



Welcome to [E-XFL.COM](https://www.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	8MHz
Connectivity	SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, LCD, POR, PWM, WDT
Number of I/O	54
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-VFQFN Exposed Pad
Supplier Device Package	64-QFN (9x9)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/atmega169pv-8mu">https://www.e-xfl.com/product-detail/microchip-technology/atmega169pv-8mu</a>

## 2.2.7 Port E (PE7:PE0)

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

Port E also serves the functions of various special features of the ATmega169P as listed on ["Alternate Functions of Port E" on page 81](#).

## 2.2.8 Port F (PF7:PF0)

Port F serves as the analog inputs to the A/D Converter.

Port F also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PF7(TDI), PF5(TMS), and PF4(TCK) will be activated even if a reset occurs.

Port F also serves the functions of the JTAG interface, see ["Alternate Functions of Port F" on page 83](#).

## 2.2.9 Port G (PG5:PG0)

Port G is a 6-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port G output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port G pins that are externally pulled low will source current if the pull-up resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port G also serves the functions of various special features of the ATmega169P as listed on [page 85](#).

## 2.2.10 $\overline{\text{RESET}}$

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. The minimum pulse length is given in [Table 28-4 on page 333](#). Shorter pulses are not guaranteed to generate a reset.

## 2.2.11 XTAL1

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

## 2.2.12 XTAL2

Output from the inverting Oscillator amplifier.

## 2.2.13 AVCC

AVCC is the supply voltage pin for Port F and the A/D Converter. It should be externally connected to  $V_{CC}$ , even if the ADC is not used. If the ADC is used, it should be connected to  $V_{CC}$  through a low-pass filter.

## 6.4.1 SPH and SPL – Stack Pointer

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	–	–	–	–	–	SP10	SP9	SP8	SPH
0x3D (0x5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
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	
	0	0	0	0	0	0	0	0	

## 6.5 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock  $\text{clk}_{\text{CPU}}$ , directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 6-2 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

**Figure 6-2.** The Parallel Instruction Fetches and Instruction Executions

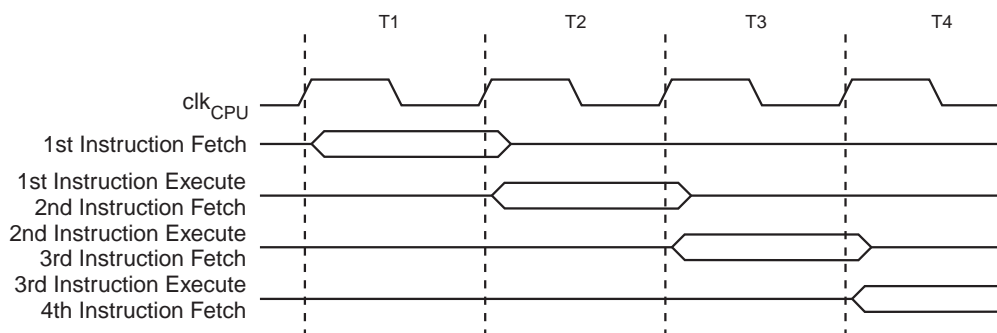
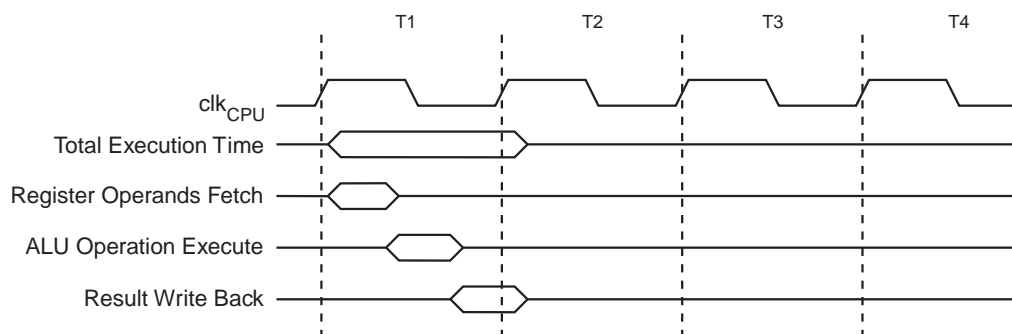


Figure 6-3 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

**Figure 6-3.** Single Cycle ALU Operation



## 9.2 Idle Mode

When the SM2..0 bits are written to 000, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing LCD controller, the SPI, USART, Analog Comparator, ADC, USI, Timer/Counters, Watchdog, and the interrupt system to continue operating. This sleep mode basically halts  $\text{clk}_{\text{CPU}}$  and  $\text{clk}_{\text{FLASH}}$ , while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow and USART Transmit Complete interrupts. If wake-up from the Analog Comparator interrupt is not required, the Analog Comparator can be powered down by setting the ACD bit in the Analog Comparator Control and Status Register – ACSR. This will reduce power consumption in Idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

## 9.3 ADC Noise Reduction Mode

When the SM2..0 bits are written to 001, the SLEEP instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the ADC, the external interrupts, the USI start condition detection, Timer/Counter2, LCD Controller, and the Watchdog to continue operating (if enabled). This sleep mode basically halts  $\text{clk}_{\text{I/O}}$ ,  $\text{clk}_{\text{CPU}}$ , and  $\text{clk}_{\text{FLASH}}$ , while allowing the other clocks to run.

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart from the ADC Conversion Complete interrupt, only an External Reset, a Watchdog Reset, a Brown-out Reset, an LCD controller interrupt, USI start condition interrupt, a Timer/Counter2 interrupt, an SPM/EEPROM ready interrupt, an external level interrupt on INT0 or a pin change interrupt can wake up the MCU from ADC Noise Reduction mode.

## 9.4 Power-down Mode

When the SM2..0 bits are written to 010, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the external Oscillator is stopped, while the external interrupts, the USI start condition detection, and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, USI start condition interrupt, an external level interrupt on INT0, or a pin change interrupt can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. Refer to ["External Interrupts" on page 61](#) for details.

When waking up from Power-down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL Fuses that define the Reset Time-out period, as described in ["Clock Sources" on page 31](#).

```

0x0031      out    SPL,r16
0x0032      sei                      ; Enable interrupts
0x0033      <instr>  xxx
...

```

When the BOOTRST Fuse is unprogrammed, the Boot section size set to 2 Kbytes and the IVSEL bit in the MCUCR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

Address	Labels	Code	Comments
0x0000	RESET:	ldi r16,high(RAMEND);	Main program start
0x0001		out SPH,r16	; Set Stack Pointer to top of RAM
0x0002		ldi r16,low(RAMEND)	
0x0003		out SPL,r16	
0x0004		sei	; Enable interrupts
0x0005		<instr> xxx	
;			
.org 0x1C02			
0x1C02		jmp EXT_INT0	; IRQ0 Handler
0x1C04		jmp PCINT0	; PCINT0 Handler
...	...	...	;
0x1C2C		jmp SPM_RDY	; Store Program Memory Ready Handler

When the BOOTRST Fuse is programmed and the Boot section size set to 2 Kbytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

Address	Labels	Code	Comments
.org 0x0002			
0x0002		jmp EXT_INT0	; IRQ0 Handler
0x0004		jmp PCINT0	; PCINT0 Handler
...	...	...	;
0x002C		jmp SPM_RDY	; Store Program Memory Ready Handler
;			
.org 0x1C00			
0x1C00	RESET:	ldi r16,high(RAMEND);	Main program start
0x1C01		out SPH,r16	; Set Stack Pointer to top of RAM
0x1C02		ldi r16,low(RAMEND)	
0x1C03		out SPL,r16	
0x1C04		sei	; Enable interrupts
0x1C05		<instr> xxx	

When the BOOTRST Fuse is programmed, the Boot section size set to 2 Kbytes and the IVSEL bit in the MCUCR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

Address	Labels	Code	Comments
;			
.org 0x1C00			
0x1C00		jmp RESET	; Reset handler
0x1C02		jmp EXT_INT0	; IRQ0 Handler
0x1C04		jmp PCINT0	; PCINT0 Handler
...	...	...	;
0x1C2C		jmp SPM_RDY	; Store Program Memory Ready Handler

```

;
0x1C2E  RESET: ldi    r16,high(RAMEND); Main program start
0x1C2F          out    SPH,r16          ; Set Stack Pointer to top of RAM
0x1C30          ldi    r16,low(RAMEND)
0x1C31          out    SPL,r16
0x1C32          sei                      ; Enable interrupts
0x1C33          <instr>  xxx

```

## 11.2 Moving Interrupts Between Application and Boot Space

The General Interrupt Control Register controls the placement of the Interrupt Vector table, see ["MCUCR – MCU Control Register" on page 60](#).

To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

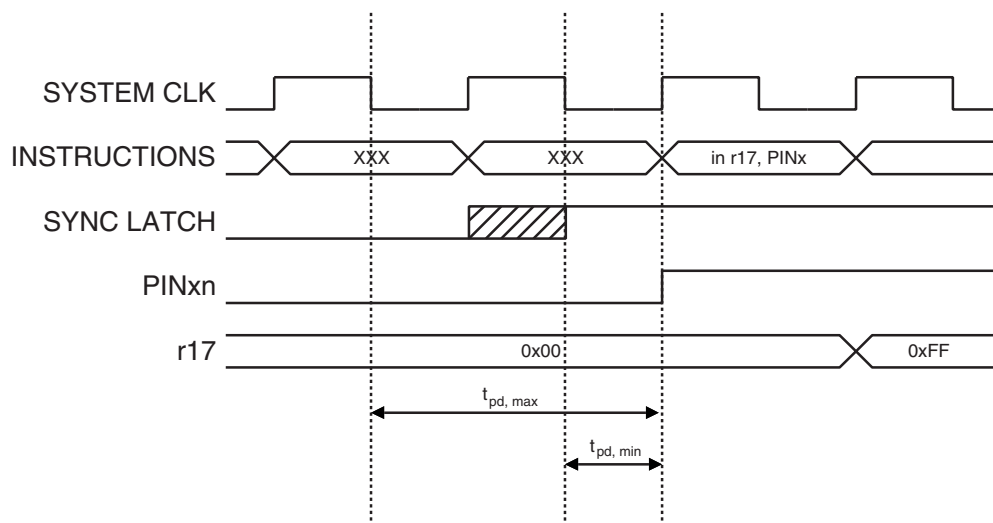
- a. Write the Interrupt Vector Change Enable (IVCE) bit to one.
- b. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the Status Register is unaffected by the automatic disabling.

**Note:** If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programmed, interrupts are disabled while executing from the Boot Loader section. Refer to the section ["Boot Loader Support – Read-While-Write Self-Programming" on page 280](#) for details on Boot Lock bits.

The following example shows how interrupts are moved.

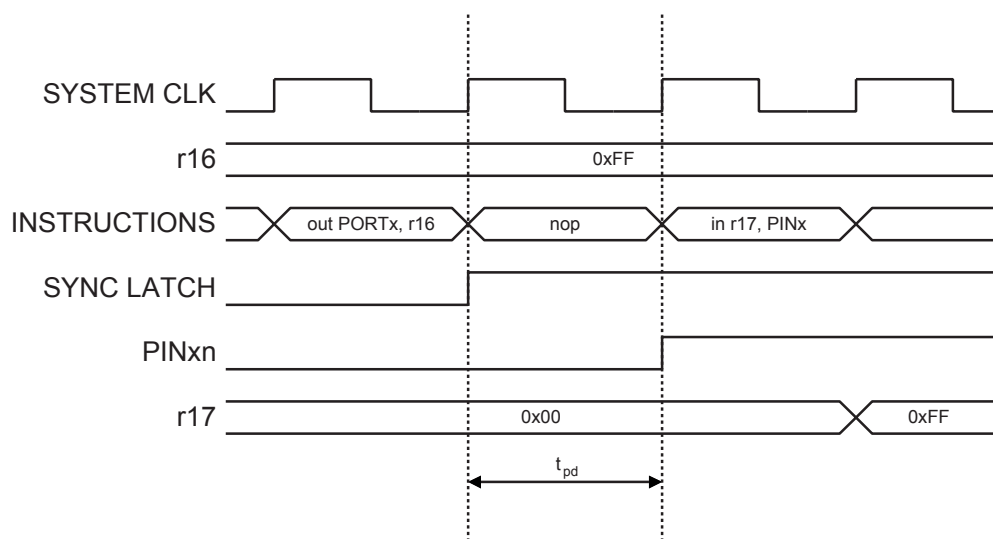
**Figure 13-3.** Synchronization when Reading an Externally Applied Pin value



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  $t_{pd, max}$  and  $t_{pd, min}$ , a single signal transition on the pin will be delayed between  $\frac{1}{2}$  and  $1\frac{1}{2}$  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 13-4](#). The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay  $t_{pd}$  through the synchronizer is 1 system clock period.

**Figure 13-4.** 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

**Table 13-2.** Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
PTOE	Port Toggle Override Enable	If PTOE is set, the PORTxn Register bit is inverted.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU state (Normal mode, sleep mode).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, sleep mode).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/Output	This is the Analog Input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.



Signal description (internal signals):

<b>count</b>	Increment or decrement TCNT0 by 1.
<b>direction</b>	Select between increment and decrement.
<b>clear</b>	Clear TCNT0 (set all bits to zero).
<b>clk<sub>Tn</sub></b>	Timer/Counter clock, referred to as clk <sub>T0</sub> in the following.
<b>top</b>	Signalize that TCNT0 has reached maximum value.
<b>bottom</b>	Signalize that TCNT0 has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk<sub>T0</sub>). clk<sub>T0</sub> can be generated from an external or internal clock source, selected by the Clock Select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk<sub>T0</sub> 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 WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0A). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare output OC0A. For more details about advanced counting sequences and waveform generation, see ["Modes of Operation" on page 96](#).

The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM01:0 bits. TOV0 can be used for generating a CPU interrupt.

## 14.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Register (OCR0A). Whenever TCNT0 equals OCR0A, the comparator signals a match. A match will set the Output Compare Flag (OCF0A) at the next timer clock cycle. If enabled (OCIE0A = 1 and Global Interrupt Flag in SREG is set), the Output Compare Flag generates an Output Compare interrupt. The OCF0A Flag is automatically cleared when the interrupt is executed. Alternatively, the OCF0A Flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the WGM01:0 bits and Compare Output mode (COM0A1:0) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (See ["Modes of Operation" on page 96](#)).

[Figure 14-3 on page 94](#) shows a block diagram of the Output Compare unit.

**Table 14-6.** Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$\text{clk}_{I/O}/(\text{No prescaling})$
0	1	0	$\text{clk}_{I/O}/8$ (From prescaler)
0	1	1	$\text{clk}_{I/O}/64$ (From prescaler)
1	0	0	$\text{clk}_{I/O}/256$ (From prescaler)
1	0	1	$\text{clk}_{I/O}/1024$ (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

## 14.9.2 TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a compare match between TCNT0 and the OCR0A Register.

## 14.9.3 OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x27 (0x47)	OCR0A[7:0]								OCR0A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0A pin.

## 14.9.4 TIMSK0 – Timer/Counter 0 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	–	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	

tion mode (WGM13:0) bits must be set before the TOP value can be written to the ICR1 Register. When writing the ICR1 Register the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

For more information on how to access the 16-bit registers refer to ["Accessing 16-bit Registers" on page 109](#).

## 15.6.1 Input Capture Trigger Source

The main trigger source for the Input Capture unit is the *Input Capture pin* (ICP1). Timer/Counter1 can alternatively use the Analog Comparator output as trigger source for the Input Capture unit. The Analog Comparator is selected as trigger source by setting the *Analog Comparator Input Capture* (ACIC) bit in the *Analog Comparator Control and Status Register* (ACSR). Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

Both the *Input Capture pin* (ICP1) and the *Analog Comparator output* (ACO) inputs are sampled using the same technique as for the T1 pin ([Figure 16-1 on page 135](#)). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in a Waveform Generation mode that uses ICR1 to define TOP.

An Input Capture can be triggered by software by controlling the port of the ICP1 pin.

## 15.6.2 Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for changing the output that in turn is used by the edge detector.

The noise canceler is enabled by setting the *Input Capture Noise Canceler* (ICNC1) bit in *Timer/Counter Control Register B* (TCCR1B). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICR1 Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

## 15.6.3 Using the Input Capture Unit

The main challenge when using the Input Capture unit is to assign enough processor capacity for handling the incoming events. The time between two events is critical. If the processor has not read the captured value in the ICR1 Register before the next event occurs, the ICR1 will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the Input Capture interrupt, the ICR1 Register should be read as early in the interrupt handler routine as possible. Even though the Input Capture interrupt has relatively high priority, the maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

Using the Input Capture unit in any mode of operation when the TOP value (resolution) is actively changed during operation, is not recommended.

Measurement of an external signal's duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR1 Register has been read. After a change of the edge, the Input Capture Flag (ICF1) must be

## 18.3 $\overline{SS}$ Pin Functionality

### 18.3.1 Slave Mode

When the SPI is configured as a Slave, the Slave Select ( $\overline{SS}$ ) pin is always input. When  $\overline{SS}$  is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When  $\overline{SS}$  is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the  $\overline{SS}$  pin is driven high.

The  $\overline{SS}$  pin is useful for packet/byte synchronization to keep the slave bit counter synchronous with the master clock generator. When the  $\overline{SS}$  pin is driven high, the SPI slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

### 18.3.2 Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the  $\overline{SS}$  pin.

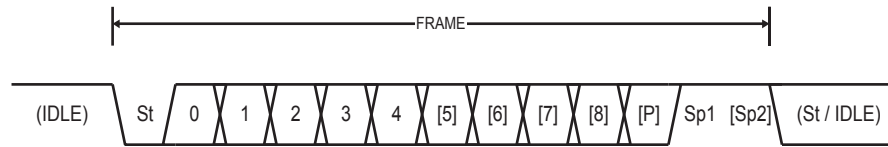
If  $\overline{SS}$  is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the  $\overline{SS}$  pin of the SPI Slave.

If  $\overline{SS}$  is configured as an input, it must be held high to ensure Master SPI operation. If the  $\overline{SS}$  pin is driven low by peripheral circuitry when the SPI is configured as a Master with the  $\overline{SS}$  pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The SPIF Flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that  $\overline{SS}$  is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI Master mode.

**Figure 19-4. Frame Formats**



<b>St</b>	Start bit, always low.
<b>(n)</b>	Data bits (0 to 8).
<b>P</b>	Parity bit. Can be odd or even.
<b>Sp</b>	Stop bit, always high.
<b>IDLE</b>	No transfers on the communication line (RxD or TxD). An IDLE line must be high.

The frame format used by the USART is set by the UCSZn2:0, UPM1n:0 and USBSn bits in UCSRnB and UCSRnC. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

The USART Character SiZe (UCSZn2:0) bits select the number of data bits in the frame. The USART Parity mode (UPM1n:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the USART Stop Bit Select (USBSn) bit. The Receiver ignores the second stop bit. An FEn (Frame Error FEn) will therefore only be detected in the cases where the first stop bit is zero.

## 19.4.1 Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows:

$$P_{even} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

$$P_{odd} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

<b>P<sub>even</sub></b>	Parity bit using even parity.
<b>P<sub>odd</sub></b>	Parity bit using odd parity.
<b>d<sub>n</sub></b>	Data bit n of the character.

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 Registers.

## Assembly Code Example<sup>(1)</sup>

```

USART_Init:
    ; Set baud rate
    sts  UBRRH0, r17
    sts  UBRRLO, r16
    ; Enable receiver and transmitter
    ldi  r16, (1<<RXEN0) | (1<<TXEN0)
    sts  UCSR0B, r16
    ; Set frame format: 8data, 2stop bit
    ldi  r16, (1<<USBS0) | (3<<UCSZ00)
    sts  UCSR0C, r16
    ret

```

## C Code Example<sup>(1)</sup>

```

#define FOSC 1843200// Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
    ...
    USART_Init ( MYUBRR );
    ...
}

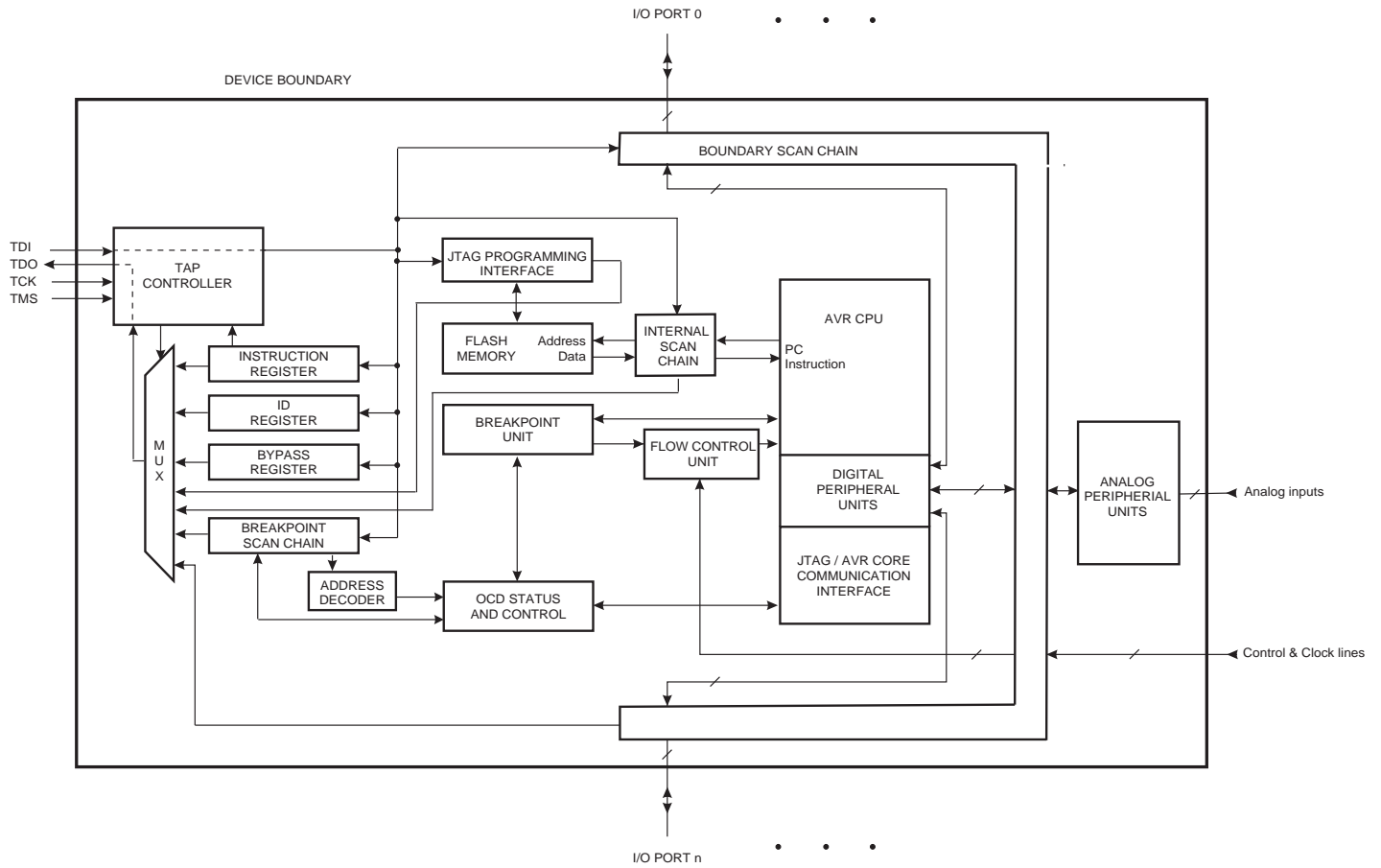
void USART_Init( unsigned int ubrr)
{
    /* Set baud rate */
    UBRRH0 = (unsigned char) (ubrr>>8);
    UBRRLO = (unsigned char)ubrr;
    /* Enable receiver and transmitter */
    UCSR0B = (1<<RXEN0) | (1<<TXEN0);
    /* Set frame format: 8data, 2stop bit */
    UCSRnC = (1<<USBS0) | (3<<UCSZ00);
}

```

Note: 1. See "About Code Examples" on page 10.

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

**Figure 24-1. Block Diagram**



- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the Shift Register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine.
- At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift Data Register – Shift-DR state. While in this state, upload the selected Data Register (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI input at the rising edge of TCK. In order to remain in the Shift-DR state, the TMS input must be held low during input of all bits except the MSB. The MSB of the data is shifted in when this state is left by setting TMS high. While the Data Register is shifted in from the TDI pin, the parallel inputs to the Data Register captured in the Capture-DR state is shifted out on the TDO pin.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine.

As shown in the state diagram, the Run-Test/Idle state need not be entered between selecting JTAG instruction and using Data Registers, and some JTAG instructions may select certain functions to be performed in the Run-Test/Idle, making it unsuitable as an Idle state.

Note: Independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS high for five TCK clock periods.

For detailed information on the JTAG specification, refer to the literature listed in ["Bibliography" on page 258](#).

## 24.4 Using the Boundary-scan Chain

A complete description of the Boundary-scan capabilities are given in the section ["IEEE 1149.1 \(JTAG\) Boundary-scan" on page 259](#).

## 24.5 Using the On-chip Debug System

As shown in [Figure 24-1 on page 254](#), the hardware support for On-chip Debugging consists mainly of:

- A scan chain on the interface between the internal AVR CPU and the internal peripheral units.
- Break Point unit.
- Communication interface between the CPU and JTAG system.

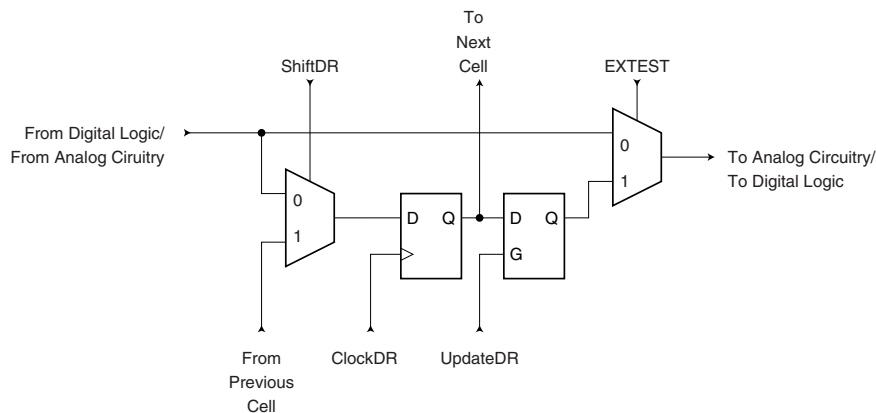
All read or modify/write operations needed for implementing the Debugger are done by applying AVR instructions via the internal AVR CPU Scan Chain. The CPU sends the result to an I/O memory mapped location which is part of the communication interface between the CPU and the JTAG system.

The Break Point Unit implements Break on Change of Program Flow, Single Step Break, two Program Memory Break Points, and two combined Break Points. Together, the four Break Points can be configured as either:

- 4 single Program Memory Break Points.
- 3 Single Program Memory Break Point + 1 single Data Memory Break Point.
- 2 single Program Memory Break Points + 2 single Data Memory Break Points.
- 2 single Program Memory Break Points + 1 Program Memory Break Point with mask ("range Break Point").
- 2 single Program Memory Break Points + 1 Data Memory Break Point with mask ("range Break Point").



**Figure 25-8.** General Boundary-scan cell Used for Signals for Comparator and ADC



**Table 25-2.** Boundary-scan Signals for the Analog Comparator

Signal Name	Direction as Seen from the Comparator	Description	Recommended Input when Not in Use	Output Values when Recommended Inputs are Used
AC_IDLE	input	Turns off Analog Comparator when true	1	Depends upon $\mu$ C code being executed
ACO	output	Analog Comparator Output	Will become input to $\mu$ C code being executed	0
ACME	input	Uses output signal from ADC mux when true	0	Depends upon $\mu$ C code being executed
ACBG	input	Bandgap Reference enable	0	Depends upon $\mu$ C code being executed

## 26.5 Boot Loader Lock Bits

If no Boot Loader capability is needed, the entire Flash is available for application code. The Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can select:

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

See [Table 26-2](#) and [Table 26-3](#) for further details. The Boot Lock bits and general Lock bits can be set in software and in Serial or Parallel Programming mode, but they can be cleared by a Chip Erase command only. The general Write Lock (Lock Bit mode 2) does not control the programming of the Flash memory by SPM instruction. Similarly, the general Read/Write Lock (Lock Bit mode 1) does not control reading nor writing by LPM/SPM, if it is attempted.

**Table 26-2.** Boot Lock Bit0 Protection Modes (Application Section)<sup>(1)</sup>

BLB0 Mode	BLB02	BLB01	Protection
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.

Note: 1. "1" means unprogrammed, "0" means programmed

**Table 26-3.** Boot Lock Bit1 Protection Modes (Boot Loader Section)<sup>(1)</sup>

BLB1 Mode	BLB12	BLB11	Protection
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Note: 1. "1" means unprogrammed, "0" means programmed

**Table 26-7.** Read-While-Write Limit<sup>(1)</sup>

Section	Pages	Address
Read-While-Write section (RWW)	112	0x0000 - 0x1BFF
No Read-While-Write section (NRWW)	16	0x1C00 - 0x1FFF

Note: 1. For details about these two section, see ["NRWW – No Read-While-Write Section" on page 281](#) and ["RWW – Read-While-Write Section" on page 281](#).

**Table 26-8.** Explanation of different variables used in [Figure 26-3 on page 286](#) and the mapping to the Z-pointer<sup>(1)</sup>

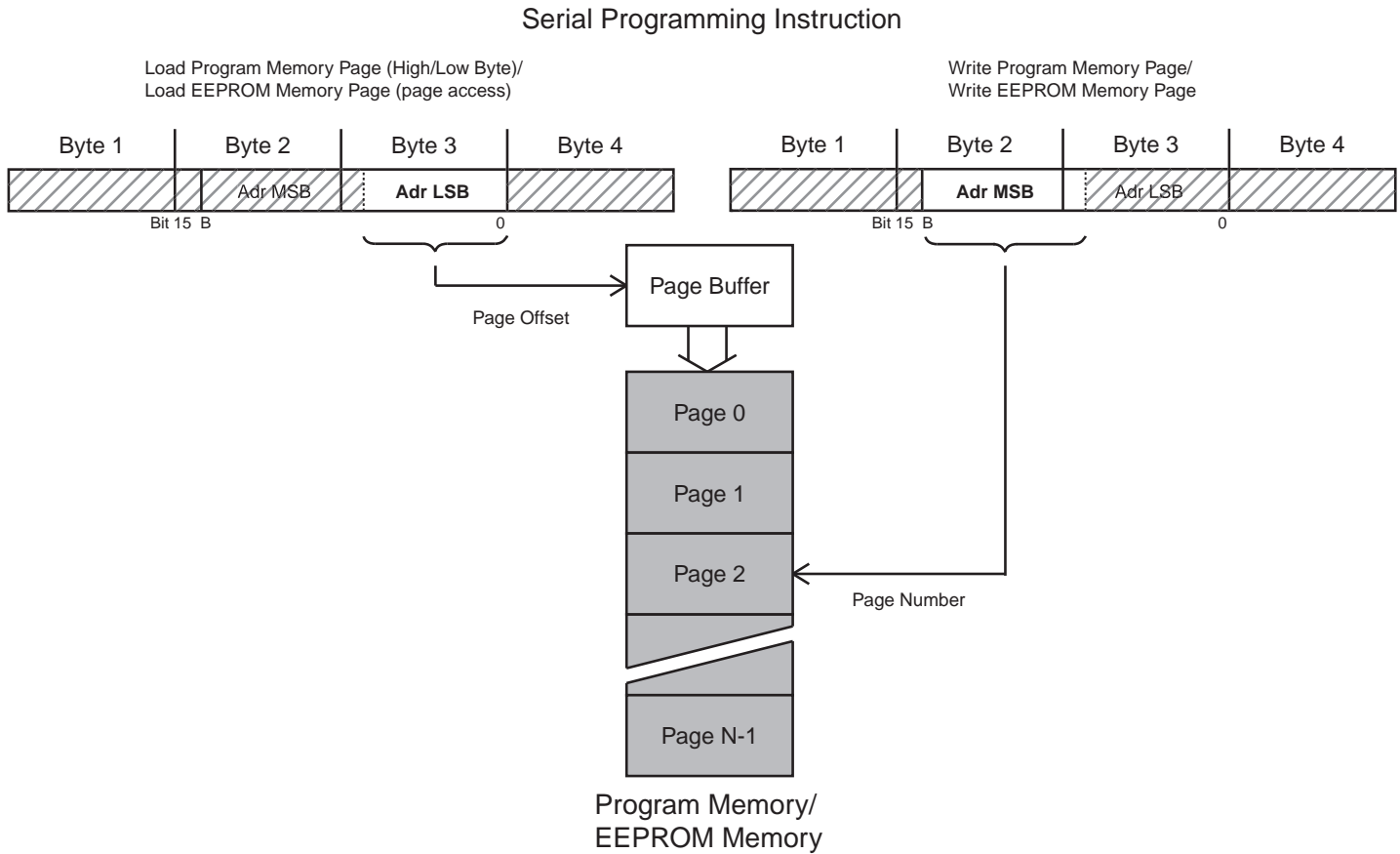
Variable		Corresponding Z-value	Description
PCMSB	12		Most significant bit in the Program Counter. (The Program Counter is 13 bits PC[12:0])
PAGEMSB	5		Most significant bit which is used to address the words within one page (64 words in a page requires six bits PC [5:0]).
ZPCMSB		Z13	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z6	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[12:6]	Z13:Z7	Program Counter page address: Page select, for Page Erase and Page Write
PCWORD	PC[5:0]	Z6:Z1	Program Counter word address: Word select, for filling temporary buffer (must be zero during Page Write operation)

Note: 1. Z15:Z14: always ignored  
 Z0: should be zero for all SPM commands, byte select for the LPM instruction.  
 See ["Addressing the Flash During Self-Programming" on page 286](#) for details about the use of Z-pointer during Self-Programming.

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 27-12](#).

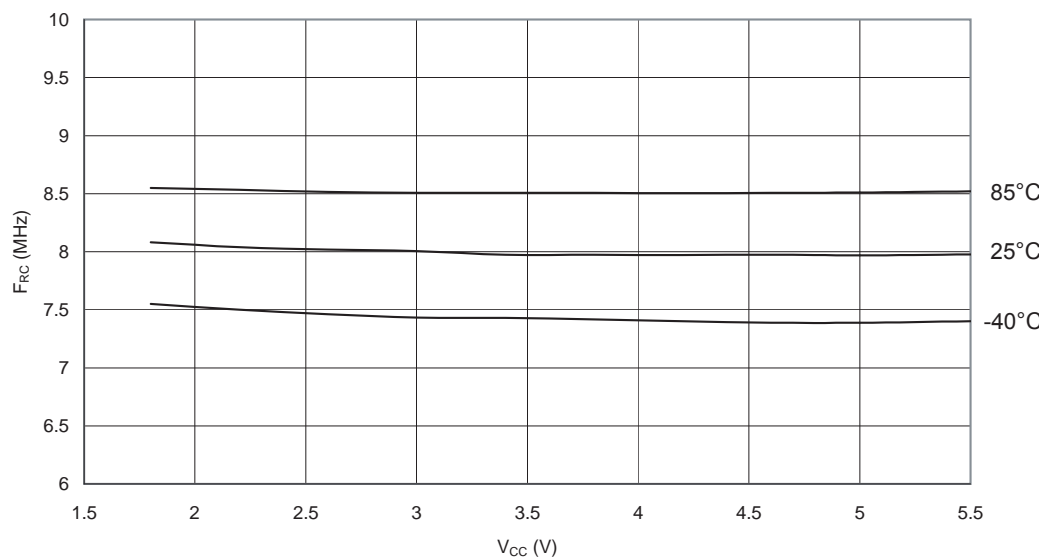
**Figure 27-12.** Serial Programming Instruction example



## 27.8.4 SPI Serial Programming Characteristics

For characteristics of the SPI module, see ["SPI Timing Characteristics" on page 334](#).

**Figure 29-56.** Calibrated 8 MHz RC Oscillator Frequency vs.  $V_{CC}$



**Figure 29-57.** Calibrated 8 MHz RC Oscillator Frequency vs. Oscal Value

