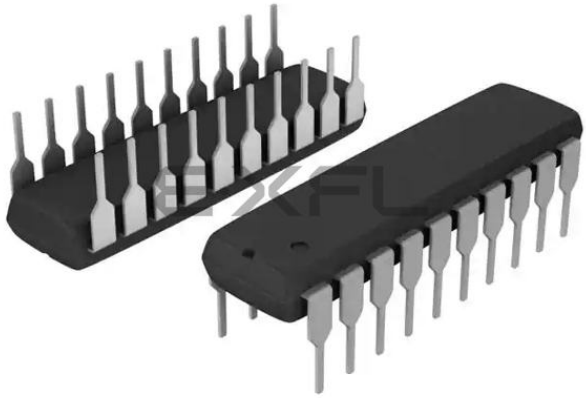


Welcome to E-XFL.COM

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	USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	20-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny861-20pu

5.5.6 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	MSB							LSB	GPIOR1
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	

5.5.7 GPIOR0 – General Purpose I/O Register 0

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	MSB							LSB	GPIOR0
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	

cleared by hardware four cycles after it is written or when the CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period, nor clear the CLKPCE bit.

- **Bits 6:4 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bits 3:0 – CLKPS3:0: Clock Prescaler Select Bits 3 - 0**

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in Table 6-13.

To avoid unintentional changes of clock frequency, a special write procedure must be followed to change the CLKPS bits:

1. Write the Clock Prescaler Change Enable (CLKPCE) bit to one and all other bits in CLKPR to zero.
2. Within four cycles, write the desired value to CLKPS while writing a zero to CLKPCE.

Interrupts must be disabled when changing prescaler setting to make sure the write procedure is not interrupted.

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 eight 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 selcted 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 6-13. 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

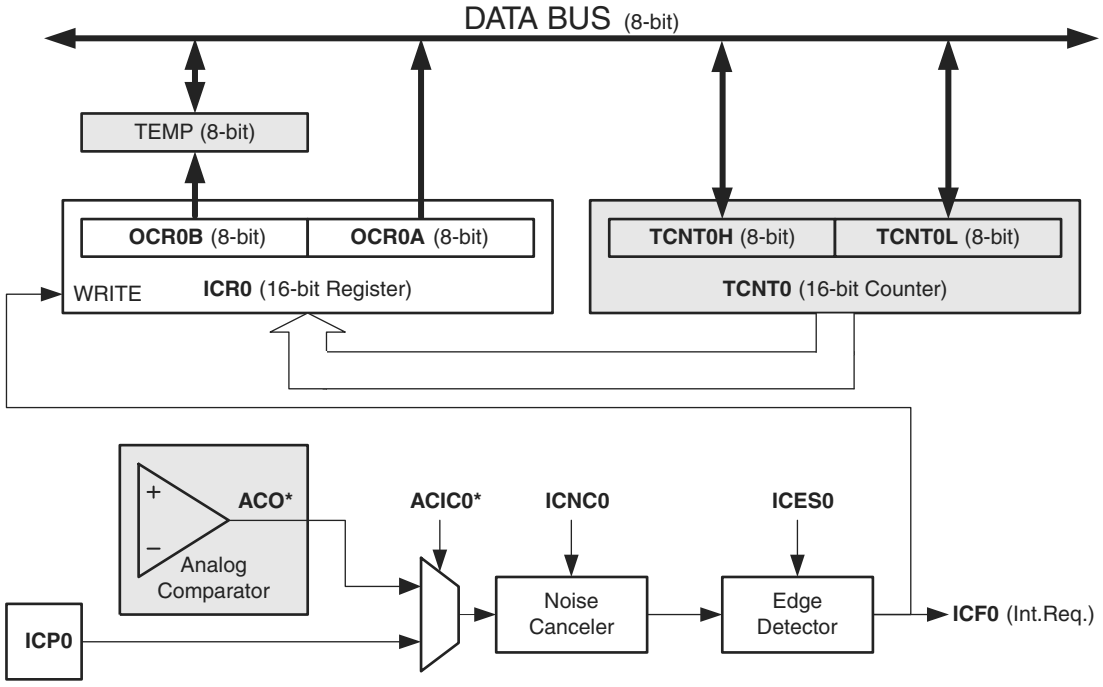
internal clock source, selected by the Clock Select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations. The Timer/Counter Overflow Flag (TOV0) is set when the counter reaches the maximum value and it can be used for generating a CPU interrupt.

11.5 Input Capture Unit

The Timer/Counter incorporates an Input Capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP0 pin or alternatively, via the analog-comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

The Input Capture unit is illustrated by the block diagram shown in Figure 11-4. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded.

Figure 11-4. Input Capture Unit Block Diagram



The Output Compare Register OCR0A is a dual-purpose register that is also used as an 8-bit Input Capture Register ICR0. In 16-bit Input Capture mode the Output Compare Register OCR0B serves as the high byte of the Input Capture Register ICR0. In 8-bit Input Capture mode the Output Compare Register OCR0B is free to be used as a normal Output Compare Register, but in 16-bit Input Capture mode the Output Compare Unit cannot be used as there are no free Output Compare Register(s). Even though the Input Capture register is called ICR0 in this section, it is refering to the Output Compare Register(s).

When a change of the logic level (an event) occurs on the *Input Capture pin* (ICP0), alternatively on the *Analog Comparator output* (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the value of the counter (TCNT0) is written to the *Input Capture Register* (ICR0). The *Input Capture Flag* (ICF0) is set at

The following code examples show how to access the 16-bit timer registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR0A/B registers.

Assembly Code Example

```
...
; Set TCNT0 to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNT0H,r17
out TCNT0L,r16
; Read TCNT0 into r17:r16
in  r16,TCNT0L
in  r17,TCNT0H
...
```

C Code Example

```
unsigned int i;
...
/* Set TCNT0 to 0x01FF */
TCNT0H = 0x01;
TCNT0L = 0xff;

/* Read TCNT0 into i */
i = TCNT0L;
i |= ((unsigned int)TCNT0H << 8);
...
```

Note: See “Code Examples” on page 6.

The assembly code example returns the TCNT0H/L value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit timer registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both 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 TCNT0 register contents. Reading any of the OCR0 register can be done by using the same principle.

Assembly Code Example

```
TIM0_ReadTCNT0:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Read TCNT0 into r17:r16
  in  r16,TCNT0L
  in  r17,TCNT0H
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

C Code Example

```
unsigned int TIM0_ReadTCNT0( void )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  _CLI();
  /* Read TCNT0 into i */
  i = TCNT0L;
  i |= ((unsigned int)TCNT0H << 8);
  /* Restore global interrupt flag */
  SREG = sreg;
  return i;
}
```

Note: See “Code Examples” on page 6.

The assembly code example returns the TCNT0H/L value in the r17:r16 register pair.

11.10 Register Description

11.10.1 TCCR0A – Timer/Counter0 Control Register A

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	TCW0	ICEN0	ICNC0	ICES0	ACIC0	–	–	CTC0	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TCW0: Timer/Counter0 Width**

When this bit is written to one 16-bit mode is selected as described Figure 11-7 on page 79. Timer/Counter0 width is set to 16-bits and the Output Compare Registers OCR0A and OCR0B are combined to form one 16-bit Output Compare Register. Because the 16-bit registers TCNT0H/L and OCR0B/A are accessed by the AVR CPU via the 8-bit data bus, special procedures must be followed. These procedures are described in section “Accessing Registers in 16-bit Mode” on page 80.

- **Bit 6 – ICEN0: Input Capture Mode Enable**

When this bit is written to onem, the Input Capture Mode is enabled.

- **Bit 5 – ICNC0: Input Capture Noise Canceler**

Setting this bit activates the Input Capture Noise Canceler. When the noise canceler is activated, the input from the Input Capture Pin (ICP0) is filtered. The filter function requires four successive equal valued samples of the ICP0 pin for changing its output. The Input Capture is therefore delayed by four System Clock cycles when the noise canceler is enabled.

- **Bit 4 – ICES0: Input Capture Edge Select**

This bit selects which edge on the Input Capture Pin (ICP0) that is used to trigger a capture event. When the ICES0 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES0 bit is written to one, a rising (positive) edge will trigger the capture. When a capture is triggered according to the ICES0 setting, the counter value is copied into the Input Capture Register. The event will also set the Input Capture Flag (ICF0), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

- **Bit 3 - ACIC0: Analog Comparator Input Capture Enable**

When written logic one, this bit enables the input capture function in Timer/Counter0 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/Counter0 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/Counter0 Input Capture interrupt, the TICIE0 bit in the Timer Interrupt Mask Register (TIMSK) must be set.

- **Bits 2:1 – Res: Reserved Bits**

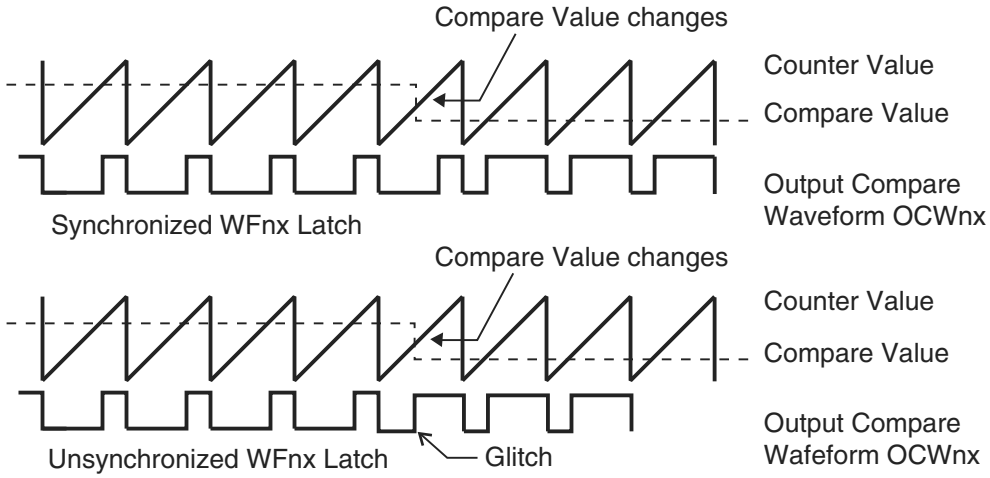
These bits are reserved and will always read zero.

- **Bit 0 – CTC0: Waveform Generation Mode**

This bit controls the counting sequence of the counter, the source for maximum (TOP) counter value, see Figure 11-7 on page 79. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter) and Clear Timer on Compare Match (CTC) mode (see “Modes of Operation” on page 77).

metrical PWM pulses, thereby making the output glitch-free. See Figure 12-6 for an example. During the time between the write and the update operation, a read from OCR1A, OCR1B, OCR1C or OCR1D will read the contents of the temporary location. This means that the most recently written value always will read out of OCR1A, OCR1B, OCR1C or OCR1D.

Figure 12-6. Effects of Unsynchronized OCR Latching



12.5.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC1x) bit. Forcing Compare Match will not set the OCF1x Flag or reload/clear the timer, but the Waveform Output (OCW1x) will be updated as if a real Compare Match had occurred (the COM1x1:0 bits settings define whether the Waveform Output (OCW1x) is set, cleared or toggled).

12.5.2 Compare Match Blocking by TCNT1 Write

All CPU write operations to the TCNT1 Register will block any Compare Match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR1x to be initialized to the same value as TCNT1 without triggering an interrupt when the Timer/Counter clock is enabled.

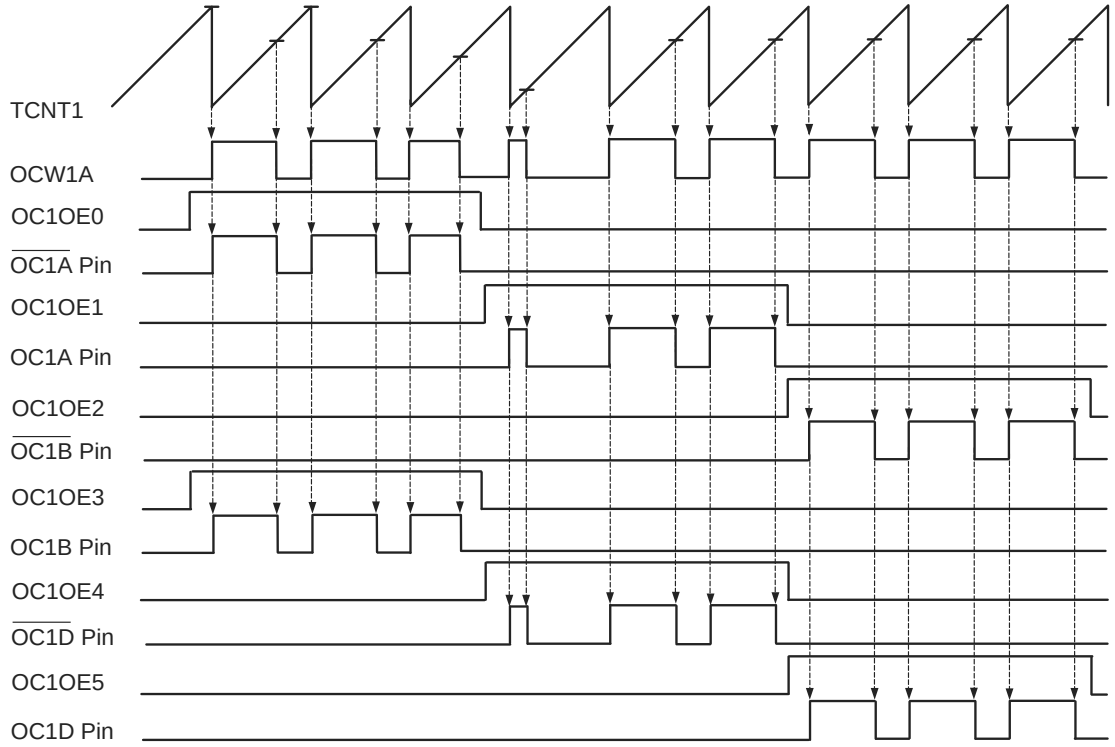
12.5.3 Using the Output Compare Unit

Since writing TCNT1 in any mode of operation will block all Compare Matches for one timer clock cycle, there are risks involved when changing TCNT1 when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1x value, the Compare Match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is down-counting.

The setup of the Waveform Output (OCW1x) should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OCW1x value is to use the Force Output Compare (FOC1x) strobe bits in Normal mode. The OC1x keeps its value even when changing between Waveform Generation modes.

Be aware that the COM1x1:0 bits are not double buffered together with the compare value. Changing the COM1x1:0 bits will take effect immediately.

Figure 12-14. PWM6 Mode, Single-slope Operation, Timing Diagram



The general I/O port function is overridden by the Output Compare value (OC1x / $\overline{OC1x}$) from the Dead Time Generator if either of the COM1x1:0 bits are set. The Output Compare pins can also be overriden by the Output Compare Override Enable bits OC1OE5:OC1OE0. If an Override Enable bit is cleared, the actual value from the port register will be visible on the port pin and, if the Override Enable bit is set, the Output Compare pin is allowed to be connected on the port pin. The Output Compare Pin configurations are described in Table 12-5, Table 12-6 and Table 12-7.

Table 12-5. Configuration of Output Compare Pins OC1A and $\overline{OC1A}$ in PWM6 Mode

COM1A1	COM1A0	$\overline{OC1A}$ Pin (PB0)	OC1A Pin (PB1)
0	0	Disconnected	Disconnected
0	1	OC1A • OC1OE0	OC1A • OC1OE1
1	0	OC1A • OC1OE0	OC1A • OC1OE1
1	1	OC1A • OC1OE0	OC1A • OC1OE1

Table 12-6. Configuration of Output Compare Pins OC1B and $\overline{OC1B}$ in PWM6 Mode

COM1B1	COM1B0	$\overline{OC1B}$ Pin (PB2)	OC1B Pin (PB3)
0	0	Disconnected	Disconnected
0	1	OC1A • OC1OE2	OC1A • OC1OE3
1	0	OC1A • OC1OE2	OC1A • OC1OE3
1	1	OC1A • OC1OE2	OC1A • OC1OE3

Note that, if 10-bit accuracy is used special procedures must be followed when accessing the internal 10-bit Output Compare Registers via the 8-bit AVR data bus. These procedures are described in section “Accessing 10-Bit Registers” on page 108.

12.12.13 TIMSK – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	OCIE1D	OCIE1A	OCIE1B	OCIE0A	OCIE0B	TOIE1	TOIE0	TICIE0	TIMSK
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 – OCIE1D: Timer/Counter1 Output Compare Interrupt Enable**

When the OCIE1D bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Compare MatchD, interrupt is enabled. The corresponding interrupt at vector $010 is executed if a compare matchD occurs. The Compare Flag in Timer/Counter1 is set (one) in the Timer/Counter Interrupt Flag Register.

- **Bit 6 – OCIE1A: Timer/Counter1 Output Compare Interrupt Enable**

When the OCIE1A bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Compare MatchA, interrupt is enabled. The corresponding interrupt at vector $003 is executed if a compare matchA occurs. The Compare Flag in Timer/Counter1 is set (one) in the Timer/Counter Interrupt Flag Register.

- **Bit 5 – OCIE1B: Timer/Counter1 Output Compare Interrupt Enable**

When the OCIE1B bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Compare MatchB, interrupt is enabled. The corresponding interrupt at vector $009 is executed if a compare matchB occurs. The Compare Flag in Timer/Counter1 is set (one) in the Timer/Counter Interrupt Flag Register.

- **Bit 2 – TOIE1: Timer/Counter1 Overflow Interrupt Enable**

When the TOIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Overflow interrupt is enabled. The corresponding interrupt (at vector $004) is executed if an overflow in Timer/Counter1 occurs. The Overflow Flag (Timer1) is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

12.12.14 TIFR – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x38 (0x58)	OCF1D	OCF1A	OCF1B	OCF0A	OCF0B	TOV1	TOV0	ICF0	TIFR
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 – OCF1D: Output Compare Flag 1D**

The OCF1D bit is set (one) when compare match occurs between Timer/Counter1 and the data value in OCR1D - Output Compare Register 1D. OCF1D is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1D is cleared, after synchronization clock cycle, by writing a logic one to the flag. When the I-bit in SREG, OCIE1D, and OCF1D are set (one), the Timer/Counter1 D compare match interrupt is executed.

- **Bit 6 – OCF1A: Output Compare Flag 1A**

The OCF1A bit is set (one) when compare match occurs between Timer/Counter1 and the data value in OCR1A - Output Compare Register 1A. OCF1A is cleared by hardware when executing

```
SPITransfer_loop:
    sts    USICR,r16
    lds    r16, USISR
    sbrs   r16, USIOIF
    rjmp   SPITransfer_loop
    lds    r16,USIDR
    ret
```

The code is size optimized using only eight instructions (plus return). The code example assumes that the DO and USCK pins have been enabled as outputs in DDRA. The value stored in register r16 prior to the function is called is transferred to the slave device, and when the transfer is completed the data received from the slave is stored back into the register r16.

The second and third instructions clear the USI Counter Overflow Flag and the USI counter value. The fourth and fifth instructions set three-wire mode, positive edge clock, count at USITC strobe, and toggle USCK. The loop is repeated 16 times.

The following code demonstrates how to use the USI as an SPI master with maximum speed ($f_{SCK} = f_{CK}/2$):

```
SPITransfer_Fast:
    out    USIDR,r16
    ldi    r16,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)
    ldi    r17,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)|(1<<USICLK)

    out    USICR,r16 ; MSB
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16 ; LSB
    out    USICR,r17

    in     r16,USIDR
ret
```

Table 13-2. Relations between the USICS1:0 and USICLK Setting (Continued)

USICS1	USICS0	USICLK	USI Data Register Clock Source	4-bit Counter Clock Source
1	1	0	External, negative edge	External, both edges
1	0	1	External, positive edge	Software clock strobe (USITC)
1	1	1	External, negative edge	Software clock strobe (USITC)

- **Bit 1 – USICLK: Clock Strobe**

Writing a one to this bit location strobes the USI Data Register to shift one step and the counter to increment by one, provided that the USICS1:0 bits are set to zero and by doing so the software clock strobe option is selected. The output will change immediately when the clock strobe is executed, i.e., in the same instruction cycle. The value shifted into the USI Data Register is sampled the previous instruction cycle. The bit will be read as zero.

When an external clock source is selected (USICS1 = 1), the USICLK function is changed from a clock strobe to a Clock Select Register. Setting the USICLK bit in this case will select the USITC strobe bit as clock source for the 4-bit counter (see Table 13-2).

- **Bit 0 – USITC: Toggle Clock Port Pin**

Writing a one to this bit location toggles the USCK/SCL value either from 0 to 1, or from 1 to 0. The toggling is independent of the setting in the Data Direction Register, but if the PORT value is to be shown on the pin the DDB2 must be set as output (to one). This feature allows easy clock generation when implementing master devices. The bit will be read as zero.

When an external clock source is selected (USICS1 = 1) and the USICLK bit is set to one, writing to the USITC strobe bit will directly clock the 4-bit counter. This allows an early detection of when the transfer is done when operating as a master device.

13.5.5 USIPP – USI Pin Position

Bit	7	6	5	4	3	2	1	0	
0x11 (0x31)	-	-	-	-	-	-	-	USIPOS	USIPP
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 0 – USIPOS: USI Pin Position**

Setting this bit to one changes the USI pin position. As default pins PB2:PB0 are used for the USI pin functions, but when writing this bit to one the USIPOS bit is set the USI pin functions are on pins PA2:PA0.

Table 15-5. ADC Prescaler Selections (Continued)

ADPS2	ADPS1	ADPS0	Division Factor
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

15.13.3 ADCL and ADCH – The ADC Data Register

15.13.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
0x05 (0x25)	–	–	–	–	–	–	ADC9	ADC8	ADCH
0x04 (0x24)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

15.13.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
0x05 (0x25)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
0x04 (0x24)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in “ADC Conversion Result” on page 153.

After $\overline{\text{RESET}}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

Table 18-9. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI	PB0	I	Serial Data in
MISO	PB1	O	Serial Data out
SCK	PB2	I	Serial Clock

Note: In Table 18-9, above, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the Serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into 0xFF.

Depending on CKSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

- Low:> 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for f_{ck} >= 12 MHz
- High:> 2 CPU clock cycles for f_{ck} < 12 MHz, 3 CPU clock cycles for f_{ck} >= 12 MHz

18.6.1 Serial Programming Algorithm

When writing serial data to the ATtiny261/461/861, data is clocked on the rising edge of SCK. When reading, data is clocked on the falling edge of SCK. See Figure 19-4 and Figure 19-5 for timing details.

To program and verify the ATtiny261/461/861 in the Serial Programming mode, the following sequence is recommended (see four byte instruction formats in Table 18-11):

1. Power-up sequence:
   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to “0”. In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse after SCK has been set to '0'. The duration of the pulse must be at least t_{RST} (the minimum pulse width on $\overline{\text{RESET}}$ pin, see Table 19-4 on page 190) plus two CPU clock cycles.
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI.
3. The serial programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (0x53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new Programming Enable command.
4. The Flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 5 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 6 MSB of the address. If polling (RDY/BSY) is not used, the user must wait at least t_{WD_FLASH} before issuing the next page. (See Table 18-10.) Accessing the serial programming

18.7.3 Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered:

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF, that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address high byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading.

18.7.4 Chip Erase

The Chip Erase will erase the Flash and EEPROM memories plus lock bits. The Lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash and/or EEPROM are reprogrammed.

1. Load Command “Chip Erase”:
   a. Set XA1, XA0 to “10”. This enables command loading.
   b. Set BS1 to “0”.
   c. Set DATA to “1000 0000”. This is the command for Chip Erase.
   d. Give XTAL1 a positive pulse. This loads the command.
   e. Give $\overline{WR}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{BSY}$ goes low.
   f. Wait until RDY/$\overline{BSY}$ goes high before loading a new command.

Note: The EEPROM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

18.7.5 Programming the Flash

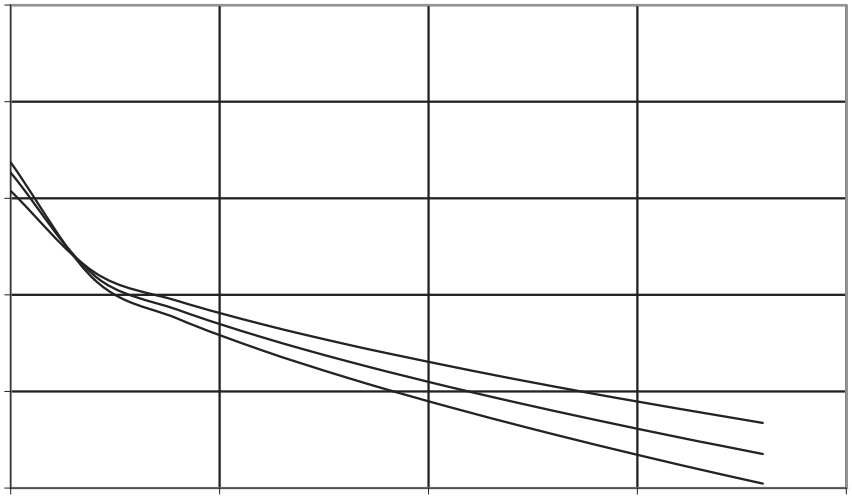
The Flash is organized in pages, see Table 18-7 on page 173. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory (see Figure 18-5 for signal waveforms):

1. Load Command “Write Flash”:
   a. Set XA1, XA0 to “10”. This enables command loading.
   b. Set BS1 to “0”.
   c. Set DATA to “0001 0000”. This is the command for Write Flash.
   d. Give XTAL1 a positive pulse. This loads the command.
2. Load Address Low byte:
   a. Set XA1, XA0 to “00”. This enables address loading.
   b. Keep BS1 at “0”. This selects low address.
   c. Set DATA = Address low byte (0x00 - 0xFF).
   d. Give XTAL1 a positive pulse. This loads the address low byte.

3. Load Data Low Byte:
   a. Set XA1, XA0 to “01”. This enables data loading.
   b. Set DATA = Data low byte (0x00 - 0xFF).
   c. Give XTAL1 a positive pulse. This loads the data byte.
4. Load Data High Byte:
   a. Set BS1 to “1”. This selects high data byte.
   b. Keep XA1, XA0 at “01”. This enables data loading.
   c. Set DATA = Data high byte (0x00 - 0xFF).
   d. Give XTAL1 a positive pulse. This loads the data byte.
5. Repeat steps 2 to 4 until the entire buffer is filled or until all data within the page is loaded.
6. Load Address High byte:
   a. Set XA1, XA0 to “00”. This enables address loading.
   b. Set BS1 to “1”. This selects high address.
   c. Set DATA = Address high byte (0x00 - 0xFF).
   d. Give XTAL1 a positive pulse. This loads the address high byte.
7. Program Page:
   a. Give $\overline{\text{WR}}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{\text{BSY}}$ goes low.
   b. Wait until RDY/$\overline{\text{BSY}}$ goes high.
8. Repeat steps 2 to 7 until the entire Flash is programmed or until all data has been programmed.
9. End Page Programming:
   a. Set XA1, XA0 to “10”. This enables command loading.
   b. Set DATA to “0000 0000”. This is the command for No Operation.
   c. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

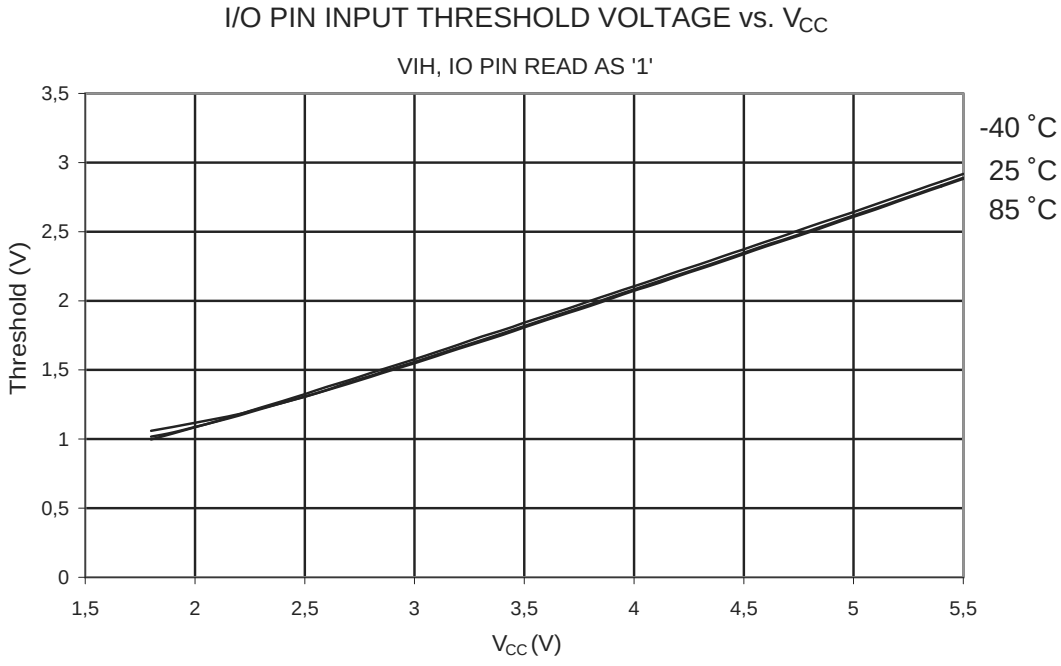
While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in Figure 18-4. Note that if less than eight bits are required to address words in the page (pagesize < 256), the most significant bit(s) in the address low byte are used to address the page when performing a Page Write.

Figure 20-26. Reset Pin Output Voltage vs. Source Current (V_{CC} = 5V)



20.7 Pin Threshold and Hysteresis

Figure 20-27. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IH}, I/O Pin Read as '1')



Note:
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 operation 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.

23. Ordering Information

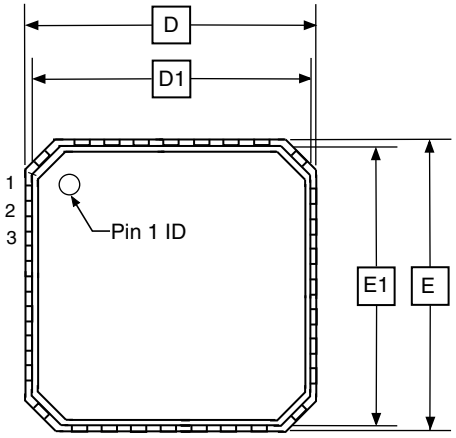
23.1 ATtiny261 - Mature

Speed (MHz)[3]	Power Supply (V)	Ordering Code[4][5]	Package[2]	Operational Range
10	1.8 - 5.5	ATtiny261V-10MU ATtiny261V-10MUR ATtiny261V-10PU ATtiny261V-10SU ATtiny261V-10SUR	32M1-A 32M1-A 20P3 20S2 20S2	Industrial (-40°C to +85°C)[1]
20	2.7 - 5.5	ATtiny261-20MU ATtiny261-20MUR ATtiny261-20PU ATtiny261-20SU ATtiny261-20SUR	32M1-A 32M1-A 20P3 20S2 20S2	Industrial (-40°C to +85°C)[1]

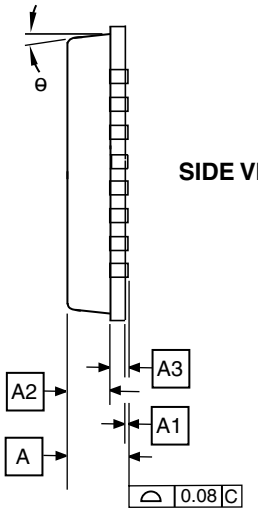
Notes:
1. These devices can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. All packages are Pb-free, halide-free and fully green and they comply with the European directive for Restriction of Hazardous Substances (RoHS).
3. For Speed vs. V_{CC}, see Figure 19.3 on page 188.
4. Code indicators:
   - U: matte tin
   - R: tape & reel
5. Mature devices, replaced by ATtiny261A.

24. Packaging Information

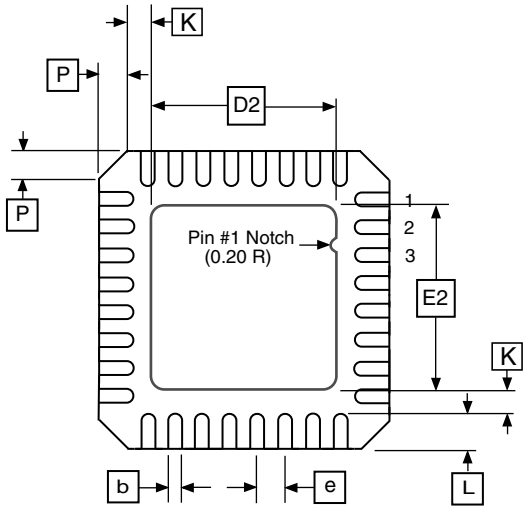
24.1 32M1-A



TOP VIEW



SIDE VIEW



BOTTOM VIEW

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A2	–	0.65	1.00	
A3		0.20 REF		
b	0.18	0.23	0.30	
D		5.00 BSC		
D1		4.75 BSC		
D2	2.95	3.10	3.25	
E		5.00 BSC		
E1		4.75BSC		
E2	2.95	3.10	3.25	
e		0.50 BSC		
L	0.30	0.40	0.50	
P	–	–	0.60	
θ	–	–	12^{o}	
K	0.20	–	–	

Note: JEDEC Standard MO-220, Fig. 2 (Anvil Singulation), VHHD-2.

8/19/04

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**32M1-A**, 32-pad, 5 x 5 x 1.0 mm Body, Lead Pitch 0.50 mm, 3.10 mm Exposed Pad, Micro Lead Frame Package (MLF)	32M1-A	D