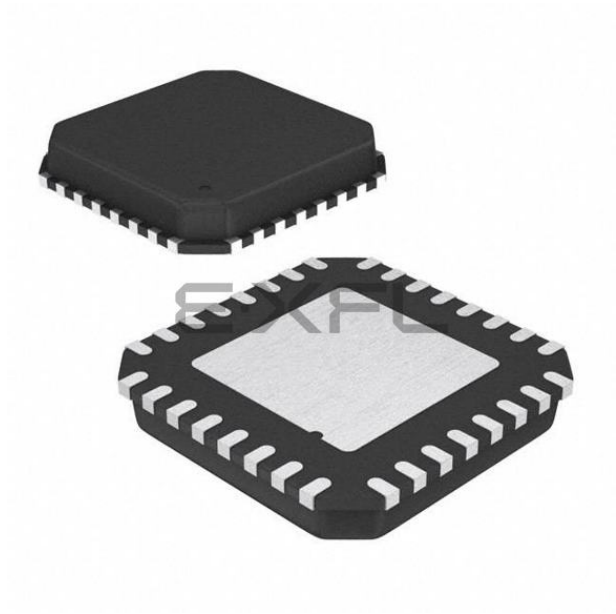


Welcome to E-XFL.COM

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"

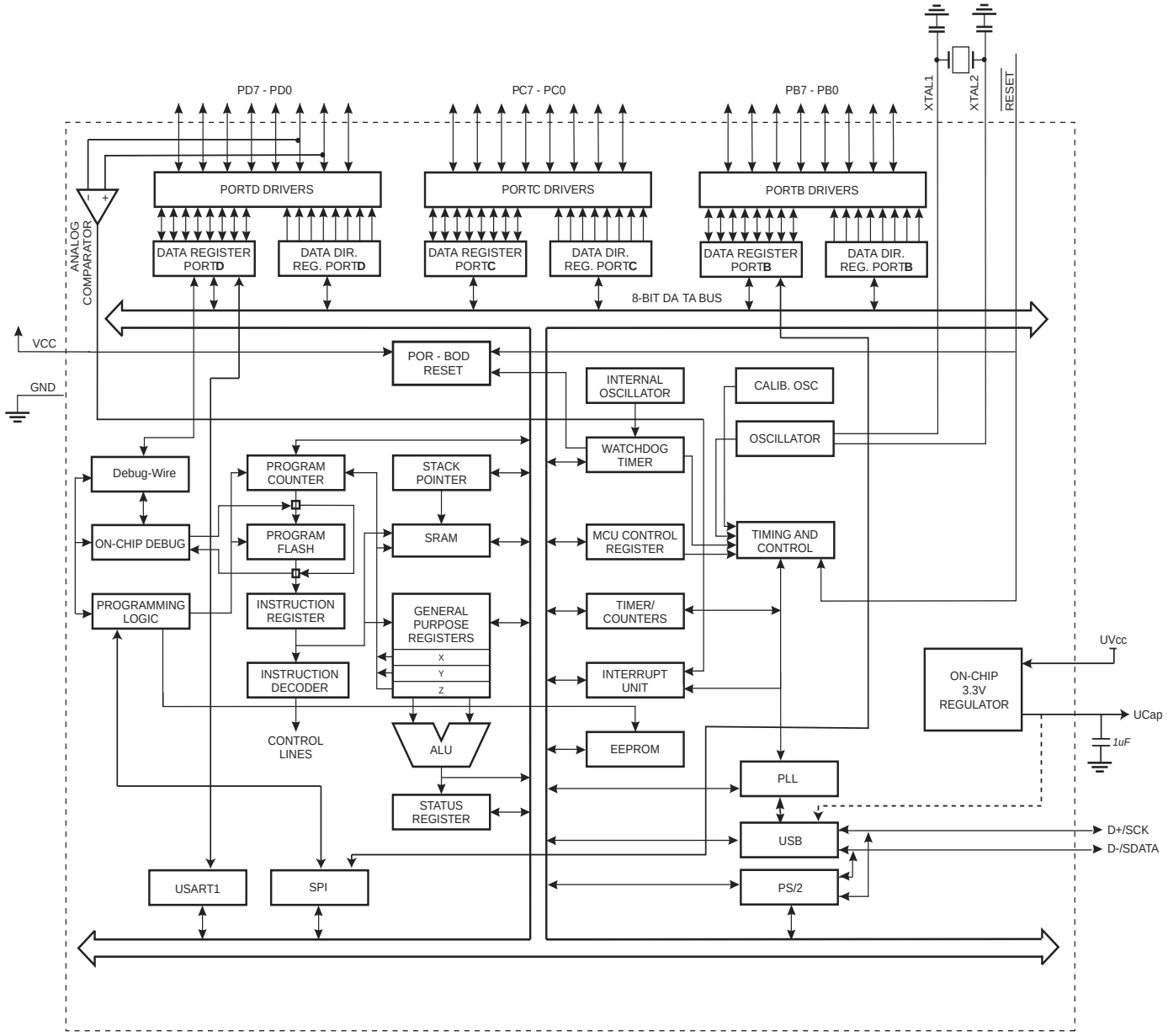
Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	22
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	-
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/atmega16u2-mu

2. Overview

The ATmega8U2/16U2/32U2 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega8U2/16U2/32U2 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting

2.2.5 Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of various special features of the ATmega8U2/16U2/32U2 as listed on page 77.

2.2.6 Port D (PD7..PD0)

Port D serves as analog inputs to the analog comparator.

Port D also serves as an 8-bit bi-directional I/O port, if the analog comparator is not used (concerns PD2/PD1 pins). Port pins can provide internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

2.2.7 D-

USB Full Speed Negative Data Upstream Port

2.2.8 D+

USB Full Speed Positive Data Upstream Port

2.2.9 UGND

USB Ground.

2.2.10 UVCC

USB Pads Internal Regulator Input supply voltage.

2.2.11 UCAP

USB Pads Internal Regulator Output supply voltage. Should be connected to an external capacitor (1µF).

2.2.12 $\overline{\text{RESET}}$/PC1/dW

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in "System Control and Reset" on page 47. Shorter pulses are not guaranteed to generate a reset. This pin alternatively serves as debugWire channel or as generic I/O. The configuration depends on the fuses RSTDISBL and DWEN.

2.2.13 XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

2.2.14 XTAL2/PC0

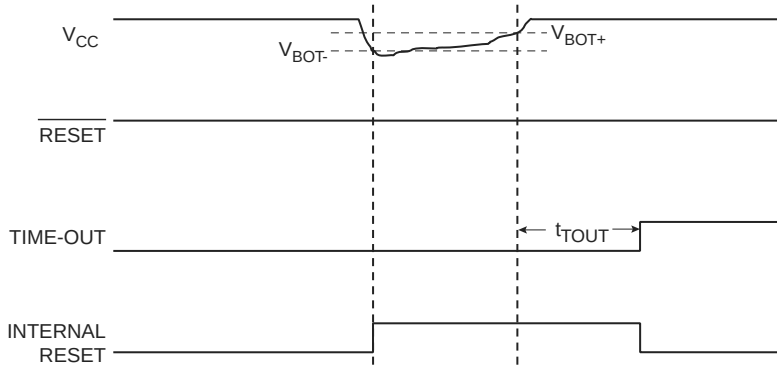
Output from the inverting Oscillator amplifier if enabled by Fuse. Also serves as a generic I/O.

10.2.3 Brown-out Detection

ATmega8U2/16U2/32U2 has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL Fuses. The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$. When the BOD is enabled, and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 10-5), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 10-5), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in "System and Reset Characteristics" on page 267.

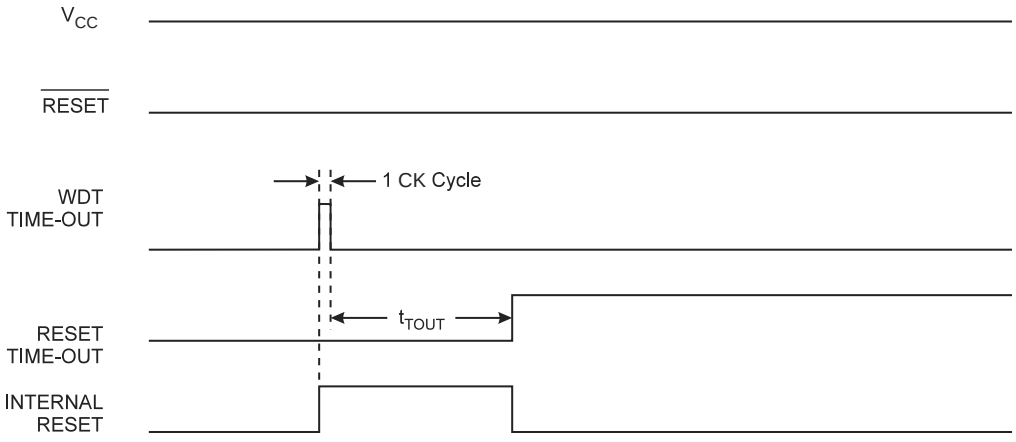
Figure 10-5. Brown-out Reset During Operation



10.2.4 Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to "Watchdog Timer" on page 51 for details on operation of the Watchdog Timer.

Figure 10-6. Watchdog Reset During Operation



10.2.5 USB Reset

When the USB macro is enabled and configured with the USB reset MCU feature enabled, and if a valid USB Reset signalling is detected, the microcontroller is reset unless the USB macro

Table 10-8. Watchdog Timer Prescale Select, DIV = 5 (CLKwdt = CLK128 / 11)

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles before 1st time-out (Early warning)	Early warning Typical Time-out at V_{CC} = 5.0V	Watchdog Reset/Interrupt Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2K (2048) cycles	88 ms	176 ms
0	0	0	1	4K (4096) cycles	176 ms	352 ms
0	0	1	0	8K (8192) cycles	352 ms	704 ms
0	0	1	1	16K (16384) cycles	704 ms	1.4 s
0	1	0	0	32K (32768) cycles	1.4 s	2.8 s
0	1	0	1	64K (65536) cycles	2.8 s	5.6 s
0	1	1	0	128K (131072) cycles	5.6 s	11.2 s
0	1	1	1	256K (262144) cycles	11.2 s	22.5 s
1	0	0	0	512K (524288) cycles	22.5 s	45 s
1	0	0	1	1024K (1048576) cycles	45s	90 s
1	0	1	0	Reserved		
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

Table 10-9. Watchdog Timer Prescale Select, DIV = 6(CLKwdt = CLK128 / 13)

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles before 1st time-out (Early warning)	Early warning Typical Time-out at V_{CC} = 5.0V	Watchdog Reset/Interrupt Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2K (2048) cycles	104 ms	208 ms
0	0	0	1	4K (4096) cycles	208 ms	416 ms
0	0	1	0	8K (8192) cycles	416 ms	832 ms
0	0	1	1	16K (16384) cycles	832 ms	1.64 s
0	1	0	0	32K (32768) cycles	1.6 s	3.3 s
0	1	0	1	64K (65536) cycles	3.3 s	6.6 s
0	1	1	0	128K (131072) cycles	6.6 s	13.3 s
0	1	1	1	256K (262144) cycles	13.3 s	26.6 s
1	0	0	0	512K (524288) cycles	26.6 s	53.2 s
1	0	0	1	1024K (1048576) cycles	53.2 s	106.4 s

11. Interrupts

11.1 Overview

This section describes the specifics of the interrupt handling as performed in ATmega8U2/16U2/32U2. For a general explanation of the AVR interrupt handling, refer to “Reset and Interrupt Handling” on page 13.

11.2 Interrupt Vectors in ATmega8U2/16U2/32U2

Table 11-1. Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, USB Reset and debugWIRE AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	PCINT0	Pin Change Interrupt Request 0
11	$0014	PCINT1	Pin Change Interrupt Request 1
12	$0016	USB General	USB General Interrupt request
13	$0018	USB Endpoint	USB Endpoint Interrupt request
14	$001A	WDT	Watchdog Time-out Interrupt
15	$001C	TIMER1 CAPT	Timer/Counter1 Capture Event
16	$001E	TIMER1 COMPA	Timer/Counter1 Compare Match A
17	$0020	TIMER1 COMPB	Timer/Counter1 Compare Match B
18	$0022	TIMER1 COMPC	Timer/Counter1 Compare Match C
19	$0024	TIMER1 OVF	Timer/Counter1 Overflow
20	$0026	TIMER0 COMPA	Timer/Counter0 Compare Match A
21	$0028	TIMER0 COMPB	Timer/Counter0 Compare match B
22	$002A	TIMER0 OVF	Timer/Counter0 Overflow
23	$002C	SPI, STC	SPI Serial Transfer Complete
24	$002E	USART1 RX	USART1 Rx Complete
25	$0030	USART1 UDRE	USART1 Data Register Empty
26	$0032	USART1TX	USART1 Tx Complete

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_{T0}).

The double buffered Output Compare Registers (OCR0A and OCR0B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC0A and OC0B). See “Output Compare Unit” on page 93. for details. The Compare Match event will also set the Compare Flag (OCF0A or OCF0B) which can be used to generate an Output Compare interrupt request.

15.2.2 Definitions

Many register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 0. A lower case “x” replaces the Output Compare Unit, in this case Compare Unit A or Compare Unit B. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in Table 15-1 are also used extensively throughout the document.

Table 15-1. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0A Register. The assignment is dependent on the mode of operation.

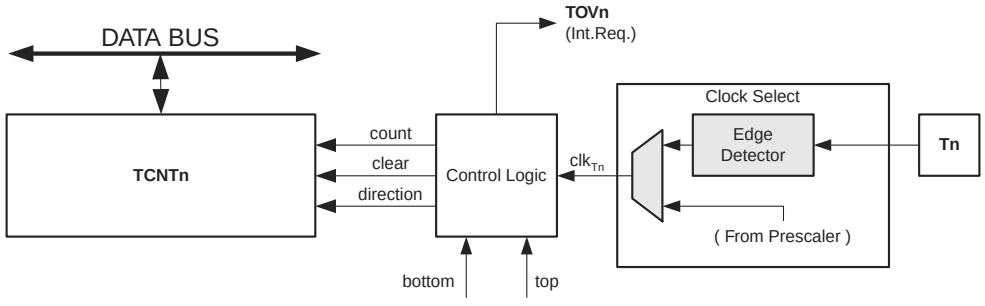
15.3 Timer/Counter Clock Sources

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 (CS02:0) bits located in the Timer/Counter Control Register (TCCR0B). For details on clock sources and prescaler, see “Timer/Counter0 and Timer/Counter1 Prescalers” on page 88.

15.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 15-2 shows a block diagram of the counter and its surroundings.

Figure 15-2. Counter Unit Block Diagram



Signal description (internal signals):

15.6.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM0x[1:0] bits differently in Normal, CTC, and PWM modes. For all modes, setting the COM0x[1:0] = 0 tells the Waveform Generator that no action on the OC0x Register is to be performed on the next Compare Match. For compare output actions in the non-PWM modes refer to Table 15-2 on page 102. For fast PWM mode, refer to Table 15-3 on page 102, and for phase correct PWM refer to Table 15-4 on page 103.

A change of the COM0x1:0 bits state will have effect at the first Compare Match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC0x strobe bits.

15.7 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM0[2:0]) and Compare Output mode (COM0x[1:0]) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM0x[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0x[1:0] bits control whether the output should be set, cleared, or toggled at a Compare Match (See “Compare Match Output Unit” on page 95.).

For detailed timing information see “Timer/Counter Timing Diagrams” on page 100.

15.7.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM0[2:0] = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

The Output Compare Unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

15.7.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM0[2:0] = 2), the OCR0A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter, hence also its resolution. This mode allows greater control of the Compare Match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 15-5. The counter value (TCNT0) increases until a Compare Match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Table 15-4 shows the COM0B[1:0] bit functionality when the WGM0[2:0] bits are set to phase correct PWM mode.

Table 15-7. Compare Output Mode, Phase Correct PWM Mode(1)

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on Compare Match when up-counting. Set OC0B on Compare Match when down-counting.
1	1	Set OC0B on Compare Match when up-counting. Clear OC0B on Compare Match when down-counting.

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 99 for more details.

- **Bits 3:2 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bits 1:0 – WGM0[1:0]: Waveform Generation Mode**

Combined with the WGM02 bit found in the TCCR0B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 15-8. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (see "Modes of Operation" on page 96).

Table 15-8. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	TOP	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

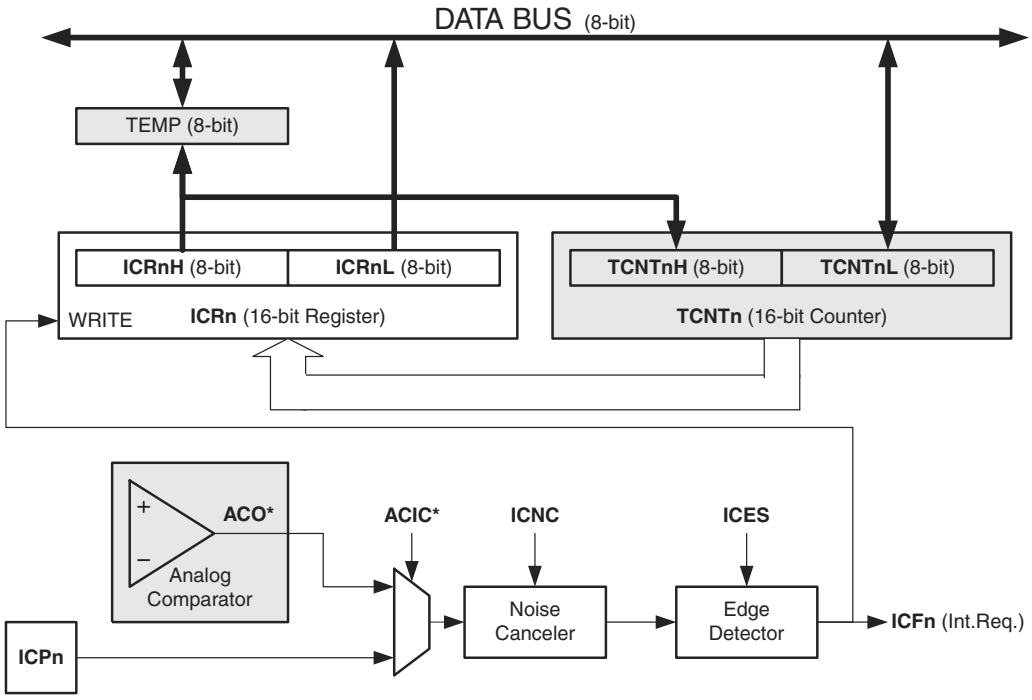
The Timer/Counter Overflow Flag (TOVn) is set according to the mode of operation selected by the WGMn3:0 bits. TOVn can be used for generating a CPU interrupt.

16.6 Input Capture Unit

The Timer/Counter incorporates an input capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICPn pin or alternatively, for the Timer/Counter1 only, via the Analog Comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

The Input Capture unit is illustrated by the block diagram shown in Figure 16-3. The elements of the block diagram that are not directly a part of the input capture unit are gray shaded. The small "n" in register and bit names indicates the Timer/Counter number.

Figure 16-3. Input Capture Unit Block Diagram



Note: The Analog Comparator Output (ACO) can only trigger the Timer/Counter1 ICP – not Timer/Counter3, 4 or 5.

When a change of the logic level (an event) occurs on the *Input Capture Pin* (ICPn), alternatively on the *analog Comparator output* (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNTn) is written to the *Input Capture Register* (ICRn). The *Input Capture Flag* (ICFn) is set at the same system clock as the TCNTn value is copied into ICRn Register. If enabled (TICIEn = 1), the input capture flag generates an input capture interrupt. The ICFn flag is automatically cleared when the interrupt is executed. Alternatively the ICFn flag can be cleared by software by writing a logical one to its I/O bit location.

Figure 16-11. Timer/Counter Timing Diagram, Setting of OCFnx, with Prescaler ($f_{clk\_I/O}/8$)

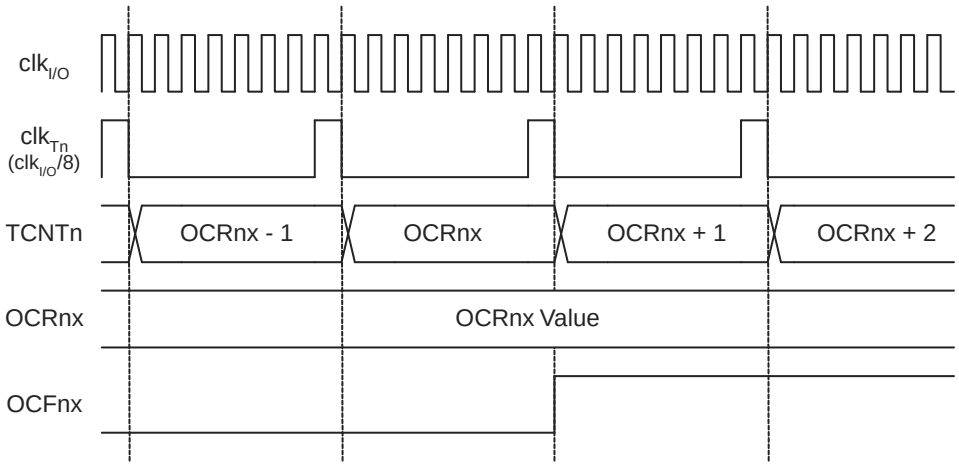


Figure 16-12 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCRnx Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOVn Flag at BOTTOM.

Figure 16-12. Timer/Counter Timing Diagram, no Prescaling

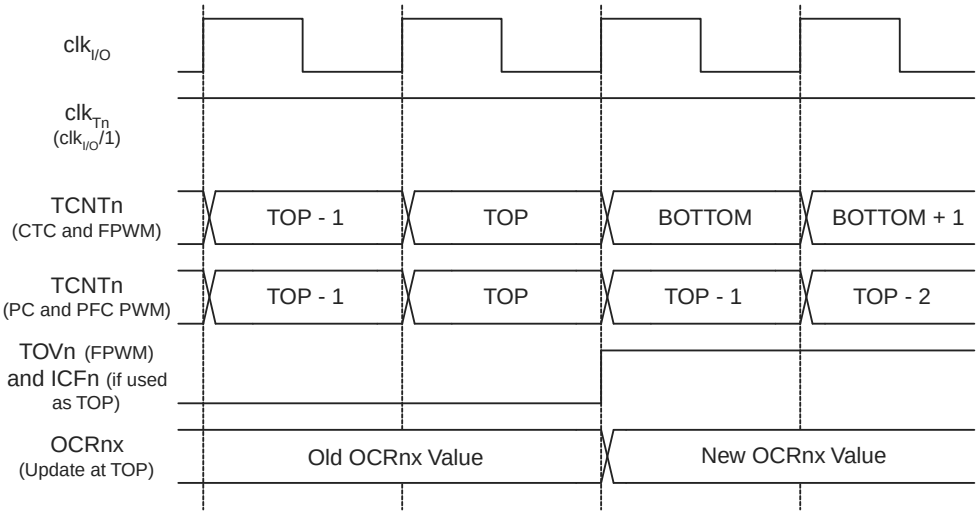


Figure 16-13 shows the same timing data, but with the prescaler enabled.

Assembly Code Example[1]

```
SPI_MasterInit:
  ; Set MOSI and SCK output, all others input
  ldi  r17,(1<<DD_MOSI)|(1<<DD_SCK)
  out  DDR_SPI,r17
  ; Enable SPI, Master, set clock rate fck/16
  ldi  r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
  out  SPCR,r17
  ret

SPI_MasterTransmit:
  ; Start transmission of data (r16)
  out  SPDR,r16
Wait_Transmit:
  ; Wait for transmission complete
  sbis SPSR,SPIF
  rjmp Wait_Transmit
  ret
```

C Code Example[1]

```
void SPI_MasterInit(void)
{
  /* Set MOSI and SCK output, all others input */
  DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
  /* Enable SPI, Master, set clock rate fck/16 */
  SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
  /* Start transmission */
  SPDR = cData;
  /* Wait for transmission complete */
  while(!(SPSR & (1<<SPIF)))
    ;
}
```

Note: 1. See “Code Examples” on page 6.

- **Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency f_{osc} is shown in the following table:

Table 17-5. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

17.5.2 SPSR – SPI Status Register

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag**

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

- **Bit 6 – WCOL: Write COLlision Flag**

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

- **Bit 5:1 – Res: Reserved Bits**

These bits are reserved bits in the ATmega8U2/16U2/32U2 and will always read as zero.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

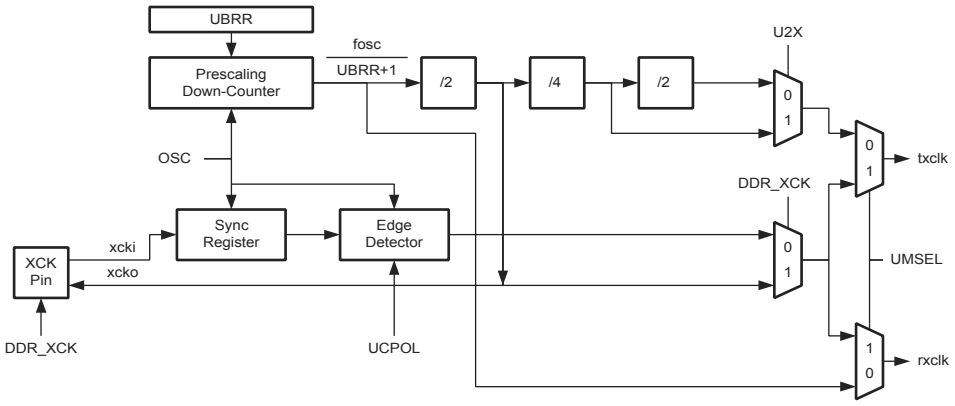
When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 17-5). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{osc}/4$ or lower.

The SPI interface on the ATmega8U2/16U2/32U2 is also used for program memory and EEPROM downloading or uploading. See page 259 for serial programming and verification.

for the XCKn pin (DDR_XCKn) controls whether the clock source is internal (Master mode) or external (Slave mode). The XCKn pin is only active when using synchronous mode.

Figure 18-2 shows a block diagram of the clock generation logic.

Figure 18-2. Clock Generation Logic, Block Diagram



Signal description:

txclk	Transmitter clock (Internal Signal).
rxclk	Receiver base clock (Internal Signal).
xcki	Input from XCK pin (internal Signal). Used for synchronous slave operation.
xcko	Clock output to XCK pin (Internal Signal). Used for synchronous master operation.
f_{osc}	XTAL pin frequency (System Clock).

18.3.1 Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master modes of operation. The description in this section refers to Figure 18-2.

The USART Baud Rate Register (UBRRn) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (f_{osc}), is loaded with the UBRRn value each time the counter has counted down to zero or when the UBRRLn Register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= f_{osc}/(UBRRn+1)). The Transmitter divides the baud rate generator clock output by 2, 8 or 16 depending on mode. The baud rate generator output is used directly by the Receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the UMSELn, U2Xn and DDR_XCKn bits.

18.5 USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the Transmitter or the Receiver depending on the usage. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXCn Flag can be used to check that the Transmitter has completed all transfers, and the RXC Flag can be used to check that there are no unread data in the receive buffer. Note that the TXCn Flag must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 Registers.

Assembly Code Example[1]

```
USART_Init:
  ; Set baud rate
  out  UBRRHn, r17
  out  UBRRLn, r16
  ; Enable receiver and transmitter
  ldi  r16, (1<<RXENn)|(1<<TXENn)
  out  UCSRnB,r16
  ; Set frame format: 8data, 2stop bit
  ldi  r16, (1<<USBSn)|(3<<UCSZn0)
  out  UCSRnC,r16
  ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRHn = (unsigned char)(baud>>8);
  UBRRLn = (unsigned char)baud;
  /* Enable receiver and transmitter */
  UCSRnB = (1<<RXENn)|(1<<TXENn);
  /* Set frame format: 8data, 2stop bit */
  UCSRnC = (1<<USBSn)|(3<<UCSZn0);
}
```

Note: 1. See “Code Examples” on page 6.

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.

18.11 Register Description

18.11.1 UDRn – USART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
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 USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDRn. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDRn Register location. Reading the UDRn Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDREn Flag in the UCSRnA Register is set. Data written to UDRn when the UDREn Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxDn pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

18.11.2 UCSRnA – USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXCn: USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXCn bit will become zero. The RXCn Flag can be used to generate a Receive Complete interrupt (see description of the RXCIEn bit).

- **Bit 6 – TXCn: USART Transmit Complete**

This flag bit 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 (UDRn). The TXCn Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXCn Flag can generate a Transmit Complete interrupt (see description of the TXCIEn bit).

- **Bit 5 – UDREn: USART Data Register Empty**

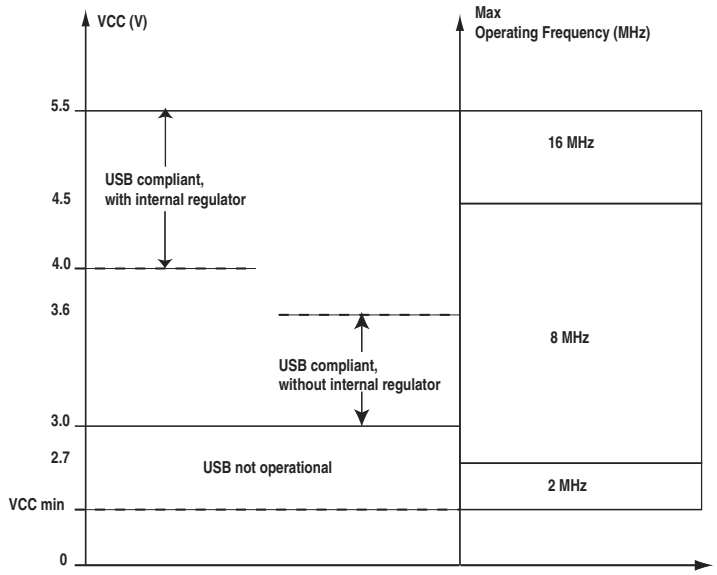
The UDREn Flag indicates if the transmit buffer (UDRn) is ready to receive new data. If UDREn is one, the buffer is empty, and therefore ready to be written. The UDREn Flag can generate a Data Register Empty interrupt (see description of the UDRIEn bit).

UDREn is set after a reset to indicate that the Transmitter is ready.

20.3 USB Module Powering Options

Depending on the selected target application power supply (V_{CC}), the ATmega8U2/16U2/32U2 USB controller requires different powering schemes, see Figure 20-2 on page 186.

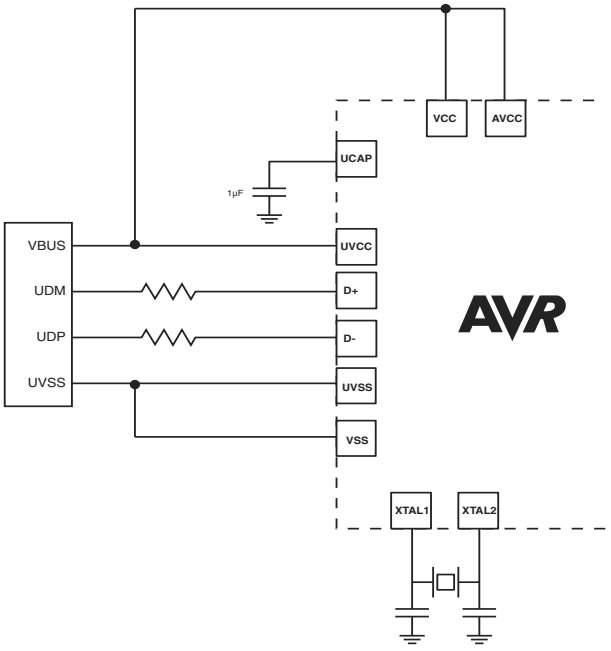
Figure 20-2. Operating modes versus frequency and power-supply



20.3.1 Bus Powered device

The following figures show typical implementations for different powering schemes.

Figure 20-3. Typical Bus powered application with 5V I/O



20.3.3 Design guidelines

The following design guidelines should be met:

- Serial resistors on USB Data lines must have 22 Ohms value (+/- 5%).
- Traces from the input USB receptacle (or from the cable connection in the case of a tethered device) to the USB microcontroller pads should be as short as possible, and follow differential traces routing rules (same length, as near as possible and avoid vias accumulation).
- Voltage transient / ESD suppressors may also be used to prevent USB pads to be damaged by external disturbances.
- U_{cap} capacitor should be 1µF (+/- 10%) for correct operation.

In addition it is highly recommended to connect a 10µF capacitor to the VBUS line

20.4 General Operating Modes

20.4.1 Introduction

The USB controller is disabled and reset after a hardware reset generated by:

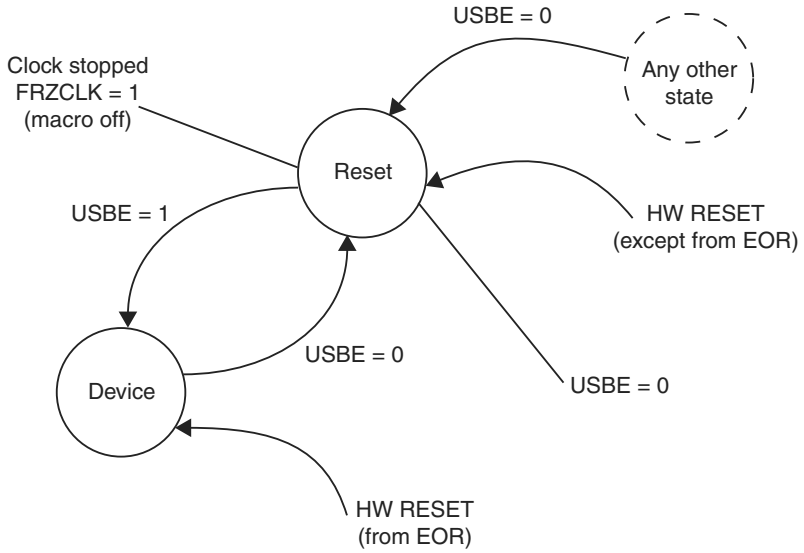
– Power on reset
– External reset
– Watchdog reset
– Brown out reset
– debugWIRE reset
– USB End Of Reset

In the case of USB End Of Reset (EOR), the USB controller is reset, but not disabled. Therefore the device remains attached.

20.4.2 Power-on and reset

Figure 20-7 on page 189 illustrates the USB controller main states on power-on:

Figure 20-7. USB controller states after reset



21.11.2 STALL handshake and Retry mechanism

The Retry mechanism has priority over the STALL handshake. A STALL handshake is sent if the STALLRQ request bit is set and if there is no retry required.

21.12 CONTROL endpoint management

A SETUP request is always ACK'ed. When a new setup packet is received, the RXSTPI interrupt is triggered (if enabled). The RXOUTI interrupt is not triggered.

The FIFOCON and RWAL fields are irrelevant with CONTROL endpoints. The firmware shall thus never use them on that endpoints. When read, their value is always 0.

CONTROL endpoints are managed by the following bits:

- RXSTPI is set when a new SETUP is received. It shall be cleared by firmware to acknowledge the packet **and to clear the endpoint bank**.
- RXOUTI is set when a new OUT data is received. It shall be cleared by firmware to acknowledge the packet **and to clear the endpoint bank**.
- TXINI is set when the bank is ready to accept a new IN packet. It shall be cleared by firmware to **send the packet and to clear the endpoint bank**.

CONTROL endpoints should not be managed by interrupts, but only by polling the status bits.

21.12.1 Control Write

The next figure shows a control write transaction. During the status stage, the controller will not necessary send a NAK at the first IN token:

- If the firmware knows the exact number of descriptor bytes that must be read, it can then anticipate on the status stage and send a ZLP for the next IN token,
- or it can read the bytes and poll NAKINI, which tells that all the bytes have been sent by the host, and the transaction is now in the status stage.

21.12.2 Control Read

The next figure shows a control read transaction. The USB controller has to manage the simultaneous write requests from the CPU and the USB host:

A NAK handshake is always generated at the first status stage command.

When the controller detect the status stage, all the data written by the CPU are erased, and clearing TXINI has no effects.

The firmware checks if the transmission is complete or if the reception is complete.

The OUT retry is always ACK'ed. This reception:

- set the RXOUTI flag (received OUT data)

- set the TXINI flag (data sent, ready to accept new data)

software algorithm:

```
set transmit ready
wait (transmit complete OR Receive complete)
if receive complete, clear flag and return
if transmit complete, continue
```

Once the OUT status stage has been received, the USB controller waits for a SETUP request. The SETUP request have priority over any other request and has to be ACK'ed. This means that any other flag should be cleared and the fifo reset when a SETUP is received.

WARNING: the byte counter is reset when a OUT Zero Length Packet is received. The firmware has to take care of this.

21.13 OUT endpoint management

OUT packets are sent by the host. All the data can be read by the CPU, which acknowledges or not the bank when it is empty.

21.13.1 Overview

The Endpoint must be configured first.

27.10 Current Consumption in Reset and Reset Pulsewidth

Figure 27-28. Reset Supply Current vs. Frequency (Excluding Current Through the Reset Pullup)