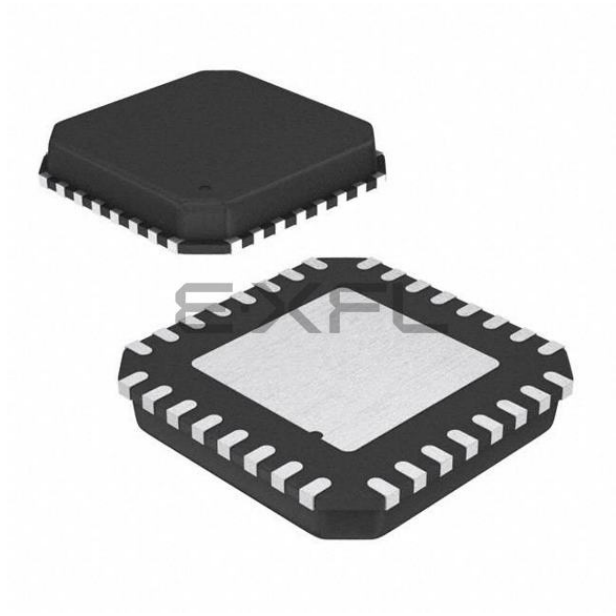


Welcome to E-XFL.COM

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-VFQFN Exposed Pad
Supplier Device Package	32-VQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny828r-mu

Figure 5. General Purpose Working Registers

7 … 0		
R0	0x00	
R1	0x01	
R2	0x02	
R3	0x03	
…	...	
R12	0x0C	
R13	0x0D	
R14	0x0E	
R15	0x0F	
R16	0x10	
R17	0x11	
…	...	
R26	0x1A	X-register Low Byte
R27	0x1B	X-register High Byte
R28	0x1C	Y-register Low Byte
R29	0x1D	Y-register High Byte
R30	0x1E	Z-register Low Byte
R31	0x1F	Z-register High Byte

Most of the instructions operating on the Register File are single cycle instructions with direct access to all registers.

As shown in Figure 5, each register is also assigned a Data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y- and Z-pointer registers can be set to index any register in the file.

4.4.1 The X-register, Y-register, and Z-register

The registers R26..R31 have added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the data space. The three indirect address registers X, Y, and Z are defined as described in Figure 6 below.

Figure 6. The X-, Y-, and Z-registers

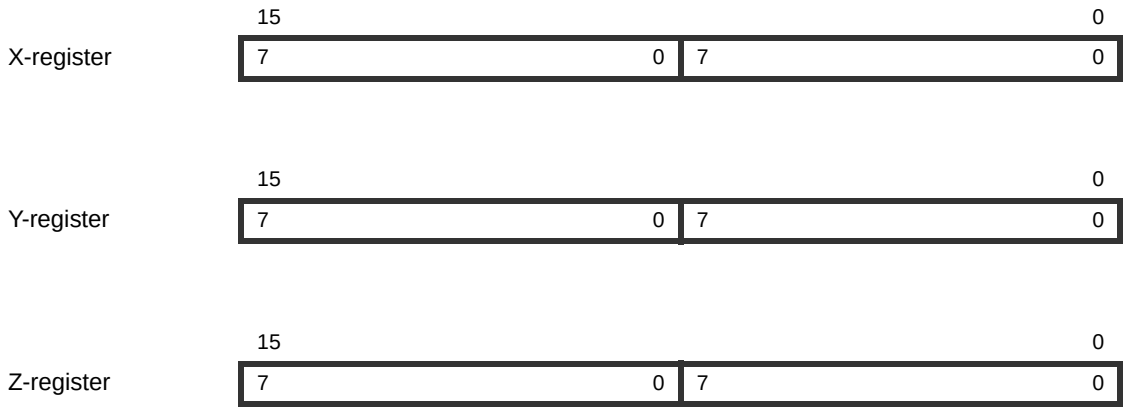
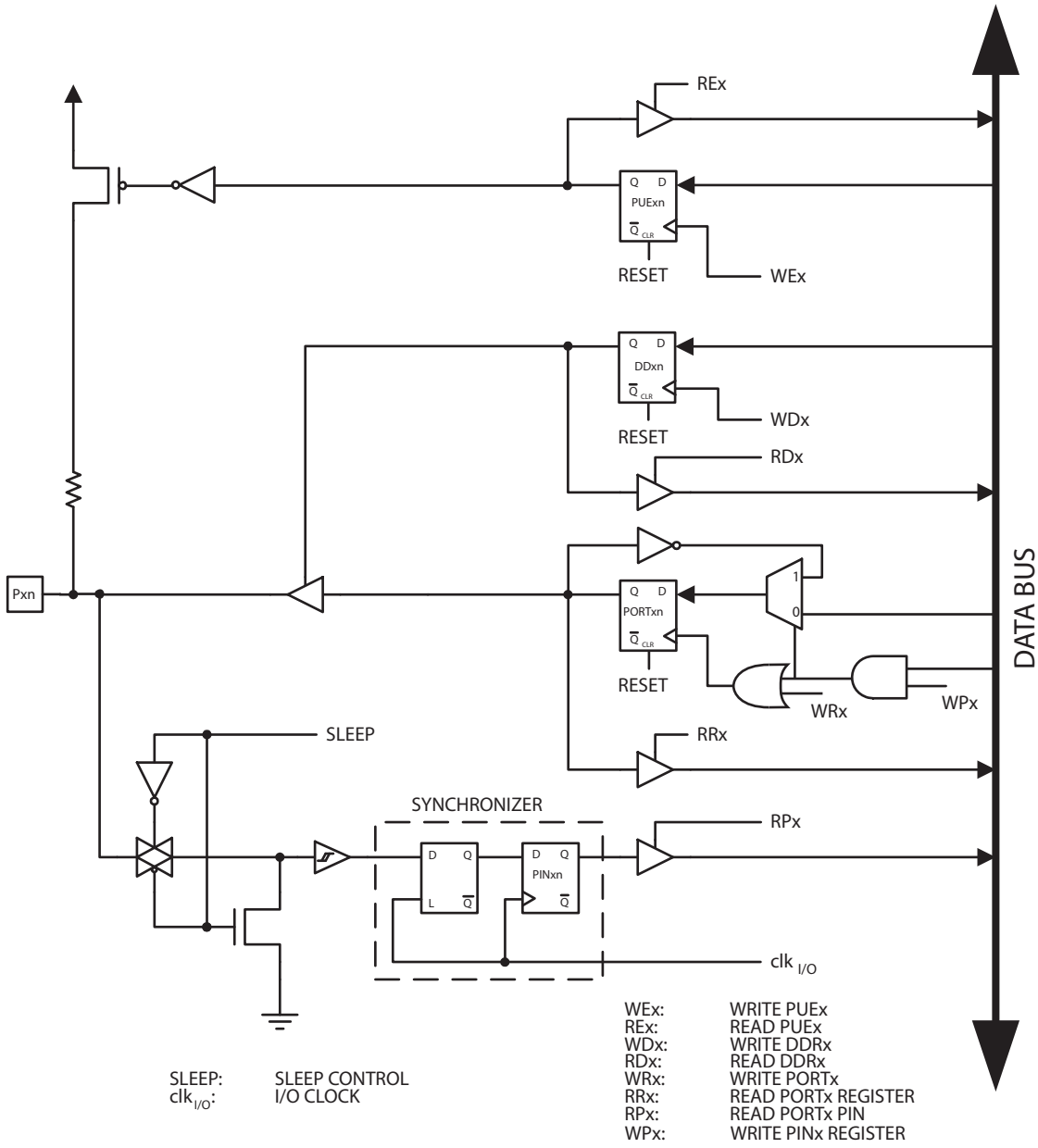


Figure 22. General Digital I/O[1]



Note: 1. 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.

10.2.1 Configuring the Pin

Each port pin consists of four register bits: DDxn, PORTxn, PUExn, and PINxn. As shown in “Register Description” on page 81, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, the PUExn bits at the PUEx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

Pin	Signal	Composition
PB1	PUOE	0
	PUOV	0
	DDOE	0
	DDOV	0
	PVOE	0
	PVOV	0
	PTOE	0
	DIEOE	(PCINT9 • PCIE1) + ADC9D
	DIEOV	PCINT9 • PCIE1
	DI	PCINT9 Input
	AIO	ADC9 Input
PB2	PUOE	0
	PUOV	0
	DDOE	0
	DDOV	0
	PVOE	0
	PVOV	0
	PTOE	0
	DIEOE	(PCINT10 • PCIE1) + ADC10D
	DIEOV	PCINT10 • PCIE1
	DI	PCINT10 Input
	AIO	ADC10 Input

10.4.13 PINB – Port B Input Pins

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

- **Bits 7:0 – PINB[7:0]: Port Input Data**

Regardless of the setting of the data direction bit, the value of the port pin PBn can be read through the PINBn bit. Writing a logic one to PINBn toggles the value of PORTBn, regardless of the value in DDBn.

10.4.14 PUEA – Port A Pull-Up Enable Control Register

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PUEA7	PUEA6	PUEA5	PUEA4	PUEA3	PUEA2	PUEA1	PUEA0	PUEA
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	

- **Bits 7:0 – PUEA[7:0]: Pull-Up Enable Bits**

When a pull-up enable bit, PUEAn, is set the pull-up resistor on the equivalent port pin, PAn, is enabled.

10.4.15 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x02 (0x22)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
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	

- **Bits 7:0 – PORTA[3:0]: Port Data Bits**

When pin PAn is configured as an output, setting PORTAn will drive PAn high. Clearing PORTAn will drive PAn low. When the pin is configured as an input the value of the PORTxn bit doesn't matter. See Table 19 on page 61.

10.4.16 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x01 (0x21)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
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	

- **Bits 7:0 – DDA[7:0]: Data Direction Bits**

When DDAn is set, the pin PAn is configured as an output. When DDAn is cleared, the pin is configured as an input.

At the very start of period 2 in Figure 32 on page 95 OCnx has a transition from high to low even though there is no Compare Match. The point of this transition is to guaratee symmetry around BOTTOM. There are two cases that give a transition without Compare Match.

- OCR0x changes its value from TOP, like in Figure 32 on page 95. When the OCR0x value is TOP the OCnx pin value is the same as the result of a down-counting Compare Match. To ensure symmetry around BOTTOM the OCnx value at TOP must correspond to the result of an up-counting Compare Match.
- The timer starts counting from a value higher than the one in OCR0x, and for that reason misses the Compare Match and hence the OCnx change that would have happened on the way up.

11.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 33 on page 96 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 33. Timer/Counter Timing Diagram, no Prescaling

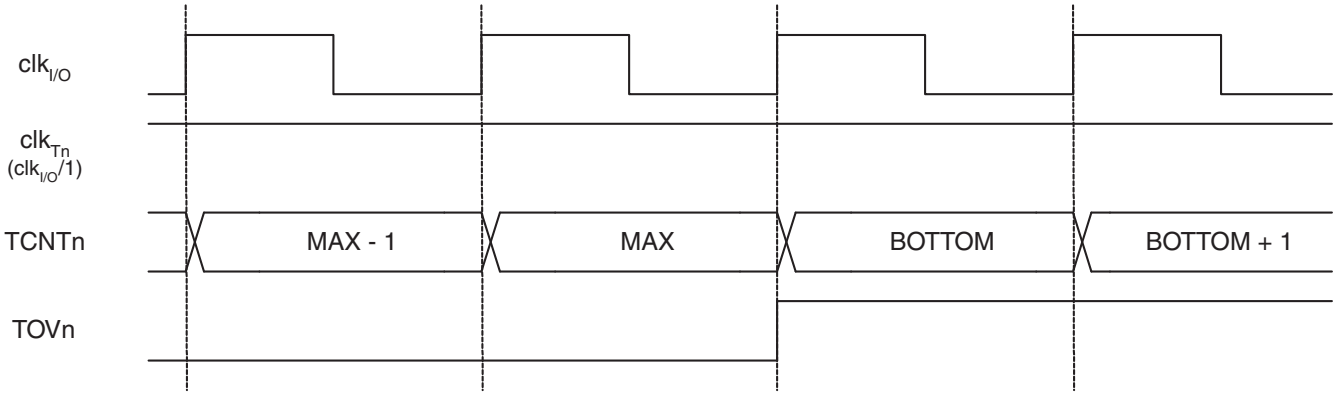


Figure 34 on page 96 shows the same timing data, but with the prescaler enabled.

Figure 34. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}$/8)

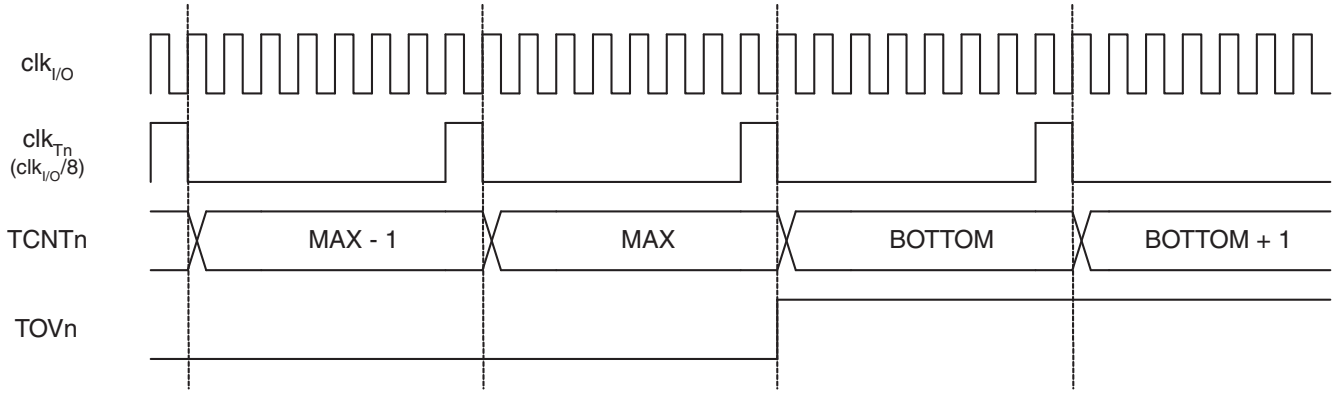


Figure 35 on page 97 shows the setting of OCF0B in all modes and OCF0A in all modes except CTC mode and PWM mode, where OCR0A is TOP.

12.11.3 TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
(0x82)	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	W	W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A: Force Output Compare for Channel A**
- **Bit 6 – FOC1B: Force Output Compare for Channel B**

The FOC1A/FOC1B bits are only active when the WGM1[3:0] bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written when operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the Waveform Generation unit. The OC1A/OC1B output is changed according to its COM1x[1:0] bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x[1:0] bits that determine the effect of the forced compare.

A FOC1A/FOC1B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCR1A as TOP.

The FOC1A/FOC1B bits are always read as zero.

- **Bits 5:0 – Res: Reserved Bit**

These bits are reserved for future use. To ensure compatibility with future devices, these bits must be set to zero when the register is written.

12.11.4 TOCPMSA1 and TOCPMSA0 – Timer/Counter Output Compare Pin Mux Selection Registers

Bit	7	6	5	4	3	2	1	0	
(0xE9)	TOCC7S1	TOCC7S0	TOCC6S1	TOCC6S0	TOCC5S1	TOCC5S0	TOCC4S1	TOCC4S0	TOCPMSA1
(0xE8)	TOCC3S1	TOCC3S0	TOCC2S1	TOCC2S0	TOCC1S1	TOCC1S0	TOCC0S1	TOCC0S0	TOCPMSA0
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	

- **Bits 7:0 – TOCCnS1 and TOCCnS0: Timer/Counter Output Compare Channel Select**

TOCCnS1 and TOCCnS bits select which Timer/Counter compare output is routed to the corresponding TOCCn pin. The two timer/counters provide four possible compare outputs that can be routed to output pins, as shown in the table below.

Table 44. Selecting Timer/Counter Compare Output for TOCCn Pins

TOCCnS1	TOCCnS0	TOCCn Output (1)
0	0	OC0A
0	1	OC0B
1	0	OC1A
1	1	OC1B

Note: 1. See “Alternative Functions of Port C” on page 73.

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).

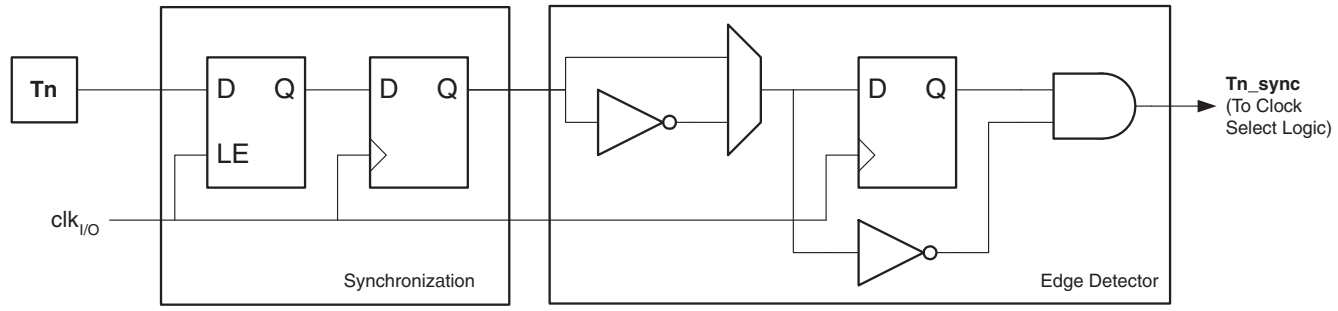
It is possible to use the Prescaler Reset for synchronizing the Timer/Counter to program execution.

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.

- **Bit 0 – MUX5: Analog Channel and Gain Selection Bit**

This bit together with MUX[4:0] in ADMUXA select which analog input is connected to the ADC. See Table 51 on page 149.

15.13.3 ADCL and ADCH – ADC Data Register

15.13.3.1ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	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	

15.13.3.2ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	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 and the MUX bits 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.

- **ADC[9:0]: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in “ADC Conversion Result” on page 148.

15.13.4 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
	/* Set baud rate */
	UBRRH = (unsigned char)(baud>>8);
	UBRRL = (unsigned char)baud;

	/* Enable receiver and transmitter */
	UCSRB = (1<<RXEN)|(1<<TXEN);

	/* Set frame format: 8 data bits, 2 stop bits */
	UCSRC = (1<<USBS)|(3<<UCSZ0);
}
```

Note: 1. See “Code Examples” on page 7.

More advanced initialization routines can be made that include frame format as parameters, disable interrupts and so on. However, many applications use a fixed setting of the baud and control registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

17.6 Data Transmission – The USART Transmitter

The USART transmitter is enabled by setting the Transmit Enable bit (TXEN) (see “UCSRB – USART Control and Status Register B” on page 185). When the transmitter is enabled, the normal port operation of the TxDn pin is overridden by the USART and given the function as the transmitter’s serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCK pin will be overridden and used as transmission clock.

17.6.1 Sending Frames with 5 to 8 Data Bits

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDR register. The buffered data in the transmit buffer will be moved to the shift register when the it is ready to send a new frame. The shift register is loaded with new data if it is in idle state (no ongoing transmission), or immediately after the last stop bit of the previous frame is transmitted. When the shift register is loaded with new data, it will transfer one complete frame at the rate given by the Baud Rate Register, the U2X bit or by XCK, depending on the mode of operation.

The following code examples show a simple USART transmit function based on polling of the Data Register Empty flag (UDRE). When using frames with less than eight bits, the most significant bits written to UDR are ignored. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in register R16

Assembly Code Example[1]

```
USART_Transmit:
	; Wait for empty transmit buffer
	sbis		UCSRA,UDRE
	rjmp		USART_Transmit

	; Put data (r16) into buffer, sends the data
	out		UDR,r16
	ret
```

17.6.5 Disabling the Transmitter

Clearing TXEN will disable the transmitter but the change will not become effective before any ongoing and pending transmissions are completed, i.e. not before the transmit shift register and transmit buffer register are cleared of data to be transmitted. When disabled, the transmitter will no longer override the TxD pin.

17.7 Data Reception – The USART Receiver

The USART receiver is enabled by writing the Receive Enable bit (RXEN) (see “UCSRB – USART Control and Status Register B” on page 185). When the receiver is enabled, the normal operation of the RxD pin is overridden by the USART and given the function as the receiver’s serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCK pin will be used as transfer clock.

17.7.1 Receiving Frames with 5 to 8 Data Bits

The receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate, or XCK clock, and then shifted into the receive shift register until the first stop bit of a frame is received. A second stop bit will be ignored by the receiver. When the first stop bit is received, i.e., a complete serial frame is present in the receive shift register, the contents of it will be moved into the receive buffer. The receive buffer can then be read by reading UDR.

The following code example shows a simple USART receive function based on polling of the Receive Complete flag (RXC). When using frames with less than eight bits the most significant bits of the data read from the UDR will be masked to zero. The USART has to be initialized before the function can be used.

Assembly Code Example[1]

```
USART_Receive:
      ; Wait for data to be received
      sbis              UCSRA, RXC
      rjmp              USART_Receive

      ; Get and return received data from buffer
      in                r16, UDR
      ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
      /* Wait for data to be received */
      while ( !(UCSRA & (1<<RXC)) )
                        ;

      /* Get and return received data from buffer */
      return UDR;
}
```

Note: 1. See “Code Examples” on page 7.

The function simply waits for data to be present in the receive buffer by checking the RXC flag, before reading the buffer and returning the value.

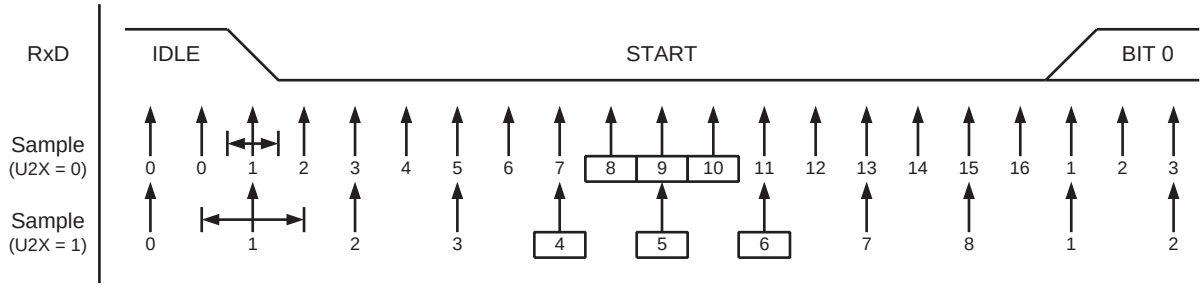
17.8 Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxD pin. The data recovery logic samples and low pass filters each incoming bit, thereby improving the noise immunity of the receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

17.8.1 Asynchronous Clock Recovery

The clock recovery logic synchronizes the internal clock to the incoming serial frames. Figure 74 illustrates the sampling process of the start bit of an incoming frame. In normal mode the sample rate is 16 times the baud rate, in double speed mode eight times. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the double speed mode of operation (U2X = 1). Samples denoted zero are samples done when the RxD line is idle (i.e., no communication activity).

Figure 74. Start Bit Sampling

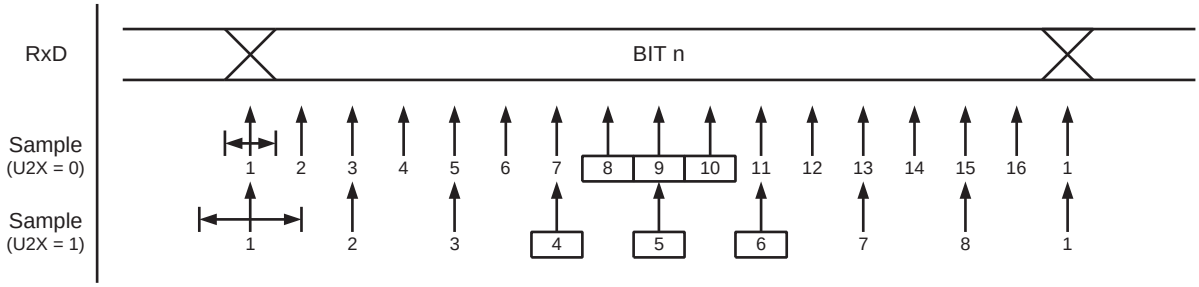


When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. In Figure 74, samples are indicated with numbers inside boxes and sample number 1 denotes the first zero-sample. The clock recovery logic then uses samples 8, 9, and 10 (in normal mode), or samples 4, 5, and 6 (in double speed mode), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the receiver starts looking for the next high to low-transition. If, however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

17.8.2 Asynchronous Data Recovery

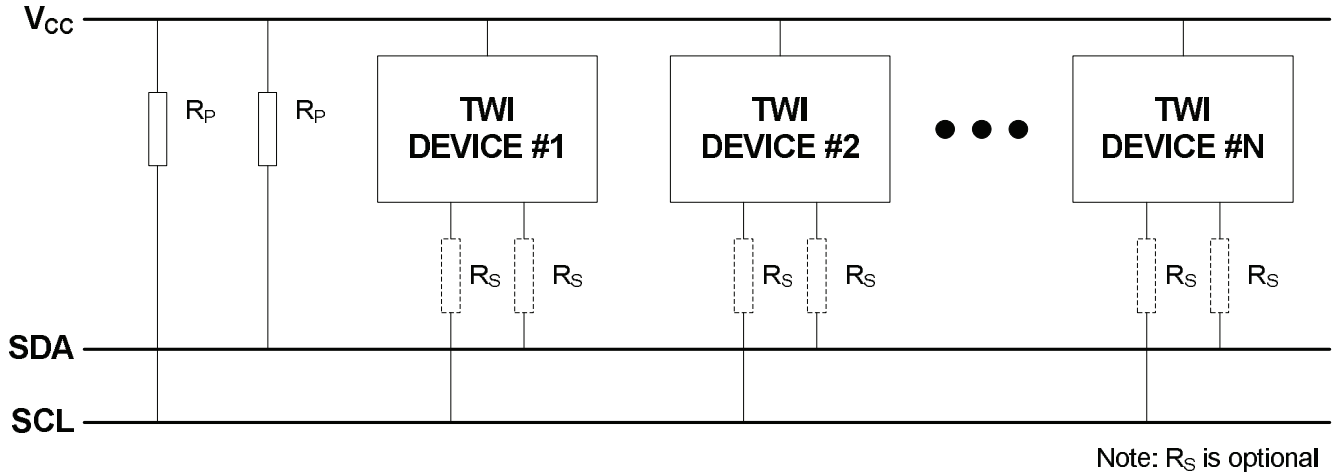
When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in normal mode and eight states for each bit in double speed mode. Figure 75 shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

Figure 75. Sampling of Data and Parity Bit



The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit. In the figure, the center samples are emphasized by having the sample number inside

Figure 78. TWI Bus Topology



A unique address is assigned to all slave devices connected to the bus, and the master will use this to address a slave and initiate a data transaction. 7-bit or 10-bit addressing can be used.

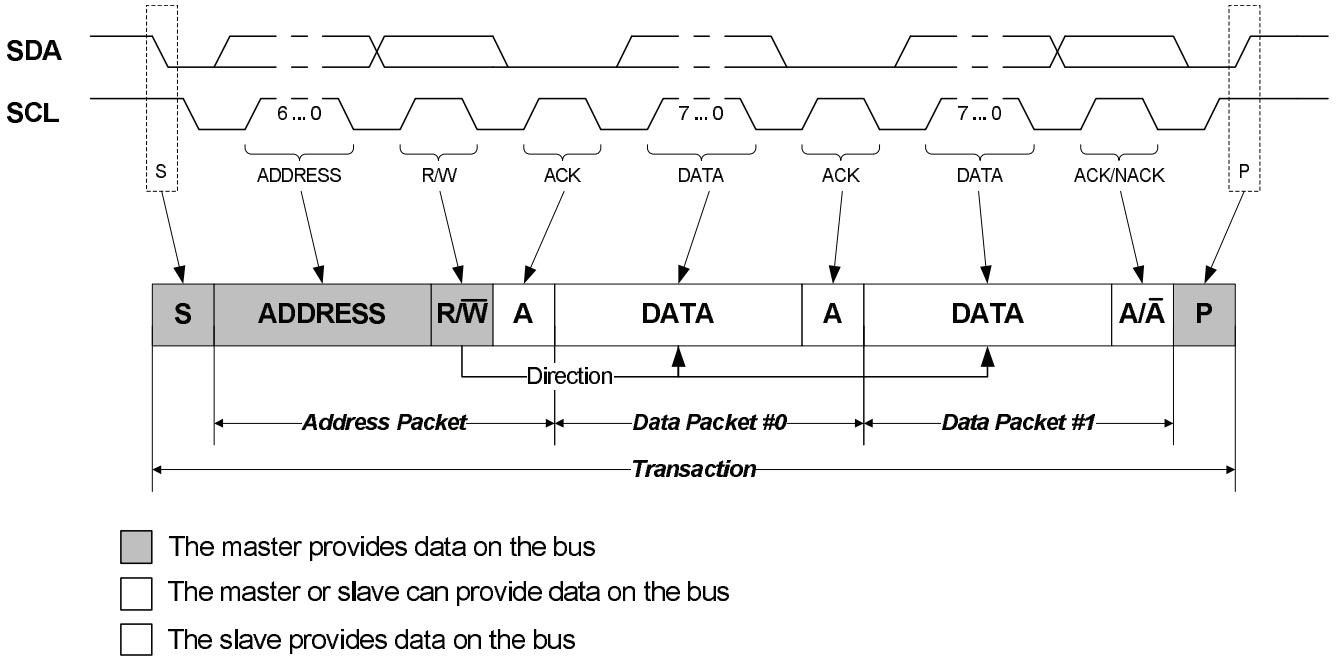
Several masters can be connected to the same bus, and this is called a multi-master environment. An arbitration mechanism is provided for resolving bus ownership between masters since only one master device may own the bus at any given time.

A device can contain both master and slave logic, and can emulate multiple slave devices by responding to more than one address.

A master indicates the start of transaction by issuing a START condition (S) on the bus. An address packet with a slave address (ADDRESS) and an indication whether the master wishes to read or write data (R/$\overline{W}$), is then sent. After all data packets (DATA) are transferred, the master issues a STOP condition (P) on the bus to end the transaction. The receiver must acknowledge (A) or not-acknowledge ($\overline{A}$) each byte received.

Figure 79 shows a TWI transaction.

Figure 79. Basic TWI Transaction Diagram Topology



19.4.1 Receiving Address Packets

When the TWI slave is properly configured, it will wait for a START condition to be detected. When this happens, the successive address byte will be received and checked by the address match logic, and the slave will ACK the correct address. If the received address is not a match, the slave will not acknowledge the address and wait for a new START condition.

The slave Address/Stop Interrupt Flag is set when a START condition succeeded by a valid address packet is detected. A general call address will also set the interrupt flag.

A START condition immediately followed by a STOP condition, is an illegal operation and the Bus Error flag is set.

The R/W Direction flag reflects the direction bit received with the address. This can be read by software to determine the type of operation currently in progress.

Depending on the R/W direction bit and bus condition one of four distinct cases (1 to 4) arises following the address packet. The different cases must be handled in software.

19.4.1.1 Case 1: Address packet accepted - Direction bit set

If the R/$\overline{W}$ Direction flag is set, this indicates a master read operation. The SCL line is forced low, stretching the bus clock. If ACK is sent by the slave, the slave hardware will set the Data Interrupt Flag indicating data is needed for transmit. If NACK is sent by the slave, the slave will wait for a new START condition and address match.

19.4.1.2 Case 2: Address packet accepted - Direction bit cleared

If the R/$\overline{W}$ Direction flag is cleared this indicates a master write operation. The SCL line is forced low, stretching the bus clock. If ACK is sent by the slave, the slave will wait for data to be received. Data, Repeated START or STOP can be received after this. If NACK is indicated the slave will wait for a new START condition and address match.

19.4.1.3 Case 3: Collision

If the slave is not able to send a high level or NACK, the Collision flag is set and it will disable the data and acknowledge output from the slave logic. The clock hold is released. A START or repeated START condition will be accepted.

19.4.1.4 Case 4: STOP condition received.

Operation is the same as case 1 or 2 above with one exception. When the STOP condition is received, the Slave Address/Stop flag will be set indicating that a STOP condition and not an address match occurred.

19.4.2 Receiving Data Packets

The slave will know when an address packet with R/$\overline{W}$ direction bit cleared has been successfully received. After acknowledging this, the slave must be ready to receive data. When a data packet is received the Data Interrupt Flag is set, and the slave must indicate ACK or NACK. After indicating a NACK, the slave must expect a STOP or Repeated START condition.

19.4.3 Transmitting Data Packets

The slave will know when an address packet, with R/$\overline{W}$ direction bit set, has been successfully received. It can then start sending data by writing to the Slave Data register. When a data packet transmission is completed, the Data Interrupt Flag is set. If the master indicates NACK, the slave must stop transmitting data, and expect a STOP or Repeated START condition.

19.5.5 TWSD – TWI Slave Data Register

Bit	7	6	5	4	3	2	1	0	
(0xBD)	TWSD[7:0]								TWSD
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 data register is used when transmitting and received data. During transfer, data is shifted from/to the TWSD register and to/from the bus. Therefore, the data register cannot be accessed during byte transfers. This is protected in hardware. The data register can only be accessed when the SCL line is held low by the slave, i.e. when TWCH is set.

When a master reads data from a slave, the data to be sent must be written to the TWSD register. The byte transfer is started when the master starts to clock the data byte from the slave. It is followed by the slave receiving the acknowledge bit from the master. The TWDIF and the TWCH bits are then set.

When a master writes data to a slave, the TWDIF and the TWCH flags are set when one byte has been received in the data register. If Smart Mode is enabled, reading the data register will trigger the bus operation, as set by the TWAA bit in TWSCRB.

Accessing TWSD will clear the slave interrupt flags and the TWCH bit.

19.5.6 TWSAM – TWI Slave Address Mask Register

Bit	7	6	5	4	3	2	1	0	
(0xBB)	TWSAM[7:1]							TWAE	TWSAM
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	

- **Bits 7:1 – TWSAM[7:1]: TWI Address Mask**

These bits can act as a second address match register, or an address mask register, depending on the TWAE setting.

If TWAE is set to zero, TWSAM can be loaded with a 7-bit slave address mask. Each bit in TWSAM can mask (disable) the corresponding address bit in the TWSA register. If the mask bit is one the address match between the incoming address bit and the corresponding bit in TWSA is ignored. In other words, masked bits will always match.

If TWAE is set to one, TWSAM can be loaded with a second slave address in addition to the TWSA register. In this mode, the slave will match on 2 unique addresses, one in TWSA and the other in TWSAM.

- **Bit 0 – TWAE: TWI Address Enable**

By default, this bit is zero and the TWSAM bits acts as an address mask to the TWSA register. If this bit is set to one, the slave address match logic responds to the two unique addresses in TWSA and TWSAM.

Table 114. Parallel Programming Characteristics, T = 25°C, V_{CC} = 5V

Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
t_{DVXH}	Data and Control Valid before CLKI High	67			ns
t_{XLXH}	CLKI Low to CLKI High	200			ns
t_{XHXL}	CLKI Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after CLKI Low	67			ns
t_{XLWL}	CLKI Low to $\overline{WR}$ Low	0			ns
t_{XLPH}	CLKI Low to PAGEL high	0			ns
t_{PLXH}	PAGEL low to CLKI high	150			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			ns
t_{PHPL}	PAGEL Pulse Width High	150			ns
t_{PLBX}	BS1 Hold after PAGEL Low	67			ns
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			ns
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			ns
t_{BVWL}	BS1 Valid to $\overline{WR}$ Low	67			ns
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			ns
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	µs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High[1]	3.7		4.5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase[2]	7.5		9	ms
t_{XLOL}	CLKI Low to $\overline{OE}$ Low	0			ns
t_{BVDV}	BS1 Valid to DATA valid	0		250	ns
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			250	ns
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			250	ns

Notes: 1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.
2. t_{WLRH_CE} is valid for the Chip Erase command.

25.5 Current Consumption of Peripheral Units

Figure 124. Current Consumption of Peripherals at 4MHz vs. V_{CC}

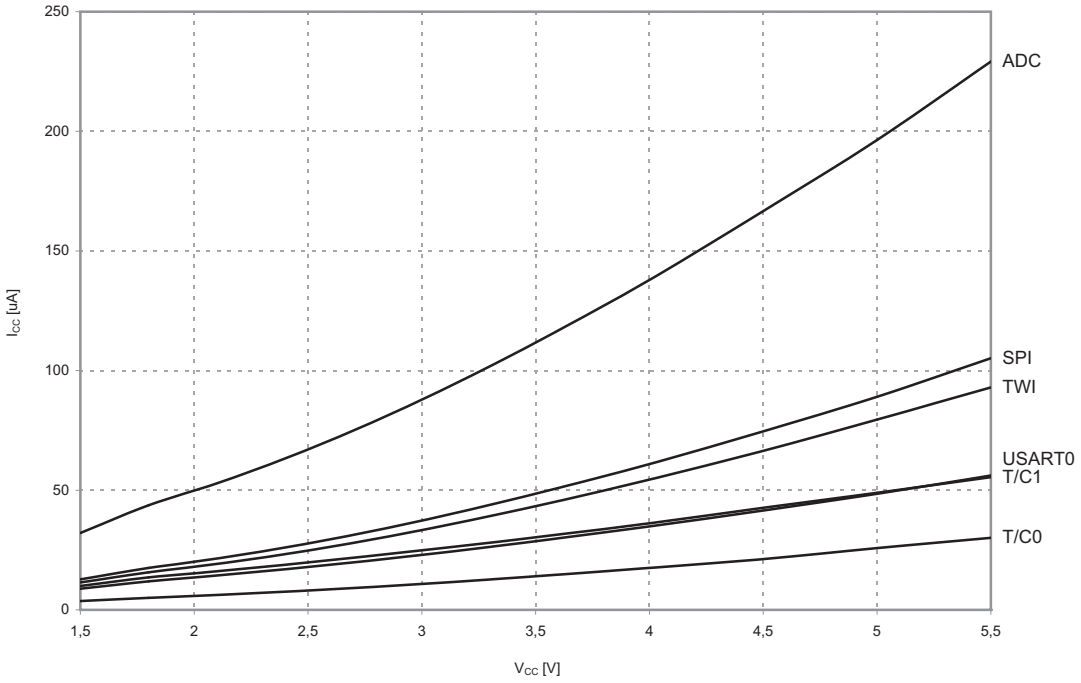


Figure 125. Current Consumption of Peripherals at 3V vs. Frequency

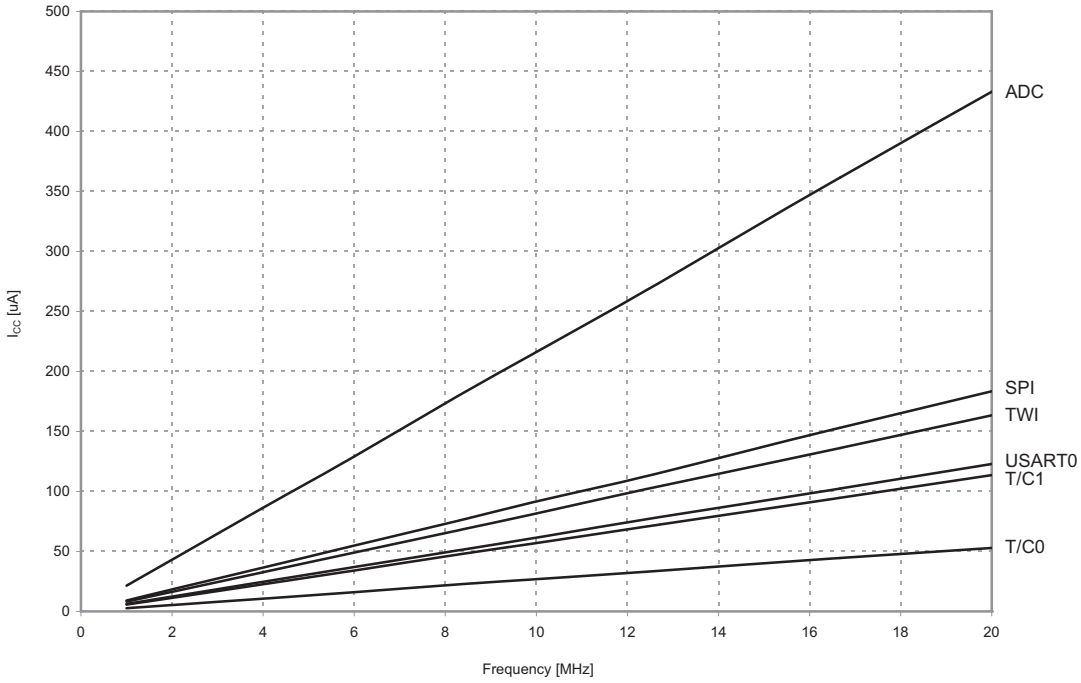
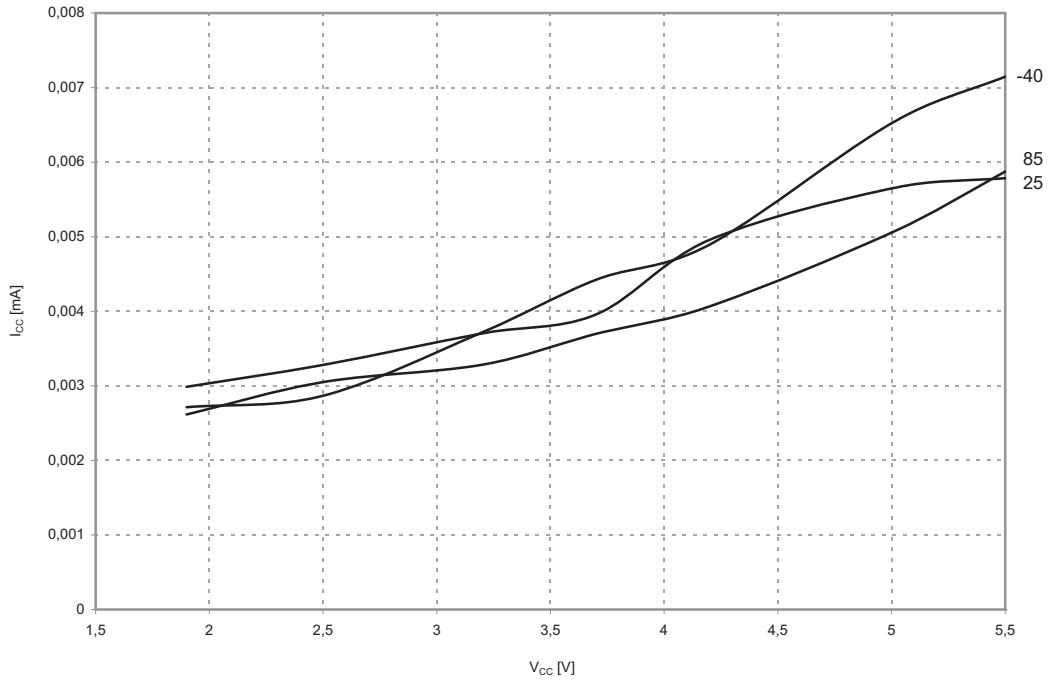


Figure 128. Sampled Brownout Detector Current vs. V_{CC}



25.6 Pull-up Resistors

25.6.1 I/O Pins

Figure 129. I/O pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 1.8V)

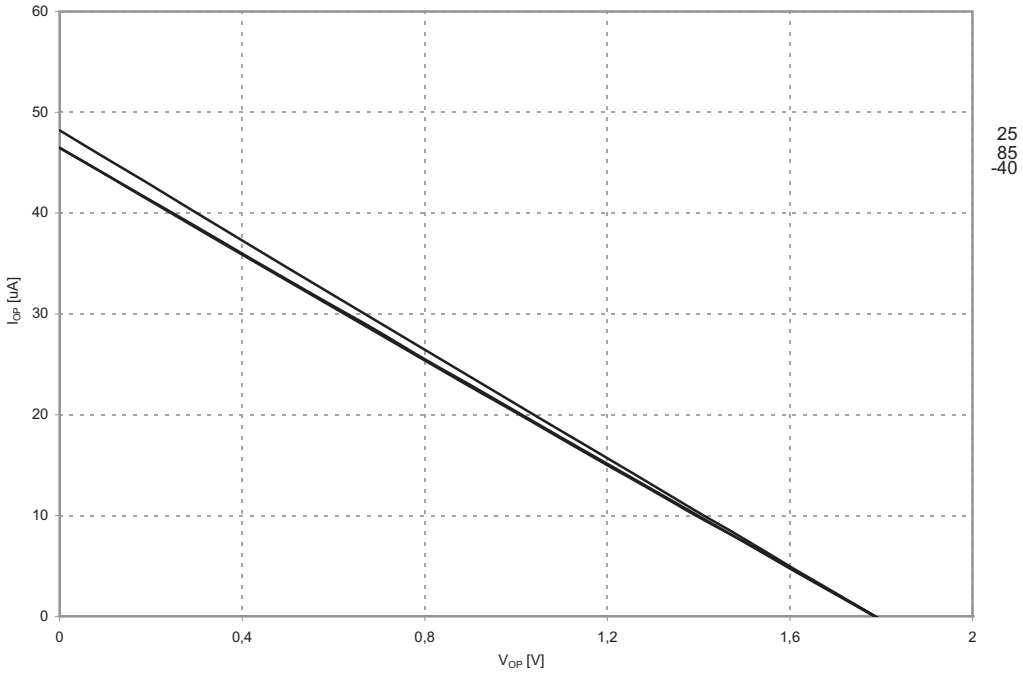


Figure 148. V_{OH}: Output Voltage vs. Source Current (I/O Pin, V_{CC} = 3V)

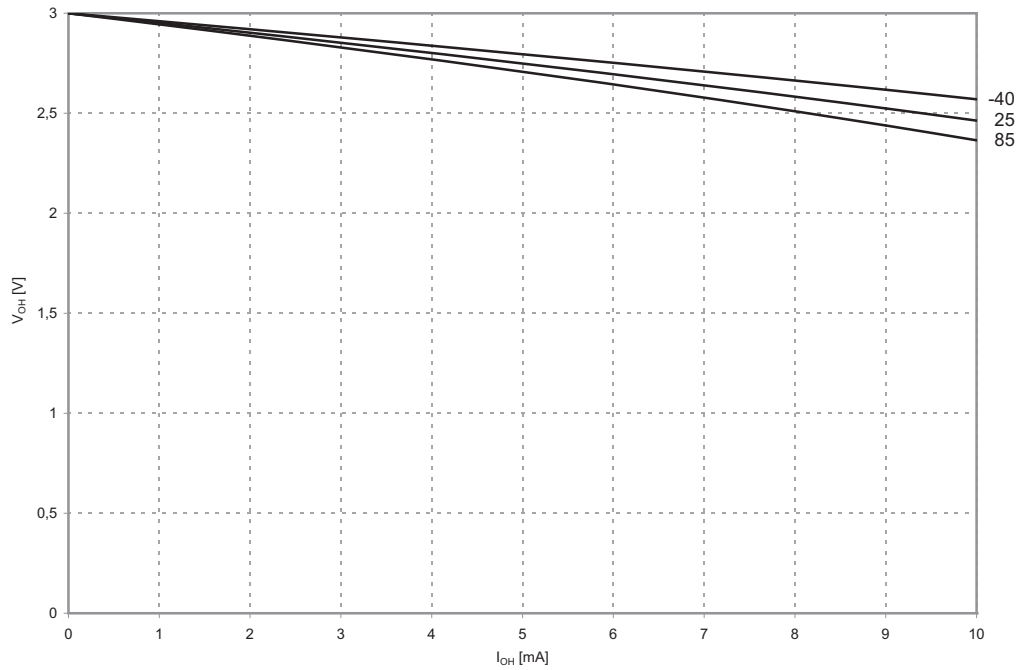


Figure 149. V_{OH}: Output Voltage vs. Source Current (I/O Pin, V_{CC} = 5V)

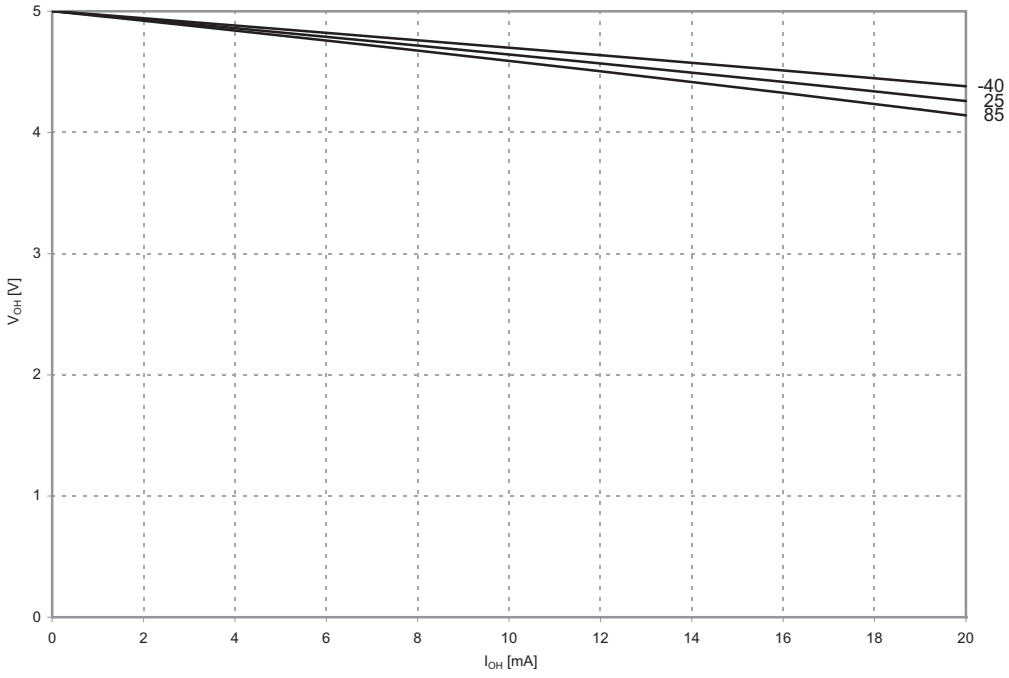


Figure 166. BOD Threshold vs Temperature (BODLEVEL = 2.7V)

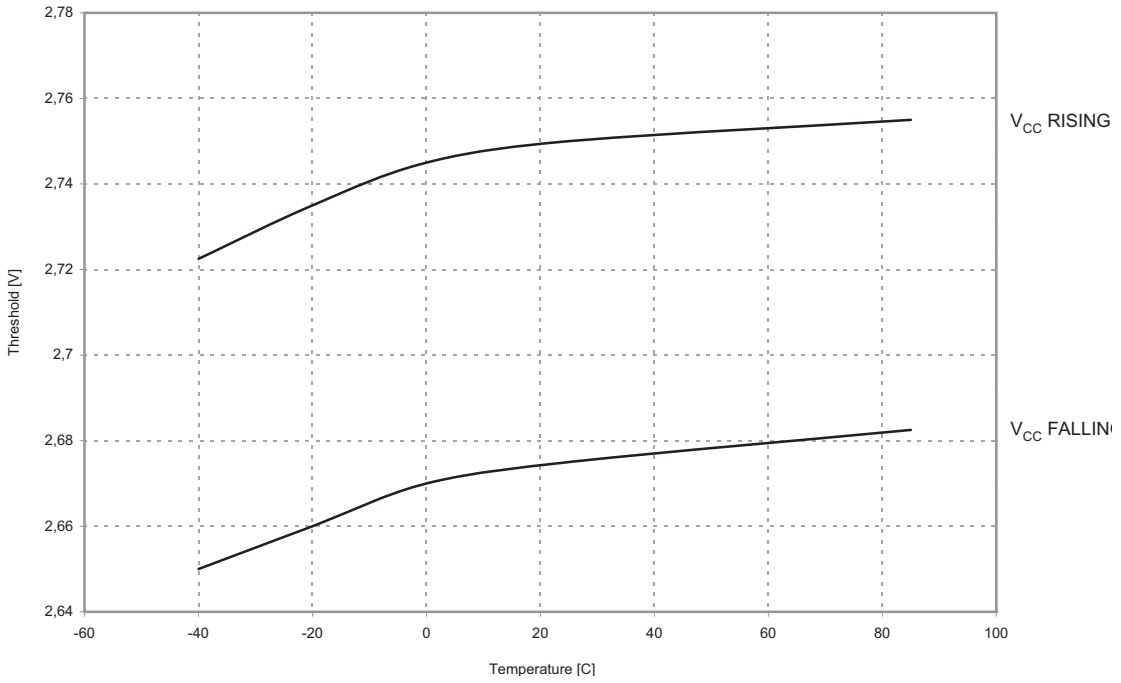


Figure 167. BOD Threshold vs Temperature (BODLEVEL = 1.8V)

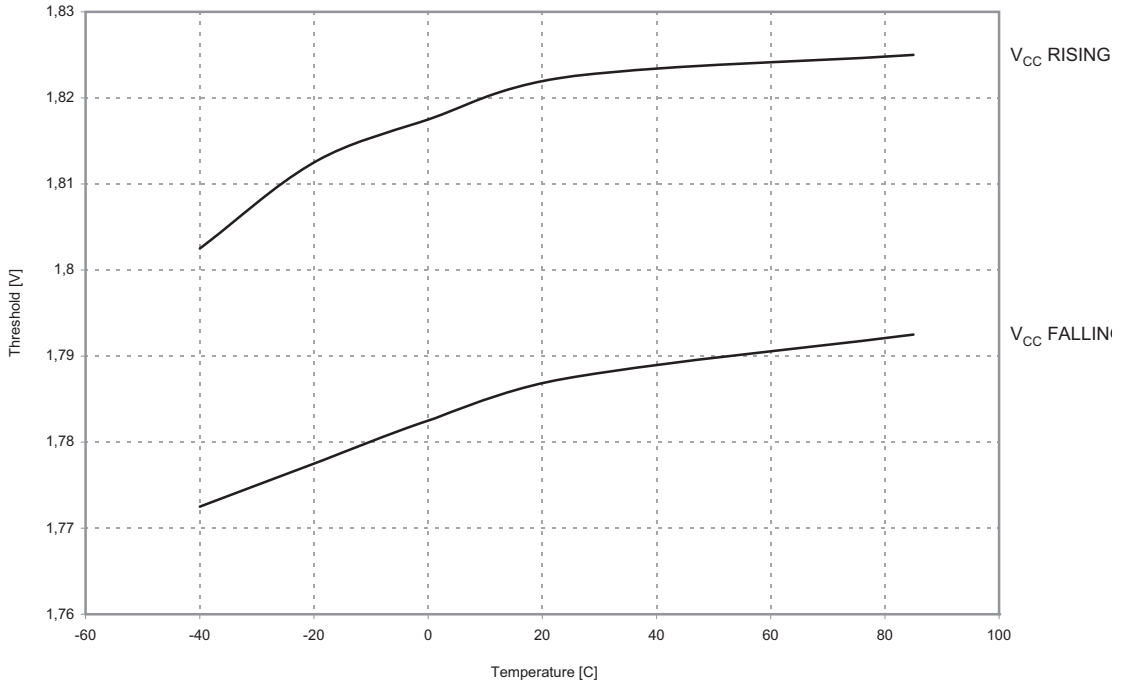
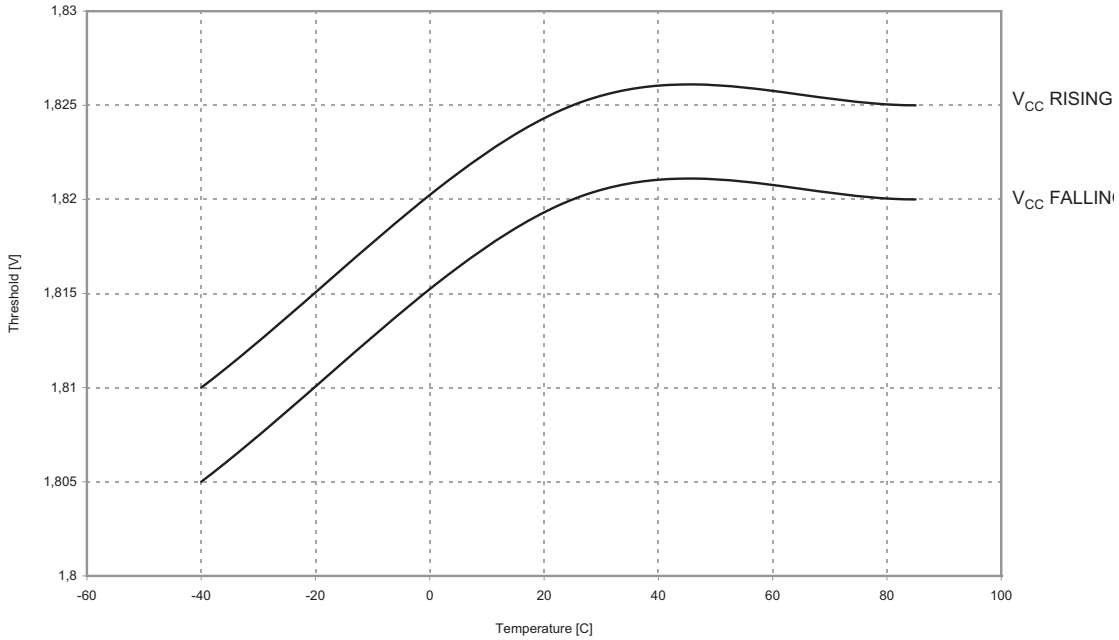


Figure 170. Sampled BOD Threshold vs Temperature (BODLEVEL = 1.8V)



25.11 Bandgap Voltage

Figure 171. Bandgap Voltage vs. Supply Voltage