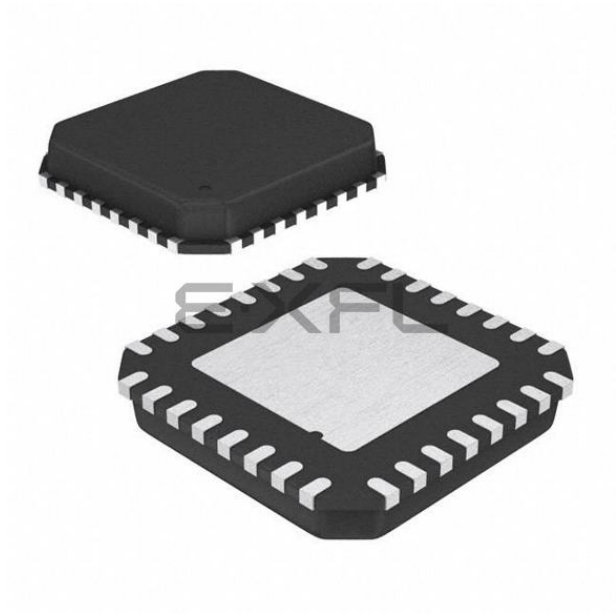


Welcome to E-XFL.COM

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/attiny828-mu

1.1 Pin Description

1.1.1 VCC

Supply voltage.

1.1.2 AVCC

AV_{CC} is the supply voltage pin for the A/D converter and a selection of I/O pins. This pin should be externally connected to V_{CC} even if the ADC is not used. If the ADC is used, it is recommended this pin is connected to V_{CC} through a low-pass filter, as described in “Noise Canceling Techniques” on page 145.

All pins of Port A and Port B are powered by AV_{CC}. All other I/O pins take their supply voltage from V_{CC}.

1.1.3 GND

Ground.

1.1.4 $\overline{\text{RESET}}$

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 and provided the reset pin has not been disabled. The minimum pulse length is given in Table 107 on page 250. Shorter pulses are not guaranteed to generate a reset.

The reset pin can also be used as a (weak) I/O pin.

1.1.5 Port A (PA7:PA0)

This is an 8-bit, bi-directional I/O port with internal pull-up resistors (selected for each bit). Output buffers have high sink and standard source capability. See Table 107 on page 250 for port drive strength.

As inputs, port pins that are externally pulled low will source current provided that pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

This port has alternative pin functions for pin change interrupts, the analog comparator, and ADC. See “Alternative Port Functions” on page 63.

1.1.6 Port B (PB7:PB0)

This is an 8-bit, bi-directional I/O port with internal pull-up resistors (selected for each bit). Output buffers have high sink and standard source capability. See Table 103 on page 247 for port drive strength.

As inputs, port pins that are externally pulled low will source current provided that pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

This port has alternative pin functions for pin change interrupts, and ADC. See “Alternative Port Functions” on page 63.

1.1.7 Port C (PC7:PC0)

This is an 8-bit, bi-directional I/O port with internal pull-up resistors (selected for each bit). Output buffers have high sink and standard source capability. Optionally, extra high sink capability can be enabled. See Table 103 on page 247 for port drive strength.

As inputs, port pins that are externally pulled low will source current provided that pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

This port has alternative pin functions for pin change interrupts, ADC, timer/counter, external interrupts, and serial interfaces. See “Alternative Port Functions” on page 63.

1.1.8 Port D (PD3:PD0)

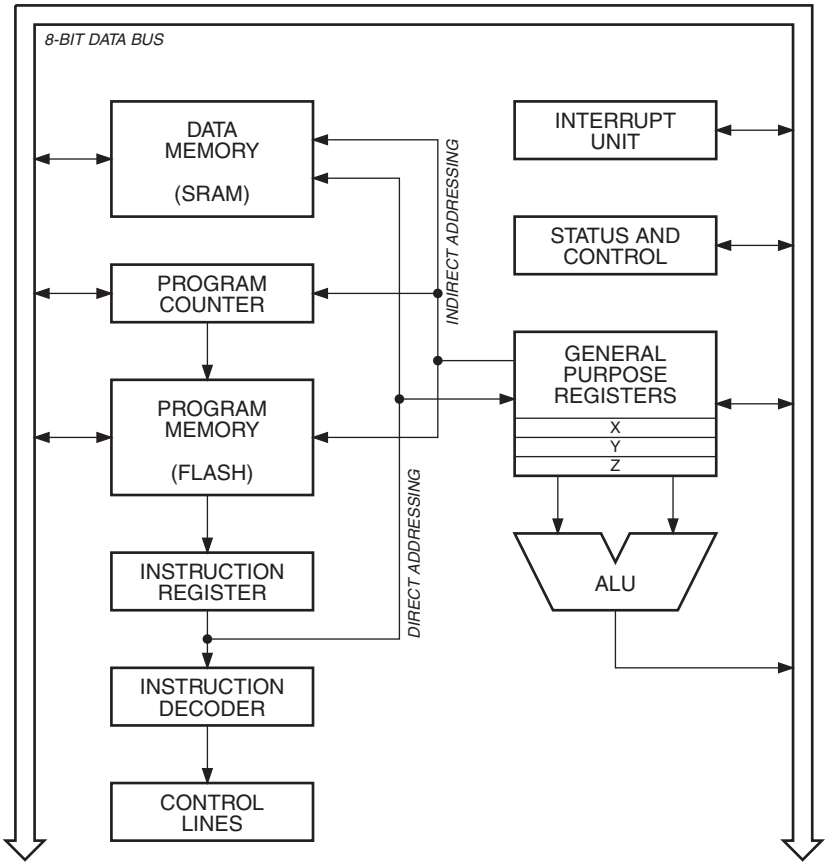
This is a 4-bit, bi-directional I/O port with internal pull-up resistors (selected for each bit). Output buffers of PD0 and PD3 have symmetrical drive characteristics, with both sink and source capability. Output buffer PD1 has high sink and

4. CPU Core

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

4.1 Architectural Overview

Figure 4. Block Diagram of the AVR Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the Program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the Program memory. This concept enables instructions to be executed in every clock cycle. The Program memory is In-System Reprogrammable Flash memory.

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash Program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
in    r16,  SREG              ; store SREG value
cli                           ; disable interrupts during timed sequence
sbi   EECR, EEMPE             ; start EEPROM write
sbi   EECR, EEPE
out   SREG, r16               ; restore SREG value (I-bit)
```

C Code Example

```
char  cSREG;

cSREG = SREG;           /* store SREG value */
_CLI();                 /* disable interrupts during timed sequence */
EECR |= (1<<EEMPE);     /* start EEPROM write */
EECR |= (1<<EEPE);
SREG = cSREG;           /* restore SREG value (I-bit) */
```

Note: See “Code Examples” on page 7.

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in the following example.

Assembly Code Example

```
sei               ; set Global Interrupt Enable
sleep             ; enter sleep, waiting for interrupt
                  ; note: will enter sleep before any pending interrupt(s)
```

C Code Example

```
_SEI();           /* set Global Interrupt Enable */
_SLEEP();         /* enter sleep, waiting for interrupt */
                  /* note: will enter sleep before any pending interrupt */
```

Note: See “Code Examples” on page 7.

4.7.1 Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the Program Vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

5.3.3 Erase

In order to prevent unintentional EEPROM writes, a specific procedure must be followed to erase memory locations. To erase 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.
- Poll the SPMEN bit in Store Program Memory Control and Status Register (SPMCSR) to make sure no self-programming opertaions are in process. If set, wait to clear. This step is relevant only if the application contains a boot loader that programs the Flash memory. If not, this step can be omitted.
- Set mode of programming to erase by writing EEPROM Programming Mode bits (EEPM0 and EEPM1) in EEPROM Control Register (EECR).
- Write target address to EEPROM Address Registers (EEARH/EEARL).
- Enable erase by setting EEPROM Master Program Enable (EEMPE) in EEPROM Control Register (EECR). Within four clock cycles, start the erase operation by setting the EEPROM Program Enable bit (EEPE) in the EEPROM Control Register (EECR). During the erase operation, the CPU is halted for two clock cycles before executing the next instruction.

The EEPE bit remains set until the erase operation has completed. While the device is busy programming, it is not possible to perform any other EEPROM operations.

5.3.4 Write

In order to prevent unintentional EEPROM writes, a specific procedure must be followed to write to memory locations.

Before writing data to EEPROM the target location must be erased. This can be done either in the same operation or as part of a split operation. Writing to an unerased EEPROM location will result in corrupted data.

To write 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.
- Poll the SPMEN bit in Store Program Memory Control and Status Register (SPMCSR) to make sure no self-programming opertaions are in process. If set, wait to clear. This step is relevant only if the application contains a boot loader that programs the Flash memory. If not, this step can be omitted.
- Set mode of programming by writing EEPROM Programming Mode bits (EEPM0 and EEPM1) in EEPROM Control Register (EECR). Alternatively, data can be written in one operation or the write procedure can be split up in erase, only, and write, only.
- Write target address to EEPROM Address Registers (EEARH/EEARL).
- Write target data to EEPROM Data Register (EEDR).
- Enable write by setting EEPROM Master Program Enable (EEMPE) in EEPROM Control Register (EECR). Within four clock cycles, start the write operation by setting the EEPROM Program Enable bit (EEPE) in the EEPROM Control Register (EECR). During the write operation, the CPU is halted for two clock cycles before executing the next instruction.

The EEPE bit remains set until the write operation has completed. While the device is busy with programming, it is not possible to do any other EEPROM operations.

5.3.5 Preventing EEPROM Corruption

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

If the watchdog timer is not going to be used in the application, it is important to go through a watchdog disable procedure in the initialization of the device. If the Watchdog is accidentally enabled, for example by a runaway pointer or brown-out condition, the device will be reset, which in turn will lead to a new watchdog reset. To avoid this situation, the application software should always clear the WDRF flag and the WDE control bit in the initialization routine.

- **Bits 5, 2:0 – WDP[3:0]: Watchdog Timer Prescaler 3 - 0**

The WDP[3:0] bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 15.

Table 15. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
0	0	0	0	512 cycles	16 ms
0	0	0	1	1K cycles	32 ms
0	0	1	0	2K cycles	64 ms
0	0	1	1	4K cycles	0.125 s
0	1	0	0	8K cycles	0.25 s
0	1	0	1	16K cycles	0.5 s
0	1	1	0	32K cycles	1.0 s
0	1	1	1	64K cycles	2.0 s
1	0	0	0	128K cycles	4.0 s
1	0	0	1	256K cycles	8.0 s
1	0	1	0	Reserved(1)	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

Note: 1. If selected, one of the valid settings below 0b1010 will be used.

To avoid unintentional changes of these bits, the following sequence must be followed:

1. Write the required signature to the CCP register. See page 14.
2. Within four instruction cycles, write the desired bit value.

- **Bit 0 – INT0: External Interrupt Request 0 Enable**

The external interrupt for pin INT0 is enabled when this bit and the I-bit in the Status Register (SREG) are set. The trigger conditions are set with the ISC0n bits.

Activity on the pin will cause an interrupt request even if INT1 has been configured as an output.

9.3.9 EIFR – External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	–	–	–	–	**INTF1**	**INTF0**	**EIFR**
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bit 1 – INTF1: External Interrupt Flag 0**

This bit is set when activity on INT1 has triggered an interrupt request. Provided that the I-bit in SREG and the INT1 bit in EIMSK are set, the MCU will jump to the corresponding interrupt vector.

The flag is cleared when the interrupt service routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

This flag is always cleared when INT1 is configured as a level interrupt.

- **Bit 0 – INTF0: External Interrupt Flag 0**

This bit is set when activity on INT0 has triggered an interrupt request. Provided that the I-bit in SREG and the INT0 bit in EIMSK are set, the MCU will jump to the corresponding interrupt vector.

The flag is cleared when the interrupt service routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

This flag is always cleared when INT0 is configured as a level interrupt.

9.3.10 PCIFR – Pin Change Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	–	–	–	–	**PCIF3**	**PCIF2**	**PCIF1**	**PCIF0**	**PCIFR**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:4 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bit 3 – PCIF3: Pin Change Interrupt Flag 3**

This bit is set when a logic change on any PCINT[27:24] pin has triggered an interrupt request. Provided that the I-bit in SREG and the PCIE3 bit in PCICR are set, the MCU will jump to the corresponding interrupt vector.

The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

Table 22. Override Signals of Port A

Pin	Signal	Composition
PA0	PUOE	0
	PUOV	0
	DDOE	0
	DDOV	0
	PVOE	0
	PVOV	0
	PTOE	0
	DIEOE	(PCINT0 • PCIE0) + ADC0D
	DIEOV	PCINT0 • PCIE0
	DI	PCINT0 Input
	AIO	ADC0 Input
PA1	PUOE	0
	PUOV	0
	DDOE	0
	DDOV	0
	PVOE	0
	PVOV	0
	PTOE	0
	DIEOE	(PCINT1 • PCIE0) + ADC1D
	DIEOV	PCINT1 • PCIE0
	DI	PCINT1 Input
	AIO	ADC1 Input
PA2	PUOE	0
	PUOV	0
	DDOE	0
	DDOV	0
	PVOE	0
	PVOV	0
	PTOE	0
	DIEOE	(PCINT2 • PCIE0) + ADC2D
	DIEOV	PCINT2 • PCIE0
	DI	PCINT2 Input
	AIO	ADC2 Input

10.4.9 PINC – Port C Input Pins

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
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 – PINC[7:0]: Port Input Data**

Regardless of the setting of the data direction bit, the value of the port pin PCn can be read through the PINCn bit. Writing a logic one to PINCn toggles the value of PORTCn, regardless of the value in DDCn.

10.4.10 PUEB – Port B Pull-Up Enable Control Register

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	PUEB7	PUEB6	PUEB5	PUEB4	PUEB3	PUEB2	PUEB1	PUEB0	PUEB
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 – PUEB[7:0]: Pull-Up Enable Bits**

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

10.4.11 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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 – PORTB[3:0]: Port Data Bits**

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

10.4.12 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

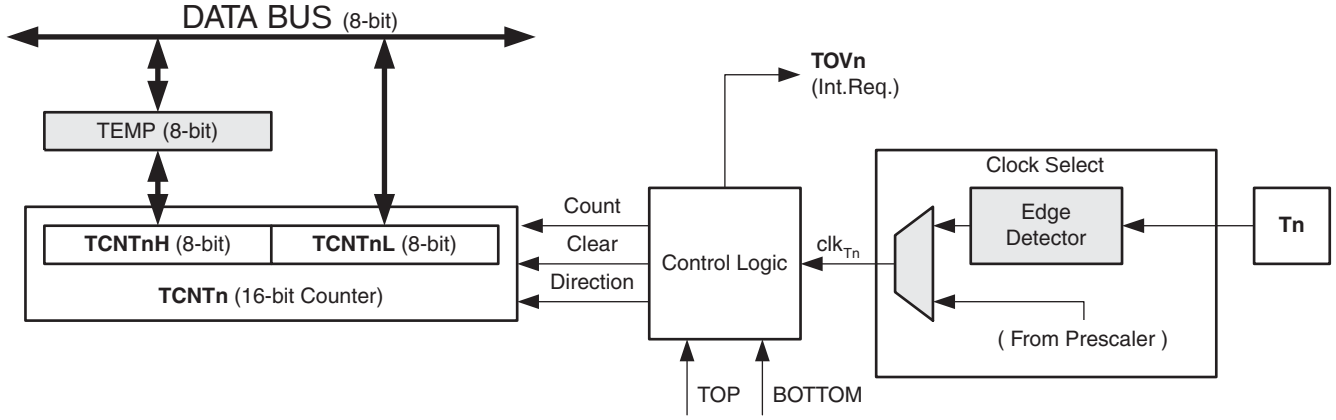
- **Bits 7:0 – DDB[7:0]: Data Direction Bits**

When DDBn is set, the pin PBn is configured as an output. When DDBn is cleared, the pin is configured as an input.

12.4 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 38 shows a block diagram of the counter and its surroundings.

Figure 38. Counter Unit Block Diagram



Description of internal signals used in Figure 38:

Count	Increment or decrement TCNT1 by 1.
Direction	Select between increment and decrement.
Clear	Clear TCNT1 (set all bits to zero).
clk_{T1}	Timer/Counter1 clock.
TOP	Signalize that TCNT1 has reached maximum value.
BOTTOM	Signalize that TCNT1 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: *Counter High* (TCNT1H) containing the upper eight bits of the counter, and Counter Low (TCNT1L) containing the lower eight bits. The TCNT1H Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when the TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNT1 Register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation the counter is cleared, incremented, or decremented at each timer clock (clk_{T1}). The clk_{T1} can be generated from an external or internal clock source, selected by the Clock Select bits (CS1[2:0]). When no clock source is selected the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether clk_{T1} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the Waveform Generation mode bits (WGM1[3:0]) located in the Timer/Counter Control Registers A and B (TCCR1A and TCCR1B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC1x. For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 111.

The Timer/Counter Overflow Flag (TOV1) is set according to the mode of operation selected by the WGM1[3:0] bits. TOV1 can be used for generating a CPU interrupt.

12.5 Input Capture Unit

The Timer/Counter incorporates an Input Capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP1 pin

The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM1x[1:0] bits are shown. When referring to the OC1x state, the reference is for the internal OC1x Register, not the OC1x pin. If a system reset occur, the OC1x Register is reset to “0”.

The general I/O port function is overridden by the Output Compare (OC1x) from the Waveform Generator if either of the COM1x[1:0] bits are set. However, the OC1x pin direction (input or output) is still controlled by the *Data Direction Register* (DDR) for the port pin. The Data Direction Register bit for the OC1x pin (DDR_OC1x) must be set as output before the OC1x value is visible on the pin. The port override function is generally independent of the Waveform Generation mode, but there are some exceptions. See Table 39 on page 124, Table 40 on page 124 and Table 41 on page 124 for details.

The design of the Output Compare pin logic allows initialization of the OC1x state before the output is enabled. Note that some COM1x[1:0] bit settings are reserved for certain modes of operation. See “Register Description” on page 123

The COM1x[1:0] bits have no effect on the Input Capture unit.

12.7.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM1x[1:0] bits differently in normal, CTC, and PWM modes. For all modes, setting the COM1x[1:0] = 0 tells the Waveform Generator that no action on the OC1x Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 39 on page 124. For fast PWM mode refer to Table 40 on page 124, and for phase correct and phase and frequency correct PWM refer to Table 41 on page 124.

A change of the COM1x[1:0] 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 FOC1x strobe bits.

12.8 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 (WGM1[3:0]) and Compare Output mode (COM1x[1:0]) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM1x[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM1x[1:0] bits control whether the output should be set, cleared or toggle at a compare match (“Compare Match Output Unit” on page 110)

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 118.

12.8.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM1[3:0] = 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 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the Timer/Counter Overflow Flag (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 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.

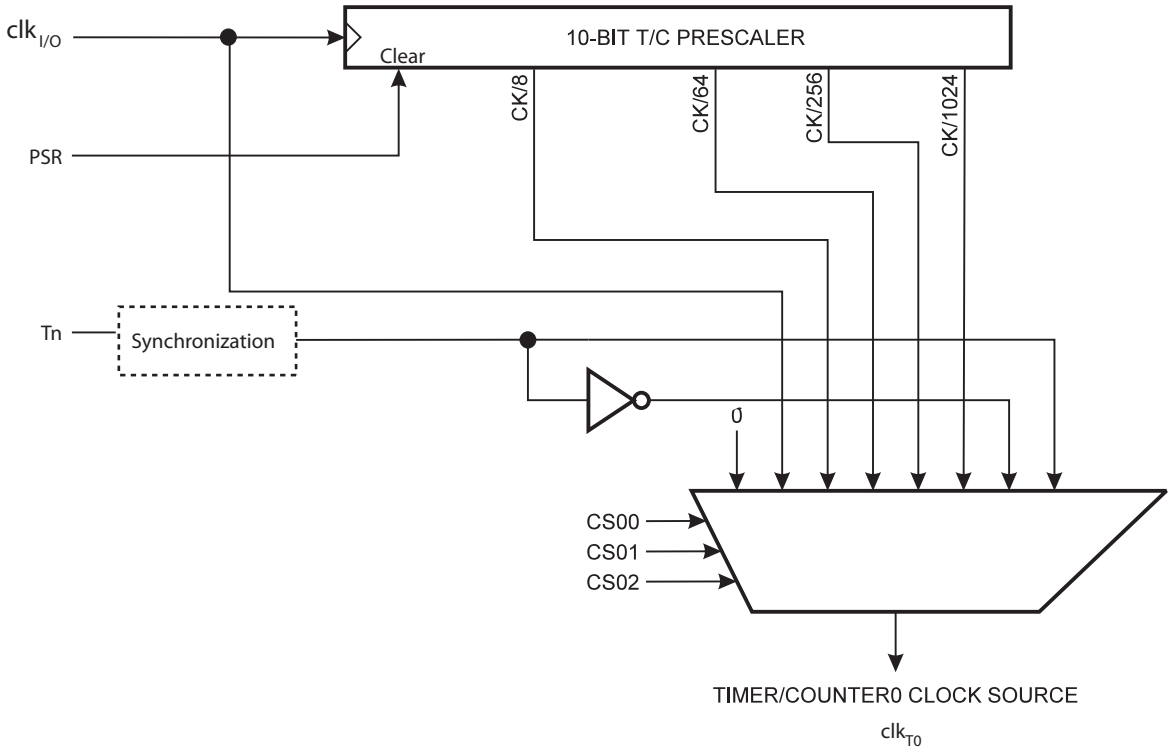
The Input Capture unit is easy to use in Normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

The Output Compare units 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.

12.8.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM1[3:0] = 4 or 12), the OCR1A or ICR1 Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM1[3:0] = 4) or the ICR1 (WGM1[3:0] = 12). The OCR1A or ICR1 define the top value for the counter, hence

Figure 51. Prescaler for Timer/Counter0



Note: 1. The synchronization logic on the input pins (T0) is shown in Figure 50 on page 131.

13.3 Register Description

13.3.1 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	**TSM**	–	–	–	–	–	–	**PSR**	**GTCCR**
Read/Write	R/W	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TSM: Timer/Counter Synchronization Mode**

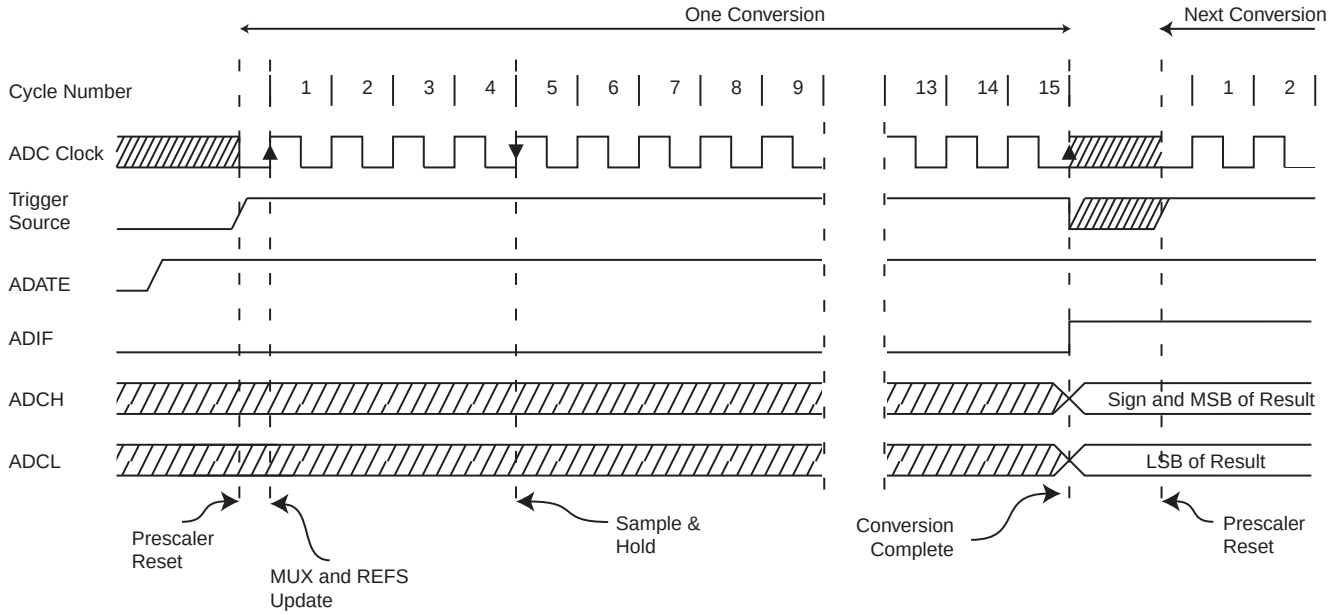
Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSR bit is kept, hence keeping the Prescaler Reset signal asserted. This ensures that the Timer/Counter is halted and can be configured without the risk of advancing during configuration.

When the TSM bit is written to zero, the PSR bit is cleared by hardware, and the Timer/Counter starts counting.

- **Bit 0 – PSR: Prescaler Reset**

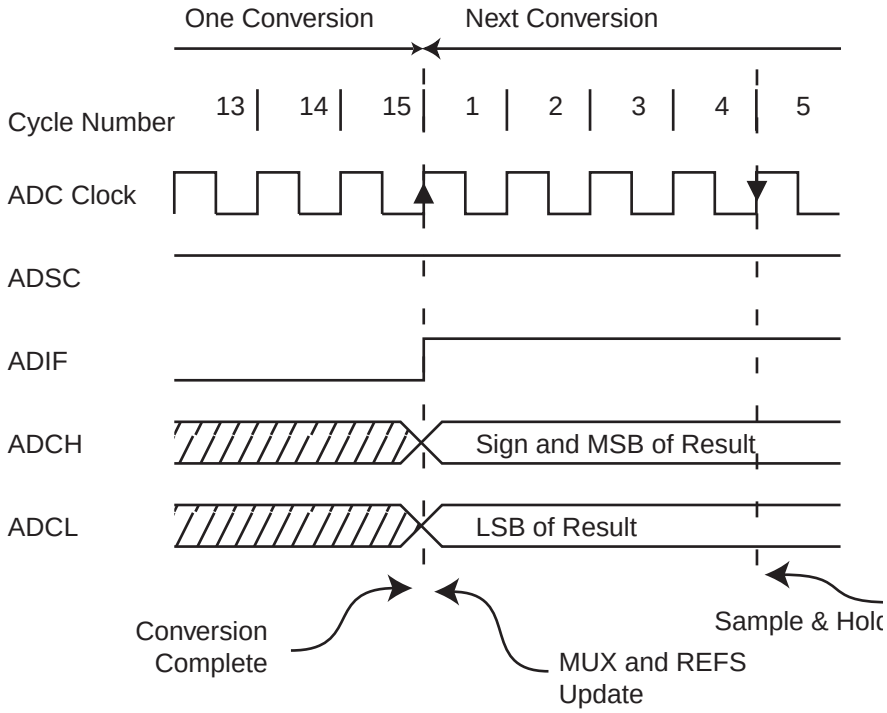
When this bit is one, the Timer/Counter prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set.

Figure 58. ADC Timing Diagram, Auto Triggered Conversion



In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. See Figure 59.

Figure 59. ADC Timing Diagram, Free Running Conversion



For a summary of conversion times, see Table 49 on page 143.

- **Bits 5:1 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

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

When this bit is set the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 59 on page 162). This means that the minimum SCK period will be two I/O clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{clk\_I/O}/4$ or lower.

16.5.3 SPDR – SPI Data Register

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	**MSB**							**LSB**	**SPDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

- Normal asynchronous mode
- Double speed asynchronous mode
- Master synchronous mode
- Slave synchronous mode

The UMSEL bit (see “UCSRC – USART Control and Status Register C” on page 186) selects between asynchronous and synchronous operation. In asynchronous mode, the speed is controlled by the U2X bit (see “UCSRA – USART Control and Status Register A” on page 184).

In synchronous mode (UMSEL = 1), the direction bit of the XCK pin (DDR_XCK) in the Data Direction Register where the XCK pin is located (DDRx) controls whether the clock source is internal (master mode), or external (slave mode). The XCK pin is active in synchronous mode, only.

17.3.1 Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used in asynchronous and synchronous master modes of operation. The description in this section refers to Figure 71 on page 165.

The USART Baud Rate Register (UBRR) 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 UBRR value each time the counter has counted down to zero, or when UBRR0L is written.

A clock is generated each time the counter reaches zero. This is the baud rate generator clock output and has a frequency of f_{osc}/(UBRR+1). Depending on the mode of operation the transmitter divides the baud rate generator clock output by 2, 8 or 16. 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 UMSEL, U2X and DDR_XCK bits.

Table 60 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRR value for each mode of operation using an internally generated clock source.

Table 60. Equations for Calculating Baud Rate Register Setting

Operating Mode	Baud Rate[1]	UBRR Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16 \times (\boldsymbol{UBRR} + 1)}$	$\boldsymbol{UBRR} = \frac{f_{OSC}}{16 \times BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8 \times (\boldsymbol{UBRR} + 1)}$	$\boldsymbol{UBRR} = \frac{f_{OSC}}{8 \times BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2 \times (\boldsymbol{UBRR} + 1)}$	$\boldsymbol{UBRR} = \frac{f_{OSC}}{2 \times BAUD} - 1$

Note: 1. Baud rate is defined as the transfer rate in bits per second (bps)

Signal description for Table 60:

BAUD	Baud rate (in bits per second, bps)
f_{osc}	System Oscillator clock frequency
UBRR	Contents of the UBRRH and UBRRL Registers, (0-4095)

Some examples of UBRR values for selected system clock frequencies are shown in Table 63 on page 181.

- **Bit 4 – FE: Frame Error**

This bit is set if the next character in the receive buffer had a frame error when received (i.e. when the first stop bit of the next character in the receive buffer is zero). This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one.

Always set this bit to zero when writing the register.

- **Bit 3 – DOR: Data OverRun**

This bit is set if a Data OverRun condition is detected. A data overrun occurs when the receive buffer is full (two characters), there is a new character waiting in the receive shift register, and a new start bit is detected. This bit is valid until the receive buffer (UDR) is read.

Always set this bit to zero when writing the register.

- **Bit 2 – UPE: USART Parity Error**

This bit is set if the next character in the receive buffer had a parity error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read.

Always set this bit to zero when writing the register.

- **Bit 1 – U2X: Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication Mode. When the bit is written to one, all the incoming frames received by the USART receiver that do not contain address information will be ignored. The transmitter is unaffected by the MPCM bit. For more detailed information, see “Multi-processor Communication Mode” on page 180.

17.11.3 UCSRB – USART Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0xC1)	**RXCIE**	**TXCIE**	**UDRIE**	**RXEN**	**TXEN**	**UCSZ2**	**RXB8**	**TXB8**	**UCSRB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RXCIE: RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC flag. A USART Receive Complete interrupt will be generated only if the RXCIE bit, the Global Interrupt Flag, and the RXC bits are set.

- **Bit 6 – TXCIE: TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC flag. A USART Transmit Complete interrupt will be generated only if the TXCIE bit, the Global Interrupt Flag, and the TXC bit are set.

- **Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE flag. A Data Register Empty interrupt will be generated only if the UDRIE bit, the Global Interrupt Flag, and the TXC bit are set.

18. USART in SPI Mode

18.1 Features

- Full Duplex, Three-wire Synchronous Data Transfer
- Master Operation
- Supports all four SPI Modes of Operation (Mode 0, 1, 2, and 3)
- LSB First or MSB First Data Transfer (Configurable Data Order)
- Queued Operation (Double Buffered)
- High Resolution Baud Rate Generator
- High Speed Operation (fXCKmax = fCK/2)
- Flexible Interrupt Generation

18.2 Overview

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) can be set to a master SPI compliant mode of operation.

Setting both UMSEL[1:0] bits to one enables the USART in MSPIM logic. In this mode of operation the SPI master control logic takes direct control over the USART resources. These resources include the transmitter and receiver shift register and buffers, and the baud rate generator. The parity generator and checker, the data and clock recovery logic, and the RX and TX control logic is disabled. The USART RX and TX control logic is replaced by a common SPI transfer control logic. However, the pin control logic and interrupt generation logic is identical in both modes of operation.

The I/O register locations are the same in both modes. However, some of the functionality of the control registers changes when using MSPIM.

18.3 Clock Generation

The clock generation logic generates the base clock for the transmitter and receiver. For USART MSPIM mode of operation only internal clock generation (i.e. master operation) is supported. Therefore, for the USART in MSPIM to operate correctly, the Data Direction Register (DDRx) where the XCK pin is located must be configured to set the pin as output (DDR_XCK = 1) . Preferably the DDR_XCK should be set up before the USART in MSPIM is enabled (i.e. before TXEN and RXEN bits are set).

The internal clock generation used in MSPIM mode is identical to the USART synchronous master mode. The baud rate or UBRR setting can therefore be calculated using the same equations, see Table 73:

Table 73. Equations for Calculating Baud Rate Register Setting

Operating Mode	Calculating Baud Rate[1]	Calculating UBRR Value
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(\boldsymbol{UBRR}+1)}$	$\boldsymbol{UBRR} = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined as the transfer rate in bits per second (bps)

BAUD	Baud rate (in bits per second, bps)
f_{osc}	System oscillator clock frequency
UBRRn	Contents of UBRRH and UBRRL, (0-4095)

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
	UBRR = 0;

	/* Setting the XCK port pin as output, enables master mode. */
	XCK_DDR |= (1<<XCK);

	/* Set MSPI mode of operation and SPI data mode 0. */
	UCSRC = (1<<UMSEL1)|(1<<UMSEL0)|(0<<UCPHA)|(0<<UCPOL);

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

	/* Set baud rate. */
	/* IMPORTANT: Baud Rate must be set after transmitter is enabled */
	UBRR = baud;
}
```

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

18.6 Data Transfer

Using the USART in MSPI mode requires the transmitter to be enabled, i.e. the TXEN bit to be set. When the transmitter is enabled, the normal port operation of the TxD pin is overridden and given the function as the transmitter's serial output. Enabling the receiver is optional and is done by setting the RXEN bit. When the receiver is enabled, the normal pin operation of the RxD pin is overridden and given the function as the receiver's serial input. The XCK will in both cases be used as the transfer clock.

After initialization the USART is ready for doing data transfers. A data transfer is initiated by writing to UDR. This is the case for both sending and receiving data since the transmitter controls the transfer clock. The data written to UDR is moved from the transmit buffer to the shift register when the shift register is ready to send a new frame.

Note: To keep the input buffer in sync with the number of data bytes transmitted, UDR must be read once for each byte transmitted. The input buffer operation is identical to normal USART mode, i.e. if an overflow occurs the character last received will be lost, not the first data in the buffer. This means that if four bytes are transferred, byte 1 first, then byte 2, 3, and 4, and the UDR is not read before all transfers are completed, then byte 3 to be received will be lost, and not byte 1.

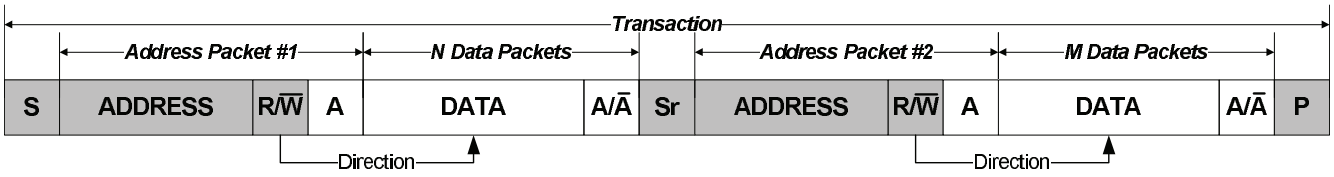
The following code examples show a simple USART in MSPIM mode transfer function based on polling of the Data Register Empty flag (UDRE) and the Receive Complete flag (RXC). The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in register R16 and the data received will be available in the same register (R16) after the function returns.

The function simply waits for the transmit buffer to be empty by checking the UDRE flag, before loading it with new data to be transmitted. The function then waits for data to be present in the receive buffer by checking the RXC flag, before reading the buffer and returning the value..

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

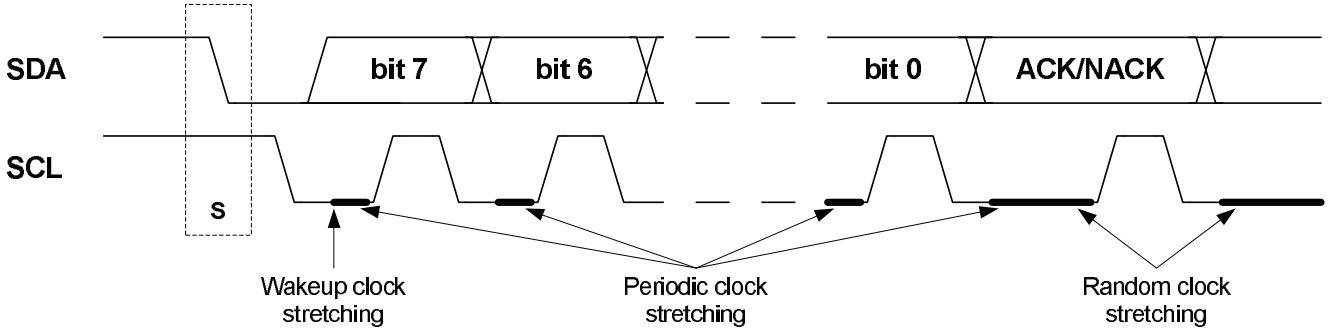


19.3.7 Clock and Clock Stretching

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.

21.7 Boot Loader Lock Bits

The boot loader has two separate sets of lock bits, which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can choose any of the following:

- Protect the entire Flash from a software update by the MCU
- Protect only the Boot Loader Flash section from a software update by the MCU
- Protect only the Application Flash section from a software update by the MCU
- Allow software update in the entire Flash.

For more details on lock bits, see “Lock Bits” on page 225.

21.7.1 Programming Boot Loader Lock Bits by SPM

To set boot loader and general lock bits:

- Write the desired data to R0. A cleared bit indicates the corresponding lock bit is to be programmed. See bit mapping of R0 below
- Write “X0001001” to SPMCSR
- Execute SPM within four clock cycles after writing SPMCSR

During lock bit programming, the contents of R0 is treated as shown below.

Bit	7	6	5	4	3	2	1	0
R0	**1**	**1**	**BLB12**	**BLB11**	**BLB02**	**BLB01**	**LB2**	**LB1**

During the operation, the value of Z-pointer is ignored, but for future compatibility it is recommended to load the Z-pointer with 0x0001 (same as used for reading the lock bits). For future compatibility, it is also recommended to set bits 7 and 6 in R0 to “1” when writing the Lock bits.

When programming the Lock bits the entire Flash can be read during the operation.

See Table 87 on page 226 and Table 88 on page 226 for how different settings of the boot loader lock bits affect Flash access. See “Lock Bits” on page 225 for lock bit layout.

21.7.2 Updating the BLS

Special care must be taken if boot lock bit BLB11 is left unprogrammed, allowing the BLS to be updated. An accidental write to the BLS can corrupt the entire boot loader, making further software updates impossible. If boot loader software does not need to be updated, it is recommended to program boot lock bit BLB11 to protect the BLS from being changed by software.

21.8 Self-Programming the Flash

The device provides a self-programming mechanism for downloading and uploading program code by the MCU itself. Self-Programming can use any available data interface and associated protocol to read code and write (program) that code into program memory.

Program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the Page Erase command or between a Page Erase and a Page Write operation:

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

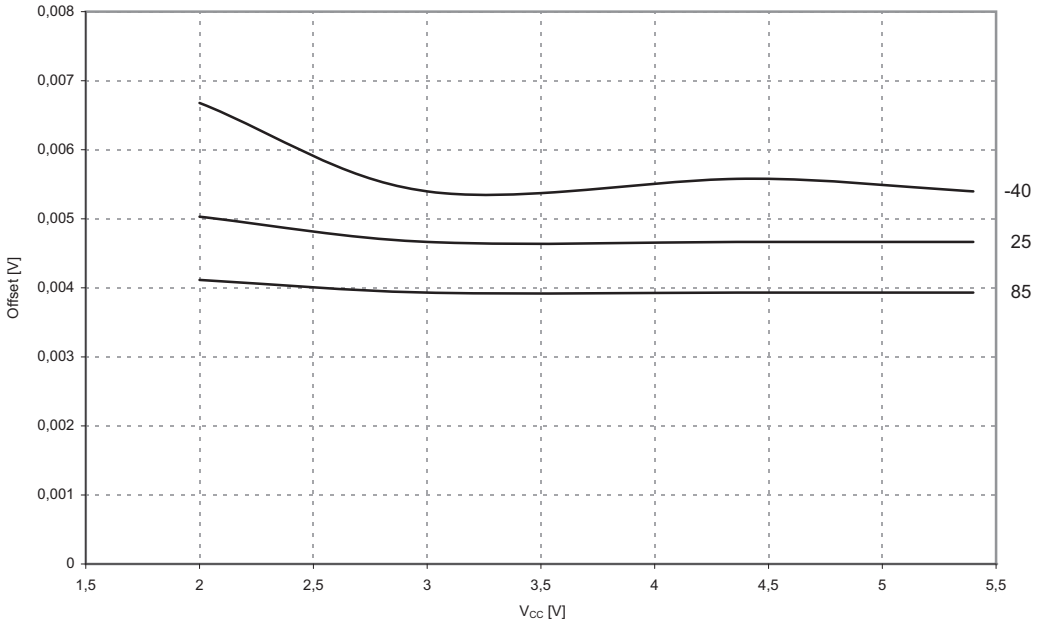


Figure 177. Analog Comparator Hysteresis vs. V_{IN} (V_{CC} = 5.0V)