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	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	28
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	1.7V ~ 5.5V
Data Converters	A/D 28x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-TQFP
Supplier Device Package	32-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny828-au

5.3 Data Memory (EEPROM)

ATtiny828 contains 256 bytes of non-volatile data memory. This EEPROM is organized as a separate data space, in which single bytes can be read and written. All access registers are located in the I/O space.

The EEPROM memory layout is summarised in Table 4, below.

Table 4. Size of Non-Volatile Data Memory (EEPROM)

Device	EEPROM Size	Address Range
ATtiny828	256B	0x00 – 0xFF

The internal 8MHz oscillator is used to time EEPROM operations. The frequency of the oscillator must be within the requirements described in “OSCCAL0 – Oscillator Calibration Register” on page 32.

When powered by heavily filtered supplies, the supply voltage, V_{CC}, is likely to rise or fall slowly on power-up and power-down. Slow rise and fall times may put the device in a state where it is running at supply voltages lower than specified. To avoid problems in situations like this, see “Preventing EEPROM Corruption” on page 20.

The EEPROM has a minimum endurance of 100,000 write/erase cycles.

5.3.1 Programming Methods

There are two methods for EEPROM programming:

- Atomic byte programming. This is the simple mode of programming, where target locations are erased and written in a single operation. In this mode of operation the target is guaranteed to always be erased before writing but programmin times are longer.
- Split byte programming. It is possible to split the erase and write cycle in two different operations. This is useful when short access times are required, for example when supply voltage is falling. In order to take advantage of this method target locations must be erased before writing to them. This can be done at times when the system allows time-critical operations, typically at start-up and initialisation.

The programming method is selected using the EEPROM Programming Mode bits (EEPM1 and EEPM0) in EEPROM Control Register (EECR). See Table 5 on page 24. Write and erase times are given in the same table.

Since EEPROM programming takes some time the application must wait for one operation to complete before starting the next. This can be done by either polling the EEPROM Program Enable bit (EEPE) in EEPROM Control Register (EECR), or via the EEPROM Ready Interrupt. The EEPROM interrupt is controlled by the EEPROM Ready Interrupt Enable (EERIE) bit in EECR.

5.3.2 Read

To read an EEPROM memory location follow the procedure below:

- Poll the EEPROM Program Enable bit (EEPE) in EEPROM Control Register (EECR) to make sure no other EEPROM operations are in process. If set, wait to clear.
- Write target address to EEPROM Address Registers (EEARH/EEARL).
- Start the read operation by setting the EEPROM Read Enable bit (EERE) in the EEPROM Control Register (EECR). During the read operation, the CPU is halted for four clock cycles before executing the next instruction.
- Read data from the EEPROM Data Register (EEDR).

To ensure stable operation of the MCU it is required to avoid sudden changes in the external clock frequency . A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. It is required to ensure that the MCU is kept in Reset during such changes in the clock frequency.

Stable operation for large step changes in system clock frequency is guaranteed when using the system clock prescaler. See “System Clock Prescaler” on page 30.

6.2.2 Calibrated Internal 8MHz Oscillator

The internal 8MHz oscillator operates with no external components and, by default, provides a clock source with an approximate frequency of 8MHz. Though voltage and temperature dependent, this clock can be very accurately calibrated by the user. See Table 104 on page 249 and “Internal Oscillator Speed” on page 293 for more details.

During reset, hardware loads the pre-programmed calibration value into the OSCCAL0 register and thereby automatically calibrates the oscillator. The accuracy of this calibration is referred to as “Factory Calibration” in Table 104 on page 249. For more information on automatic loading of pre-programmed calibration value, see section “Calibration Bytes” on page 229.

It is possible to reach higher accuracies than factory defaults, especially when the application allows temperature and voltage ranges to be narrowed. The firmware can reprogram the calibration data in OSCCAL0 either at start-up or during run-time. The continuous, run-time calibration method allows firmware to monitor voltage and temperature and compensate for any detected variations. See “OSCCAL0 – Oscillator Calibration Register” on page 32, “Temperature Measurement” on page 148, and Table 52 on page 150. The accuracy of this calibration is referred to as “User Calibration” in Table 104 on page 249.

The oscillator temperature calibration registers, OSCTCAL0A and OSCTCAL0B, can be used for one-time temperature calibration of oscillator frequency. See “OSCTCAL0A – Oscillator Temperature Calibration Register A” on page 33 and “OSCTCAL0B – Oscillator Temperature Calibration Register B” on page 33.

When this oscillator is used as the chip clock, it will still be used for the Watchdog Timer and for the Reset Time-out.

Start-up time for this clock source is determined by the SUT fuse bit, as shown in Table 7 on page 30.

6.2.3 Internal 32kHz Ultra Low Power (ULP) Oscillator

The internal 32kHz oscillator is a low power oscillator that operates with no external components. It provides a clock source with an approximate frequency of 32kHz. The frequency depends on supply voltage, temperature and batch variations. See Table 105 on page 250 for accuracy details.

During reset, hardware loads the pre-programmed calibration value into the OSCCAL1 register and thereby automatically calibrates the oscillator. The accuracy of this calibration is referred to as “Factory Calibration” in Table 105 on page 250. For more information on automatic loading of pre-programmed calibration value, see section “Calibration Bytes” on page 229.

Start-up time for this clock source is determined by the SUT fuse bit, as shown in Table 7 on page 30.

6.2.4 Default Clock Settings

The device is shipped with following fuse settings:

- Calibrated Internal 8MHz Oscillator (see CKSEL fuse bits in Table 6 on page 28)
- Longest possible start-up time (see SUT fuse bits in Table 7 on page 30)
- System clock prescaler set to 8 (see CKDIV8 fuse bit on page 32)

The default setting gives a 1MHz system clock and ensures all users can make their desired clock source setting using an in-system or high-voltage programmer.

the ACD bit in ACSRA. See “ACSRA – Analog Comparator Control and Status Register” on page 134. This will reduce power consumption in Idle mode.

If the ADC is enabled, a conversion starts automatically when this mode is entered.

7.1.2 ADC Noise Reduction Mode

This sleep mode halts $clk_{I/O}$, clk_{CPU}, and clk_{FLASH}, while allowing other clocks to run. In ADC Noise Reduction mode, the CPU is stopped but the following peripherals continue to operate:

- Watchdog (if enabled), and external interrupts
- ADC
- USART start frame detector, and TWI

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered.

The following events can wake up the MCU:

- Watchdog reset, external reset, and brown-out reset
- External level interrupt on INT0, and pin change interrupt
- ADC conversion complete interrupt, and SPM/EEPROM ready interrupt
- USART start frame detection, and TWI slave address match

7.1.3 Power-Down Mode

This sleep mode halts all generated clocks, allowing operation of asynchronous modules, only. In Power-down Mode the oscillator is stopped, while the following peripherals continue to operate:

- Watchdog (if enabled), external interrupts

The following events can wake up the MCU:

- Watchdog reset, external reset, and brown-out reset
- External level interrupt on INT0, and pin change interrupt
- USART start frame detection, and TWI slave address match

7.2 Power Reduction Register

The Power Reduction Register (PRR), see “PRR – Power Reduction Register” on page 37, provides a method to reduce power consumption by stopping the clock to individual peripherals. When the clock for a peripheral is stopped then:

- The current state of the peripheral is frozen.
- The associated registers can not be read or written.
- Resources used by the peripheral will remain occupied.

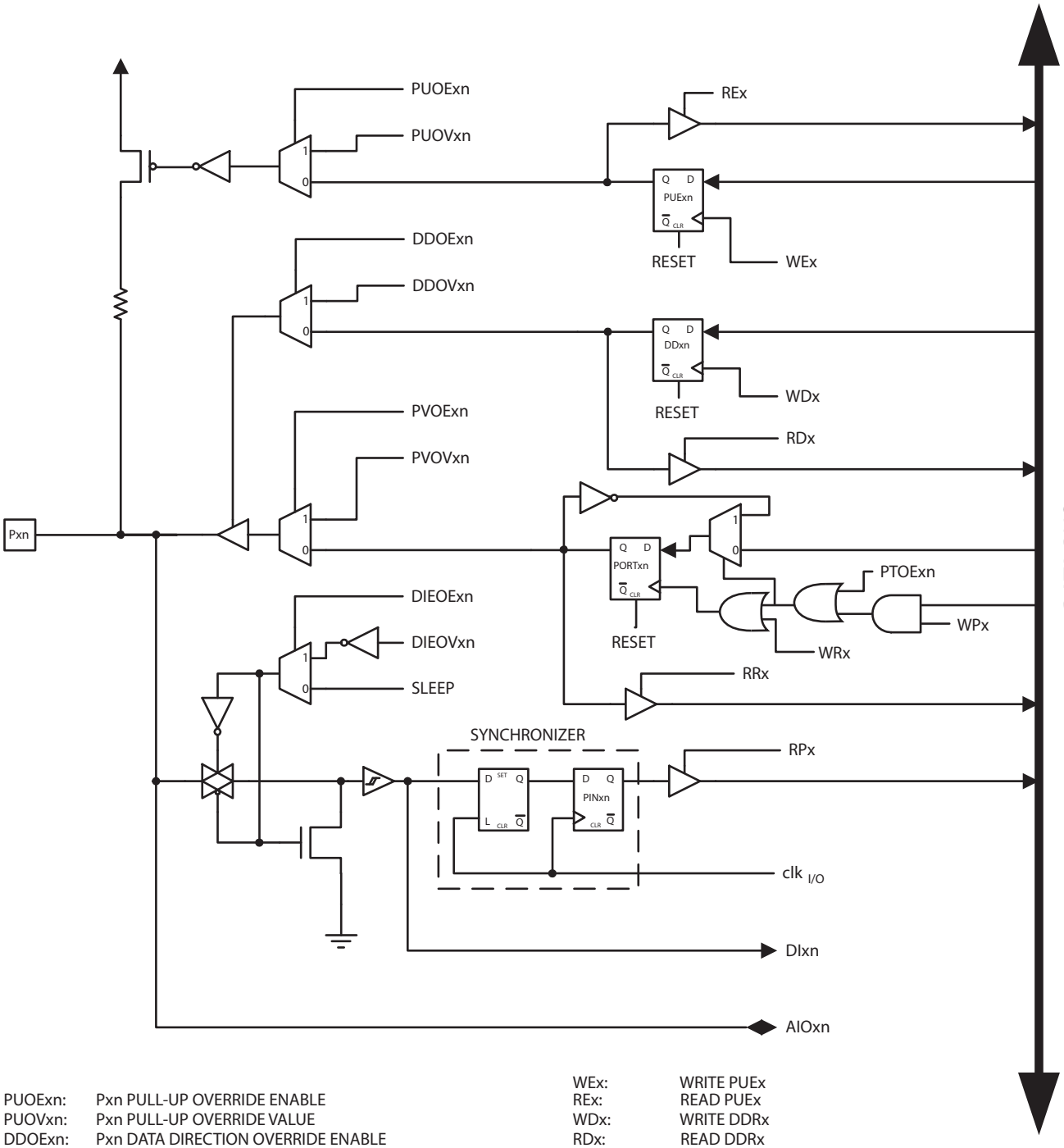
The peripheral should in most cases be disabled before stopping the clock. Clearing the PRR bit wakes up the peripheral and puts it in the same state as before shutdown.

Peripheral shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. See “Current Consumption of Peripheral Units” on page 266 for examples. In all other sleep modes, the clock is already stopped.

7.3 Minimizing Power Consumption

There are several issues to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as

Figure 25. Alternative Port Functions



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

WEx:	WRITE PUEx
REx:	READ PUEx
WDx:	WRITE DDRx
RDx:	READ DDRx
RRx:	READ PORTx REGISTER
WRx:	WRITE PORTx
RPx:	READ PORTx PIN
WPx:	WRITE PINx
$clk_{I/O}$:	I/O CLOCK
DIxn:	DIGITAL INPUT PIN n ON PORTx
AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

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

For actual placement of I/O pins, refer to Figure 1 on page 2 (MLF), and Figure 2 on page 2 (TQFP). Also, see "TOCPMSA1 and TOCPMSA0 – Timer/Counter Output Compare Pin Mux Selection Registers" on page 127, and "TOCPMCOE – Timer/Counter Output Compare Pin Mux Channel Output Enable" on page 128.

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 channel. 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.

12.2.1 Registers

The Timer/Counter (TCNT1), Output Compare Registers (OCR1A/B), and Input Capture Register (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in section "Accessing 16-bit Registers" on page 120. The Timer/Counter Control Registers (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T1 pin. The Clock Select logic block controls which clock source and edge 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_{T1}).

The double buffered Output Compare Registers (OCR1A/B) are compared with the Timer/Counter value at all time. 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 (OC1A/B). See "Output Compare Units" on page 108. The compare match event will also set the Compare Match Flag (OCF1A/B) which can be used to generate an Output Compare interrupt request.

The Input Capture Register can capture the Timer/Counter value at a given external (edge triggered) event on either the Input Capture pin (ICP1) or on the Analog Comparator pins (See "Analog Comparator" on page 133). The Input Capture unit includes a digital filtering unit (Noise Canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR1A Register, the ICR1 Register, or by a set of fixed values. When using OCR1A as TOP value in a PWM mode, the OCR1A Register can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICR1 Register can be used as an alternative, freeing the OCR1A to be used as PWM output.

12.2.2 Definitions

The following definitions are used extensively throughout the section:

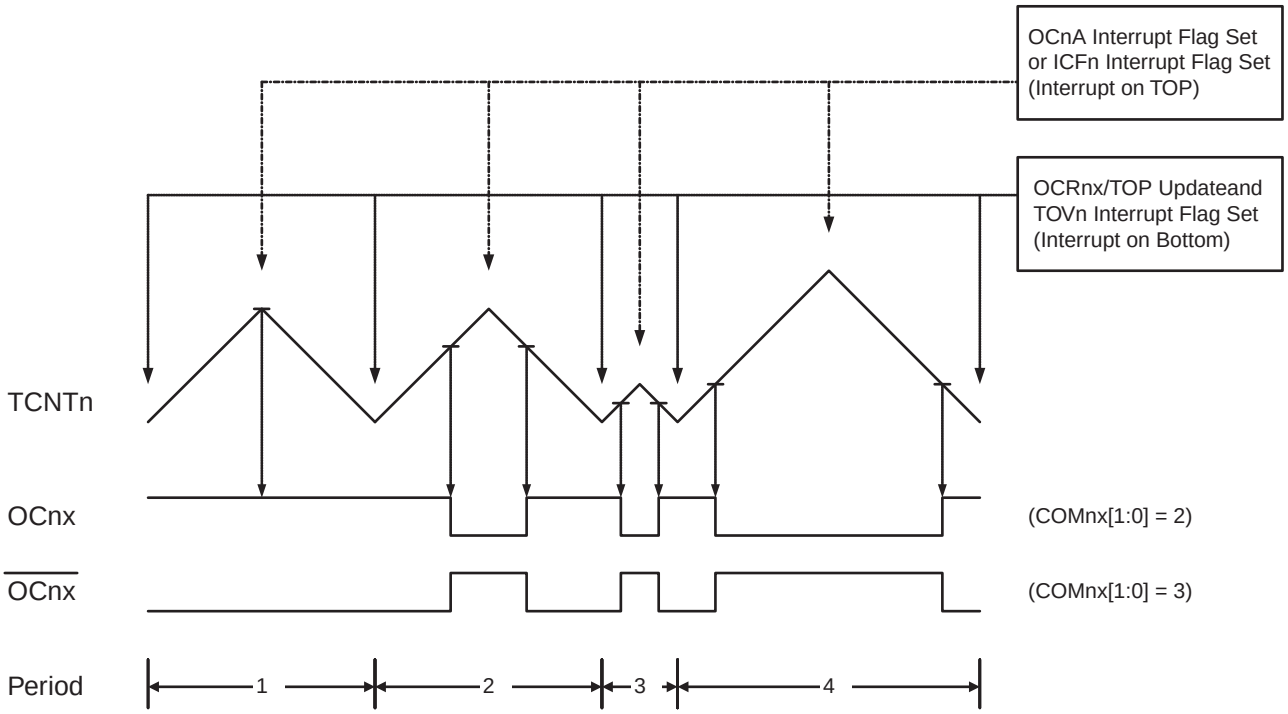
Table 38. Definitions

Constant	Description
BOTTOM	The counter reaches BOTTOM when it becomes 0x00
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255)
TOP	The counter reaches 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), the value stored in the OCR1A register, or the value stored in the ICR1 register. The assignment depends on the mode of operation

12.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS1[2:0]) bits located in the Timer/Counter control Register B (TCCR1B). For details on clock sources and prescaler, see "Timer/Counter Prescaler" on page 131.

Figure 45. Phase and Frequency Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set at the same timer clock cycle as the OCR1x Registers are updated with the double buffer value (at BOTTOM). When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 flag set when TCNT1 has reached TOP. The interrupt flags can then be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1x.

As Figure 45 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1x Registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct.

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 COM1x[1:0] bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x[1:0] to three (See Table 41 on page 124). 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

13. Timer/Counter Prescaler

Timer/Counter0 and Timer/Counter1 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to both Timer/Counters. Tn is used as a general name, n = 0, 1.

The Timer/Counter can be clocked directly by the system clock (by setting the CSn[2:0] = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK\_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK\_I/O}/8$, $f_{CLK\_I/O}/64$, $f_{CLK\_I/O}/256$, or $f_{CLK\_I/O}/1024$.

13.1 Prescaler Reset

The prescaler is free running, i.e., operates independently of the Clock Select logic of the Timer/CounterCounter, and it is shared by the Timer/Counter Tn. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (CSn[2:0] = 2, 3, 4, or 5). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

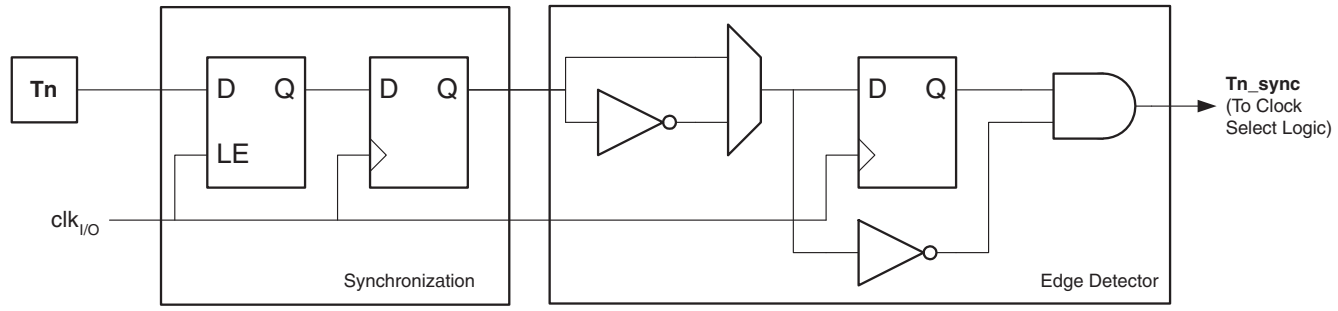
It is possible to use the Prescaler Reset for synchronizing the Timer/Counter to program execution.

13.2 External Clock Source

An external clock source applied to the Tn pin can be used as Timer/Counter clock (clk_{Tn}). The Tn pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 50 shows a functional equivalent block diagram of the Tn synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock ($clk_{I/O}$). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{T0} pulse for each positive (CSn[2:0] = 7) or negative (CSn[2:0] = 6) edge it detects.

Figure 50. T0 Pin Sampling



The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the Tn pin to the counter is updated.

Enabling and disabling of the clock input must be done when Tn has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk\_I/O}/2$) given a 50/50% duty cycle. Since the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by oscillator source tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{clk\_I/O}/2.5$.

An external clock source can not be prescaled.

Table 45. 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 these bits, the analog comparator interrupt must be disabled. Otherwise, an interrupt can occur when the bits are changed.

14.1.2 ACSRB – Analog Comparator Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x2F (0x4F)	**HSEL**	**HLEV**	**–**	**–**	**ACNMUX1**	**ACNMUX0**	**ACPMUX1**	**ACPMUX0**	**ACSRB**
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – HSEL: Hysteresis Select**

When this bit is written logic one, the hysteresis of the analog comparator is enabled. The level of hysteresis is selected by the HLEV bit.

- **Bit 6 – HLEV: Hysteresis Level**

When enabled via the HSEL bit, the level of hysteresis can be set using the HLEV bit, as shown in Table 46.

Table 46. Selecting Level of Analog Comparator Hysteresis

HSEL	HLEV	Hysteresis of Analog Comparator
0	X	Not enabled
1	0	20 mV
	1	50 mV

- **Bit 4 – Res: Reserved Bit**

This bit is reserved and will always read as zero.

- **Bits 3:2 – ACNMUX[1:0]: Analog Comparator Negative Input Multiplexer**

These bits select the source for the negative input of the analog comparator, as shown in Table 47, below.

C Code Example[1][2]

```
void USART_Transmit( unsigned int data )
{
      /* Wait for empty transmit buffer */
      while ( !( UCSRA & (1<<UDRE)) )
                        ;

      /* Copy 9th bit to TXB8 */
      UCSRB &= ~(1<<TXB8);
      if ( data & 0x0100 )
            UCSRB |= (1<<TXB8);

      /* Put data into buffer, sends the data */
      UDR = data;
}
```

Notes: 1. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRB is static. For example, only the TXB8 bit of UCSRB is used after initialization.

2. See “Code Examples” on page 7.

The ninth bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

17.6.3 Transmitter Flags and Interrupts

The USART transmitter has two flags that indicate its state: USART Data Register Empty (UDRE) and Transmit Complete (TXC). Both flags can be used for generating interrupts.

The Data Register Empty flag (UDRE) indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not yet been moved into the shift register. For compatibility with future devices, always write this bit to zero when writing UCSRA.

When the Data Register Empty Interrupt Enable bit (UDRIE) is set, the USART Data Register Empty Interrupt will be executed as long as UDRE is set (and provided that global interrupts are enabled). UDRE is cleared by writing UDR. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to UDR in order to clear UDRE or disable the Data Register Empty interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

The Transmit Complete flag (TXC) is set when the entire frame in the transmit shift register has been shifted out and there are no new data currently present in the transmit buffer. The TXC flag is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its location. The TXC flag is useful in half-duplex communication interfaces (like the RS-485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmit Compete Interrupt Enable bit (TXCIE) is set, the USART Transmit Complete Interrupt will be executed when the TXC flag becomes set (and provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXC flag, since this is done automatically when the interrupt is executed.

17.6.4 Parity Generator

The parity generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPM1 = 1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

Baud Rate (bps)	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max. (1)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

1. UBRR = 0, Error = 0.0%

Table 66. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

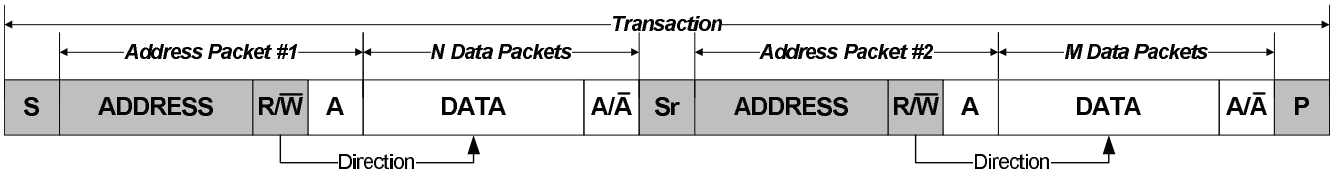
Baud Rate (bps)	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max. (1)	1 Mbps		2 Mbps		1.152 Mbps		2.304 Mbps		1.25 Mbps		2.5 Mbps	

1. UBRR = 0, Error = 0.0%

Given that the slave acknowledges the address, the master can start receiving data from the slave. There are no limitations to the number of data packets that can be transferred. The slave transmits the data while the master signals ACK or NACK after each data byte. The master terminates the transfer with a NACK before issuing a STOP condition.

Figure 84 illustrates a combined transaction. A combined transaction consists of several read and write transactions separated by a Repeated START conditions (Sr).

Figure 84. Combined Transaction

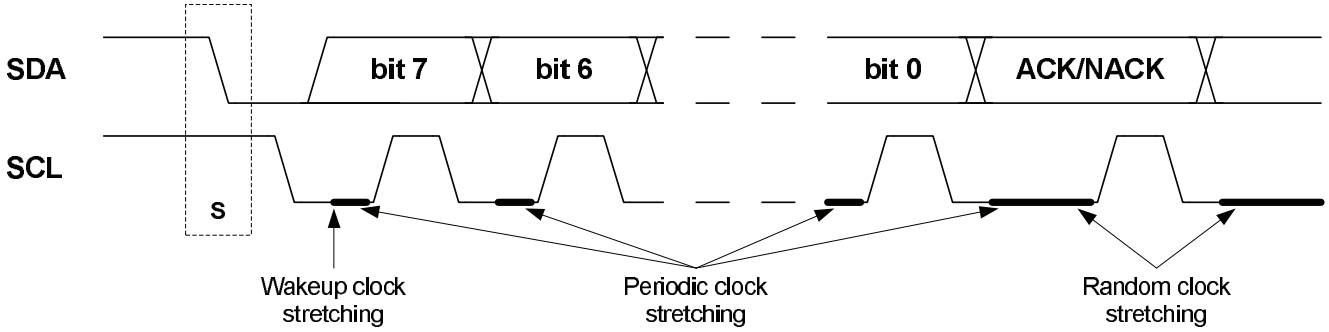


19.3.7 Clock and Clock Stretching

All devices connected to the bus are allowed to stretch the low period of the clock to slow down the overall clock frequency or to insert wait states while processing data. A device that needs to stretch the clock can do this by holding/forcing the SCL line low after it detects a low level on the line.

Three types of clock stretching can be defined as shown in Figure 85.

Figure 85. Clock Stretching



If the device is in a sleep mode and a START condition is detected the clock is stretched during the wake-up period for the device.

A slave device can slow down the bus frequency by stretching the clock periodically on a bit level. This allows the slave to run at a lower system clock frequency. However, the overall performance of the bus will be reduced accordingly. Both the master and slave device can randomly stretch the clock on a byte level basis before and after the ACK/NACK bit. This provides time to process incoming or prepare outgoing data, or performing other time critical tasks.

In the case where the slave is stretching the clock the master will be forced into a wait-state until the slave is ready and vice versa.

19.3.8 Arbitration

A master can only start a bus transaction if it has detected that the bus is idle. As the TWI bus is a multi master bus, it is possible that two devices initiate a transaction at the same time. This results in multiple masters owning the bus simultaneously. This is solved using an arbitration scheme where the master loses control of the bus if it is not able to transmit a high level on the SDA line. The masters who lose arbitration must then wait until the bus becomes idle (i.e. wait for a STOP condition) before attempting to reacquire bus ownership. Slave devices are not involved in the arbitration procedure.

19.5.2 TWSCRB – TWI Slave Control Register B

Bit	7	6	5	4	3	2	1	0	
(0xB9)	–	–	–	–	–	TWAA	TWCMD1	TWCMD0	TWSCRB
Read/Write	R	R	R	R	R	R/W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 2 – TWAA: TWI Acknowledge Action**

This bit defines the slave's acknowledge behavior after an address or data byte has been received from the master. Depending on the TWSME bit in TWSCRA the Acknowledge Action is executed either when a valid command has been written to TWCMDn bits, or when the data register has been read. Acknowledge action is also executed if clearing TWAIF flag after address match or TWDIF flag during master transmit. See Table 77 for details.

Table 77. Acknowledge Action of TWI Slave

TWAA	Action	TWSME	When
0	Send ACK	0	When TWCMDn bits are written to 10 or 11
		1	When TWSD is read
1	Send NACK	0	When TWCMDn bits are written to 10 or 11
		1	When TWSD is read

- **Bits 1:0 – TWCMD[1:0]: TWI Command**

Writing these bits triggers the slave operation as defined by Table 78. The type of operation depends on the TWI slave interrupt flags, TWDIF and TWASIF. The Acknowledge Action is only executed when the slave receives data bytes or address byte from the master.

Table 78. TWI Slave Command

TWCMD[1:0]	TWDIR	Operation
00	X	No action
01	X	Reserved
10	Used to complete transaction	
	0	Execute Acknowledge Action, then wait for any START (S/Sr) condition
	1	Wait for any START (S/Sr) condition

In addition to the application and boot loader areas, the Flash is also divided into two fixed sections; the Read-While-Write (RWW) and the No Read-While-Write (NRWW) section. The main difference between the RWW and NRWW sections are:

- When erasing or writing a page located inside the RWW section, the NRWW section can be read during the operation
- When erasing or writing a page located inside the NRWW section, the CPU is halted during the entire operation

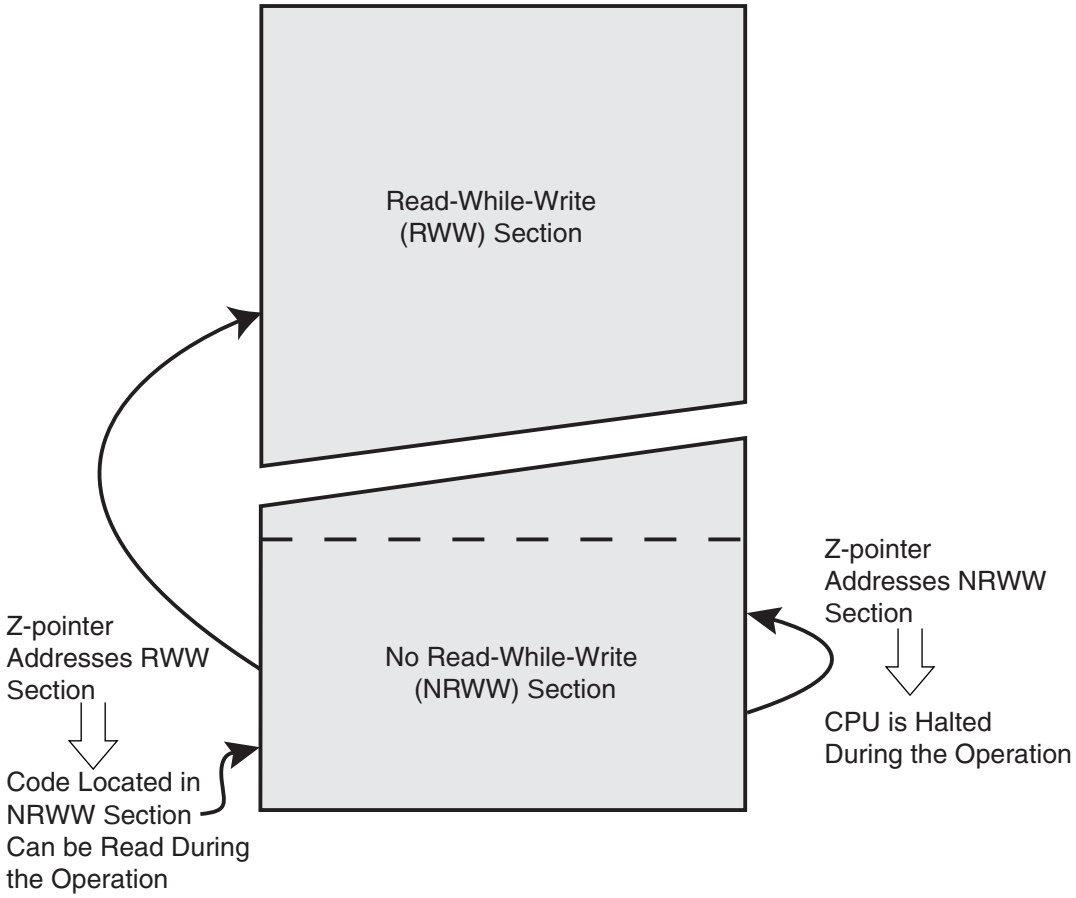
Note that during a boot loader operation, software can never read code located in the RWW section. See Table 79, below.

Table 79. Read-While-Write Features

Section that Z-pointer addresses during programming	Section that can be read during programming?	CPU halted	Read-while-write supported
RWW	NRWW	No	Yes
NRWW	None	Yes	No

The term “read-while-write section” refers to the section being programmed (i.e. erased, or written), not the section that is being read during a software update by the boot loader.

Figure 90. Read-While-Write vs. No Read-While-Write



RWW and NRWW sections are defined in Table 82 on page 217 and illustrated in Figure 90.

Bit #	Bit Name	Use	See	Default Value
3	–	–		1 (unprogrammed)
2	BOOTSZ1	Sets size of boot loader section	Page 216	1 (unprogrammed)
1	BOOTSZ0			1 (unprogrammed)
0	BOOTRST	Defines boot reset vector	Page 216	1 (unprogrammed)

Table 90. High Fuse Byte

Bit #	Bit Name	Use	See	Default Value
7	RSTDISBL	Disables external reset (1)	Page 79	1 (unprogrammed)
6	DWEN	Enables debugWIRE (1)	Page 212	1 (unprogrammed)
5	SPIEN	Enables serial programming and downloading of data to device (2)		0 (programmed) (3)
4	WDTON	Sets watchdog timer permanently on	Page 46	1 (unprogrammed)
3	EESAVE	Preserves EEPROM memory during Chip Erase operation	Page 235	1 (unprogrammed) (4)
2	BODLEVEL2	Sets BOD trigger level	Page 251	1 (unprogrammed)
1	BODLEVEL1			1 (unprogrammed)
0	BODLEVEL0			1 (unprogrammed)

Notes:
1. Programming this fuse bit will change the functionality of the $\overline{\text{RESET}}$ pin and render further programming via the serial interface impossible. The fuse bit can be unprogrammed using the parallel programming algorithm (see page 232).
2. This fuse bit is not accessible in serial programming mode.
3. This setting enables SPI programming.
4. This setting does not preserve EEPROM.

Table 91. Low Fuse Byte

Bit #	Bit Name	Use	See	Default Value
7	CKDIV8	Divides clock by 8 (1)	Page 30	0 (programmed)
6	CKOUT	Outputs system clock on port pin	Page 30	1 (unprogrammed)
5	SUT1	Sets system start-up time	Page 30	1 (unprogrammed) (2)
4	SUT0			0 (programmed) (2)
3	–	–		1 (unprogrammed)
2	–	–		1 (unprogrammed)
1	CKSEL1	Selects clock source	Page 28	1 (unprogrammed) (3)
0	CKSEL0			0 (programmed) (3)

Note:
1. Unprogramming this fuse at low voltages may result in overclocking. See Section 24.3 on page 249 for device speed versus supply voltage.
2. This setting results in maximum start-up time for the default clock source.

22.3.1 Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode, also when the device is locked.

Signature bytes can also be read by the device firmware. See section “Reading Lock, Fuse and Signature Data from Software” on page 229.

The three signature bytes reside in a separate address space called the device signature imprint table. The signature data for ATtiny828 is given in Table 93.

Table 93. Device Signature Bytes

Part	Signature Byte 0	Signature Byte 1	Signature Byte 0
ATtiny828	0x1E	0x93	0x14

22.3.2 Calibration Bytes

The device signature imprint table of ATtiny828 contains calibration data for the internal oscillators, as shown in Table 92 on page 228. During reset, calibration data is automatically copied to the calibration registers (OSCCAL0, OSCCAL1) to ensure correct frequency of the calibrated oscillators. See “OSCCAL0 – Oscillator Calibration Register” on page 32, and “OSCCAL1 – Oscillator Calibration Register” on page 33.

Calibration bytes can also be read by the device firmware. See section “Reading Lock, Fuse and Signature Data from Software” on page 229.

22.4 Reading Lock, Fuse and Signature Data from Software

Fuse and lock bits can be read by device firmware. Programmed fuse and lock bits read zero. unprogrammed as one. See “Lock Bits” on page 225 and “Fuse Bits” on page 226.

In addition, firmware can also read data from the device signature imprint table. See “Device Signature Imprint Table” on page 228.

22.4.1 Lock Bit Read

Lock bit values are returned in the destination register after an LPM instruction has been issued within three CPU cycles after RWFLB and SPMEN bits have been set in SPMCSR (see page 223). The RWFLB and SPMEN bits automatically clear upon completion of reading the lock bits, or if no LPM instruction is executed within three CPU cycles, or if no SPM instruction is executed within four CPU cycles. When RWFLB and SPMEN are cleared LPM functions normally.

To read the lock bits, follow the below procedure:

1. Load the Z-pointer with 0x0001.
2. Set RWFLB and SPMEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.
4. Read the lock bits from the LPM destination register.

If successful, the contents of the destination register are as follows.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	BLB12	BLB11	BLB02	BLB01	LB2	LB1

See section “Lock Bits” on page 225 for more information.

5. Wait until V_{CC} actually reaches 4.5 – 5.5V before giving any parallel programming commands.
6. Exit programming mode by powering the device down or by bringing $\overline{\text{RESET}}$ pin to 0V.

23.2.2 Considerations for Efficient Programming

Loaded commands and addresses are retained in the device during programming. For efficient programming, the following should be considered.

- When writing or reading multiple memory locations, the command needs only be loaded once
- Do not write the data value 0xFF, since this already is the contents of the entire Flash and EEPROM (unless the EESAVE Fuse is programmed) 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 also applies to reading signature bytes

23.2.3 Chip Erase

A Chip Erase must be performed before the Flash and/or EEPROM are reprogrammed. The Chip Erase command will erase all Flash and EEPROM plus lock bits. If the EESAVE fuse is programmed, the EEPROM is not erased.

Lock bits are not reset until the program memory has been completely erased. Fuse bits are not changed.

The Chip Erase command is loaded as follows:

1. Set XA1, XA0 to “10”. This enables command loading
2. Set BS1 to “0”
3. Set DATA to “1000 0000”. This is the command for Chip Erase
4. Give CLKI a positive pulse. This loads the command
5. Give $\overline{\text{WR}}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{\text{BSY}}$ goes low
6. Wait until RDY/$\overline{\text{BSY}}$ goes high before loading a new command

23.2.4 Programming the Flash

Flash is organized in pages, as shown in Table 94 on page 232. When programming the Flash, the program data is first 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:

A. Load Command “Write Flash”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “0001 0000”. This is the command for Write Flash.
4. Give CLKI a positive pulse. This loads the command.

B. Load Address Low byte

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “0”. This selects low address.
3. Set DATA = Address low byte (0x00 – 0xFF).
4. Give CLKI a positive pulse. This loads the address low byte.

C. Load Data Low Byte

1. Set XA1, XA0 to “01”. This enables data loading.
2. Set DATA = Data low byte (0x00 – 0xFF).

25.2 Current Consumption in Idle Mode

Figure 114. Idle Supply Current vs. Low Frequency (0.1 - 1.0 MHz)

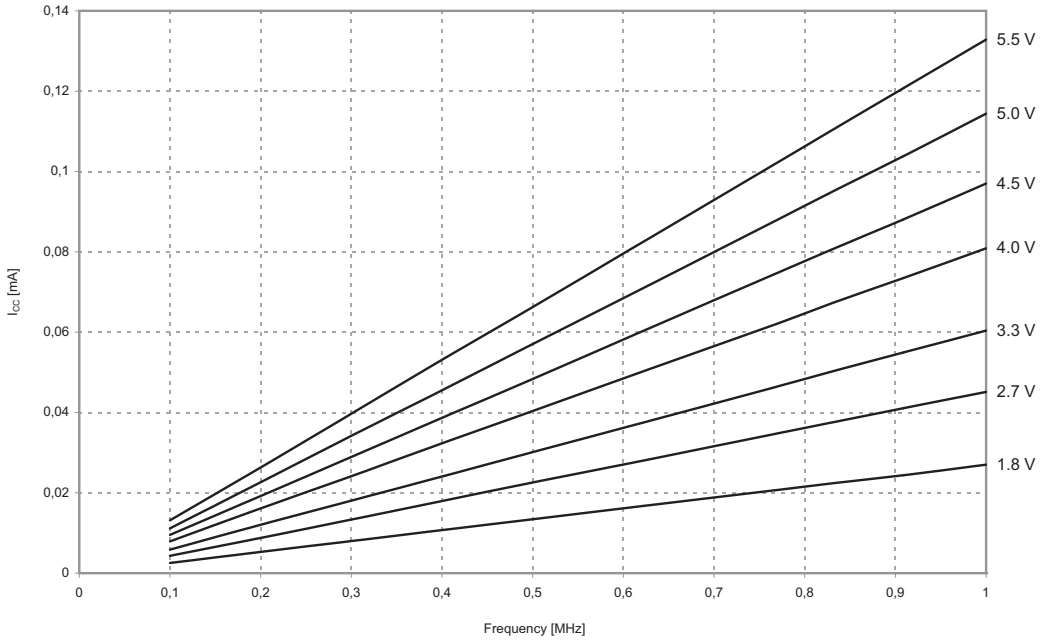


Figure 115. Idle Supply Current vs. Frequency (1 - 20 MHz)

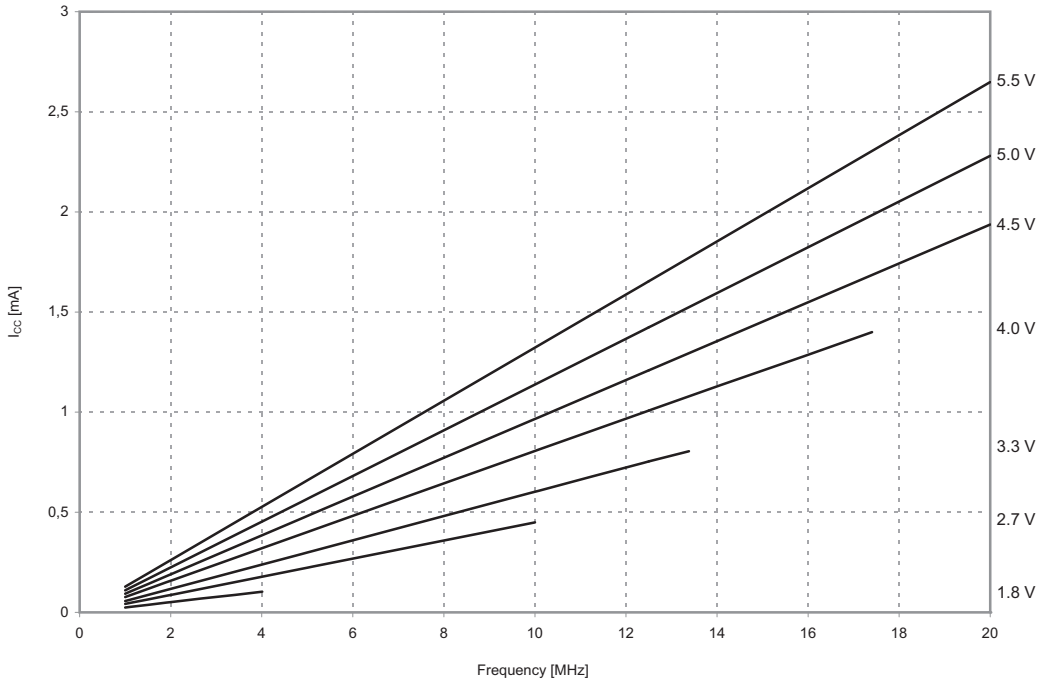


Figure 168. Sampled BOD Threshold vs Temperature (BODLEVEL = 4.3V)

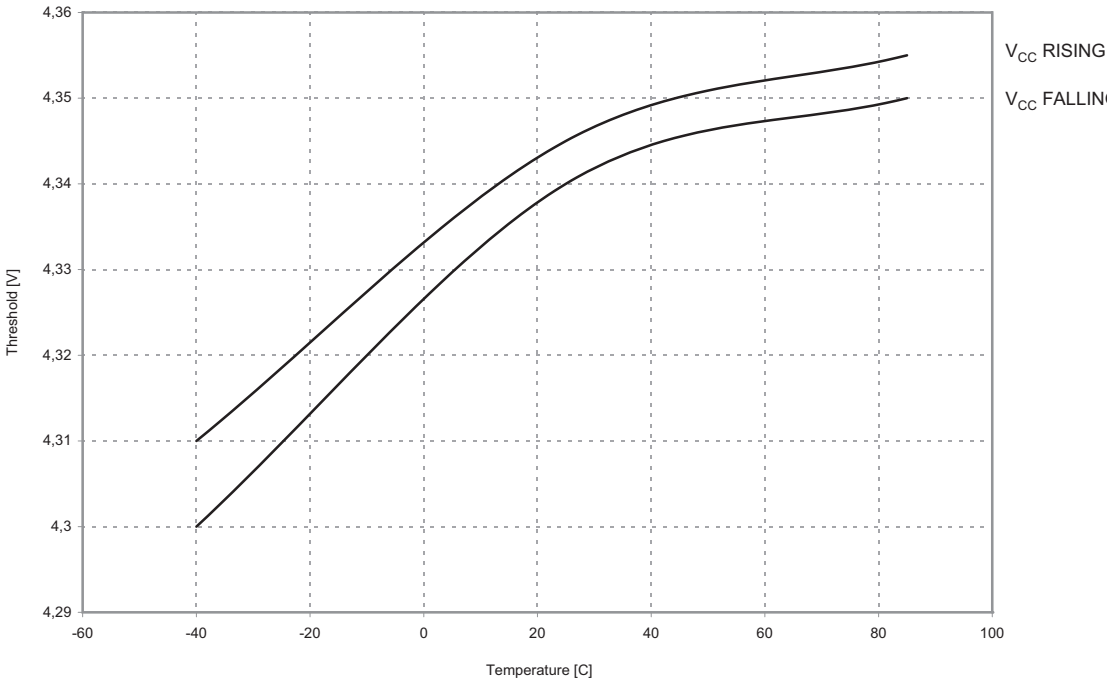


Figure 169. Sampled BOD Threshold vs Temperature (BODLEVEL = 2.7V)

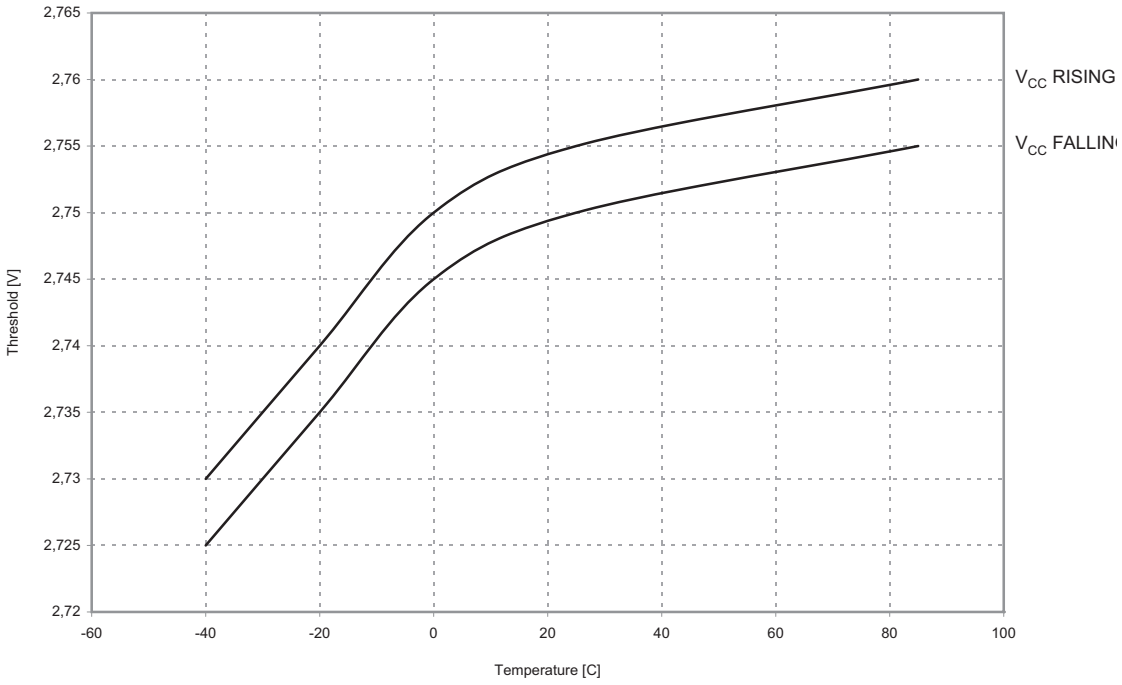


Figure 176. Analog Comparator Offset vs. V_{CC} (V_{IN} = 1.1V)

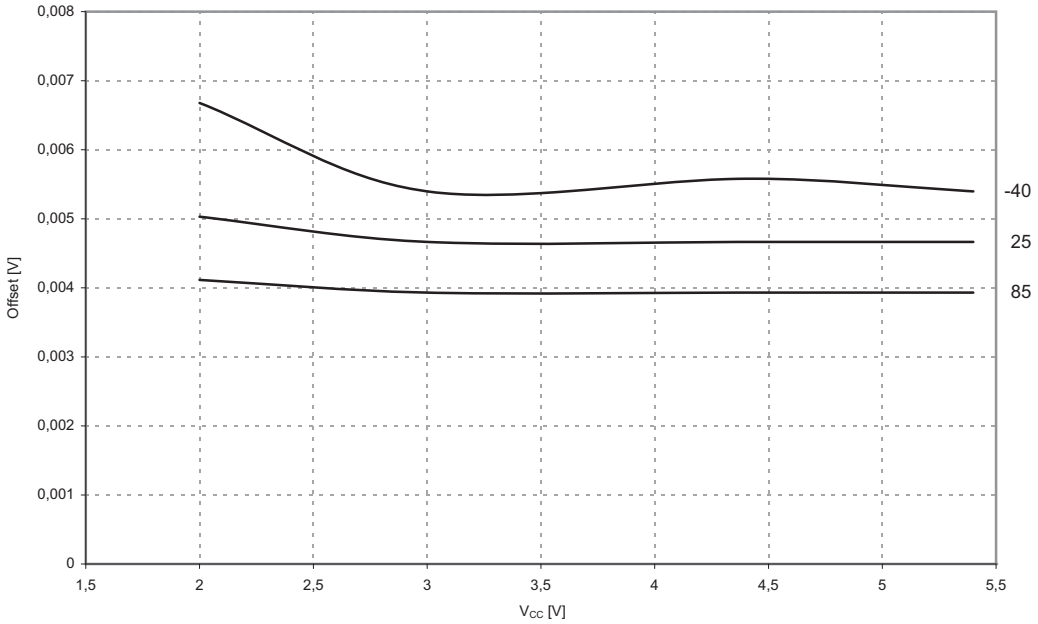
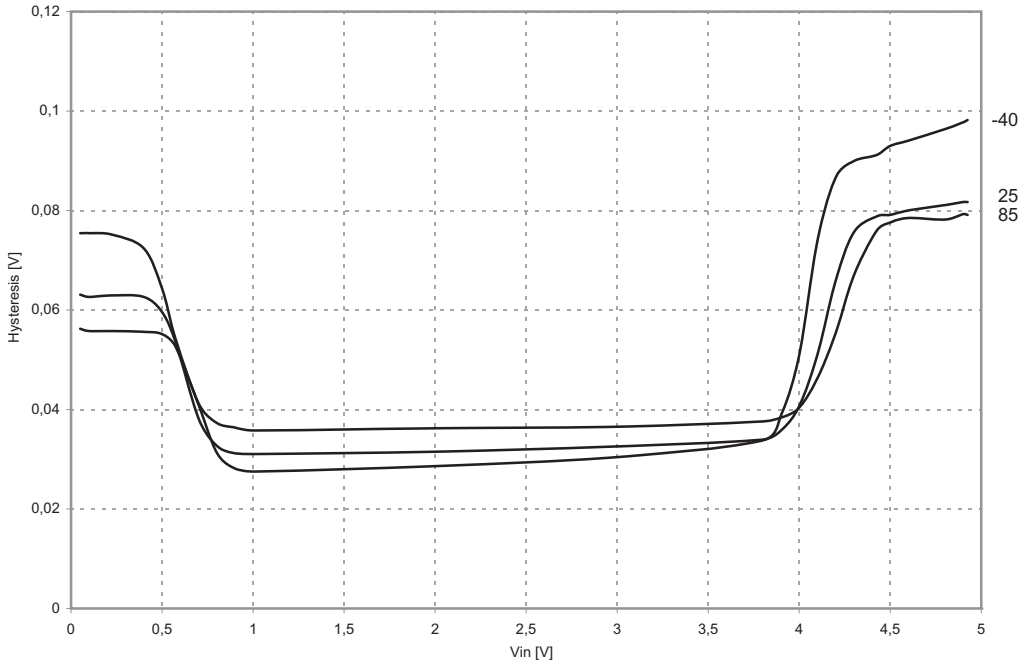


Figure 177. Analog Comparator Hysteresis vs. V_{IN} (V_{CC} = 5.0V)



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page(s)
0x15 (0x35)	TIFR0	–	–	–	–	–	OCF0B	OCF0A	TOV0	Page 103
0x14 (0x34)	PHDE	–	–	–	–	–	PHDEC	–	–	Page 81
0x13 (0x33)	Reserved	–	–	–	–	–	–	–	–	
0x12 (0x32)	Reserved	–	–	–	–	–	–	–	–	
0x11 (0x31)	Reserved	–	–	–	–	–	–	–	–	
0x10 (0x30)	Reserved	–	–	–	–	–	–	–	–	
0x0F (0x2F)	PUED	–	–	–	–	PUED3	PUED2	PUED1	PUED0	Page 82
0x0E (0x2E)	PORTD	–	–	–	–	PORTD3	PORTD2	PORTD1	PORTD0	Page 82
0x0D (0x2D)	DDRD	–	–	–	–	DDD3	DDD2	DDD1	DDD0	Page 82
0x0C (0x2C)	PIND	–	–	–	–	PIND3	PIND2	PIND1	PIND0	Page 83
0x0B (0x2B)	PUEC	PUEC7	PUEC6	PUEC5	PUEC4	PUEC3	PUEC2	PUEC1	PUEC0	Page 83
0x0A (0x2A)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	Page 83
0x09 (0x29)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	Page 83
0x08 (0x28)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	Page 84
0x07 (0x27)	PUEB	PUEB7	PUEB6	PUEB5	PUEB4	PUEB3	PUEB2	PUEB1	PUEB0	Page 84
0x06 (0x26)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	Page 84
0x05 (0x25)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	Page 84
0x04 (0x24)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	Page 85
0x03 (0x23)	PUEA	PUEA7	PUEA6	PUEA5	PUEA4	PUEA3	PUEA2	PUEA1	PUEA0	Page 85
0x02 (0x22)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	Page 85
0x01 (0x21)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	Page 85
0x00 (0x20)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	Page 86

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.