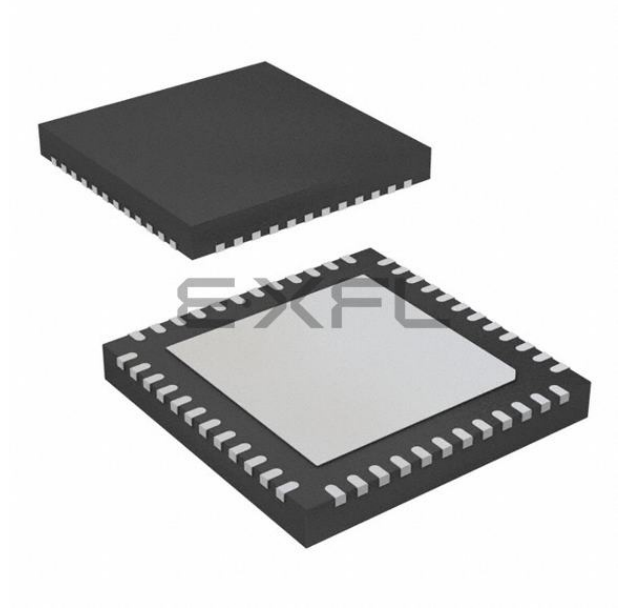


Welcome to E-XFL.COM

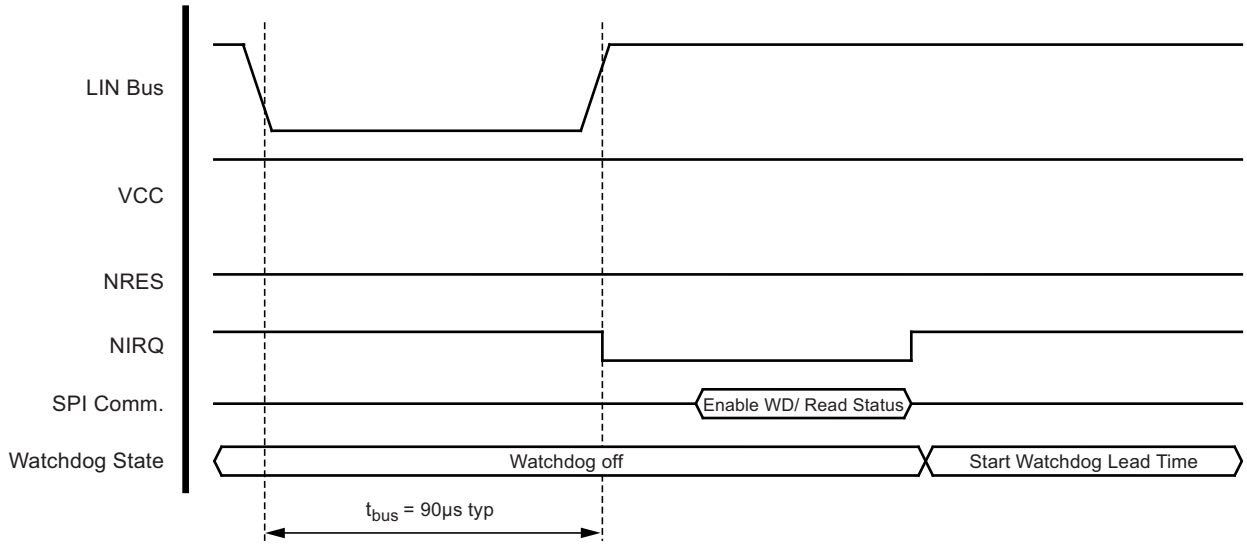
What is "Embedded - Microcontrollers"?

"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	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I²C, LINbus, SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TC)
Mounting Type	Surface Mount
Package / Case	48-VFQFN Exposed Pad
Supplier Device Package	48-QFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/ata664251-wgqw

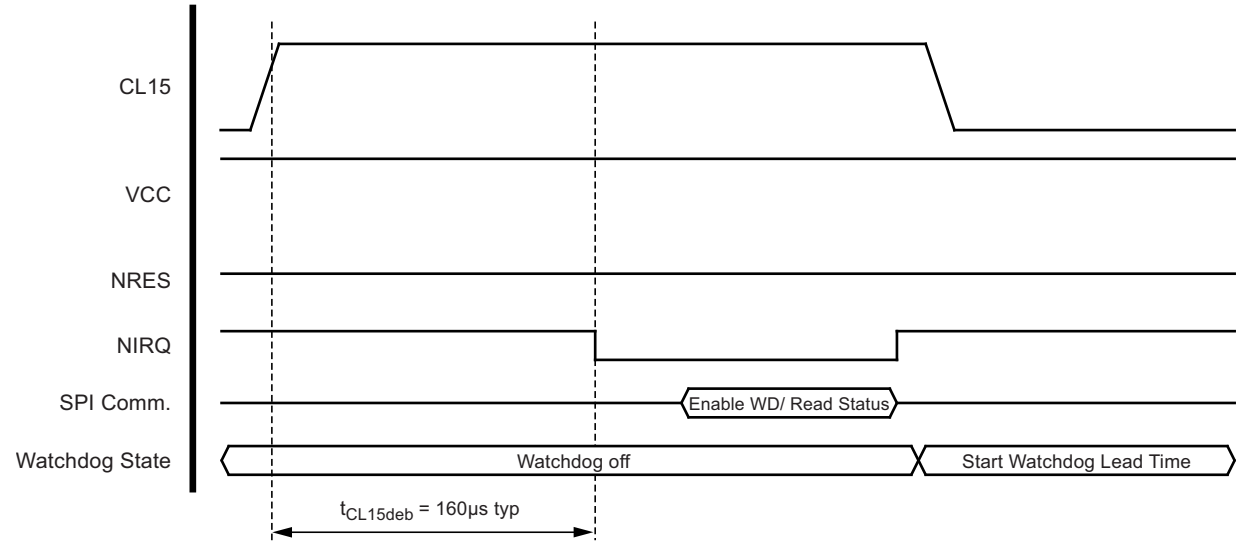
Figure 4-9. LIN Wake-up from Active Low-power Mode



The negative edge on the NIRQ pin indicates a change of conditions, in this case a wake-up request at the LIN bus. The microcontroller can check the IRQ source by assessing the “IRQS1” and “IRQS0” bits in the status register. Note that if a watchdog operation is desired, it must be enabled via the configuration register.

The behavior can be transferred to a wake-up over CL15 pin from Active Low-power Mode.

Figure 4-10. CL15 Wake-up from Active Low-power Mode



Apart from the LIN transceiver and the CL15 input, the high-voltage I/O ports CS1 to CS8 can also be used to generate interrupts while in Active Low-power Mode. This can be done by enabling the current sources so that they can generate an interrupt with the corresponding CSEx- and CSIEx bits in the configuration register. As long as the current source is not enabled (CSCx=’0’ and PWMy low), the IC stays in Active Low-power Mode (if all other conditions are met, such as disabled watchdog). The PWMy pin has to be set to high by the microcontroller, for example, controlled via a PWM timer unit, in order to check the condition of the connected switch. Because the switch interface unit is enabled, current consumption increases drastically. This “switch scanning phase” can be short compared to the interceding idle time so the mean current consumption of the IC remains close to the Active Low-power Mode current consumption. For more information , see Section 4.8.1 “Current Sources” on page 24 and Section 4.8.2 “Switch Inputs” on page 26 for further details.

5.11.6.2 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM0A1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM0A1:0 = 0 tells the Waveform Generator that no action on the OC0A Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 5-29 on page 129. For fast PWM mode, refer to Table 5-30 on page 129, and for phase correct PWM refer to Table 5-31 on page 130.

A change of the COM0A1:0 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 FOC0A strobe bits.

5.11.7 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 (WGM01:0) and Compare Output mode (COM0A1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM0A1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0A1:0 bits control whether the output should be set, cleared, or toggled at a compare match (See Section 5.11.6 “Compare Match Output Unit” on page 121).

For detailed timing information refer to Section 5.11.8 “Timer/Counter Timing Diagrams” on page 125.

5.11.7.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM01:0 = 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.

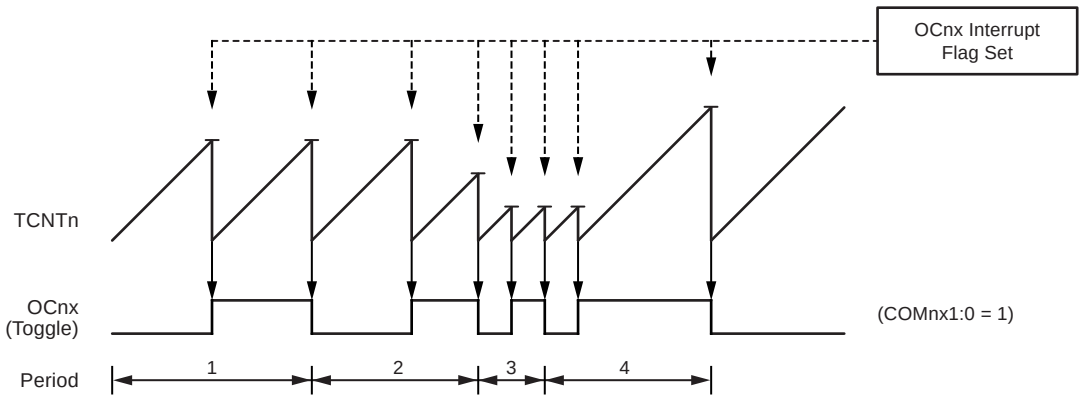
The Output Compare unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

5.11.7.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM01:0 = 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 5-34. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

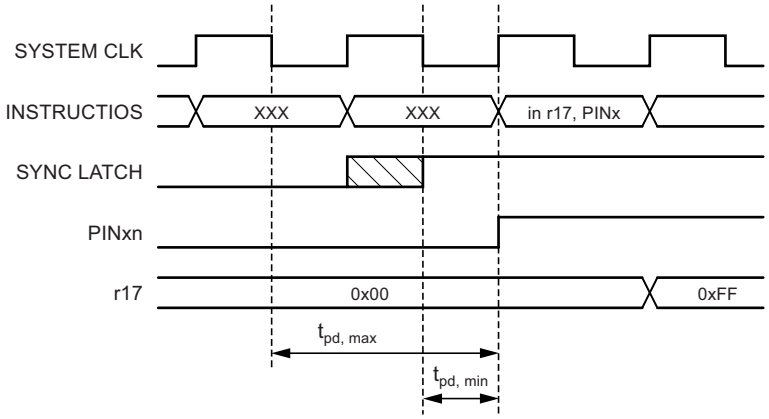
Figure 5-34. CTC Mode, Timing Diagram



5.10.2.5 Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Figure 5-25 on page 101, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 5-27 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

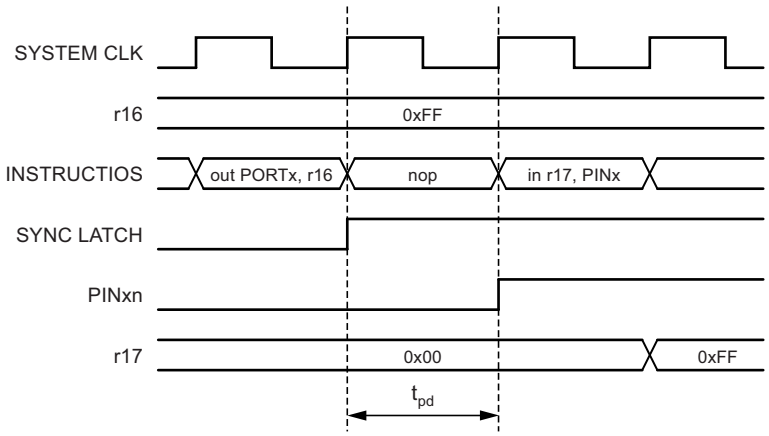
Figure 5-27. Synchronization when Reading an Externally Applied Pin Value



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

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 5-28. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is 1 system clock period.

Figure 5-28. Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a nop instruction is included to be able to read back the value recently assigned to some of the pins.

5.11.9 Asynchronous Operation of Timer/Counter0

When Timer/Counter0 operates asynchronously, some considerations must be taken.

- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter0, the timer registers TCNT0, OCR0A, and TCCR0A might be corrupted. A safe procedure for switching clock source is:
  a) Disable the Timer/Counter0 interrupts by clearing OCIE0A and TOIE0.
  b) Select clock source by setting AS0 and EXCLK as appropriate.
  c) Write new values to TCNT0, OCR0A, and TCCR0A.
  d) To switch to asynchronous operation: Wait for TCN0UB, OCR0UB, and TCR0UB.
  e) Clear the Timer/Counter0 interrupt flags.
  f) Enable interrupts, if needed.
- If an 32.768kHz watch crystal is used, the CPU main clock frequency must be more than four times the Oscillator or external clock frequency.
- When writing to one of the registers TCNT0, OCR0A, or TCCR0A, the value is transferred to a temporary register, and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the three mentioned registers have their individual temporary register, which means that e.g. writing to TCNT0 does not disturb an OCR0A write in progress. To detect that a transfer to the destination register has taken place, the Asynchronous Status Register – ASSR has been implemented.
- When entering Power-save mode after having written to TCNT0, OCR0A, or TCCR0A, the user must wait until the written register has been updated if Timer/Counter0 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if the Output Compare0 interrupt is used to wake up the device, since the Output Compare function is disabled during writing to OCR0A or TCNT0. If the write cycle is not finished, and the MCU enters sleep mode before the OCR0UB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
- If Timer/Counter0 is used to wake the device up from Power-save mode, precautions must be taken if the user wants to re-enter one of these modes: The interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and re-entering sleep mode is less than one TOSC1 cycle, the interrupt will not occur, and the device will fail to wake up. If the user is in doubt whether the time before re-entering Power-save mode is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:
  a) Write a value to TCCR0A, TCNT0, or OCR0A.
  b) Wait until the corresponding Update Busy flag in ASSR returns to zero.
  c) Enter Power-save or ADC Noise Reduction mode.
- When the asynchronous operation is selected, the oscillator for Timer/Counter0 is always running, except in Power-down mode. After a Power-up Reset or wake-up from Power-down mode, the user should be aware of the fact that this oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter0 after power-up or wake-up from Power-down mode. The contents of all Timer/Counter0 Registers must be considered lost after a wake-up from Power-down mode due to unstable clock signal upon start-up, no matter whether the oscillator is in use or a clock signal is applied to the XTAL1 pin.
- Description of wake up from Power-save mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.
- Reading of the TCNT0 Register shortly after wake-up from Power-save may give an incorrect result. Since TCNT0 is clocked on the asynchronous clock, reading TCNT0 must be done through a register synchronized to the internal I/O clock domain (CPU main clock). Synchronization takes place for every rising XTAL1 edge. When waking up from Power-save mode, and the I/O clock ($clk_{I/O}$) again becomes active, TCNT0 will read as the previous value (before entering sleep) until the next rising XTAL1 edge. The phase of the XTAL1 clock after waking up from Power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading TCNT0 is thus as follows:
  a) Write any value to either of the registers OCR0A or TCCR0A.
  b) Wait for the corresponding Update Busy Flag to be cleared.
  c) Read TCNT0.

5.13 16-bit Timer/Counter1

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement. The main features are:

5.13.1 Features

- True 16-bit Design (i.e., Allows 16-bit PWM)
- Two independent Output Compare Units
- Four Controlled Output Pins per Output Compare Unit
- Double Buffered Output Compare Registers
- One Input Capture Unit
- Input Capture Noise Canceler
- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Variable PWM Period
- Frequency Generator
- External Event Counter
- Four independent interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)

5.13.2 Overview

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 1. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT1 for accessing Timer/Counter1 counter value and so on.
- A lower case “x” replaces the Output Compare unit channel, in this case A or B. However, when using the register or bit defines in a program, the precise form must be used, i.e., OCR1A for accessing Timer/Counter1 output compare channel A value and so on.
- A lower case “i” replaces the index of the Output Compare output pin, in this case U, V, W or X. However, when using the register or bit defines in a program, the precise form must be used.

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 5-44 on page 137. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in Section 5.13.11 “16-bit Timer/Counter Register Description” on page 156.

The following code examples show how to do an atomic write of the TCNT1 Register contents. Writing any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

Assembly Code Example[1]

```
TIM16_WriteTCNT1:
        ; Save global interrupt flag
        in      r18,SREG
        ; Disable interrupts
        cli
        ; Set TCNT1 to r17:r16
        sts     TCNT1H,r17
        sts     TCNT1L,r16
        ; Restore global interrupt flag
        out     SREG,r18
        ret
```

C Code Example[1]

```
void TIM16_WriteTCNT1(unsigned int i)
{
        unsigned char sreg;
        unsigned int i;
        /* Save global interrupt flag */
        sreg = SREG;
        /* Disable interrupts */
        _CLI();
        /* Set TCNT1 to i */
        TCNT1 = i;
        /* Restore global interrupt flag */
        SREG = sreg;
}
```

Note: 1. The example code assumes that the part specific header file is included1

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT1.

5.13.3.2 Reusing the Temporary High Byte Register

If writing to more than one 16-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

5.13.4 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 (CS12:0) bits located in the Timer/Counter control Register B (TCCR1B). For details on clock sources and prescaler, see Section 5.12 “Timer/Counter1 Prescaler” on page 134.

5.15 USI – Universal Serial Interface

5.15.1 Features

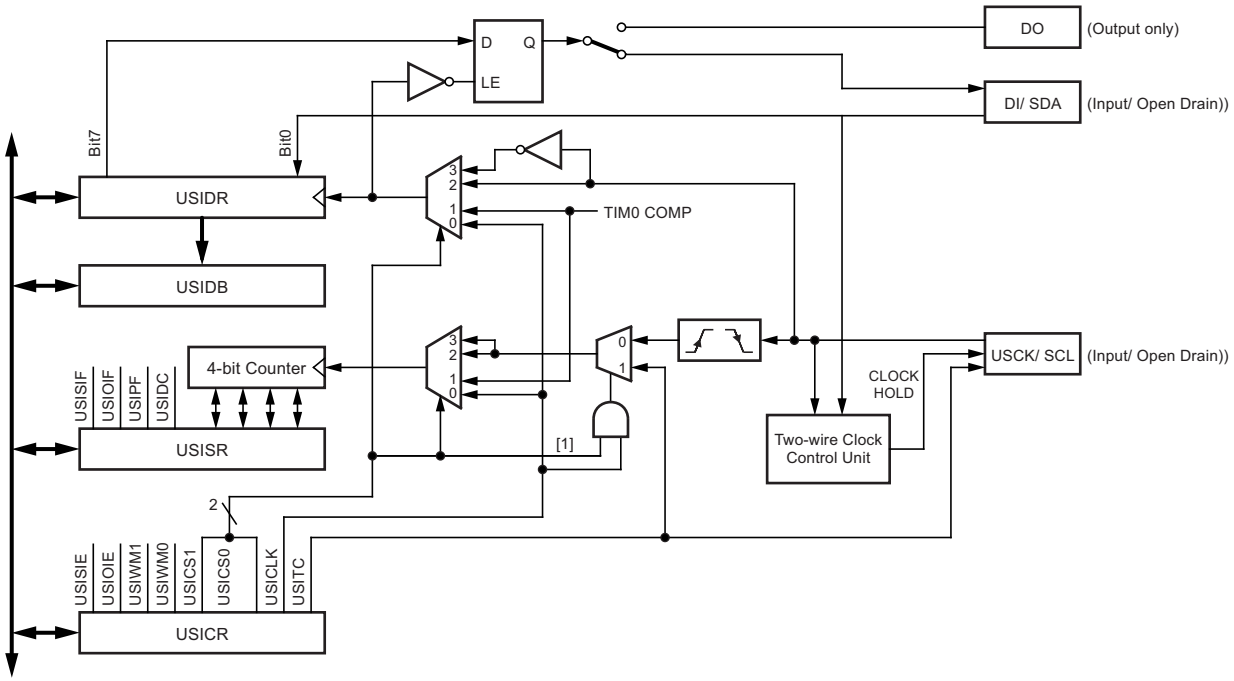
- Two-wire Synchronous Data Transfer (Master or Slave)
- Three-wire Synchronous Data Transfer (Master or Slave)
- Data Received Interrupt
- Wakeup from Idle Mode
- In Two-wire Mode: Wake-up from All Sleep Modes, Including Power-down Mode
- Two-wire Start Condition Detector with Interrupt Capability

5.15.2 Overview

The Universal Serial Interface, or USI, provides the basic hardware resources needed for serial communication. Combined with a minimum of control software, the USI allows significantly higher transfer rates and uses less code space than solutions based on software only. Interrupts are included to minimize the processor load.

A simplified block diagram of the USI is shown on Figure 5-62. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in Section 5.15.5 “Register Descriptions” on page 177.

Figure 5-62. Universal Serial Interface, Block Diagram



The 8-bit USI Data Register is directly accessible via the data bus and contains the incoming and outgoing data. The register has no buffering so the data must be read as quickly as possible to ensure that no data is lost. The USI Data Register is a serial shift register and the most significant bit that is the output of the serial shift register is connected to one of two output pins depending of the wire mode configuration. A transparent latch is inserted between the USI Data Register Output and output pin, which delays the change of data output to the opposite clock edge of the data input sampling. The serial input is always sampled from the Data Input (DI) pin independent of the configuration.

The 4-bit counter can be both read and written via the data bus, and can generate an overflow interrupt. Both the USI Data Register and the counter are clocked simultaneously by the same clock source. This allows the counter to count the number of bits received or transmitted and generate an interrupt when the transfer is complete. Note that when an external clock source is selected the counter counts both clock edges. In this case the counter counts the number of edges, and not the number of bits. The clock can be selected from three different sources: The USCK pin, Timer/Counter0 Compare Match or from software. The Two-wire clock control unit can generate an interrupt when a start condition is detected on the Two-wire bus. It can also generate wait states by holding the clock pin low after a start condition is detected, or after the counter overflows.

5.16.5.6 Bit Timing

Baud rate Generator

The baud rate is defined to be the transfer rate in bits per second (bps):

- BAUD: Baud rate (in bps)
- $fclk_{i/o}$: System I/O clock frequency
- LDIV[11..0]: Contents of LINBRRH & LINBRRL registers - (0-4095), the pre-scaler receives $clk_{i/o}$ as input clock
- LBT[5..0]: Least significant bits of - LINBTR register- (0-63) is the number of samplings in a LIN or UART bit (default value 32)

Equation for calculating baud rate:

$$BAUD = fclk_{i/o} / LBT[5..0] \times (LDIV[11..0] + 1)$$

Equation for setting LINDIV value:

$$LDIV[11..0] = (fclk_{i/o} / LBT[5..0] \times BAUD) - 1$$

Note that in reception a majority vote on three samplings is made.

Re-synchronization in LIN Mode

When waiting for Rx Header, LBT[5..0] = 32 in LINBTR register. The re-synchronization begins when the BREAK is detected. If the BREAK size is not in the range (10.5 bits min., 28 bits max. — 13 bits nominal), the BREAK is refused. The re-synchronization is done by adjusting LBT[5..0] value to the SYNCH field of the received header (0x55). Then the PROTECTED IDENTIFIER is sampled using the new value of LBT[5..0]. The re-synchronization implemented in the controller tolerates a clock deviation of ± 20% and adjusts the baud rate in a ±2% range.

The new LBT[5..0] value will be used up to the end of the response. Then, the LBT[5..0] will be reset to 32 for the next header.

The LINBTR register can be used to (software) re-calibrate the clock oscillator.

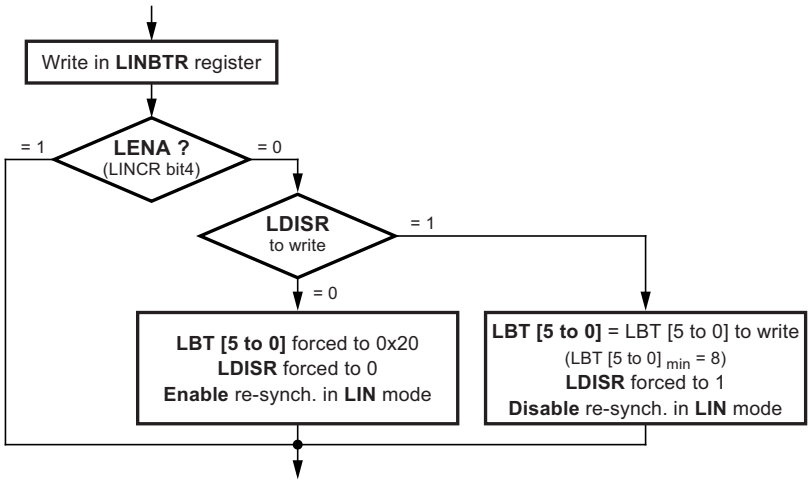
The re-synchronization is not performed if the LIN node is enabled as a master.

Handling LBT[5..0]

- LDISR bit of LINBTR register is used to:
- Disable the re-synchronization (for instance in the case of LIN MASTER node),
- To enable the setting of LBT[5..0] (to manually adjust the baud rate especially in the case of UART mode). A minimum of 8 is required for LBT[5..0] due to the sampling operation.

Note that the LENA bit of LINCR register is important for this handling (see Figure 5-75 on page 190).

Figure 5-75. Handling LBT[5..0]



5.16.6.2 LIN Status and Interrupt Register - LINSIR

Bit	7	6	5	4	3	2	1	0	
	LIDST2	LIDST1	LIDST0	LBUSY	LERR	LIDOK	LTXOK	LRXOK	LINSIR
Read/Write	R	R	R	R	R/Wone	R/Wone	R/Wone	R/Wone	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:5 - LIDST[2:0]: Identifier Status**
  - 0xx = no specific identifier,
  - 100 = Identifier 60 (0x3C),
  - 101 = Identifier 61 (0x3D),
  - 110 = Identifier 62 (0x3E),
  - 111 = Identifier 63 (0x3F).
- **Bit 4 - LBUSY: Busy Signal**
  - 0 = Not busy,
  - 1 = Busy (receiving or transmitting).
- **Bit 3 - LERR: Error Interrupt**

  It is a logical OR of LINERR register bits. This bit generates an interrupt if its respective enable bit - LENERR - is set in LINENIR.
  - 0 = No error,
  - 1 = An error has occurred.

  The user clears this bit by writing 1 in order to reset this interrupt. Resetting LERR also resets all LINERR bits.

  In UART mode, this bit is also cleared by reading LINDAT.
- **Bit 2 - LIDOK: Identifier Interrupt**

  This bit generates an interrupt if its respective enable bit - LENIDOK - is set in LINENIR.
  - 0 = No identifier,
  - 1 = Slave task: Identifier present, master task: Tx Header complete.

  The user clears this bit by writing 1, in order to reset this interrupt.
- **Bit 1 - LTXOK: Transmit Performed Interrupt**

  This bit generates an interrupt if its respective enable bit - LENTXOK - is set in LINENIR.
  - 0 = No Tx,
  - 1 = Tx Response complete.

  The user clears this bit by writing 1, in order to reset this interrupt.

  In UART mode, this bit is also cleared by writing LINDAT.
- **Bit 0 - LRXOK: Receive Performed Interrupt**

  This bit generates an interrupt if its respective enable bit - LENRXOK - is set in LINENIR.
  - 0 = No Rx
  - 1 = Rx Response complete.

  The user clears this bit by writing 1, in order to reset this interrupt.

  In UART mode, this bit is also cleared by reading LINDAT.

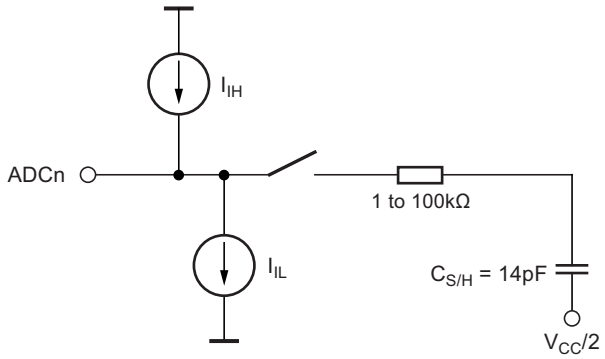
5.18.7.1 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 5-89. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10 kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedant sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

Signal components higher than the Nyquist frequency ($f_{ADC}/2$) should not be present to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 5-89. Analog Input Circuitry



5.18.7.2 Analog Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

a. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.

b. Use the ADC noise canceler function to reduce induced noise from the CPU.

c. If any port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

5.18.12.3 ADCL and ADCH – The ADC Data Register

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	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	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	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 in ADMUX, and the MUXn bits in ADMUX 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.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in Section 5.18.8 “ADC Conversion Result” on page 215.

5.21.2.7 Simple Assembly Code Example for a Boot Loader

Note that the RWWSB bit will always be read as zero in Atmel® ATtiny87/167. Nevertheless, it is recommended to check this bit as shown in the code example, to ensure compatibility with devices supporting Read-While-Write.

```
        ;-      The routine writes one page of data from RAM to Flash
        ;       the first data location in RAM is pointed to by the Y-pointer
        ;       the first data location in Flash is pointed to by the Z-pointer
        ;-      Error handling is not included
        ;-      Registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
        ;                       loophi (r25), spmcsrval (r20)
        ; -     Storing and restoring of registers is not included in the routine
        ;       register usage can be optimized at the expense of code size

.equ            PAGESIZEB = PAGESIZE*2     ; AGESIZEB is page size in BYTES, not words
.org            SMALLBOOTSTART

Write_page:
        ;       Page Erase
        ldi     spmcsrval, (1<<PGERS) | (1<<SELFPGEN)
        rcall   Do_spm

        ;       Clear temporary page buffer
        ldi     spmcsrval, (1<<CPTB) | (1<<SELFPGEN)
        rcall   Do_spm

        ;       Transfer data from RAM to Flash temporary page buffer
        ldi     looplo, low(PAGESIZEB)      ; init loop variable
        ldi     loophi, high(PAGESIZEB)     ; not required for PAGESIZEB<=256

Wrloop:
        ld       r0, Y+
        ld      r1, Y+
        ldi     spmcsrval, (1<<SELFPGEN)
        rcall   Do_spm
        adiw    ZH:ZL, 2
        sbiw    loophi:looplo, 2            ; use subi for PAGESIZEB<=256
        brne    Wrloop

        ;       Execute Page Write
        subi    ZL, low(PAGESIZEB)          ; restore pointer
        sbci    ZH, high(PAGESIZEB)         ; not required for PAGESIZEB<=256
        ldi     spmcsrval, (1<<PGWRT) | (1<<SELFPGEN)
        rcall   Do_spm

        ;       Clear temporary page buffer
        ldi     spmcsrval, (1<<CPTB) | (1<<SELFPGEN)
        rcall   Do_spm

        ;       Read back and check, optional
        ldi     looplo, low(PAGESIZEB)      ; init loop variable
        ldi     loophi, high(PAGESIZEB)     ; not required for PAGESIZEB<=256
        subi    YL, low(PAGESIZEB)          ; restore pointer
        sbci    YH, high(PAGESIZEB)
```

5.22.5 Page Size

Table 5-72. Number of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATtiny87	4K words	64 words	PC[5:0]	64	PC[11:6]	11
ATtiny167	8K words	64 words	PC[5:0]	128	PC[12:6]	12

Table 5-73. Number of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATtiny87 ATtiny167	512bytes	4bytes	EEA[1:0]	128	EEA[8:2]	8

5.22.6 Parallel Programming Parameters, Pin Mapping, and Commands

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the Atmel® ATtiny87/167. Pulses are assumed to be at least 250 ns unless otherwise noted.

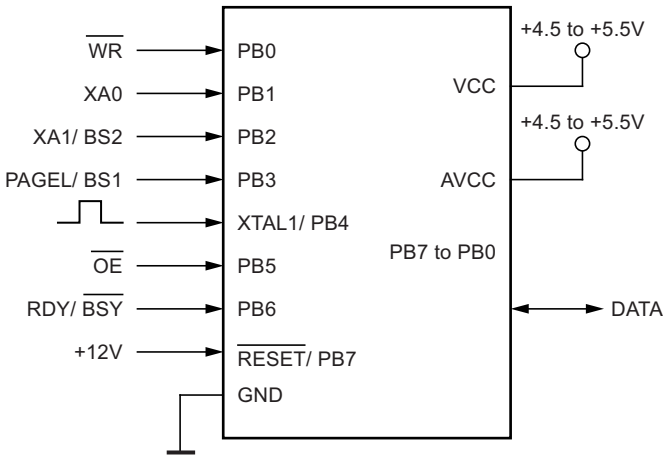
5.22.6.1 Signal Names

In this section, some pins of the Atmel ATtiny87/167 are referenced by signal names describing their functionality during parallel programming, see Figure 5-97 and Figure 5-98 on page 239. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in Figure 5-75 on page 239.

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different commands are shown in Figure 5-76 on page 240.

Figure 5-97. Parallel Programming



Note: Vcc – 0.3V < AVcc < Vcc + 0.3V, however, AVcc should always be within 4.5 to 5.5V

Table 5-76. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000 $_b$	Chip Erase
0100 0000 $_b$	Write Fuse bits
0010 0000 $_b$	Write Lock bits
0001 0000 $_b$	Write Flash
0001 0001 $_b$	Write EEPROM
0000 1000 $_b$	Read Signature bytes and Calibration byte
0000 0100 $_b$	Read Fuse and Lock bits
0000 0010 $_b$	Read Flash
0000 0011 $_b$	Read EEPROM

5.22.7 Parallel Programming

5.22.7.1 Enter Programming Mode

The following algorithm puts the device in parallel programming mode:

1. Apply 4.5V - 5.5V between Vcc and GND.
2. Set RESET to “0” and toggle XTAL1 at least six times.
3. Set the Prog_enable pins listed in Table 5-74 on page 239 to “0000 $_b$” and wait at least 100 ns.
4. Apply 11.5V - 12.5V to RESET. Any activity on Prog_enable pins within 100ns after +12V has been applied to RESET, will cause the device to fail entering programming mode.
5. Wait at least 50µs before sending a new command.

5.22.7.2 Considerations for Efficient Programming

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Skip writing the data value 0xFF, that is the contents of the entire EEPROM (unless the EESAVE Fuse is programmed) and Flash after a Chip Erase.
- Address high byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This consideration also applies to Signature bytes reading.

5.25.7 Internal Oscillator Speed

Figure 5-132. Calibrated 8.0MHz RC Oscillator Frequency versus Vcc

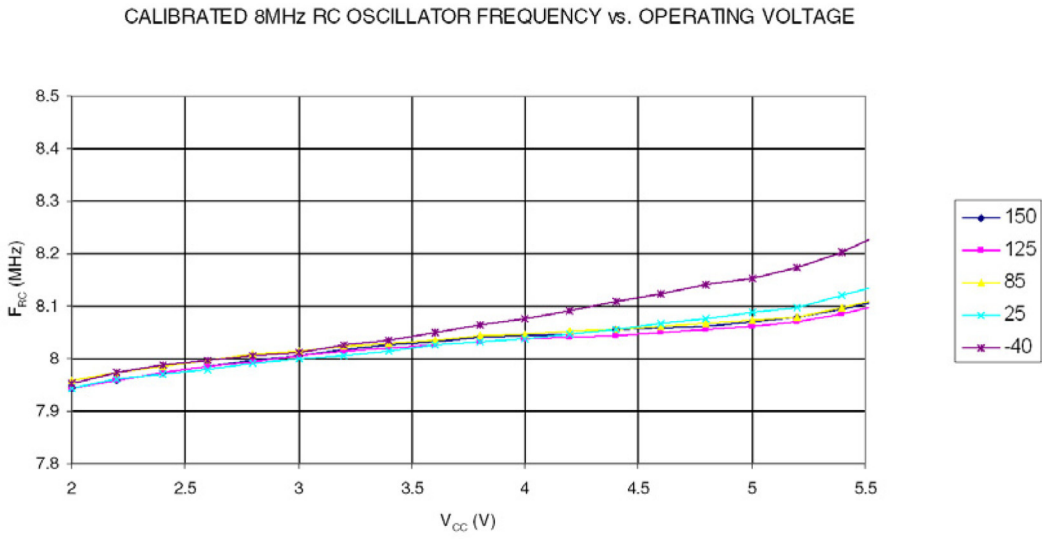


Figure 5-133. Calibrated 8.0MHz RC Oscillator Frequency versus OSCCAL Value