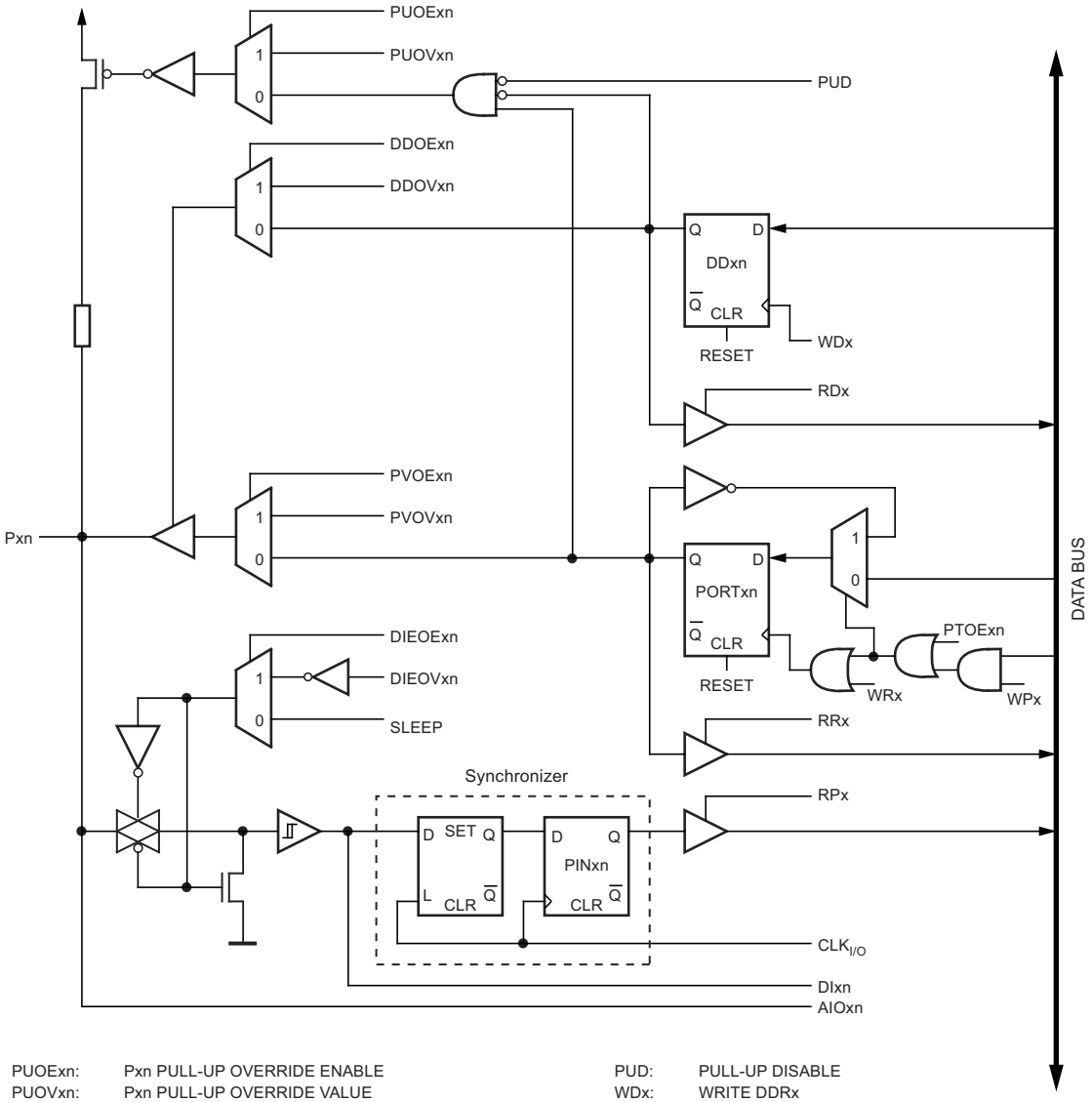
Table 9-16. Clock Prescaler Select

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock Division Factor
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256
1	0	0	1	Reserved
1	0	1	0	Reserved
1	0	1	1	Reserved
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

14.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 14-5 shows how the port pin control signals from the simplified Figure 14-2 on page 58 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.

Figure 14-5. Alternate Port Functions(1)



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE
DIEOExn:	Pxn DIGITAL INPUT ENABLE OVERRIDE ENABLE
DIEOVxn:	Pxn DIGITAL INPUT ENABLE OVERRIDE VALUE
SLEEP:	SLEEP CONTROL
PTOExn:	Pxn, PORT TOGGLE OVERRIDE ENABLE

PUD:	PULL-UP DISABLE
WDx:	WRITE DDRx
RDx:	READ DDRx
RRx:	READ PORTx REGISTER
WRx:	WRITE PORTx
RPx:	READ PORTx PIN
WPx:	WRITE PINx
CLK$_{I/O}$:	I/O CLOCK
DIxn:	DIGITAL INPUT PIN n ON PORTx
AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

- **ADC7:0/PCINT7:0 – Port A, Bit 7:0**

ADC7:0, analog to digital converter, channels 7:0.

PCINT7:0, pin change interrupt source 7:0: The PA7:0 pins can serve as external interrupt sources.

Table 14-4 and Table 14-5 relate the alternate functions of port A to the overriding signals shown in Figure 14-5 on page 62.

Table 14-4. Overriding Signals for Alternate Functions in PA7:PA4

Signal Name	PA7/ADC7/PCINT7	PA6/ADC6/PCINT6	PA5/ADC5/PCINT5	PA4/ADC4/PCINT4
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	PCINT7 × PCIE0 + ADC7D	PCINT6 × PCIE0 + ADC6D	PCINT5 × PCIE0 + ADC5D	PCINT4 × PCIE0 + ADC4D
DIEOV	PCINT7 × PCIE0	PCINT6 × PCIE0	PCINT5 × PCIE0	PCINT4 × PCIE0
DI	PCINT7 INPUT	PCINT6 INPUT	PCINT5 INPUT	PCINT4 INPUT
AIO	ADC7 INPUT	ADC6 INPUT	ADC5 INPUT	ADC4 INPUT

Table 14-5. Overriding Signals for Alternate Functions in PA3:PA0

Signal Name	PA3/ADC3/PCINT3	PA2/ADC2/PCINT2	PA1/ADC1/PCINT1	PA0/ADC0/PCINT0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	PCINT3 × PCIE0 + ADC3D	PCINT2 × PCIE0 + ADC2D	PCINT1 × PCIE0 + ADC1D	PCINT0 × PCIE0 + ADC0D
DIEOV	PCINT3 × PCIE0	PCINT2 × PCIE0	PCINT1 × PCIE0	PCINT0 × PCIE0
DI	PCINT3 INPUT	PCINT2 INPUT	PCINT1 INPUT	PCINT0 INPUT
AIO	ADC3 INPUT	ADC2 INPUT	ADC1 INPUT	ADC0 INPUT

- **SCL/PCINT16 – Port C, Bit 0**

SCL, 2-wire serial bus clock line.

PCINT16, pin change interrupt source 16: The PC0 pin can serve as an external interrupt source.

Table 14-10 and Table 14-11 relate the alternate functions of port C to the overriding signals shown in Figure 14-5 on page 62.

Table 14-10. Overriding Signals for Alternate Functions in PC7:PC4

Signal Name	PC7/TOSC2/PCINT23	PC6/TOSC1/PCINT22	PC5/TDI/PCINT21	PC4/TDO/PCINT20
PUOE	AS2 × $\overline{\text{EXCLK}}$	AS2	JTAGEN	JTAGEN
PUOV	0	0	1	1
DDOE	AS2 × $\overline{\text{EXCLK}}$	AS2	JTAGEN	JTAGEN
DDOV	0	0	0	SHIFT_IR + SHIFT_DR
PVOE	0	0	0	JTAGEN
PVOV	0	0	0	TDO
DIEOE	AS2 × $\overline{\text{EXCLK}}$ + PCINT23 × PCIE2	AS2 + PCINT22 × PCIE2	JTAGEN + PCINT21 × PCIE2	JTAGEN + PCINT20 × PCIE2
DIEOV	$\overline{\text{AS2}}$	EXCLK + $\overline{\text{AS2}}$	$\overline{\text{JTAGEN}}$	$\overline{\text{JTAGEN}}$
DI	PCINT23 INPUT	PCINT22 INPUT	PCINT21 INPUT	PCINT20 INPUT
AIO	T/C2 OSC OUTPUT	T/C2 OSC INPUT	TDI INPUT	–

Table 14-11. Overriding Signals for Alternate Functions in PC3:PC0

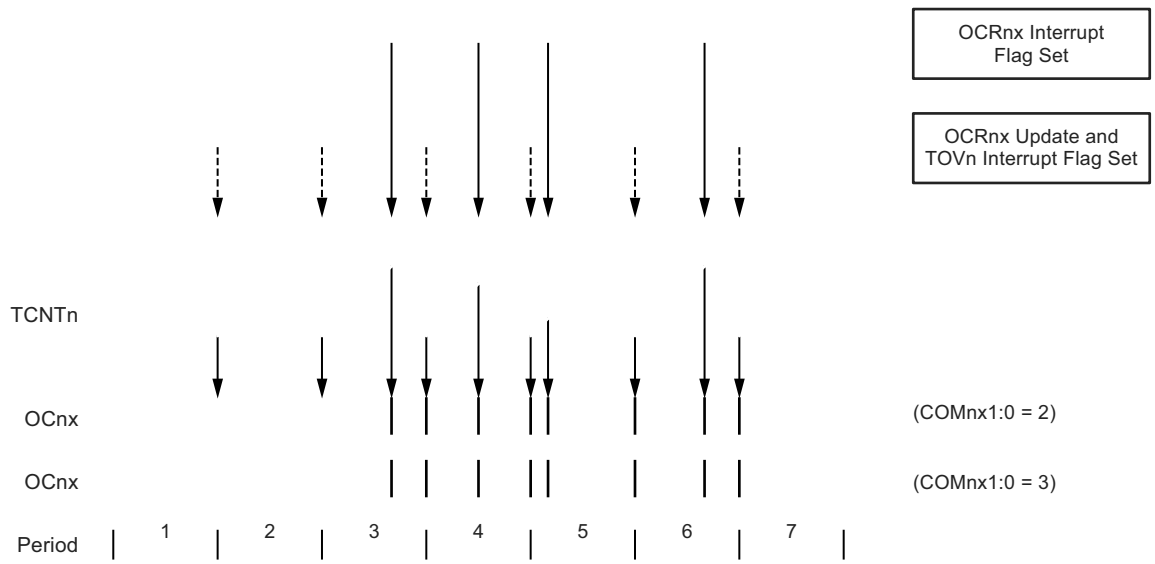
Signal Name	PC3/TMS/PCINT19	PC2/TCK/PCINT18	PC1/SDA/PCINT17	PC0/SCL/PCINT16
PUOE	JTAGEN	JTAGEN	TWEN	TWEN
PUOV	1	1	PORTC1 × $\overline{\text{PUD}}$	PORTC0 × $\overline{\text{PUD}}$
DDOE	JTAGEN	JTAGEN	TWEN	TWEN
DDOV	0	0	0	0
PVOE	0	0	TWEN	TWEN
PVOV	0	0	SDA OUT	SCL OUT
DIEOE	JTAGEN + PCINT19 × PCIE2	JTAGEN + PCINT18 × PCIE2	PCINT17 × PCIE2	PCINT16 × PCIE2
DIEOV	$\overline{\text{JTAGEN}}$	$\overline{\text{JTAGEN}}$	1	1
DI	PCINT19 INPUT	PCINT18 INPUT	PCINT17 INPUT	PCINT16 INPUT
AIO	TMS INPUT	TCK INPUT	SDA INPUT	SCL INPUT

15.7.3 Fast PWM Mode

The fast pulse width modulation or fast PWM mode (WGM02:0 = 3 or 7) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. TOP is defined as 0xFF when WGM2:0 = 3, and OCR0A when WGM2:0 = 7. In non-inverting compare output mode, the output compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at BOTTOM. In inverting compare output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 15-6. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-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 15-6. Fast PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV0) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

In fast 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 and an inverted PWM output can be generated by setting the COM0x1:0 to three: Setting the COM0A1:0 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 15-3 on page 85). 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 setting (or clearing) the OC0x register at the compare match between OCR0x and TCNT0, and clearing (or setting) the OC0x register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

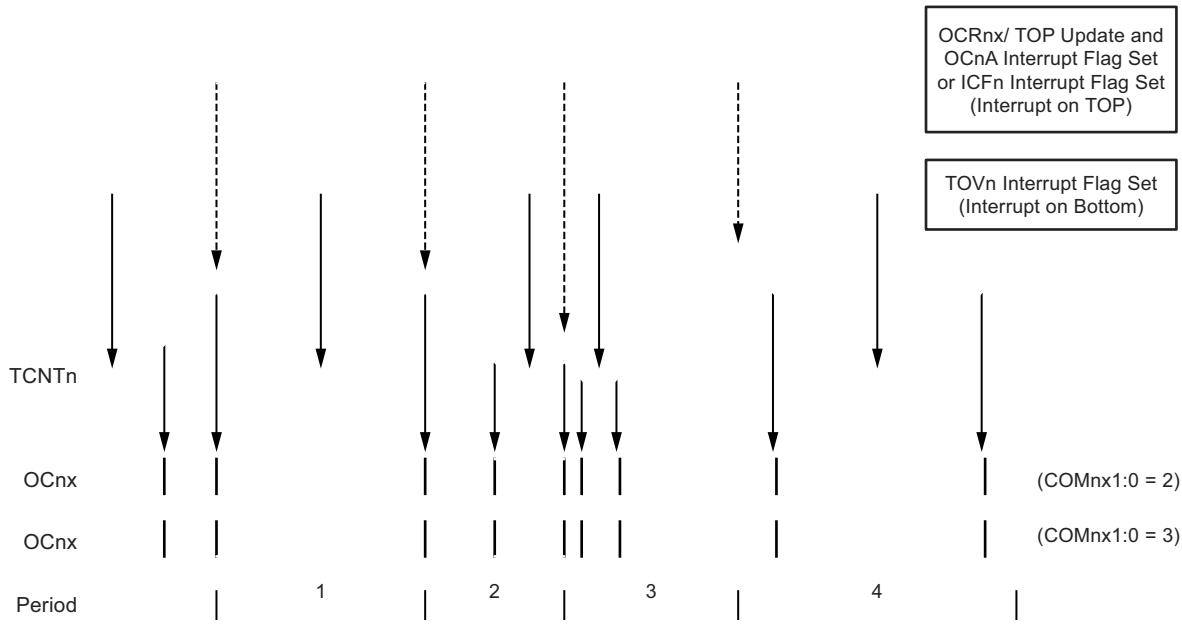
The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{\mathrm{clk\_I/O}}}{N \cdot 256}$$

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

The extreme values for the OCR0A register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM0A1:0 bits.)

Figure 16-8. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOVn) is set each time the counter reaches BOTTOM. When either OCRnA or ICRn is used for defining the TOP value, the OCnA or ICFn flag is set accordingly at the same timer clock cycle as the OCRnx registers are updated with the double buffer value (at TOP). The interrupt flags can be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNTn and the OCRnx. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCRnx registers are written. As the third period shown in Figure 16-8 illustrates, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an unsymmetrical output. The reason for this can be found in the time of update of the OCRnx register. Since the OCRnx update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values differ the two slopes of the period will differ in length. The difference in length gives the unsymmetrical result on the output.

It is recommended to use the phase and frequency correct mode instead of the phase correct mode when changing the TOP value while the Timer/Counter is running. When using a static TOP value there are practically no differences between the two modes of operation.

In phase correct PWM mode, the compare units allow generation of PWM waveforms on the OCnx pins. Setting the COMnx1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COMnx1:0 to three (See Table on page 110). The actual OCnx value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OCnx). The PWM waveform is generated by setting (or clearing) the OCnx register at the compare match between OCRnx and TCNTn when the counter increments, and clearing (or setting) the OCnx register at compare match between OCRnx and TCNTn when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{\text{clk\_I/O}}}{2 \cdot N \cdot TOP}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCRnx register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCRnx is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 11) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

Table 17-4 shows the COM2A1:0 bit functionality when the WGM22:0 bits are set to phase correct PWM mode.

Table 17-4. Compare Output Mode, Phase Correct PWM Mode(1)

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	WGM22 = 0: Normal port operation, OC2A disconnected. WGM22 = 1: Toggle OC2A on compare match.
1	0	Clear OC2A on compare match when up-counting. Set OC2A on compare match when down-counting.
1	1	Set OC2A on compare match when up-counting. Clear OC2A on compare match when down-counting.

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 17.7.4 “Phase Correct PWM Mode” on page 127 for more details.

- **Bits 5:4 – COM2B1:0: Compare Match Output B Mode**

These bits control the output compare pin (OC2B) behavior. If one or both of the COM2B1:0 bits are set, the OC2B 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 OC2B pin must be set in order to enable the output driver.

When OC2B is connected to the pin, the function of the COM2B1:0 bits depends on the WGM22:0 bit setting. Table 17-5 shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to a normal or CTC mode (non-PWM).

Table 17-5. Compare Output Mode, non-PWM Mode

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Toggle OC2B on compare match
1	0	Clear OC2B on compare match
1	1	Set OC2B on compare match

Table 17-6 shows the COM2B1:0 bit functionality when the WGM22:0 bits are set to fast PWM mode.

Table 17-6. Compare Output Mode, Fast PWM Mode(1)

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Reserved
1	0	Clear OC2B on compare match, set OC2B at BOTTOM, (non-inverting mode).
1	1	Set OC2B on compare match, clear OC2B at BOTTOM, (inverting mode).

Note: 1. A special case occurs when OCR2B equals TOP and COM2B1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See Section 17.7.3 “Fast PWM Mode” on page 126 for more details.

After a repeated START condition (state 0x10) the 2-wire serial interface can access the same slave again, or a new slave without transmitting a STOP condition. Repeated START enables the master to switch between slaves, master transmitter mode and master receiver mode without losing control over the bus.

Table 21-4. Status codes for Master Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	Next Action Taken by TWI Hardware
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+R or	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
		Load SLA+W	0	0	1	X	SLA+W will be transmitted logic will switch to master transmitter mode
0x38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	X	2-wire serial bus will be released and not addressed slave mode will be entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free
0x40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
0x48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
0x50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
0x58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset

Table 21-5. Status Codes for Slave Receiver Mode (Continued)

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
0x90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
0x98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
0xA0	A STOP condition or repeated START condition has been received while still addressed as slave	No action	0	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
			1	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
			1	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

Figure 23-6. ADC Timing Diagram, Auto Triggered Conversion

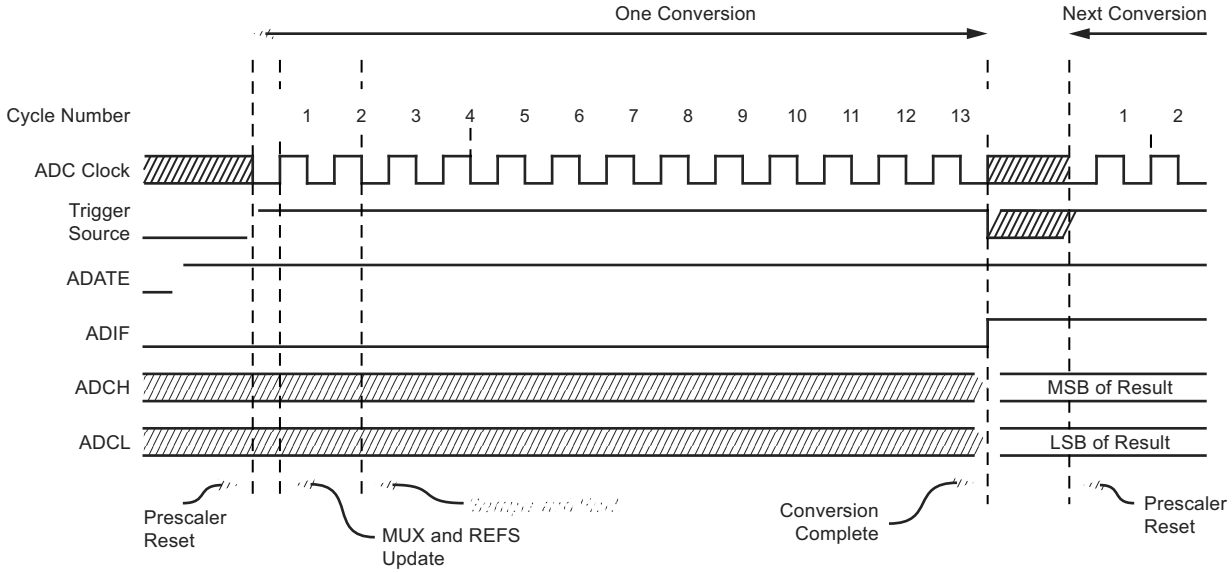


Figure 23-7. ADC Timing Diagram, Free Running Conversion

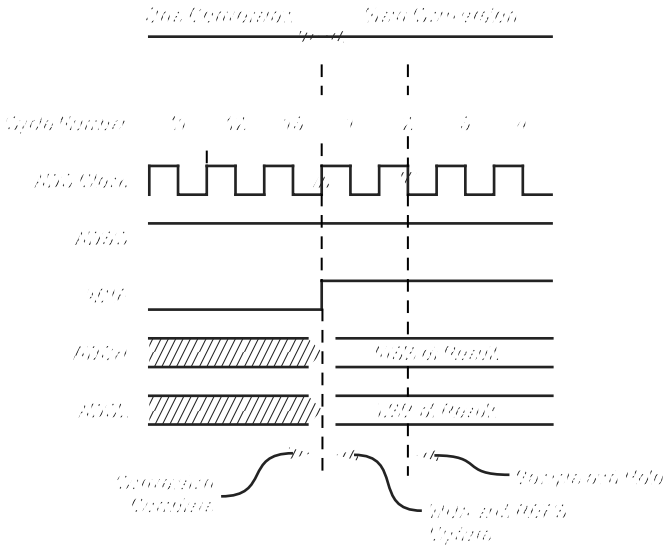


Table 23-1. ADC Conversion Time

Condition	Sample and Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	14.5	25
Normal conversions, single ended	1.5	13
Auto triggered conversions	2	13.5
Normal conversions, differential	1.5/2.5	13/14

Table 23-2. Correlation between Input Voltage and Output Codes

V_{ADCn}	Read code	Corresponding Decimal Value
V_{ADCm} + V_{REF}/GAIN	0x1FF	511
V_{ADCm} + 0.999 V_{REF}/GAIN	0x1FF	511
V_{ADCm} + 0.998 V_{REF}/GAIN	0x1FE	510
...	...	...
V_{ADCm} + 0.001 V_{REF}/GAIN	0x001	1
V_{ADCm}	0x000	0
V_{ADCm} – 0.001 V_{REF}/GAIN	0x3FF	-1
...	...	...
V_{ADCm} – 0.999 V_{REF}/GAIN	0x201	–511
V_{ADCm} – V_{REF}/GAIN	0x200	–512

Example: ADMUX = 0xED (ADC3 - ADC2, 10x gain, 2.56V reference, left adjusted result)
Voltage on ADC3 is 300mV, voltage on ADC2 is 500mV.
ADCR = 512 × 10 × (300 – 500) / 2560 = –400 = 0x270
ADCL will thus read 0x00, and ADCH will read 0x9C. Writing zero to ADLAR right adjusts the result: ADCL = 0x70, ADCH = 0x02.

23.9 Register Description

23.9.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	**REFS1**	**REFS0**	**ADLAR**	**MUX4**	**MUX3**	**MUX2**	**MUX1**	**MUX0**	**ADMUX**
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	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 23-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 23-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{REF} turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Internal 1.1V voltage reference with external capacitor at AREF pin
1	1	Internal 2.56V voltage reference with external capacitor at AREF pin

Note: If differential channels are selected, only 2.56V should be used as internal voltage reference.

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC data register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC data register immediately, regardless of any ongoing conversions. For a complete description of this bit, see Section 23.9.3 “ADCL and ADCH – The ADC Data Register” on page 223.

25.5 Boundary-scan Chain

The boundary-scan chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connection.

25.5.1 Scanning the Digital Port Pins

Figure 25-3 on page 235 shows the boundary-scan cell for a bi-directional port pin. The pull-up function is disabled during boundary-scan when the JTAG IC contains EXTEST or SAMPLE_PRELOAD. The cell consists of a bi-directional pin cell that combines the three signals output control - OCxn, output data - ODxn, and input data - IDxn, into only a two-stage shift register. The port and pin indexes are not used in the following description

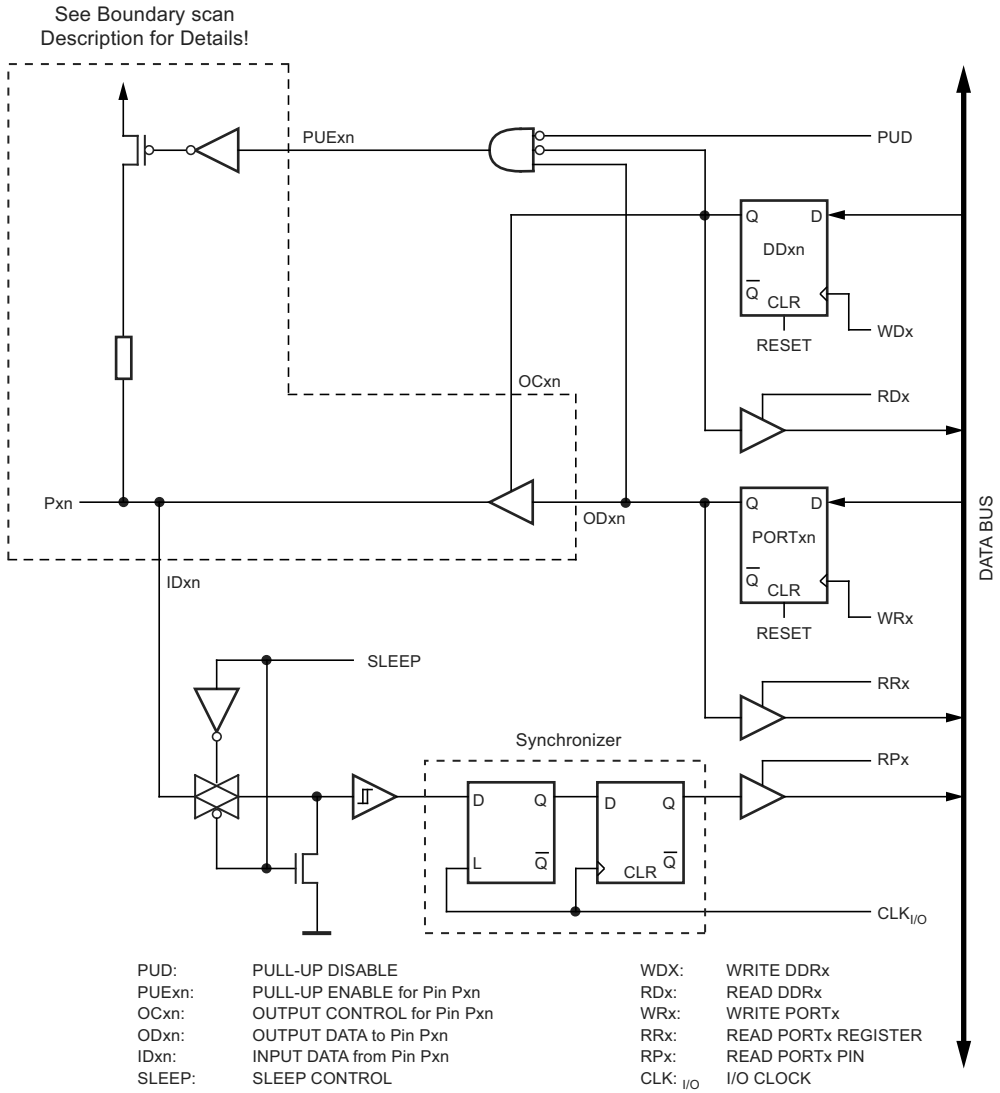
The boundary-scan logic is not included in the figures in the datasheet. Figure 25-4 on page 236 shows a simple digital port pin as described in the Section 14. “I/O-Ports” on page 57. The boundary-scan details from Figure 25-3 on page 235 replaces the dashed box in Figure 25-4 on page 236.

When no alternate port function is present, the input data - ID - corresponds to the PINxn register value (but ID has no synchronizer), output data corresponds to the PORT register, output control corresponds to the data direction - DD register, and the pull-up enable - PUExn - corresponds to logic expression $\overline{PUD} \cdot \overline{DDxn} \cdot PORTxn$.

Digital alternate port functions are connected outside the dotted box in Figure 25-4 on page 236 to make the scan chain read the actual pin value. For analog function, there is a direct connection from the external pin to the analog circuit. There is no scan chain on the interface between the digital and the analog circuitry, but some digital control signal to analog circuitry are turned off to avoid driving contention on the pads.

When JTAG IR contains EXTEST or SAMPLE_PRELOAD the clock is not sent out on the port pins even if the CKOUT fuse is programmed. Even though the clock is output when the JTAG IR contains SAMPLE_PRELOAD, the clock is not sampled by the boundary scan.

Figure 25-4. General Port Pin Schematic Diagram



26.8.10 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 26-5 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.

Table 26-5. 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 3V	0x0001
RC oscillator calibration byte 5V	0x0003

Note: All other addresses are reserved for future use.

26.8.11 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.

26.8.12 Programming Time for Flash when Using SPM

The calibrated RC oscillator is used to time flash accesses. Table 26-6 shows the typical programming time for flash accesses from the CPU.

Table 26-6. SPM Programming Time(1)

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

Note: 1. Minimum and maximum programming times is per individual operation.

- **Bit 3 – BLBSET: Boot Lock Bit Set**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles sets boot lock bits, according to the data in R0. The data in R1 and the address in the Z-pointer are ignored. The BLBSET bit will automatically be cleared upon completion of the lock bit set, or if no SPM instruction is executed within four clock cycles.

An (E)LPM instruction within three cycles after BLBSET and SPMEN are set in the SPMCSR register, will read either the lock bits or the fuse bits (depending on Z0 in the Z-pointer) into the destination register. See Section 26.8.9 “Reading the Fuse and Lock Bits from Software” on page 247 for details.

- **Bit 2 – PGWRT: Page Write**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes page write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a page write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

- **Bit 1 – PGERS: Page Erase**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes page erase. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a page erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

- **Bit 0 – SPMEN: Store Program Memory Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either RWWSRE, BLBSET, PGWRT’ or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z-pointer. The LSB of the Z-pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. During page erase and page write, the SPMEN bit remains high until the operation is completed.

Writing any other combination than “10001”, “01001”, “00101”, “00011” or “00001” in the lower five bits will have no effect.

Note: Only one SPM instruction should be active at any time.

27.10.10 Programming Command Register

The programming command register is a 15-bit register. This register is used to serially shift in programming commands, and to serially shift out the result of the previous command, if any. The JTAG programming instruction Set is shown in Table 27-18 on page 279. The state sequence when shifting in the programming commands is illustrated in Figure 27-16 on page 282.

Figure 27-15. Programming Command Register

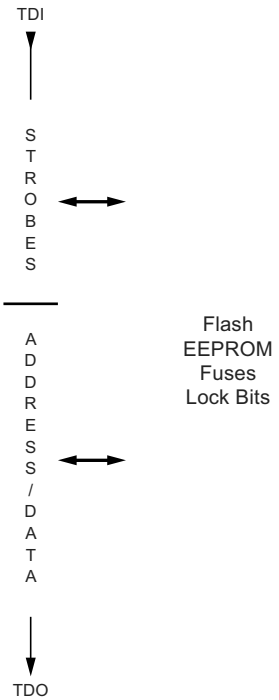


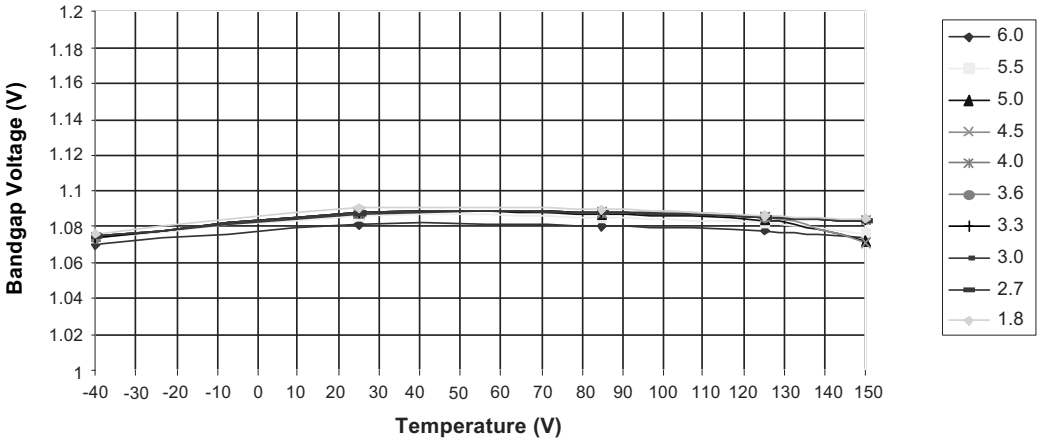
Table 27-18. JTAG Programming Instruction

Set a = address high bits, b = address low bits, c = address extended bits, H = 0 - Low byte, 1 - High byte, o = data out, i = data in, x = don't care

Instruction	TDI Sequence	TDO Sequence	Notes
1a. Chip erase	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
1b. Poll for chip erase complete	0110011_10000000	xxxxxox_xxxxxxxx	(2)
2a. Enter flash write	0100011_00010000	xxxxxxx_xxxxxxxx	
2b. Load address extended high byte	0001011_cccccccc	xxxxxxx_xxxxxxxx	(10)
2c. Load address high byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	
2d. Load address low byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
2e. Load data low byte	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
2f. Load data high byte	0010111_iiiiiiii	xxxxxxx_xxxxxxxx	
2g. Latch data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2h. Write flash page	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2i. Poll for page write complete	0110111_00000000	xxxxxox_xxxxxxxx	(2)
3a. Enter flash read	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Load address extended high byte	0001011_cccccccc	xxxxxxx_xxxxxxxx	(10)
3c. Load address high byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	
3d. Load address low byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
3e. Read data low and high byte	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo xxxxxxx_oooooooo	Low byte High byte
4a. Enter EEPROM write	0100011_00010001	xxxxxxx_xxxxxxxx	
4b. Load address high byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(10)
4c. Load address low byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
4d. Load data byte	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	

Notes:
1. This command sequence is not required if the seven MSB are correctly set by the previous command sequence (which is normally the case).
2. Repeat until **o** = "1".
3. Set bits to "0" to program the corresponding Fuse, "1" to unprogram the fuse.
4. Set bits to "0" to program the corresponding lock bit, "1" to leave the lock bit unchanged.
5. "0" = programmed, "1" = unprogrammed.
6. The bit mapping for fuses extended byte is listed in Table 27-3 on page 256
7. The bit mapping for fuses high byte is listed in Table 27-4 on page 257
8. The bit mapping for fuses low byte is listed in Table 27-5 on page 257
9. The bit mapping for lock bits byte is listed in Table 27-1 on page 255
10. Address bits exceeding PCMSB and EEAMSB (Table 27-7 on page 258 and Table 27-8 on page 258) are don't care
11. All TDI and TDO sequences are represented by binary digits (0b...).

Figure 29-45. ATmega324P-B: Bandgap Voltage versus Temperature



29.2.8 Internal Oscillator Speed

Figure 29-46. ATmega324P-B: Watchdog Oscillator Frequency versus Temperature

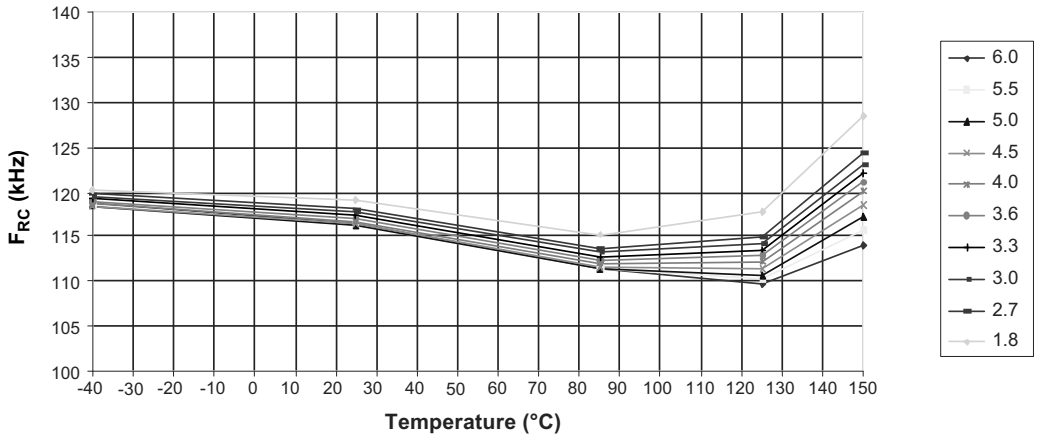
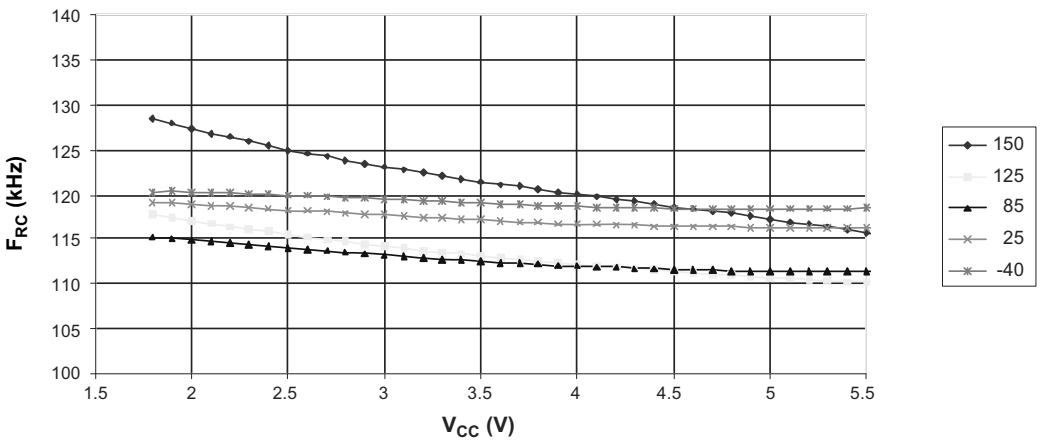


Figure 29-47. ATmega324P-B: Watchdog Oscillator Frequency versus V_{CC}



30. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xDF)	Reserved	-	-	-	-	-	-	-	-	
(0xDE)	Reserved	-	-	-	-	-	-	-	-	
(0xDD)	Reserved TCCR2B	-	-	-	-	-	-	-	-	
(0xDC)	Reserved	-	-	-	-		-	-	-	
(0xDB)	Reserved	-	-	-	-	-	-	-	-	
(0xDA)	Reserved	-	-	-	-	-	-	-	-	
(0xD9)	Reserved	-	-	-	-	-	-	-	-	
(0xD8)	Reserved	-	-	-	-	-	-	-	-	
(0xD7)	Reserved	-	-	-	-	-	-	-	-	
(0xD6)	Reserved	-	-	-	-	-	-	-	-	
(0xD5)	Reserved	-	-	-	-	-	-	-	-	
(0xD4)	Reserved	-	-	-	-	-	-	-	-	
(0xD3)	Reserved	-	-	-	-	-	-	-	-	
(0xD2)	Reserved	-	-	-	-	-	-	-	-	
(0xD1)	Reserved	-	-	-	-	-	-	-	-	
(0xD0)	Reserved	-	-	-	-	-	-	-	-	
(0xCF)	Reserved	-	-	-	-	-	-	-	-	
(0xCE)	UDR1	USART1 I/O data register USART0 I/O data register								163
(0xCD)	UBRR1H	-	-	-	-	USART1 baud rate register high byte				166/175
(0xCC)	UBRR1L	USART1 baud rate Register low byte								166/175
(0xCB)	Reserved	-	-	-	-	-	-	-	-	
(0xCA)	UCSR1C	UMSEL11	UMSEL10	UPM11	UPM10	USBS1	UCSZ11/UDORD0(5)	UCSZ10/UCPHA0(5)	UCPOL1	165/175
(0xC9)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	164/174
(0xC8)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	163/173
(0xC7)	Reserved	-	-	-	-	-	-	-	-	
(0xC6)	UDR0	USART0 I/O data register USART0 I/O data register								163
(0xC5)	UBRR0H	-	-	-	-	USART0 baud rate register high byte				166/175
(0xC4)	UBRR0L	USART0 baud rate register low byte								166/175
(0xC3)	Reserved	-	-	-	-	-	-	-	-	
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01/UDORD0(5)	UCSZ00/UCPHA0(5)	UCPOL0	165/175
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	164/174
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	163/173
(0xBF)	Reserved	-	-	-	-	-	-	-	-	
(0xBE)	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 $00 - $1F 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 the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. 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 $00 - $3F must be used. When addressing I/O registers as data space using LD and ST instructions, $20 must be added to these addresses.
   The ATmega164P-B/324P-B/644P-B 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 $60 - $FF, only the ST/STS/STD and LD/LDS/LDD instructions can be used.
5. USART in SPI master mode.

31. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
LPM	Rd, Z	Load program memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load program memory and post-inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store program memory	(Z) ← R1:R0	None	-
IN	Rd, P	In port	Rd ← P	None	1
OUT	P, Rr	Out port	P ← Rr	None	1
PUSH	Rr	Push register on stack	STACK ← Rr	None	2
POP	Rd	Pop register from stack	Rd ← STACK	None	2
MCU Control Instructions					
NOP		No operation		None	1
SLEEP		Sleep	(see specific descr. for sleep function)	None	1
WDR		Watchdog reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For on-chip debug only	None	N/A