Microchip Technology - ATMEGA3250P-20AU Datasheet

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	20MHz
Connectivity	SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	69
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 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega3250p-20au

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_write
  ; Set up address (r18:r17) in address register
  out EEARH, r18
  out EEARL, r17
  ; Write data (r16) to Data Register
  out EEDR,r16
  ; Write logical one to EEMWE
  sbi EECR,EEMWE
  ; Start eeprom write by setting EEWE
  sbi EECR,EEWE
  ret
```

C Code Example

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up address and Data Registers */
  EEAR = uiAddress;
  EEDR = ucData;
  /* Write logical one to EEMWE */
  EECR |= (1<<EEMWE);
  /* Start eeprom write by setting EEWE */
  EECR |= (1<<EEWE);
}
```

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

symmetry around BOTTOM the OCn value at MAX must correspond to the result of an up-counting Compare Match.

- The timer starts counting from a value higher than the one in OCR0A, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.

14.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 4 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 4. Timer/Counter Timing Diagram, no Prescaling

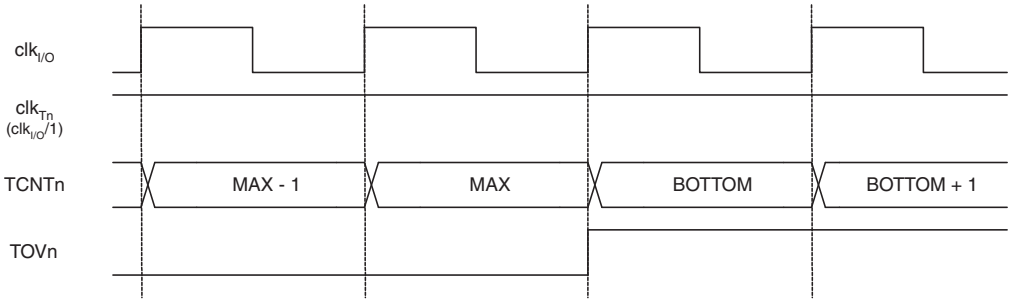


Figure 5 shows the same timing data, but with the prescaler enabled.

Figure 5. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

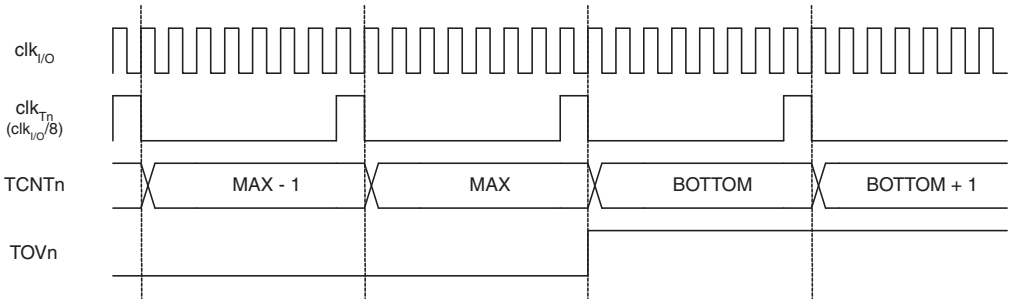
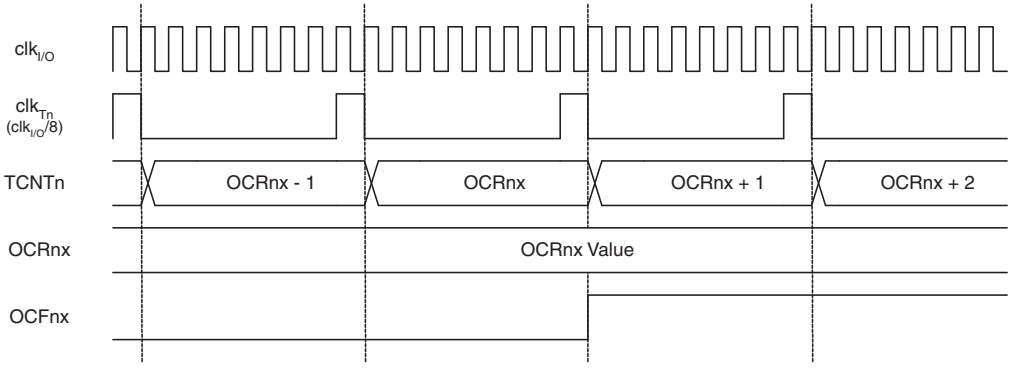


Figure 6 shows the setting of OCF0A in all modes except CTC mode.

Figure 6. Timer/Counter Timing Diagram, Setting of OCF0A, with Prescaler ($f_{clk\_I/O}/8$)



15. 16-bit Timer/Counter1

15.1 Features

- **True 16-bit Design (i.e., Allows 16-bit PWM)**
- **Two independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **One Input Capture Unit**
- **Input Capture Noise Canceler**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **External Event Counter**
- **Four independent interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)**

15.2 Overview

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement. Most register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, and a lower case “x” replaces the Output Compare unit. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT1 for accessing Timer/Counter1 counter value and so on.

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 15-1. For the actual placement of I/O pins, refer to ”Pinout ATmega3250P” on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the ”Register Description” on page 128.

The PRTIM1 bit in ”PRR – Power Reduction Register” on page 44 must be written to zero to enable the Timer/Counter1 module.

the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to do an atomic read of the TCNT1 Register contents. Reading any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

Assembly Code Example[1]

```
TIM16_ReadTCNT1:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Read TCNT1 into r17:r16
  in  r16,TCNT1L
  in  r17,TCNT1H
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

C Code Example[1]

```
unsigned int TIM16_ReadTCNT1( void )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  __disable_interrupt();
  /* Read TCNT1 into i */
  i = TCNT1;
  /* Restore global interrupt flag */
  SREG = sreg;
  return i;
}
```

Note: 1. See Section “5.” on page 10.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

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 OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to three (See Table 1 on page 129). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x Register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

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

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

The extreme values for the OCR1x Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x 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.

15.9.5 Phase and Frequency Correct PWM Mode

The *phase and frequency correct Pulse Width Modulation,* or phase and frequency correct PWM mode (WGM13:0 = 8 or 9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while counting up, and set on the compare match while counting down. In inverting Compare Output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCR1x Register is updated by the OCR1x Buffer Register, (see Figure 15-8 and Figure 15-9).

The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and

Using the ICR1 Register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A Register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to three (See Table 1 on page 129). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x Register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

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

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

The extreme values for the OCR1x Register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be set to 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 = 9) and COM1A1:0 = 1, the OC1A output will toggle with a 50% duty cycle.

15.10 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T1}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set, and when the OCR1x Register is updated with the OCR1x buffer value (only for modes utilizing double buffering). Figure 15-10 shows a timing diagram for the setting of OCF1x.

Figure 15-10. Timer/Counter Timing Diagram, Setting of OCF1x, no Prescaling

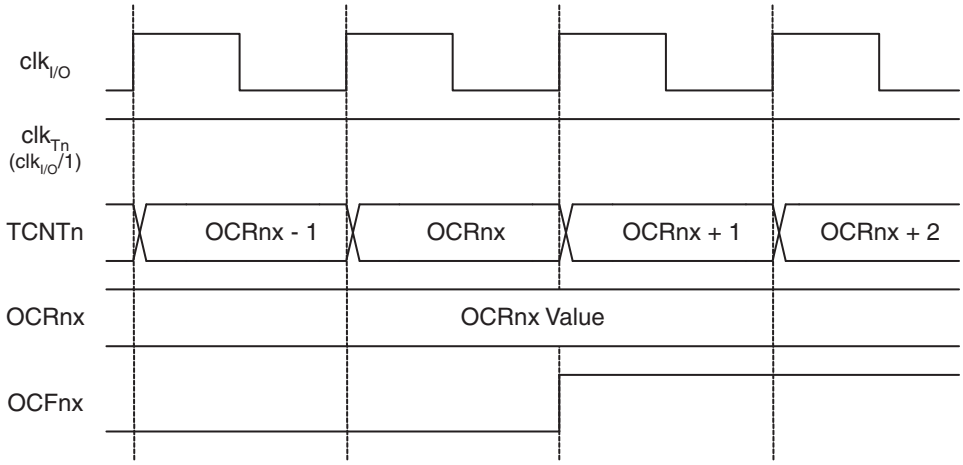


Figure 15-11 shows the same timing data, but with the prescaler enabled.

16. 8-bit Timer/Counter2 with PWM and Asynchronous Operation

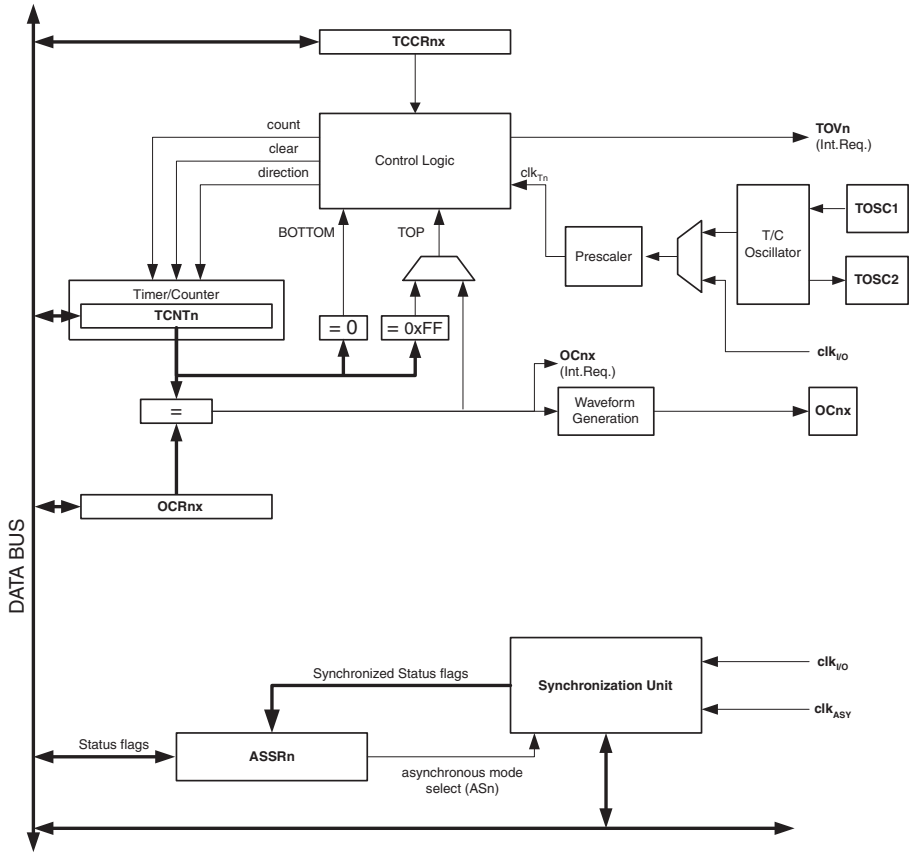
16.1 Features

- **Single Compare Unit Counter**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Frequency Generator**
- **10-bit Clock Prescaler**
- **Overflow and Compare Match Interrupt Sources (TOV2 and OCF2A)**
- **Allows Clocking from External 32 kHz Watch Crystal Independent of the I/O Clock**

16.2 Overview

Timer/Counter2 is a general purpose, single compare unit, 8-bit Timer/Counter module. A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 16-1. For the actual placement of I/O pins, refer to ”Pinout ATmega3250P” on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the ”Register Description” on page 149.

Figure 16-1. 8-bit Timer/Counter Block Diagram



16.2.1 Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2A) are 8-bit registers. Interrupt request (shorten as Int.Req.) signals are all visible in the Timer Interrupt Flag Register (TIFR2). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK2). TIFR2 and TIMSK2 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or asynchronously clocked from the TOSC1/2 pins, as detailed later in this section. The asynchronous operation is controlled by the Asynchronous Status Register (ASSR). The Clock Select logic block controls which clock source the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T2}).

The double buffered Output Compare Register (OCR2A) is compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pin (OC2A). See Section “16.5” on page 137. for details. The compare match event will also set the Compare Flag (OCF2A) which can be used to generate an Output Compare interrupt request.

16.2.2 Definitions

Many register and bit references in this document are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 2. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT2 for accessing Timer/Counter2 counter value and so on.

The definitions in Table 16-1 are also used extensively throughout the section.

Table 16-1. Definitions of Timer/Counter values.

BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2A Register. The assignment is dependent on the mode of operation.

16.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source clk_{T2} is by default equal to the MCU clock, $clk_{I/O}$. When the AS2 bit in the ASSR Register is written to logic one, the clock source is taken from the Timer/Counter Oscillator connected to TOSC1 and TOSC2. For details on asynchronous operation, see ”ASSR – Asynchronous Status Register” on page 151. For details on clock sources and prescaler, see ”Timer/Counter Prescaler” on page 148.

symmetry around BOTTOM the OCn value at MAX must correspond to the result of an up-counting Compare Match.

- The timer starts counting from a value higher than the one in OCR2A, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.

16.8 Timer/Counter Timing Diagrams

The following figures show the Timer/Counter in synchronous mode, and the timer clock (clk_{T2}) is therefore shown as a clock enable signal. In asynchronous mode, $clk_{I/O}$ should be replaced by the Timer/Counter Oscillator clock. The figures include information on when Interrupt Flags are set. Figure 16-8 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 16-8. Timer/Counter Timing Diagram, no Prescaling

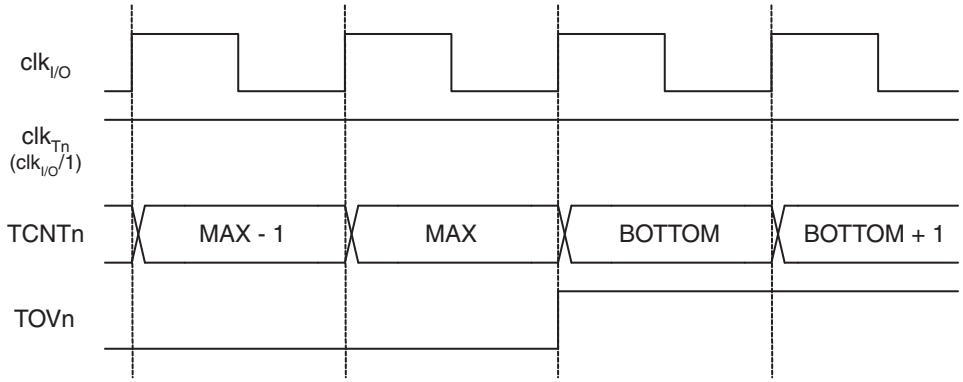


Figure 16-9 shows the same timing data, but with the prescaler enabled.

Figure 16-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

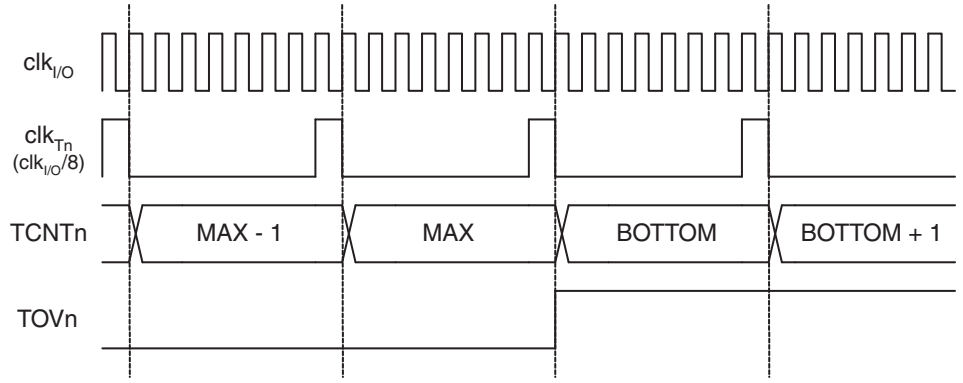
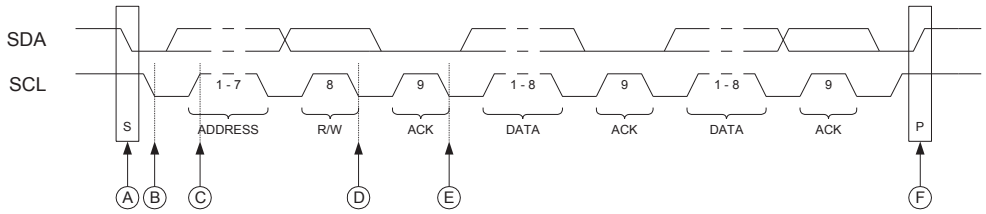


Figure 16-10 shows the setting of OCF2A in all modes except CTC mode.

Figure 19-5. Two-wire Mode, Typical Timing Diagram

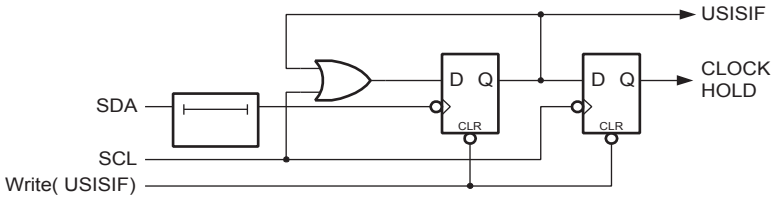


Referring to the timing diagram (Figure 19-5.), a bus transfer involves the following steps:

1. The a start condition is generated by the Master by forcing the SDA low line while the SCL line is high (A). SDA can be forced low either by writing a zero to bit 7 of the Shift Register, or by setting the corresponding bit in the PORT Register to zero. Note that the Data Direction Register bit must be set to one for the output to be enabled. The slave device's start detector logic (Figure 19-6) detects the start condition and sets the USISIF Flag. The flag can generate an interrupt if necessary.
2. In addition, the start detector will hold the SCL line low after the Master has forced an negative edge on this line (B). This allows the Slave to wake up from sleep or complete its other tasks before setting up the Shift Register to receive the address. This is done by clearing the start condition flag and reset the counter.
3. The Master set the first bit to be transferred and releases the SCL line (C). The Slave samples the data and shift it into the Serial Register at the positive edge of the SCL clock.
4. After eight bits are transferred containing slave address and data direction (read or write), the Slave counter overflows and the SCL line is forced low (D). If the slave is not the one the Master has addressed, it releases the SCL line and waits for a new start condition.
5. If the Slave is addressed it holds the SDA line low during the acknowledgment cycle before holding the SCL line low again (i.e., the Counter Register must be set to 14 before releasing SCL at (D)). Depending of the R/W bit the Master or Slave enables its output. If the bit is set, a master read operation is in progress (i.e., the slave drives the SDA line) The slave can hold the SCL line low after the acknowledge (E).
6. Multiple bytes can now be transmitted, all in same direction, until a stop condition is given by the Master (F). Or a new start condition is given.

If the Slave is not able to receive more data it does not acknowledge the data byte it has last received. When the Master does a read operation it must terminate the operation by force the acknowledge bit low after the last byte transmitted.

Figure 19-6. Start Condition Detector, Logic Diagram



- **Bit 4 – ACI: Analog Comparator Interrupt Flag**

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

- **Bit 3 – ACIE: Analog Comparator Interrupt Enable**

When the ACIE bit is written logic one and the I-bit in the Status Register is set, the Analog Comparator interrupt is activated. When written logic zero, the interrupt is disabled.

- **Bit 2 – ACIC: Analog Comparator Input Capture Enable**

When written logic one, this bit enables the Input Capture function in Timer/Counter1 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the Input Capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When written logic zero, no connection between the Analog Comparator and the Input Capture function exists. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the ICIE1 bit in the Timer Interrupt Mask Register (TIMSK1) must be set.

- **Bits 1, 0 – ACIS1, ACIS0: Analog Comparator Interrupt Mode Select**

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 20-2.

Table 20-2. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle.
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge.
1	1	Comparator Interrupt on Rising Output Edge.

When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

20.3.3 DIDR1 – Digital Input Disable Register 1

Bit	7	6	5	4	3	2	1	0	
(0x7F)	–	–	–	–	–	–	AIN1D	AIN0D	DIDR1
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1, 0 – AIN1D, AIN0D: AIN1, AIN0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN1/0 pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN1/0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

Table 23-6. ATmega325P Boundary-scan Order, 64-pin (Continued)

Bit Number	Signal Name	Module
157	PE0.Data	Port E
156	PE0.Control	
155	PE0.Pull-up_Enable	
154	PE1.Data	
153	PE1.Control	
152	PE1.Pull-up_Enable	
151	PE2.Data	
150	PE2.Control	
149	PE2.Pull-up_Enable	
148	PE3.Data	
147	PE3.Control	
146	PE3.Pull-up_Enable	
145	PE4.Data	
144	PE4.Control	
143	PE4.Pull-up_Enable	
142	PE5.Data	
141	PE5.Control	
140	PE5.Pull-up_Enable	
139	PE6.Data	
138	PE6.Control	
137	PE6.Pull-up_Enable	
136	PE7.Data	
135	PE7.Control	
134	PE7.Pull-up_Enable	

Serial Programming Instruction set Table 25-16 and Figure 25-12 on page 290 describes the Instruction set.

Table 25-16. Serial Programming Instruction Set

Instruction/Operation	Instruction Format			
	Byte 1	Byte 2	Byte 3	Byte4
Programming Enable	$AC	$53	$00	$00
Chip Erase (Program Memory/EEPROM)	$AC	$80	$00	$00
Poll RDY/$\overline{BSY}$	$F0	$00	$00	data byte out
Load Instructions				
Load Extended Address byte(1)	$4D	$00	Extended adr	$00
Load Program Memory Page, High byte	$48	$00	adr LSB	high data byte in
Load Program Memory Page, Low byte	$40	$00	adr LSB	low data byte in
Load EEPROM Memory Page (page access)	$C1	$00	0000 00aa / 0000 0aaa	data byte in
Read Instructions				
Read Program Memory, High byte	$28	adr MSB	adr LSB	high data byte out
Read Program Memory, Low byte	$20	adr MSB	adr LSB	low data byte out
Read EEPROM Memory	$A0	0000 00aa / 0000 0aaa	aaaa aaaa	data byte out
Read Lock bits	$58	$00	$00	data byte out
Read Signature Byte	$30	$00	0000 000aa	data byte out
Read Fuse bits	$50	$00	$00	data byte out
Read Fuse High bits	$58	$08	$00	data byte out
Read Extended Fuse Bits	$50	$08	$00	data byte out
Read Calibration Byte	$38	$00	$00	data byte out
Write Instructions(6)				
Write Program Memory Page	$4C	adr MSB	adr LSB	$00
Write EEPROM Memory	$C0	0000 00aa / 0000 0aaa	aaaa aaaa	data byte in
Write EEPROM Memory Page (page access)	$C2	0000 00aa / 0000 0aaa	aaaa aa00 / aaaa a000	$00
Write Lock bits	$AC	$E0	$00	data byte in
Write Fuse bits	$AC	$A0	$00	data byte in
Write Fuse High bits	$AC	$A8	$00	data byte in
Write Extended Fuse Bits	$AC	$A4	$00	data byte in

Table 25-17. JTAG Programming Instruction Set

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

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	xxxxx**o**x_xxxxxxxx	(2)
2a. Enter Flash Write	0100011_00010000	xxxxxxx_xxxxxxxx	
2b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(9)
2c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
2d. Load Data Low Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	
2e. Load Data High Byte	0010111_**iiiiiiii**	xxxxxxx_xxxxxxxx	
2f. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2g. Write Flash Page	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2h. Poll for Page Write Complete	0110111_00000000	xxxxx**o**x_xxxxxxxx	(2)
3a. Enter Flash Read	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(9)
3c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
3d. 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	(9)
4c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
4d. Load Data Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	
4e. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4f. Write EEPROM Page	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4g. Poll for Page Write Complete	0110011_00000000	xxxxx**o**x_xxxxxxxx	(2)
5a. Enter EEPROM Read	0100011_00000011	xxxxxxx_xxxxxxxx	
5b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(9)

Table 25-17. JTAG Programming Instruction Set (Continued)

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

Instruction	TDI Sequence	TDO Sequence	Notes
8f. Read Fuses and Lock Bits	0111010_00000000 0111110_00000000 0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo** xxxxxxx_**oooooooo** xxxxxxx_**oooooooo** xxxxxxx_**oooooooo**	(5) Fuse Ext. byte Fuse High byte Fuse Low byte Lock bits
9a. Enter Signature Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
9b. Load Address Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
9c. Read Signature Byte	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
10a. Enter Calibration Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
10b. Load Address Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
10c. Read Calibration Byte	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
11a. Load No Operation Command	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx 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 25-3 on page 272
7. The bit mapping for Fuses High byte is listed in Table 25-4 on page 273
8. The bit mapping for Fuses Low byte is listed in Table 25-5 on page 273
9. The bit mapping for Lock bits byte is listed in Table 25-1 on page 271
10. Address bits exceeding PCMSB and EEAMSB (Table 25-11 and Table 25-12) are don't care
11. All TDI and TDO sequences are represented by binary digits (0b...).

3. Load the page address using programming instructions 3b and 3c. PCWORD (refer to Table 25-11 on page 276) is used to address within one page and must be written as 0.
4. Enter JTAG instruction PROG_PAGEREAD.
5. Read the entire page (or Flash) by shifting out all instruction words in the page (or Flash), starting with the LSB of the first instruction in the page (Flash) and ending with the MSB of the last instruction in the page (Flash). The Capture-DR state both captures the data from the Flash, and also auto-increments the program counter after each word is read. Note that Capture-DR comes before the shift-DR state. Hence, the first byte which is shifted out contains valid data.
6. Enter JTAG instruction PROG_COMMANDS.
7. Repeat steps 3 to 6 until all data have been read.

25.8.18 Programming the EEPROM

Before programming the EEPROM a Chip Erase must be performed, see “Performing Chip Erase” on page 301.

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable EEPROM write using programming instruction 4a.
3. Load address High byte using programming instruction 4b.
4. Load address Low byte using programming instruction 4c.
5. Load data using programming instructions 4d and 4e.
6. Repeat steps 4 and 5 for all data bytes in the page.
7. Write the data using programming instruction 4f.
8. Poll for EEPROM write complete using programming instruction 4g, or wait for t_{WLRH} (refer to Table 25-13 on page 284).
9. Repeat steps 3 to 8 until all data have been programmed.

Note that the PROG_PAGELOAD instruction can not be used when programming the EEPROM.

25.8.19 Reading the EEPROM

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable EEPROM read using programming instruction 5a.
3. Load address using programming instructions 5b and 5c.
4. Read data using programming instruction 5d.
5. Repeat steps 3 and 4 until all data have been read.

Note that the PROG_PAGEREAD instruction can not be used when reading the EEPROM.

25.8.20 Programming the Fuses

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Fuse write using programming instruction 6a.
3. Load data high byte using programming instructions 6b. A bit value of “0” will program the corresponding fuse, a “1” will unprogram the fuse.
4. Write Fuse High byte using programming instruction 6c.
5. Poll for Fuse write complete using programming instruction 6d, or wait for t_{WLRH} (refer to Table 25-13 on page 284).
6. Load data low byte using programming instructions 6e. A “0” will program the fuse, a “1” will unprogram the fuse.
7. Write Fuse low byte using programming instruction 6f.

26.4 Clock Characterizations

26.4.1 Calibrated Internal RC Oscillator Accuracy

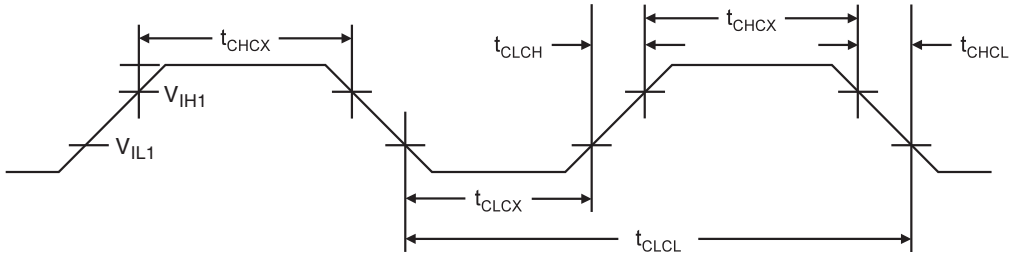
Table 26-1. Calibration Accuracy of Internal RC Oscillator

	Frequency	V_{CC}	Temperature	Calibration Accuracy
Factory Calibration	8.0 MHz	3V	25·C	±10%
User Calibration	7.3 - 8.1 MHz	1.8V - 5.5V(1) 2.7V - 5.5V(2)	-40·C - 85·C	±1%

Notes: 1. Voltage range for ATmega325PV/3250PV/645PV/6450PV.
2. Voltage range for ATmega325P/3250P.

26.4.2 External Clock Drive Waveforms

Figure 26-3. External Clock Drive Waveforms



26.4.3 External Clock Drive

Table 26-2. External Clock Drive

Symbol	Parameter	V_{CC}=1.8-5.5V		V_{CC}=2.7-5.5V		V_{CC}=4.5-5.5V		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$1/t_{CLCL}$	Oscillator Frequency	0	4	0	10	0	20	MHz
t_{CLCL}	Clock Period	1000		100		50		ns
t_{CHCX}	High Time	400		40		20		ns
t_{CLCX}	Low Time	400		40		20		ns
t_{CLCH}	Rise Time		2.0		1.6		0.5	µs
t_{CHCL}	Fall Time		2.0		1.6		0.5	µs
Δt_{CLCL}	Change in period from one clock cycle to the next		2		2		2	%

30.2 ATmega3250P

Speed (MHz)[3]	Power Supply	Ordering Code[2]	Package Type[1]	Operational Range
10	1.8 - 5.5V	ATmega3250PV-10AU	100A	Industrial (-40·C to 85·C)
20	2.7 - 5.5V	ATmega3250P-20AU	100A	Industrial (-40·C to 85·C)

Notes:
1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. For Speed vs. V_{CC} see Figure 26-1 on page 306 and Figure 26-2 on page 306.

Package Type	
100A	100-lead, 14 x 14 x 1.0 mm, 0.5 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)