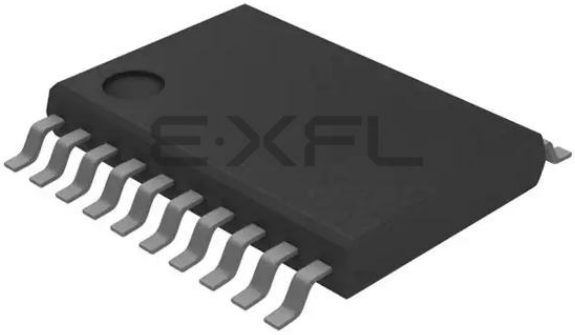


Welcome to E-XFL.COM

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, Temp Sensor, WDT
Number of I/O	16
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	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 150°C (TA)
Mounting Type	Surface Mount
Package / Case	20-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	20-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny167-15xd

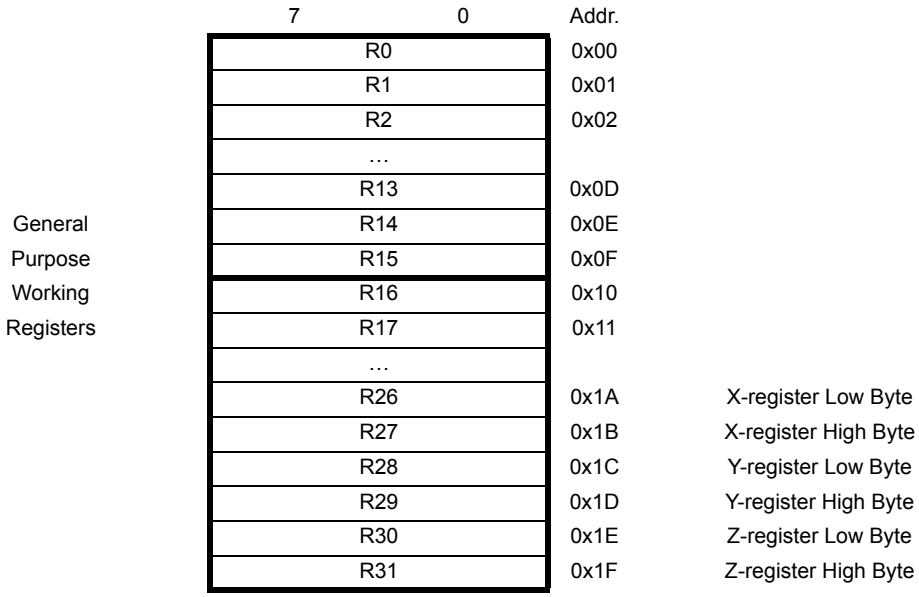
2.4 General Purpose Register File

The register file is optimized for the AVR® enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the register file:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Figure 2-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 2-2. AVR CPU General Purpose Working Registers



Most of the instructions operating on the register file have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 2-2, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user data space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y- and Z-pointer registers can be set to index any register in the file.

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

The registers R26.R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the data space. The three indirect address registers X, Y, and Z are defined as described in Figure 2-3 on page 12.

The following code examples show one assembly and one C function for erase, write, or atomic write of the EEPROM. The examples assume that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic   EECR,EEPE
  rjmp   EEPROM_write
  ; Set Programming mode
  ldi    r16, (0<<EEPM1)|(0<<EEPM0)
  out    EECR, r16
  ; Set up address (r18:r17) in address register
  out    EEARH, r18
  out    EEARL, r17
  ; Write data (r16) to data register
  out    EEDR, r16
  ; Write logical one to EEMPE
  sbi    EECR,EEMPE
  ; Start eeprom write by setting EEPE
  sbi    EECR,EEPE
  ret
```

C Code Example

```
void EEPROM_write(unsigned char ucAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEPE))
    ;
  /* Set Programming mode */
  EECR = (0<<EEPM1)|(0<<EEPM0);
  /* Set up address and data registers */
  EEAR = ucAddress;
  EEDR = ucData;
  /* Write logical one to EEMPE */
  EECR |= (1<<EEMPE);
  /* Start eeprom write by setting EEPE */
  EECR |= (1<<EEPE);
}
```

4.1.1 CPU Clock – clk_{CPU}

The CPU clock is routed to parts of the system concerned with the AVR® core operation. Examples of such modules are the general purpose register file, the status register and the data memory holding the stack pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

4.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like synchronous timer/counter. The I/O clock is also used by the external interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted.

4.1.3 Flash Clock – clk_{FLASH}

The flash clock controls operation of the flash interface. The flash clock is usually active simultaneously with the CPU clock.

4.1.4 Asynchronous Timer Clock – clk_{ASY}

The asynchronous timer clock allows the asynchronous timer/counter to be clocked directly from an external clock or an external low frequency crystal. The dedicated clock domain allows using this timer/counter as a real-time counter even when the device is in sleep mode.

4.1.5 ADC Clock – clk_{ADC}

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

4.2 Clock Sources

The device has the following clock source options, selectable by flash fuse bits (default) or by the CLKSELR register (dynamic clock switch circuit) as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 4-1. Device Clocking Options Select(1) versus PB4 and PB5 Functionality

Device Clocking Option	CKSEL3..0 (2) CSEL3..0 (3)	PB4	PB5
External Clock	0000_b	CLKI	CLKO - I/O
Calibrated Internal RC Oscillator 8.0MHz	0010_b	I/O	CLKO - I/O
Watchdog Oscillator 128kHz	0011_b	I/O	CLKO - I/O
External Low-frequency Oscillator	$01xx_b$	XTAL1	XTAL2
External Crystal/Ceramic Resonator (0.4 - 0.9MHz)	$100x_b$	XTAL1	XTAL2
External Crystal/Ceramic Resonator (0.9 - 3.0MHz)	$101x_b$	XTAL1	XTAL2
External Crystal/Ceramic Resonator (3.0 - 8.0MHz)	$110x_b$	XTAL1	XTAL2
External Crystal/Ceramic Resonator (8.0 - 16.0MHz)	$111x_b$	XTAL1	XTAL2

Notes: 1. For all fuses “1” means unprogrammed while “0” means programmed.
2. Flash fuse bits.
3. CLKSELR register bits.

4.3.2 CLKSELR Register

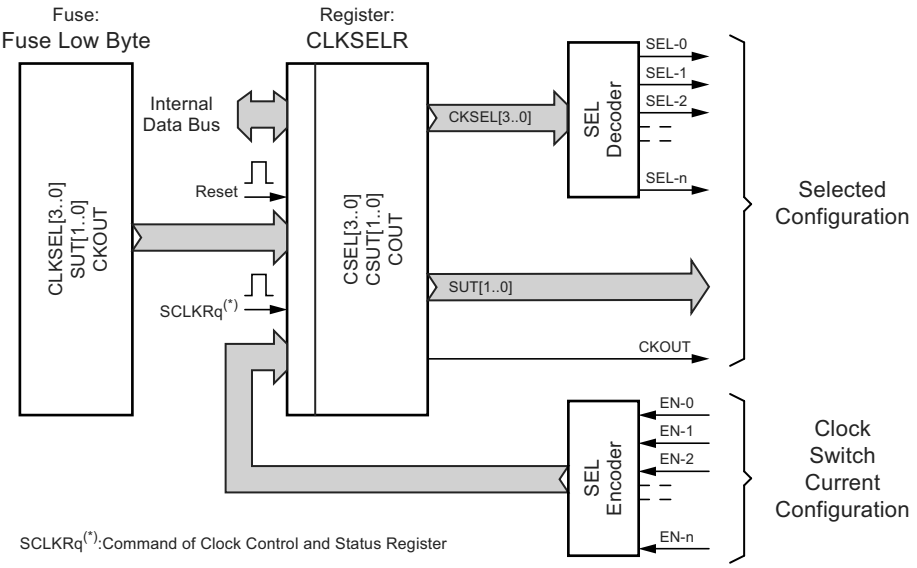
4.3.2.1 Fuses Substitution

At reset, bits of the low fuse byte are copied into the CLKSELR register. The content of this register can subsequently be user modified to overwrite the default values from the low fuse byte. CKSEL3..0, SUT1..0 and CKOUT fuses correspond respectively to CSEL3..0, CSUT1:0 and ~(COUT) bits of the CLKSELR register as shown in Figure 4-5 on page 33.

4.3.2.2 Source Selection

The available codes of clock source are given in Table 4-1 on page 26.

Figure 4-5. Fuses substitution and Clock Source Selection



The CLKSELR register contains the CSEL, CSUT and COUT values which will be used by the ‘enable/disable clock source’, ‘request for clock availability’ or ‘clock source switching’ commands.

4.3.2.3 Source Recovering

The ‘*recover system clock source*’ command updates the CKSEL field of CLKSELR register (Section 4.3.6 “System Clock Source Recovering” on page 34).

4.3.3 Enable/Disable Clock Source

The ‘*enable clock source*’ command selects and enables a clock source configured by the settings in the CLKSELR register. CSEL3..0 will select the clock source and CSUT1:0 will select the start-up time (just as CKSEL and SUT fuse bits do). To be sure that a clock source is operating, the ‘*request for clock availability*’ command must be executed after the ‘*enable clock source*’ command. This will indicate via the CLKRDY bit in the CLKCSR register that a valid clock source is available and operational.

The ‘*disable clock source*’ command disables the clock source indicated by the settings of CLKSELR register (only CSEL3..0). If the clock source indicated is currently the one that is used to drive the system clock, the command is not executed.

Because the selected configuration is latched at clock source level, it is possible to enable many clock sources at a given time (ex: the internal RC oscillator for system clock + an oscillator with external crystal). The user (code) is responsible of this management.

4.3.4 COUT Command

The ‘*CKOUT* ’ command allows to drive the CLKO pin. Refer to Section 4.2.7 “Clock Output Buffer” on page 32 for using.

7. Interrupts

This section describes the specifics of the interrupt handling as performed in Atmel® ATtiny87/167. For a general explanation of the AVR® interrupt handling, refer to “Reset and Interrupt Handling” on page 13.

7.1 Interrupt Vectors in ATtiny87/167

Table 7-1. Reset and Interrupt Vectors in ATtiny87/167

Vector Nb.	Program Address		Source	Interrupt Definition
	ATtiny87	ATtiny167		
1	0x0000	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0001	0x0002	INT0	External Interrupt Request 0
3	0x0002	0x0004	INT1	External Interrupt Request 1
4	0x0003	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0004	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x0005	0x000A	WDT	Watchdog Time-out Interrupt
7	0x0006	0x000C	TIMER1 CAPT	Timer/Counter1 Capture Event
8	0x0007	0x000E	TIMER1 COMPA	Timer/Counter1 Compare Match A
9	0x0008	0x0010	TIMER1 COMPB	Timer/Coutner1 Compare Match B
10	0x0009	0x0012	TIMER1 OVF	Timer/Counter1 Overflow
11	0x000A	0x0014	TIMER0 COMPA	Timer/Counter0 Compare Match A
12	0x000B	0x0016	TIMER0 OVF	Timer/Counter0 Overflow
13	0x000C	0x0018	LIN TC	LIN/UART Transfer Complete
14	0x000D	0x001A	LIN ERR	LIN/UART Error
15	0x000E	0x001C	SPI, STC	SPI Serial Transfer Complete
16	0x000F	0x001E	ADC	ADC Conversion Complete
17	0x0010	0x0020	EE READY	EEPROM Ready
18	0x0011	0x0022	ANALOG COMP	Analog Comparator
19	0x0012	0x0024	USI START	USI Start Condition Detection
20	0x0013	0x0026	USI OVF	USI Counter Overflow

7.2 Program Setup in ATtiny87

The most typical and general program setup for the reset and interrupt vector addresses in Atmel® ATtiny87 is (2-byte step - using “rjmp” instruction):

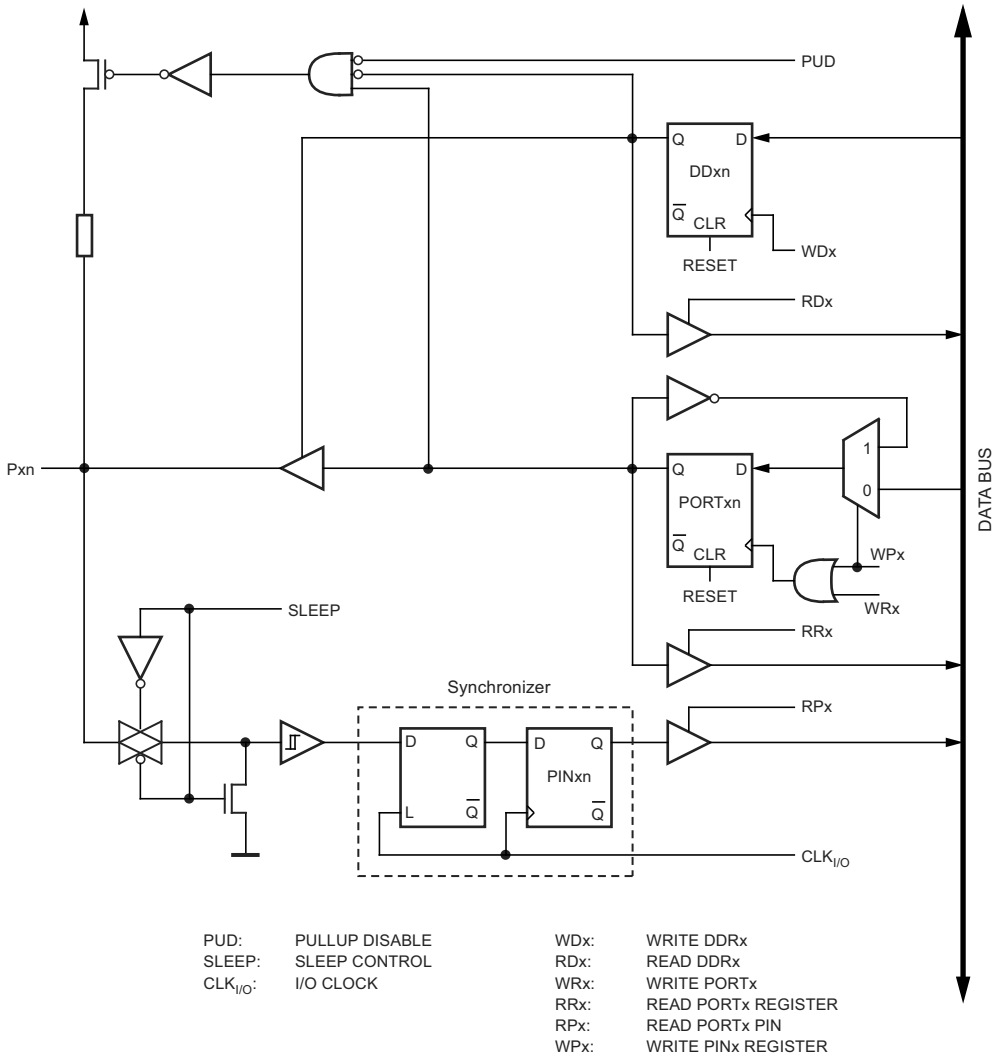
```
Address(1) Label Code                 Comments
0x0000           rjmp   RESET          ; Reset Handler
0x0001           rjmp   INT0addr       ; IRQ0 Handler
0x0002           rjmp   INT1addr       ; IRQ1 Handler
0x0003           rjmp   PCINT0addr     ; PCINT0 Handler
0x0004           rjmp   PCINT1addr     ; PCINT1 Handler
0x0005           rjmp   WDTaddr        ; Watchdog Timer Handler
0x0006           rjmp   ICP1addr       ; Timer1 Capture Handler
0x0007           rjmp   OC1Aaddr       ; Timer1 Compare A Handler
0x0008           rjmp   OC1Baddr       ; Timer1 Compare B Handler
0x0009           rjmp   OVF1addr       ; Timer1 Overflow Handler
0x000A           rjmp   OC0Aaddr       ; Timer0 Compare A Handler
0x000B           rjmp   OVF0addr       ; Timer0 Overflow Handler
0x000C           rjmp   LINTCaddr      ; LIN Transfer Complete Handler
0x000D           rjmp   LINERRaddr     ; LIN Error Handler
0x000E           rjmp   SPIaddr        ; SPI Transfer Complete Handler
0x000F           rjmp   ADCCaddr       ; ADC Conversion Complete Handler
0x0010           rjmp   ERDYaddr       ; EEPROM Ready Handler
0x0011           rjmp   ACIaddr        ; Analog Comparator Handler
0x0012           rjmp   USISTARTaddr   ; USI Start Condition Handler
0x0013           rjmp   USIOVFaddr     ; USI Overflow Handler

0x0014 RESET:    ldi    r16, high(RAMEND); Main program start
0x0015           out    SPH,r16          ; Set Stack Pointer to top of RAM
0x0016           ldi    r16, low(RAMEND)
0x0017           out    SPL,r16
0x0018           sei                     ; Enable interrupts
0x0019           <instr>  xxx
   ...     ...    ...    ...
```

Note: 1. 16-bit address

9.2 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 9-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 9-2. General Digital I/O(1)



Notes: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

9.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in Section 9.4 “Register Description for I/O Ports” on page 82, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

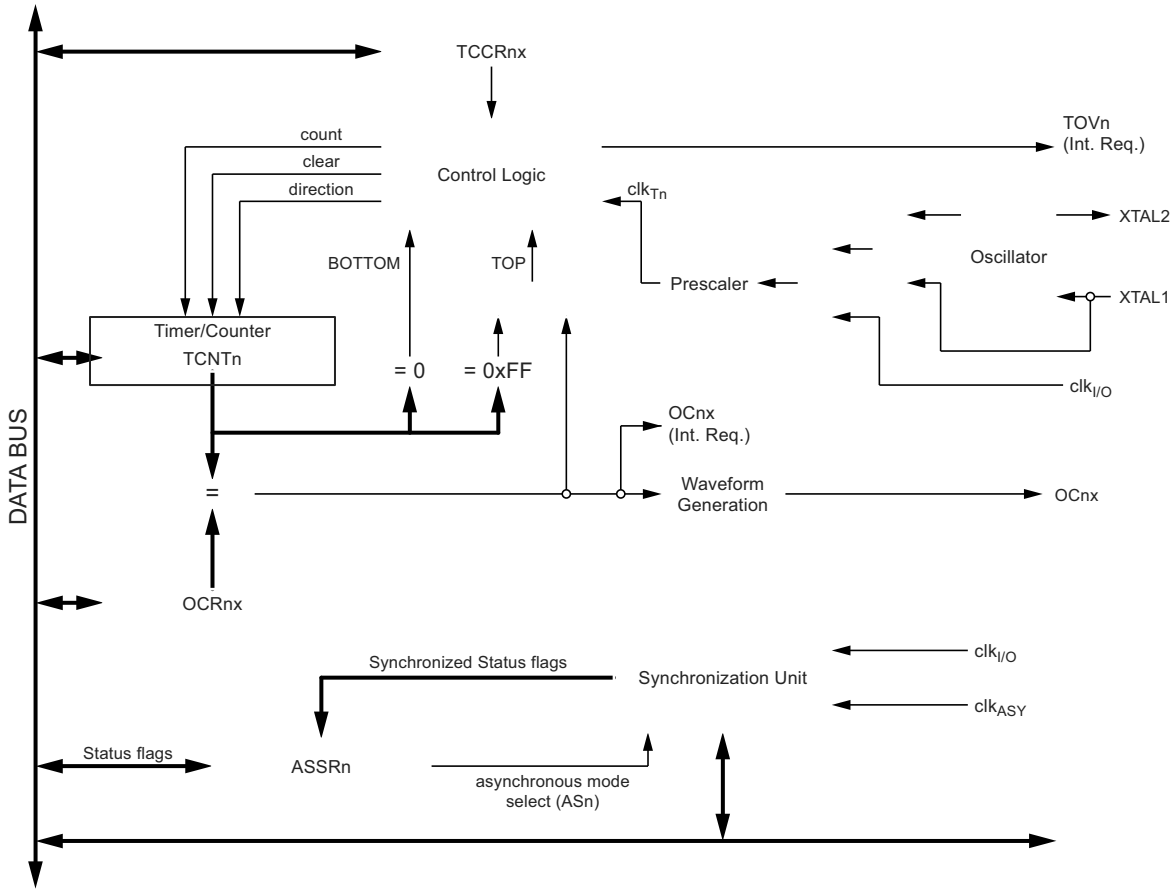
The DDxn bit in the DDRx register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when reset condition becomes active, even if no clocks are running.

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

A simplified block diagram of the 8-bit timer/counter is shown in Figure 10-1. For the actual placement of I/O pins, refer to Section 1.6 “Pin Configuration” on page 6. 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 the Section 10.11 “8-bit Timer/Counter Register Description” on page 95.

Figure 10-1. 8-bit Timer/Counter0 Block Diagram



The timer/counter (TCNT0) and output compare register (OCR0A) are 8-bit registers. Interrupt request (shorten as Int.Req.) signals are all visible in the timer interrupt flag register (TIFR0). All interrupts are individually masked with the timer interrupt mask register (TIMSK0). TIFR0 and TIMSK0 are not shown in the figure.

The timer/counter can be clocked internally, via the prescaler, or asynchronously clocked from the XTAL1/2 pins, as detailed later in this section. The asynchronous operation is controlled by the asynchronous status register (ASSR). The clock select logic block controls which clock source the timer/counter uses to increment (or decrement) its value. The timer/counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_T0).

The double buffered output compare register (OCR0A) is 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 pin (OC0A). Section 10.5 “Output Compare Unit” on page 86 for details. The compare match event will also set the compare flag (OCF0A) which can be used to generate an output compare interrupt request.

- **Bit 3 – OCR0AUB: Output Compare 0 Register A Update Busy**

When timer/counter0 operates asynchronously and OCR0A is written, this bit becomes set. When OCR0A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR0A is ready to be updated with a new value.

- **Bit 2 – Res: Reserved Bit**

This bit is reserved in the Atmel® ATtiny87/167 and will always read as zero.

- **Bit 1 – TCR0AUB: Timer/Counter0 Control Register A Update Busy**

When timer/counter0 operates asynchronously and TCCR0A is written, this bit becomes set. When TCCR0A has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR0A is ready to be updated with a new value.

- **Bit 0 – TCR0BUB: Timer/Counter0 Control Register B Update Busy**

When timer/counter0 operates asynchronously and TCCR0B is written, this bit becomes set. When TCCR0B has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR0B is ready to be updated with a new value.

If a write is performed to any of the four timer/counter0 registers while its update busy flag is set, the updated value might get corrupted and cause an unintentional interrupt to occur.

The mechanisms for reading TCNT0, OCR0A, TCCR0A and TCCR0B are different. When reading TCNT0, the actual timer value is read. When reading OCR0A, TCCR0A or TCCR0B the value in the temporary storage register is read.

10.11.5 Timer/Counter0 Interrupt Mask Register – TIMSK0

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:2 – Res: Reserved Bits**

These bits are reserved in the Atmel ATtiny87/167 and will always read as zero.

- **Bit 1 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one and the I-bit in the status register is set (one), the timer/counter0 compare match A interrupt is enabled. The corresponding interrupt is executed if a compare match in timer/counter0 occurs, i.e., when the OCF0A bit is set in the timer/counter0 interrupt flag register – TIFR0.

- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one and the I-bit in the status register is set (one), the timer/counter0 overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in timer/counter0 occurs, i.e., when the TOV0 bit is set in the timer/counter0 interrupt flag register – TIFR0.

10.11.6 Timer/Counter0 Interrupt Flag Register – TIFR0

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:2 – Res: Reserved Bits**

These bits are reserved in the Atmel ATtiny87/167 and will always read as zero.

15. LIN/UART - Local Interconnect Network Controller or UART

The LIN (local interconnect network) is a serial communications protocol which efficiently supports the control of mechatronics nodes in distributed automotive applications. The main properties of the LIN bus are:

- Single master with multiple slaves concept
- Low cost silicon implementation based on common UART/SCI interface
- Self synchronization with on-chip oscillator in slave node
- Deterministic signal transmission with signal propagation time computable in advance
- Low cost single-wire implementation
- Speed up to 20Kbit/s.

LIN provides a cost efficient bus communication where the bandwidth and versatility of CAN are not required. The specification of the line driver/receiver needs to match the ISO9141 NRZ-standard.

If LIN is not required, the controller alternatively can be programmed as universal asynchronous serial receiver and transmitter (UART).

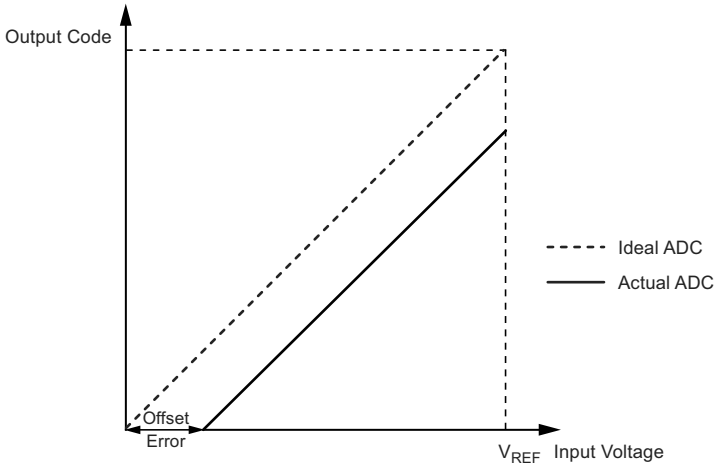
15.1 LIN Features

- Hardware implementation of LIN 2.1 (LIN 1.3 compatibility)
- Small, CPU efficient and independent master/slave routines based on “LIN Work Flow Concept” of LIN 2.1 specification
- Automatic LIN header handling and filtering of irrelevant LIN frames
- Automatic LIN response handling
- Extended LIN error detection and signaling
- Hardware frame time-out detection
- “Break-in-data” support capability
- Automatic re-synchronization to ensure proper frame integrity
- Fully flexible extended frames support capabilities

15.2 UART Features

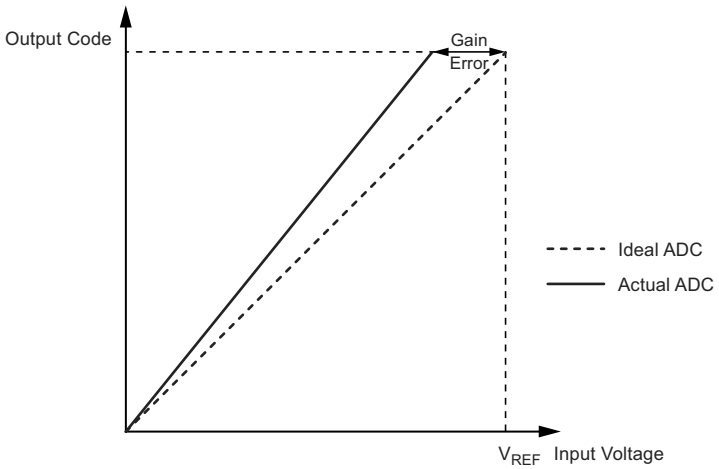
- Full duplex operation (independent serial receive and transmit processes)
- Asynchronous operation
- High resolution baud rate generator
- Hardware support of 8 data bits, odd/even/no parity Bit, 1 stop bit frames
- Data over-run and framing error detection

Figure 17-9. Offset Error



- Gain Error: After adjusting for offset, the gain error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 17-10. Gain Error



17.8 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC result registers (ADCL, ADCH). The form of the conversion result depends on the type of the conversion as there are three types of conversions: single ended conversion, unipolar differential conversion and bipolar differential conversion.

17.8.1 Single Ended Conversion

For single ended conversion, the result is:

$$\mathrm{ADC} = \frac{\mathrm{V_{IN}} \times 1024}{\mathrm{V_{REF}}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 17-4 on page 188 and Table 17-5 on page 189). 0x000 represents analog ground, and 0x3FF represents the selected voltage reference minus one LSB. The result is presented in one-sided form, from 0x3FF to 0x000.

17.8.2 Unipolar Differential Conversion

If differential channels and an unipolar input mode are used, the result is:

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot 1024}{V_{REF}} \cdot GAIN$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference (see Table 17-4 on page 188 and Table 17-5 on page 189). The voltage on the positive pin must always be larger than the voltage on the negative pin or otherwise the voltage difference is saturated to zero. The result is presented in one-sided form, from 0x000 (0_d) to 0x3FF ($+1023_d$). The GAIN is either 8x or 20x.

17.8.3 Bipolar Differential Conversion

As default the ADC converter operates in the unipolar input mode, but the bipolar input mode can be selected by writing the BIN bit in the ADCSRB register to one. In the bipolar input mode two-sided voltage differences are allowed and thus the voltage on the negative input pin can also be larger than the voltage on the positive input pin. If differential channels and a bipolar input mode are used, the result is:

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot 512}{V_{REF}} \cdot GAIN$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (-512_d) through 0x000 ($+0_d$) to 0x1FF ($+511_d$). The GAIN is either 8x or 20x.

However, if the signal is not bipolar by nature (9 bits + sign as the 10th bit), this scheme loses one bit of the converter dynamic range. Then, if the user wants to perform the conversion with the maximum dynamic range, the user can perform a quick polarity check of the result and use the unipolar differential conversion with selectable differential input pair. When the polarity check is performed, it is sufficient to read the MSB of the result (ADC9 in ADCH register). If the bit is one, the result is negative, and if this bit is zero, the result is positive.

18.1.3 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
	ADC7D / AIN1D	ADC6D / AIN0D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
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 7,6 – AIN1D, AIN0D: AIN1D and AIN0D Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding analog compare pin is disabled. The corresponding PIN register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN0/1 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

18.2 Analog Comparator Inputs

18.2.1 Analog Compare Positive Input

It is possible to select any of the inputs of the ADC positive input multiplexer to replace the positive input to the analog comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the analog comparator multiplexer enable bit (ACME in ADCSRB register) is set and the ADC is switched off (ADEN in ADCSRA register is zero), MUX[4..0] in ADMUX register select the input pin to replace the positive input to the analog comparator, as shown in Table 18-2 on page 196. If ACME is cleared or ADEN is set, AIN1 pin is applied to the positive input to the analog comparator.

Table 18-2. Analog Comparator Positive Input

ACME	ADEN	MUX[4..0]	Analog Comparator Positive Input - Comment	
0	x	x xxxx$_b$	AIN1	ADC Switched On
x	1	x xxxx$_b$	AIN1	
1	0	0 0000$_b$	ADC0	ADC Switched Off.
1	0	0 0001$_b$	ADC1	
1	0	0 0010$_b$	ADC2	
1	0	0 0011$_b$	ADC3 / ISRC	
1	0	0 0100$_b$	ADC4	
1	0	0 0101$_b$	ADC5	
1	0	0 0110$_b$	ADC6	
1	0	0 0111$_b$	ADC7	
1	0	0 1000$_b$	ADC8	
1	0	0 1001$_b$	ADC9	
1	0	0 1010$_b$	ADC10	
1	0	Other	This doesn't make sense - Don't use.	

19. DebugWIRE On-chip Debug System

19.1 Features

- Complete program flow control
- Emulates all on-chip functions, both digital and analog, except RESET pin
- Real-time operation
- Symbolic debugging support (both at C and assembler source level, or for other HLLs)
- Unlimited number of program break points (using software break points)
- Non-intrusive operation
- Electrical characteristics identical to real device
- Automatic configuration system
- High-speed operation
- Programming of non-volatile memories

19.2 Overview

The debugWIRE on-chip debug system uses a One-wire, bi-directional interface to control the program flow, execute AVR® instructions in the CPU and to program the different non-volatile memories.

19.3 Physical Interface

When the debugWIRE enable (DWEN) fuse is programmed and lock bits are unprogrammed, the debugWIRE system within the target device is activated. The RESET port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

Figure 19-1. The debugWIRE Setup

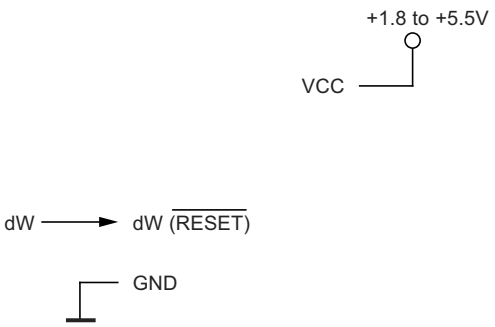
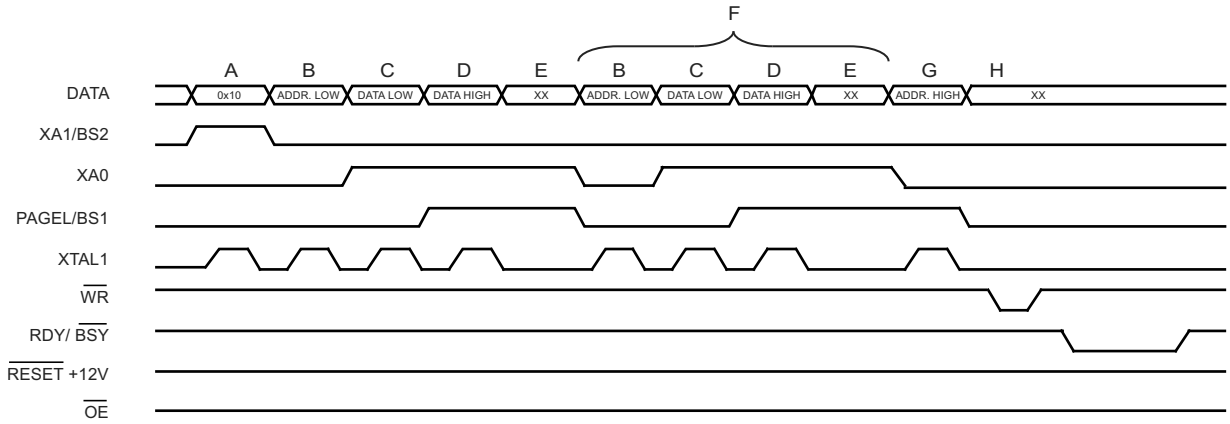


Figure 19-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL fuses.

When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-up resistors on the dW/(RESET) line must not be smaller than 10kΩ. The pull-up resistor is not required for debugWIRE functionality.
- Connecting the RESET pin directly to Vcc will not work.
- Capacitors connected to the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

Figure 21-3. Programming the Flash Waveforms (1)



Note: 1. “XX” is don’t care. The letters refer to the programming description above.

21.7.5 Programming the EEPROM

The EEPROM is organized in pages, see Table 21-8 on page 210. When programming the EEPROM, the program data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (refer to Section 21.7.4 “Programming the Flash” on page 212 for details on command, address and data loading):

A: Load command “0001 0001 $_b$”.

G: Load address high byte (0x00 - 0xFF).

B: Load address low byte (0x00 - 0xFF).

C: Load data (0x00 - 0xFF).

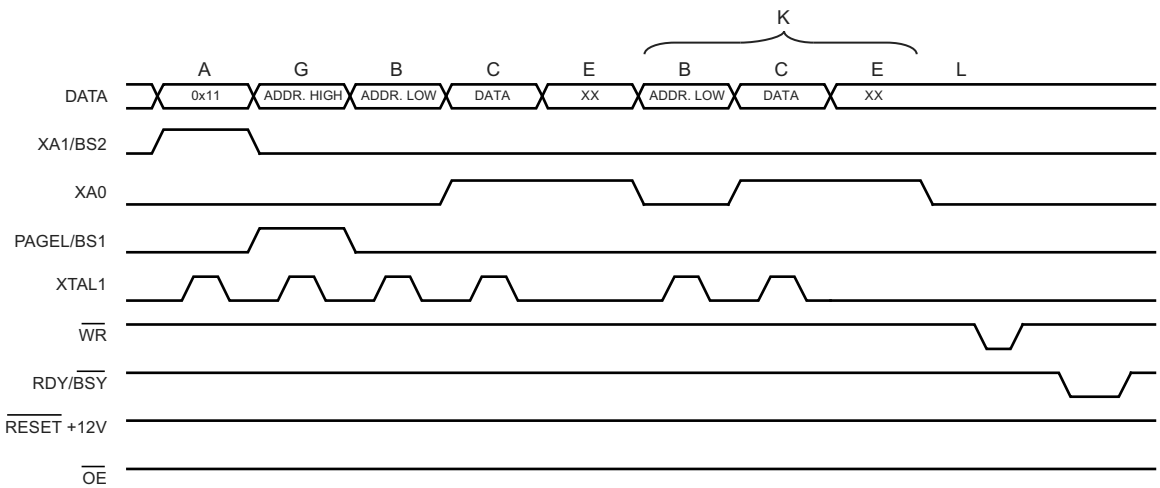
E: Latch data (give PAGEL a positive pulse).

K: Repeat A through E until the entire buffer is filled.

L: Program EEPROM page

1. Set BS1 to “0”.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the EEPROM page. RDY/$\overline{BSY}$ goes low.
3. Wait until to RDY/$\overline{BSY}$ goes high before programming the next page (See Figure 21-4 for signal waveforms).

Figure 21-4. Programming the EEPROM Waveforms



21.7.6 Reading the Flash

The algorithm for reading the flash memory is as follows (refer to Section 21.7.4 “Programming the Flash” on page 212 for details on command and address loading):

1. **A:** Load command “0000 0010 $_b$”.
2. **G:** Load address high byte (0x00 - 0xFF).
3. **B:** Load address low byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The flash word low byte can now be read at DATA.
5. Set BS1 to “1”. The flash word high byte can now be read at DATA.
6. Set $\overline{OE}$ to “1”.

21.7.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Section 21.7.4 “Programming the Flash” on page 212 for details on command and address loading):

1. **A:** Load command “0000 0011 $_b$”.
2. **G:** Load address high byte (0x00 - 0xFF).
3. **B:** Load address low byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The EEPROM data byte can now be read at DATA.
5. Set $\overline{OE}$ to “1”.

21.7.8 Programming the Fuse Low Bits

The algorithm for programming the fuse low bits is as follows (refer to Section 21.7.4 “Programming the Flash” on page 212 for details on command and data loading):

1. **A:** Load command “0100 0000 $_b$”.
2. **C:** Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

21.7.9 Programming the Fuse High Bits

The algorithm for programming the fuse high bits is as follows (refer to Section 21.7.4 “Programming the Flash” on page 212 for details on command and data loading):

1. **A:** Load command “0100 0000 $_b$”.
2. **C:** Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS1 to “0”. This selects low data byte.

21.7.10 Programming the Extended Fuse Bits

The algorithm for programming the extended fuse bits is as follows (refer to Section 21.7.4 “Programming the Flash” on page 212 for details on command and data loading):

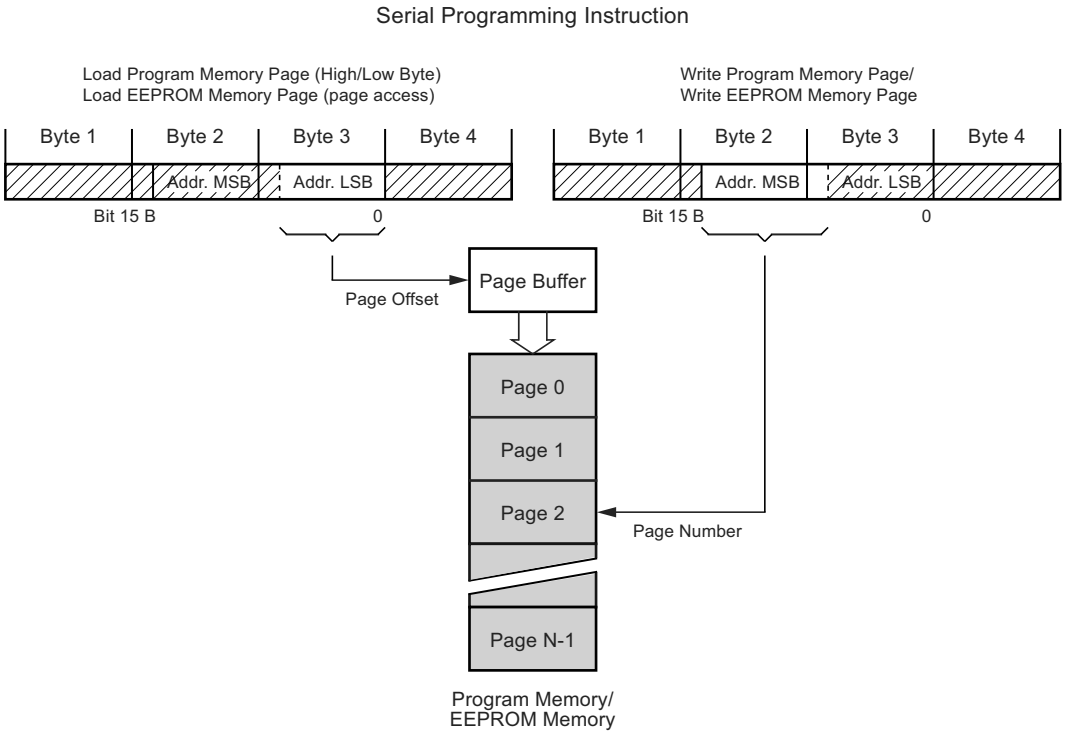
1. **A:** Load command “0100 0000 $_b$”.
2. **C:** Load data low byte. Bit n = “0” programs and bit n = “1” erases the fuse bit.
3. Set BS1 to “0” and BS2 to “1”. This selects extended data byte.
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
5. Set BS2 to “0”. This selects low data byte.

If the LSB in RDY/BSY data byte out is ‘1’, a programming operation is still pending. Wait until this bit returns ‘0’ before the next instruction is carried out.

Within the same page, the low data byte must be loaded prior to the high data byte.

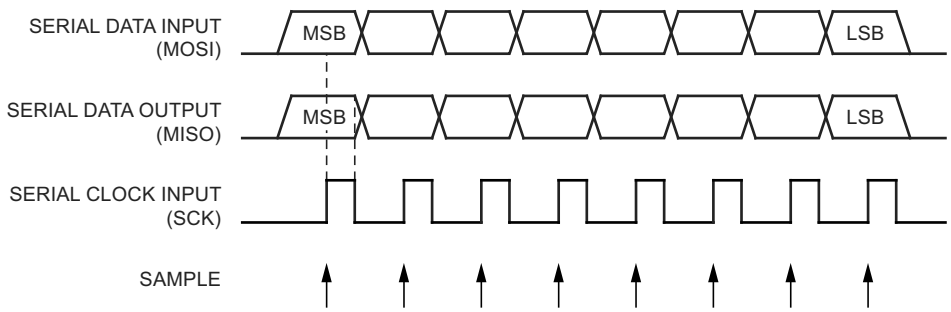
After data is loaded to the page buffer, program the EEPROM page, see Figure 21-8.

Figure 21-8. Serial programming Instruction Example



21.9 Serial Programming Characteristics

Figure 21-9. Serial Programming Waveforms



For characteristics of the SPI module, see Section 22.10 “SPI Timing Characteristics” on page 231

22.2 DC Characteristics (Continued)

T_A = -40°C to +125°C, Vcc = 2.7V to 5.5V (unless otherwise noted)

Parameter	Condition	Symbol	Min.	Typ.(1)	Max.	Units
Power supply current(6) Active mode (external clock)	16MHz, Vcc = 5V	I_{CC}		10	13	mA
	8MHz, Vcc = 5V			5.5	7.0	mA
	8MHz, Vcc = 3V			2.8	3.5	mA
	4MHz, Vcc = 3V			1.8	2.5	mA
Power supply current(6) Idle mode (external clock)	16MHz, Vcc = 5V			3.5	5.0	mA
	8MHz, Vcc = 5V			1.8	2.5	mA
	8MHz, Vcc = 3V			1	1.5	mA
	4MHz, Vcc = 3V			0.5	0.8	mA
Power supply current(7) Power-down mode	WDT enabled, Vcc = 5V			7	100	µA
	WDT disabled, Vcc = 5V			0.18	70	µA
	WDT enabled, Vcc = 3V			5	70	µA
	WDT disabled, Vcc = 3V			0.15	45	µA
Analog comparator Input offset voltage	Vcc = 5V V_{in} = Vcc/2	V_{ACIO}	-10	10	40	mV
Analog comparator Input leakage current	Vcc = 5V V_{in} = Vcc/2	I_{ACLK}	-50		50	nA
Analog comparator Propagation delay Common mode Vcc/2	Vcc = 2.7V	t_{ACID}		170		ns
	Vcc = 5.0V			180		ns

Notes:
1. “Typ.”, typical values at 25°C. Maximum values are characterized values and not test limits in production.
2. “Max.” means the highest value where the pin is guaranteed to be read as low.
3. “Min.” means the lowest value where the pin is guaranteed to be read as high.
4. Although each I/O port can sink more than the test conditions (10mA at Vcc= 5V, 5mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed: The sum of all IOL, for all ports, should not exceed 120mA. If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
5. Although each I/O port can source more than the test conditions (10mA at Vcc = 5V, 5mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed: The sum of all IOH, for all ports, should not exceed 120mA. If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
6. Values using methods described in Section 5.8 “Minimizing Power Consumption” on page 44. Power reduction is enabled (PRR = 0xFF) and there is no I/O drive.
7. BOD disabled.

Figure 24-16. I/O Pin Output Voltage versus Source Current (Vcc = 3V)

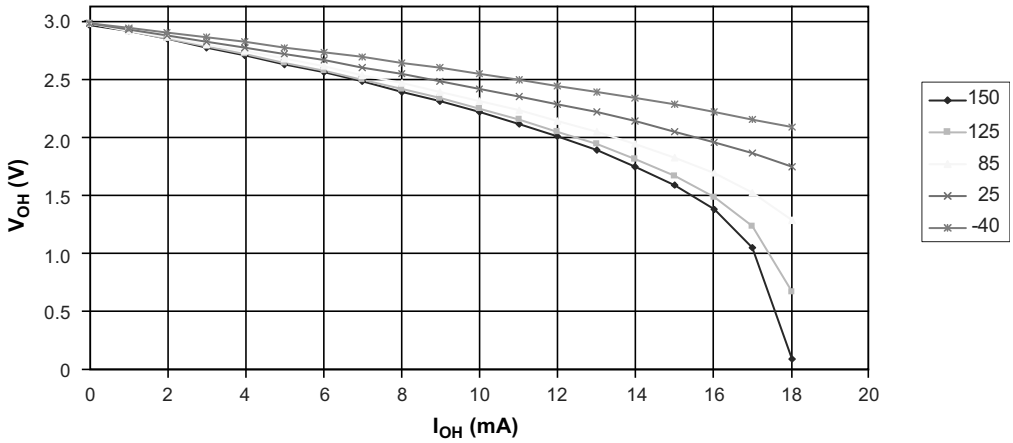
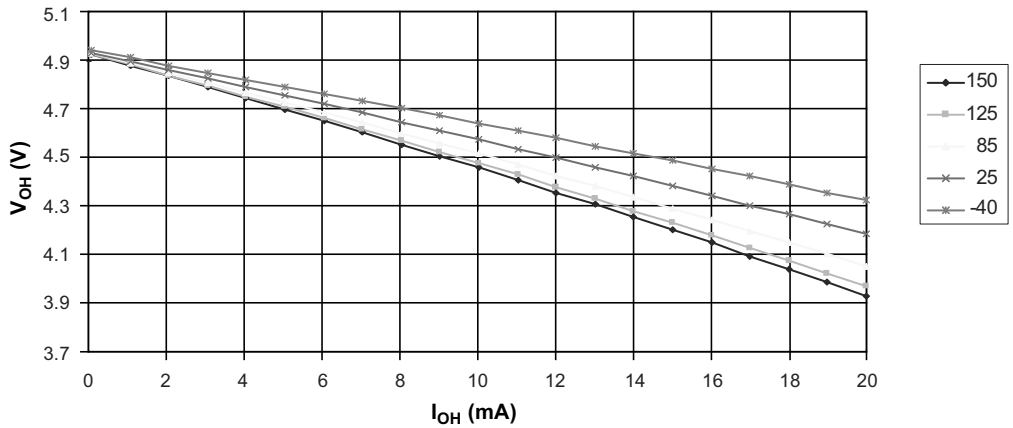


Figure 24-17. I/O Pin Output Voltage versus Source Current (Vcc = 5V)



24.7 Internal Oscillator Speed

Figure 24-18. Calibrated 8.0MHz RC Oscillator Frequency versus Vcc

4. Comparison between ADC inputs and voltage references

   In the analog comparator module, comparing any ADC input (ADC[10..0]) with voltage references (2.56V, 1.28V, 1.10V, 0.64V or 0.32V) fails.

   Regardless, AIN1 input can be compared with the voltage references and any ADC input can be compared with AIN0 input.

   Problem Fix/Workaround

   Do not use this configuration.

5. Register bits of DIDR1

   ADC8D, ADC9D and ADC10D (digital input disable) initially located at bit 4 up to 6 are instead located at bit 0 up to 2. These register bits are also in write only mode.

   Problem Fix/Workaround

   Allow for the change in bit locations and the access mode restriction.

6. LIN Break Delimiter

   In SLAVE MODE, a BREAK field detection error can occur under following conditions.

   The problem occurs if 2 conditions occur simultaneously:

   a. The DOMINANT part of the BREAK is (N+0.5)*Tbit long with N=13, 14,15, ...
   b. The RECESSIVE part of the BREAK (BREAK DELIMITER) is equal to 1*Tbit. (see note below)

   The BREAK_high is not detected, and the 2nd bit of the SYNC field is interpreted as the BREAK DELIMITER.

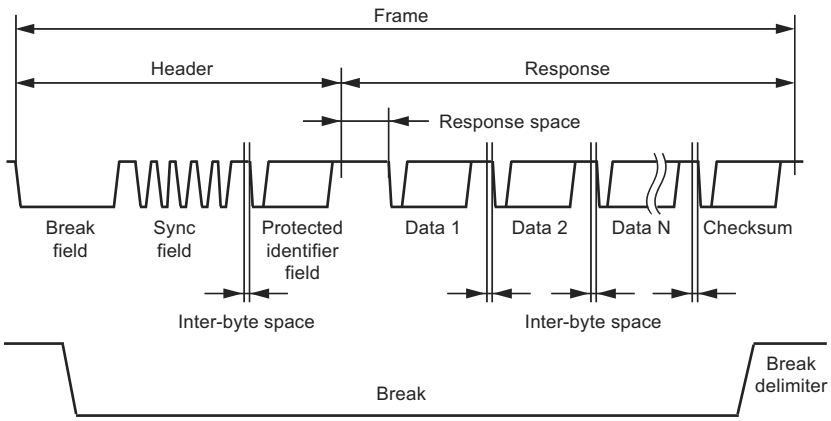
   The error is detected as a framing error on the first bits of the PID or on subsequent Data or a Checksum error.

   There is no error if BREAK_high is greater than 1 × Tbit + 18%.

   There is no problem in master mode.

Note: LIN2.1 protocol specification paragraph 2.3.1.1 Break field says: “A break field is always generated by the master task(in the master node) and it shall be at least 13 nominal bit times of dominant value, followed by a break delimiter, as shown in Figure 29-1. **The break delimiter shall be at least one nominal bit time long.**”

Figure 29-1. The Break Field



Workaround

None