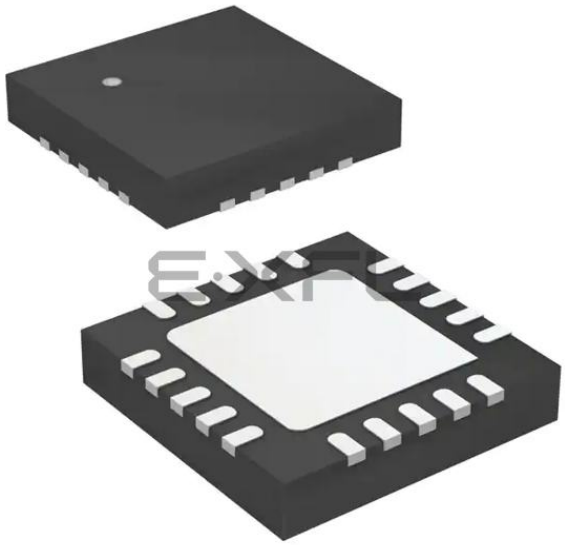


Welcome to E-XFL.COM

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	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 ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	20-WFQFN Exposed Pad
Supplier Device Package	20-WQFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny861-15maz

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.

Assembly Code Example

```
EEPROM_read:
        ; Wait for completion of previous write
        sbic    EECR,EEPE
        rjmp    EEPROM_read
        ; Set up address (r18:r17) in address register
        out     EEARH, r18
        out     EEARL, r17
        ; Start eeprom read by writing EERE
        sbi     EECR,EERE
        ; Read data from data register
        in      r16,EEDR
        ret
```

C Code Example

```
unsigned char EEPROM_read(unsigned char ucAddress)
{
        /* Wait for completion of previous write */
        while(EECR & (1<<EEPE))
        ;
        /* Set up address register */
        EEAR = ucAddress;
        /* Start eeprom read by writing EERE */
        EECR |= (1<<EERE);
        /* Return data from data register */
        return EEDR;
}
```

6.3.6 Preventing EEPROM Corruption

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

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

EEPROM data corruption can easily be avoided by following this design recommendation:

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 detection level of the internal BOD does not match the needed detection level, 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.

- **Bit 2 – EEMPE: EEPROM Master Program Enable**

The EEMPE bit determines whether writing EEPE to one will have effect or not.

When EEMPE is set, setting EEPE within four clock cycles will program the EEPROM at the selected address. If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to one by software, hardware clears the bit to zero after four clock cycles.

- **Bit 1 – EEPE: EEPROM Program Enable**

The EEPROM program enable signal EEPE is the programming enable signal to the EEPROM. When EEPE is written, the EEPROM will be programmed according to the EEPMn bits setting. The EEMPE bit must be written to one before a logical one is written to EEPE, otherwise no EEPROM write takes place. When the write access time has elapsed, the EEPE bit is cleared by hardware. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM read enable signal – EERE – is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be written to one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed. The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR register.

6.5.4 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x0C (0x2C)	**MSB**							**LSB**	**GPIOR2**
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	

6.5.5 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	

6.5.6 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	

8.7.4 Internal Voltage Reference

The internal voltage reference will be enabled when needed by the brown-out detection, the analog comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to
Section 9.7 “Internal Voltage Reference” on page 42 for details on the start-up time.

8.7.5 Watchdog Timer

If the watchdog timer is not needed in the application, this module should be turned off. If the watchdog timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to Section 9.8 “Watchdog Timer” on page 42 for details on how to configure the Watchdog Timer.

8.7.6 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ($clk_{I/O}$) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to Section 12.2.5 “Digital Input Enable and Sleep Modes” on page 56 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or has an analog signal level close to $V_{CC}/2$, the input buffer will use excessive power.

For analog input pins, the digital input buffer should be disabled at all times. An analog signal level close to $V_{CC}/2$ on an input pin can cause significant current even in active mode. Digital input buffers can be disabled by writing to the digital input disable registers (DIDR0, DIDR1). Refer to Section 19.10.5 “DIDR0 – Digital Input Disable Register 0” on page 149 or Section 19.10.6 “DIDR1 – Digital Input Disable Register 1” on page 149 for details.

8.8 Register Description

8.8.1 MCUCR – MCU Control Register

The MCU control register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	**–**	**PUD**	**SE**	**SM1**	**SM0**	**—**	**ISC01**	**ISC00**	**MCUCR**
Read/Write	R	R/W	R/W	R/W	R/W	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

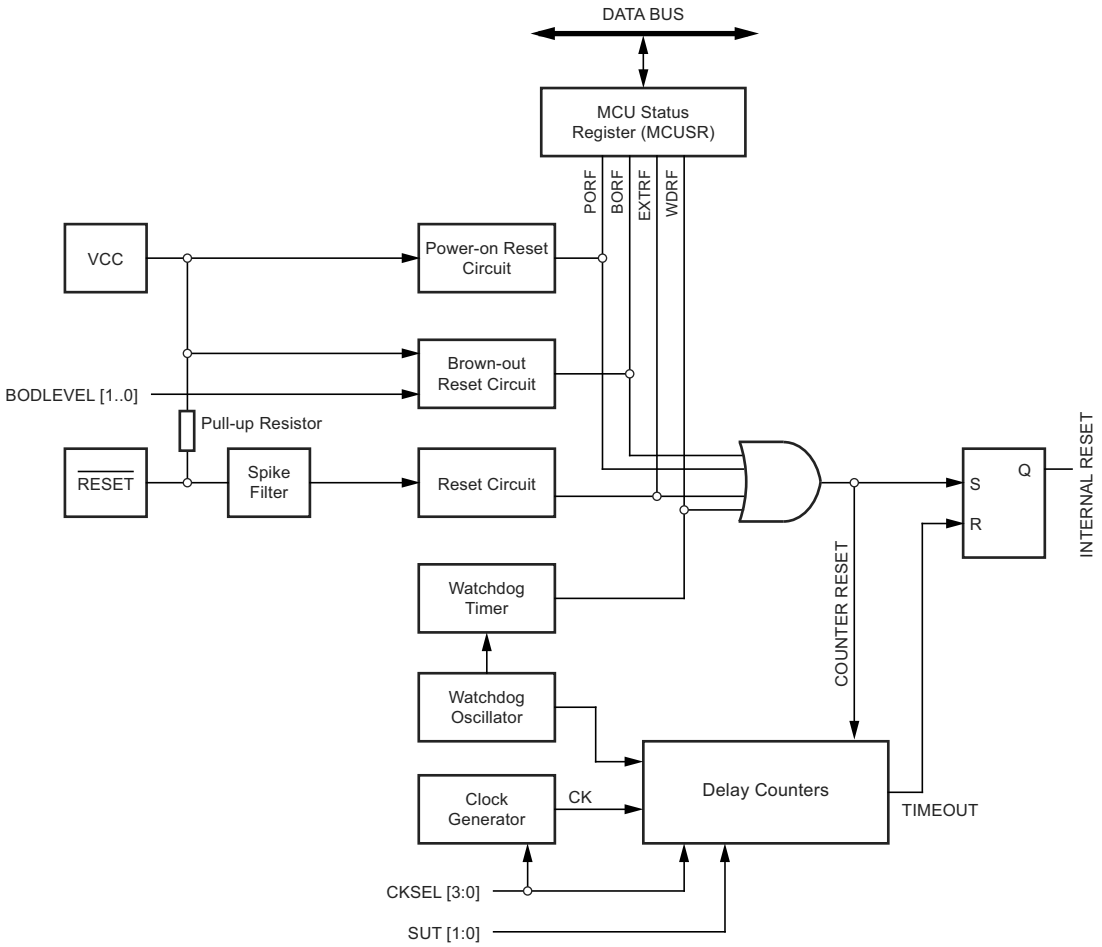
- **Bit 5 – SE: Sleep Enable**

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer’s purpose, it is recommended to write the sleep enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

- **Bits 4, 3 – SM1:0: Sleep Mode Select Bits 2..0**

These bits select between the three available sleep modes as shown in Table 8-2 on page 37.

Figure 9-1. Reset Logic



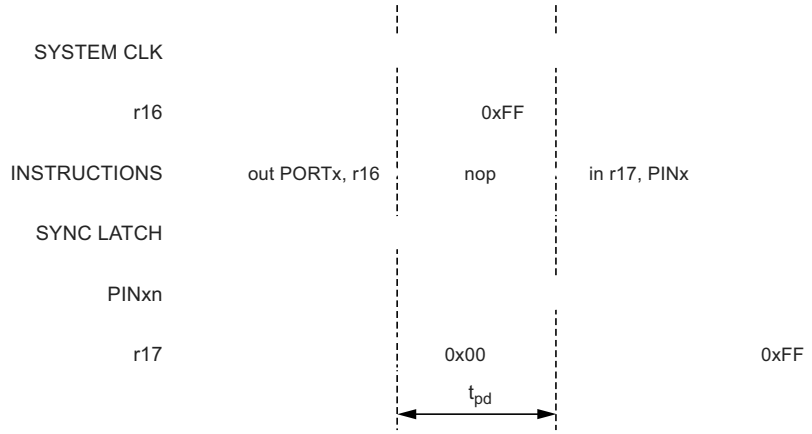
If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The most typical and general program setup for the reset and interrupt vector addresses in Atmel® ATtiny261/461/861 is:

```
Address      Labels Code                Comments
0x0000              rjmp   RESET        ; Reset Handler
0x0001              rjmp   EXT_INT0     ; IRQ0 Handler
0x0002              rjmp   PCINT        ; PCINT Handler
0x0003              rjmp   TIM1_COMPA   ; Timer1 CompareA Handler
0x0004              rjmp   TIM1_COMPB   ; Timer1 CompareB Handler
0x0005              rjmp   TIM1_OVF     ; Timer1 Overflow Handler
0x0006              rjmp   TIM0_OVF     ; Timer0 Overflow Handler
0x0007              rjmp   USI_START    ; USI Start Handler
0x0008              rjmp   USI_OVF      ; USI Overflow Handler
0x0009              rjmp   EE_RDY       ; EEPROM Ready Handler
0x000A              rjmp   ANA_COMP     ; Analog Comparator Handler
0x000B              rjmp   ADC          ; ADC Conversion Handler
0x000C              rjmp   WDT          ; WDT Interrupt Handler
0x000D              rjmp   EXT_INT1     ; IRQ1 Handler
0x000E              rjmp   TIM0_COMPA   ; Timer0 CompareA Handler
0x000F              rjmp   TIM0_COMPB   ; Timer0 CompareB Handler
0x0010              rjmp   TIM0_CAPT    ; Timer0 Capture Event Handler
0x0011              rjmp   TIM1_COMPD   ; Timer1 CompareD Handler
0x0012              rjmp   FAULT_PROTECTION; Timer1 Fault Protection
0x0013       RESET: ldi    r16, low(RAMEND); Main program start
0x0014              ldi    r17, high(RAMEND); Tiny861 have also SPH
0x0015              out    SPL, r16     ; Set Stack Pointer to top of RAM
0x0016              out    SPH, r17     ; Tiny861 have also SPH
0x0017              sei                 ; Enable interrupts
0x0018       <instr> xxx
...    ...    ...    ...
```

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

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 12-4. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is one system clock period.

Figure 12-4. Synchronization when Reading a Software Assigned Pin Value



- **Port B, Bit 4 - XTAL1/ CLKI/ OC1B/ ADC7/ PCINT12**

XTAL1/CLKI: chip clock oscillator pin 1. Used for all chip clock sources except internal calibrated RC oscillator. When used as a clock pin, the pin can not be used as an I/O pin.

$\overline{OC1D}$: Inverted output compare match output: The PB4 pin can serve as an external output for the Timer/Counter1 compare match D when configured as an output (DDA0 set). The $\overline{OC1D}$ pin is also the inverted output pin for the PWM mode timer function.

ADC7: ADC input channel 7. Note that ADC input channel 7 uses analog power.

PCINT12: Pin change interrupt source 12.

- **Port B, Bit 3 - OC1B/ PCINT11**

OC1B, output compare match output: The PB3 pin can serve as an external output for the Timer/Counter1 compare match B. The PB3 pin has to be configured as an output (DDB3 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

PCINT11: Pin change interrupt source 11.

- **Port B, Bit 2 - SCK/ USCK/ SCL/ $\overline{OC1B}$/ PCINT10**

SCK: Master clock output, slave clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB2. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB2. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB2 bit.

USCK: Three-wire mode universal serial interface clock.

SCL: Two-wire mode serial clock for USI two-wire mode.

$\overline{OC1B}$: Inverted output compare match output: The PB2 pin can serve as an external output for the Timer/Counter1 compare match B when configured as an output (DDB2 set). The $\overline{OC1B}$ pin is also the inverted output pin for the PWM mode timer function.

PCINT10: Pin change interrupt source 10.

- **Port B, Bit 1 - MISO/ DO/ OC1A/ PCINT9**

MISO: Master data input, slave data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDB1. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB1. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB1 bit.

DO: Three-wire mode universal serial interface data output. Three-wire mode data output overrides PORTB1 value and it is driven to the port when data direction bit DDB1 is set (one). PORTB1 still enables the pull-up, if the direction is input and PORTB1 is set (one).

OC1A: Output compare match output: The PB1 pin can serve as an external output for the Timer/Counter1 compare match B when configured as an output (DDB1 set). The OC1A pin is also the output pin for the PWM mode timer function.

PCINT9: Pin change interrupt source 9.

- **Port B, Bit 0 - MOSI/ DI/ SDA/ $\overline{OC1A}$/ PCINT8**

MOSI: SPI master data output, slave data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB0. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB0. When the pin is forced by the SPI to be an input, the pull-up can still be controlled by the PORTB0 bit.

DI: Data input in USI three-wire mode. USI three-wire mode does not override normal port functions, so pin must be configure as an input for DI function.

SDA: Two-wire mode serial interface data.

$\overline{OC1A}$: Inverted output compare match output: The PB0 pin can serve as an external output for the Timer/Counter1 compare match B when configured as an output (DDB0 set). The $\overline{OC1A}$ pin is also the inverted output pin for the PWM mode timer function.

PCINT8: Pin change interrupt source 8.

Figure 16-2. Timer/Counter1 Synchronization Register Block Diagram

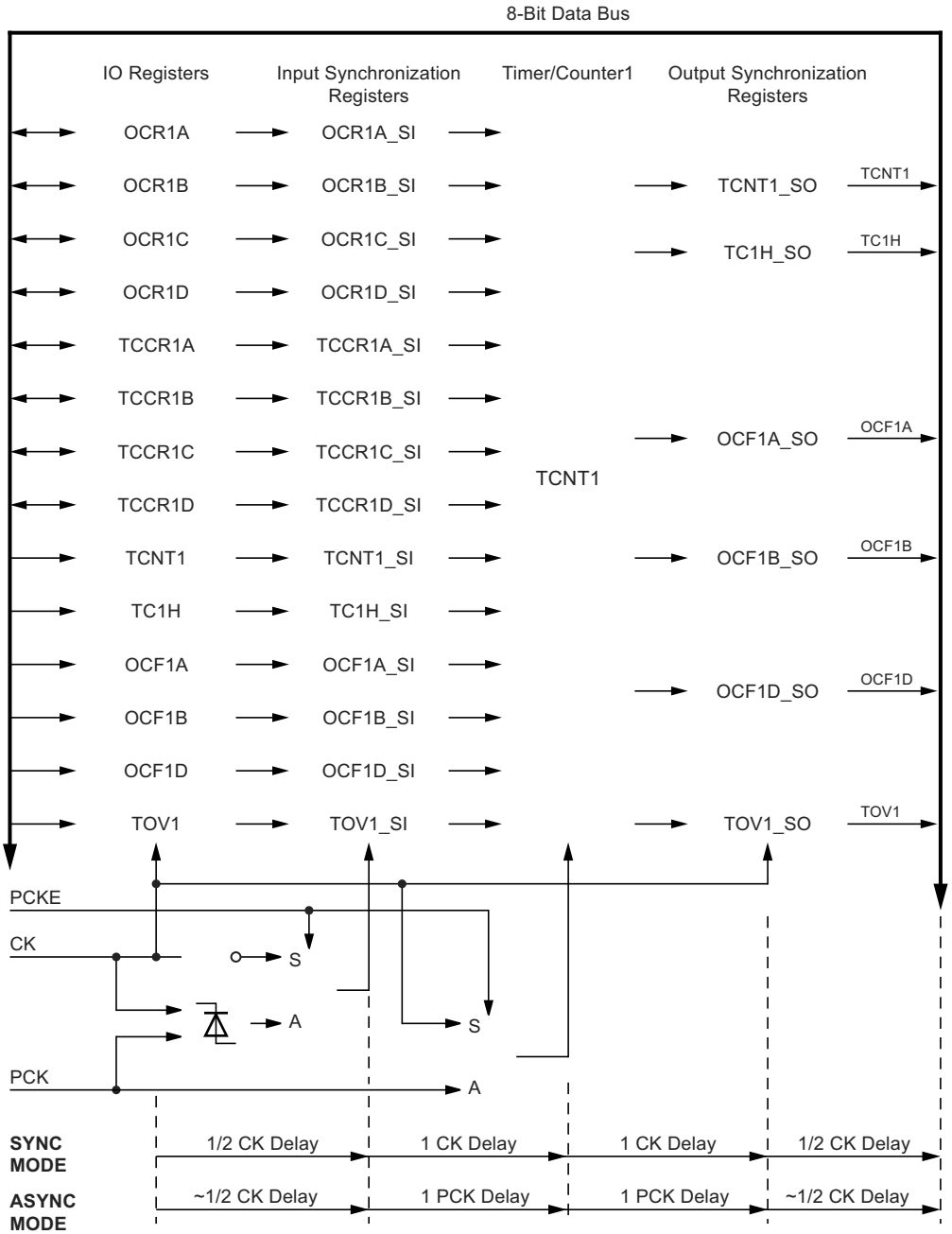
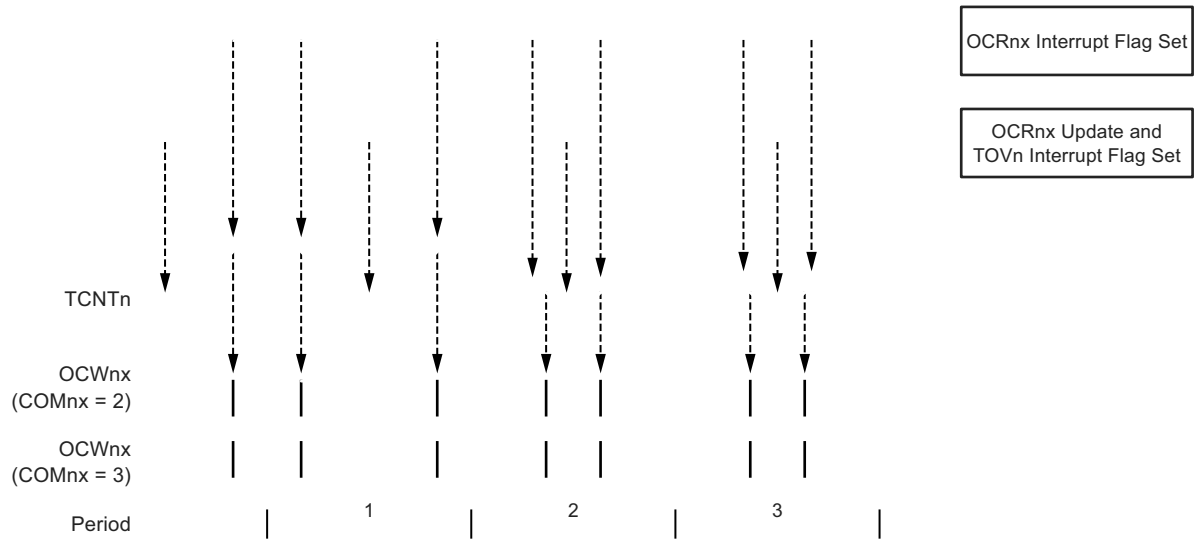


Figure 16-12. Phase and Frequency Correct PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV1) is set each time the counter reaches BOTTOM. The interrupt flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In the phase and frequency correct PWM mode, the compare unit allows generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and setting the COM1x1:0 to three will produce an inverted PWM output. Setting the COM1A1:0 bits to one will enable complementary compare output mode and produce both the non-inverted (OC1x) and inverted output ($\overline{OC1x}$). The actual values will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the waveform output (OCW1x) at the compare match between OCR1x and TCNT1 when the counter increments, and setting (or clearing) the waveform output at compare match when the counter decrements. The PWM frequency for the output when using the phase and frequency correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{\mathrm{clkT1}}}{N}$$

The *N* variable represents the number of steps in dual-slope operation. The value of *N* equals to the TOP value.

The extreme values for the OCR1C register represent special cases when generating a PWM waveform output in the phase and frequency correct PWM mode. If the OCR1C is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

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 and the data direction register bits for the OC1X and $\overline{OC1X}$ pins are set as an output. If the COM1x1:0 bits are cleared, the actual value from the port register will be visible on the port pin. The configurations of the output compare pins are described in Table 16-4.

Table 16-4. Output Compare pin configurations in Phase and Frequency Correct PWM Mode

COM1x1	COM1x0	$\overline{OC1x}$ Pin	OC1x Pin
0	0	Disconnected	Disconnected
0	1	OC1x	OC1x
1	0	Disconnected	OC1x
1	1	Disconnected	OC1x

The 4-bit counter can be both read and written via the data bus, and can generate an overflow interrupt. Both the USI data register and the counter are clocked simultaneously by the same clock source. This allows the counter to count the number of bits received or transmitted and generate an interrupt when the transfer is complete. Note that when an external clock source is selected the counter counts both clock edges. In this case the counter counts the number of edges, and not the number of bits. The clock can be selected from three different sources: The USCK pin, Timer/Counter0 compare match or from software.

The Two-wire clock control unit can generate an interrupt when a start condition is detected on the two-wire bus. It can also generate wait states by holding the clock pin low after a start condition is detected, or after the counter overflows.

17.3 Functional Descriptions

17.3.1 Three-wire Mode

The USI three-wire mode is compliant to the serial peripheral interface (SPI) mode 0 and 1, but does not have the slave select (SS) pin functionality. However, this feature can be implemented in software if necessary. Pin names used by this mode are: DI, DO, and USCK.

Figure 17-2. Three-wire Mode Operation, Simplified Diagram

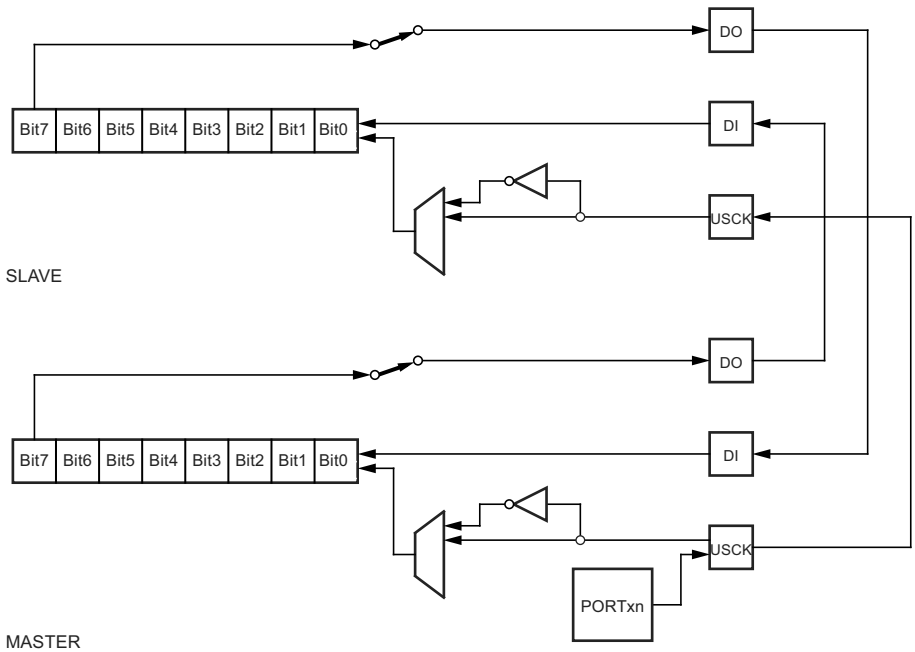


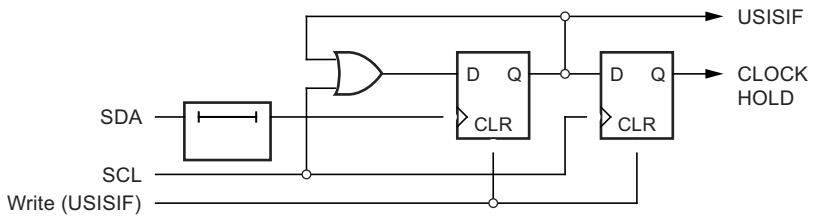
Figure 17-2 shows two USI units operating in three-wire mode, one as master and one as slave. The two USI data register are interconnected in such way that after eight USCK clocks, the data in each register are interchanged. The same clock also increments the USI's 4-bit counter. The counter overflow (interrupt) flag, or USIOIF, can therefore be used to determine when a transfer is completed. The clock is generated by the master device software by toggling the USCK pin via the PORT register or by writing a one to the USITC bit in USICR.

Referring to the timing diagram (Figure 17-5 on page 123), 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 USI data register bit must be set to one for the output to be enabled. The slave device's start detector logic (Figure 17-6) 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 USI data 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 USI data 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 17-6. Start Condition Detector, Logic Diagram



17.3.5 Start Condition Detector

The start condition detector is shown in Figure 17-6 The SDA line is delayed (in the range of 50 to 300ns) to ensure valid sampling of the SCL line. The start condition detector is only enabled in two-wire mode.

The start condition detector is working asynchronously and can therefore wake up the processor from the power-down sleep mode. However, the protocol used might have restrictions on the SCL hold time. Therefore, when using this feature in this case the oscillator start-up time set by the CKSEL fuses (see Section 7.1 "Clock Systems and their Distribution" on page 24) must also be taken into the consideration. Refer to the USISIF bit description on page 126 for further details.

17.4 Alternative USI Usage

When the USI unit is not used for serial communication, it can be set up to do alternative tasks due to its flexible design.

17.4.1 Half-duplex Asynchronous Data Transfer

By utilizing the USI data register in three-wire mode, it is possible to implement a more compact and higher performance UART than by software only.

17.4.2 4-bit Counter

The 4-bit counter can be used as a stand-alone counter with overflow interrupt. Note that if the counter is clocked externally, both clock edges will generate an increment.

17.4.3 12-bit Timer/Counter

Combining the USI 4-bit counter and Timer/Counter0 allows them to be used as a 12-bit counter.

17.4.4 Edge Triggered External Interrupt

By setting the counter to maximum value (F) it can function as an additional external interrupt. The overflow flag and interrupt enable bit are then used for the external interrupt. This feature is selected by the USICS1 bit.

17.4.5 Software Interrupt

The counter overflow interrupt can be used as a software interrupt triggered by a clock strobe.

17.5 Register Descriptions

17.5.1 USIDR – USI Data Register

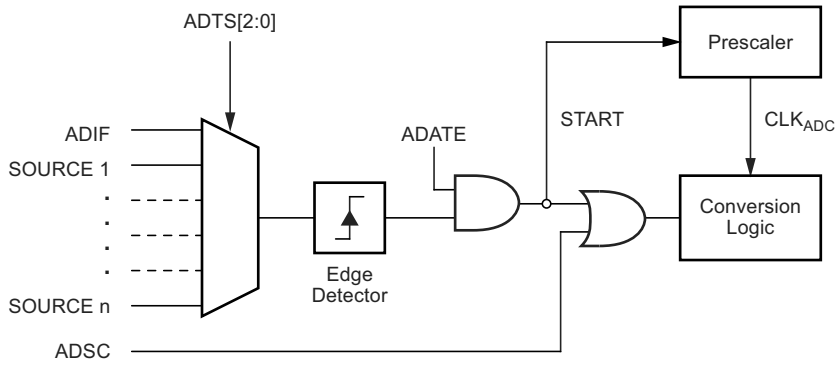
Bit	7	6	5	4	3	2	1	0	
0x0F (0x2F)	**MSB**							**LSB**	**USIDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

When accessing the USI data register (USIDR) the serial register can be accessed directly. If a serial clock occurs at the same cycle the register is written, the register will contain the value written and no shift is performed. A (left) shift operation is performed depending of the USICS1..0 bits setting. The shift operation can be controlled by an external clock edge, by a Timer/Counter0 Compare Match, or directly by software using the USICLK strobe bit. Note that even when no wire mode is selected (USIWM1..0 = 0) both the external data input (DI/SDA) and the external clock input (USCK/SCL) can still be used by the USI data register.

The output pin in use, DO or SDA depending on the wire mode, is connected via the output latch to the most significant bit (bit 7) of the data register. The output latch is open (transparent) during the first half of a serial clock cycle when an external clock source is selected (USICS1 = 1), and constantly open when an internal clock source is used (USICS1 = 0). The output will be changed immediately when a new MSB written as long as the latch is open. The latch ensures that data input is sampled and data output is changed on opposite clock edges.

Note that the corresponding data direction register to the pin must be set to one for enabling data output from the USI data register.

Figure 19-2. ADC Auto Trigger Logic

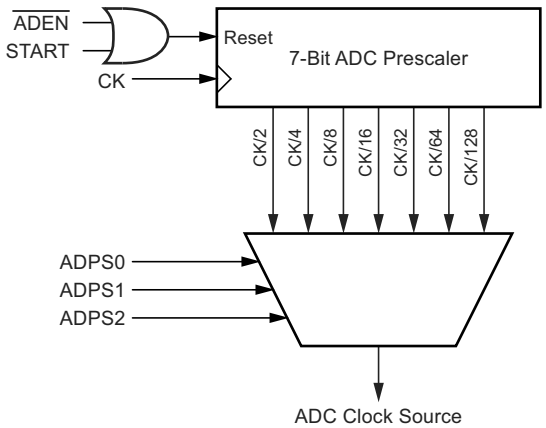


Using the ADC interrupt flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in free running mode, constantly sampling and updating the ADC data register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA. In this mode the ADC will perform successive conversions independently of whether the ADC interrupt flag, ADIF is cleared or not.

If auto triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

19.5 Prescaling and Conversion Timing

Figure 19-3. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

20.4 Software Break Points

debugWIRE supports program memory break points by the AVR® break instruction. Setting a break point in AVR Studio® will insert a BREAK instruction in the program memory. The instruction replaced by the BREAK instruction will be stored. When program execution is continued, the stored instruction will be executed before continuing from the program memory. A break can be inserted manually by putting the BREAK instruction in the program.

The flash must be re-programmed each time a break point is changed. This is automatically handled by AVR Studio through the debugWIRE interface. The use of break points will therefore reduce the flash data retention. Devices used for debugging purposes should not be shipped to end customers.

20.5 Limitations of debugWIRE

The debugWIRE communication pin (dW) is physically located on the same pin as external reset (RESET). An external reset source is therefore not supported when the debugWIRE is enabled.

The debugWIRE system accurately emulates all I/O functions when running at full speed, i.e., when the program in the CPU is running. When the CPU is stopped, care must be taken while accessing some of the I/O registers via the debugger (AVR Studio).

A programmed DWEN fuse enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption while in sleep. Thus, the DWEN fuse should be disabled when debugWire is not used.

20.6 Register Description

The following section describes the registers used with the debugWire.

20.6.1 DWDR – debugWire Data Register

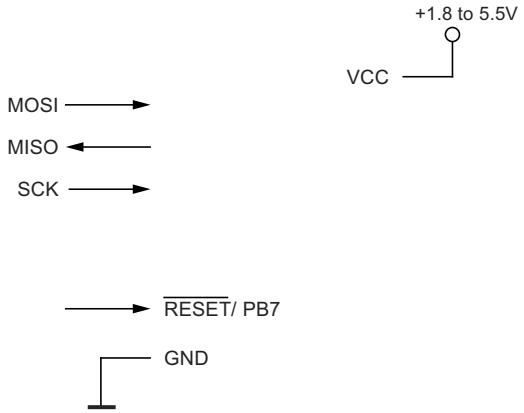
Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	**DWDR[7:0]**								**DWDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The DWDR register provides a communication channel from the running program in the MCU to the debugger. This register is only accessible by the debugWIRE and can therefore not be used as a general purpose register in the normal operations.

22.9 Serial Downloading

Both the flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{\text{RESET}}$ is set low, the programming enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 22-13 on page 167, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Figure 22-7. Serial Programming and Verify(1)



Note: 1. If the device is clocked by the internal oscillator, it is no need to connect a clock source to the CLKI pin.

Table 22-13. 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

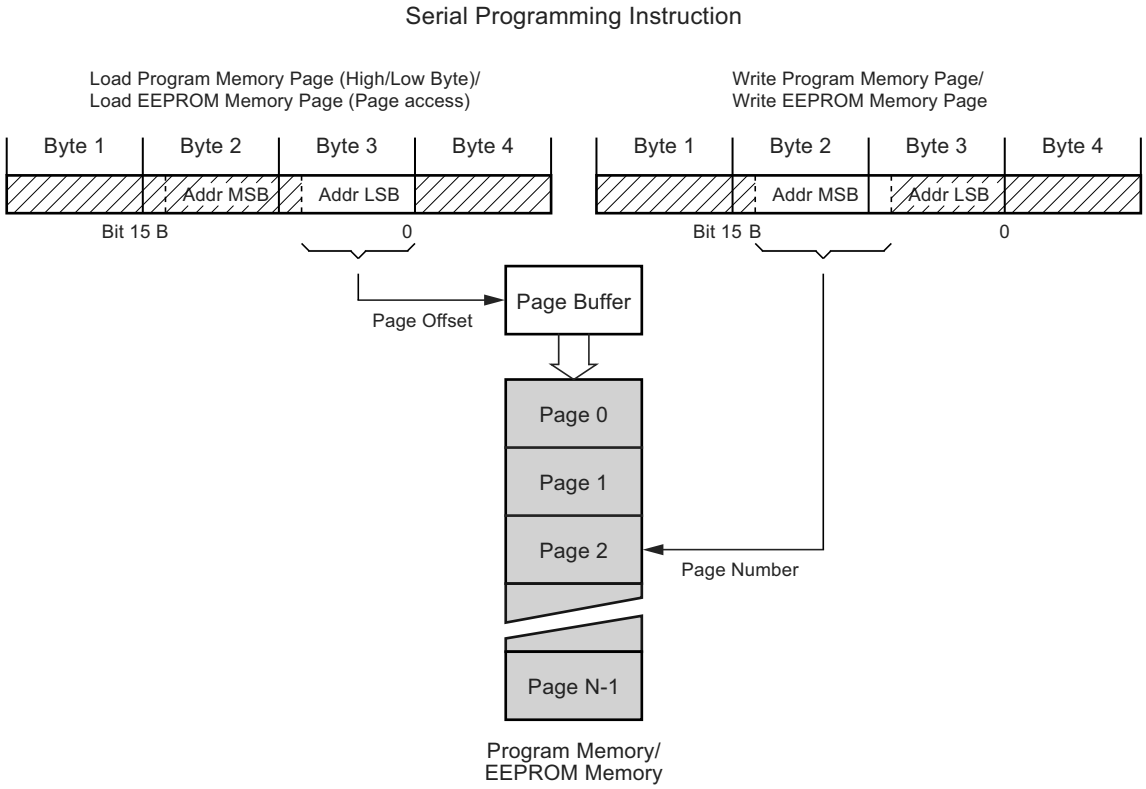
When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the chip erase instruction. The chip erase operation turns the content of every memory location in both the Program and EEPROM arrays into 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} < 12MHz, 3 CPU clock cycles for f_{ck} ≥ 12MHz

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

Figure 22-8. Serial Programming Instruction Example



24.3 Supply Current of I/O Modules

The tables and formulas below can be used to calculate the additional current consumption for the different I/O modules in Active and Idle mode. The enabling or disabling of the I/O modules are controlled by the power reduction register. See Section 8.8.2 “PRR – Power Reduction Register” on page 37 for details.

Table 24-1. Additional Current Consumption for the Different I/O Modules (Absolute Values)

PRR Bit	Typical Numbers		
	V_{CC} = 2V, F = 1MHz	V_{CC} = 3V, F = 4MHz	V_{CC} = 5V, F = 8MHz
PRTIM1	65µA	423µA	1787µA
PRTIM0	7µA	39µA	165µA
PRUSI	5µA	25µA	457µA
PRADC	18µA	111µA	102µA

Table 24-2. Additional Current Consumption (Percentage) in Active and Idle Mode

PRR Bit	Additional Current Consumption Compared to Active with External Clock (see Figure 24-1 and Figure 24-2)	Additional Current Consumption Compared to Idle with External Clock
PRTIM1	26.9%	103.7%
PRTIM0	2.6%	10.0%
PRUSI	1.7%	6.5%
PRADC	7.1%	27.3%

It is possible to calculate the typical current consumption based on the numbers from Table 24-1 for other V_{CC} and frequency settings than listed in Table 24-2.

24.3.1 Example

Calculate the expected current consumption in idle mode with TIMER0, ADC, and USI enabled at V_{CC} = 2.0V and F = 1MHz. From Table 24-2, third column, we see that we need to add 10% for the TIMER0, 27.3% for the ADC, and 6.5% for the USI module. Reading from Figure 24-3 on page 181, we find that the idle current consumption is ~0,085mA at V_{CC} = 2.0V and F = 1MHz. The total current consumption in idle mode with TIMER0, ADC, and USI enabled, gives:

$ICCtotal \approx 0.085mA \times (1 + 0.10 + 0.273 + 0.065) \approx 0.122mA$

Figure 24-10. I/O Pin Output Voltage versus Source Current (V_{CC} = 3V)

Figure 24-11. I/O Pin Output Voltage versus Source Current (V_{CC} = 5V)

24.7 Pin Threshold and Hysteresis

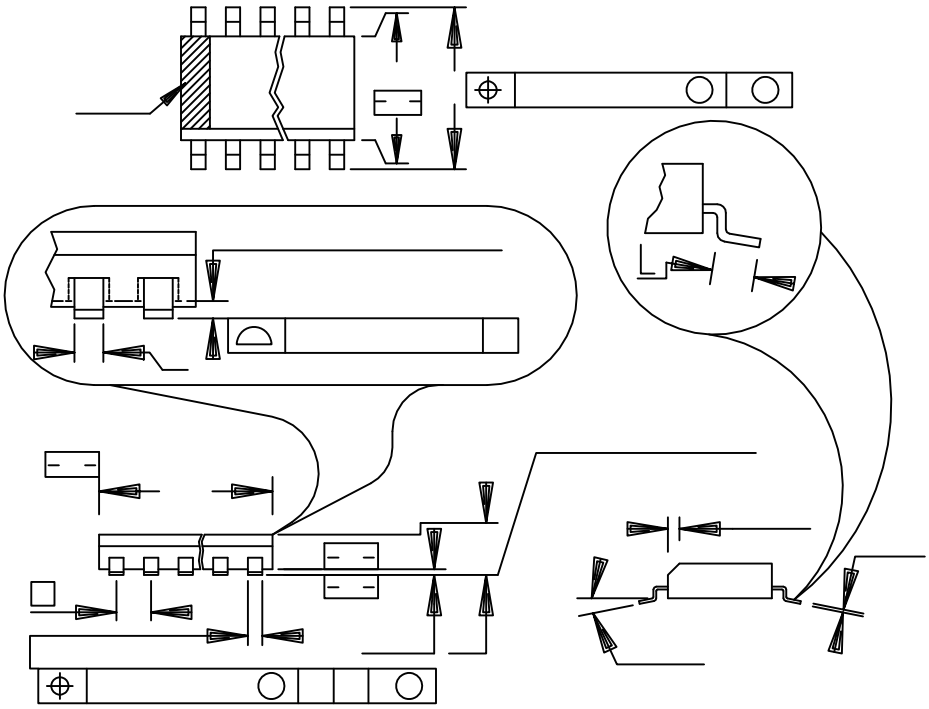
Figure 24-12. I/O Pin Input Threshold Voltage versus V_{CC} (V_{IH}, I/O Pin Read as '1')

24.9 Internal Oscillator Speed

Figure 24-19. Watchdog Oscillator Frequency versus Temperature

Figure 24-20. Calibrated 8.0MHz RC Oscillator Frequency versus Temperature

Figure 28-2. TG



09/10/07

Atmel Package Drawing Contact: packagedrawings@atmel.com	TITLE TG, 20 Lead, 0.300" Body Width Plastic Gull Wing Small outline Package (SOIC)	GPC	DRAWING NO. TG	REV. N