Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	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-TQFP
Supplier Device Package	64-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega169pv-8au

7.5 General Purpose I/O Registers

The ATmega169P contains three General Purpose I/O Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags. General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

7.5.1 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	MSB							LSB	GPIOR2
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	

7.5.2 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
0x2A (0x4A)	MSB							LSB	GPIOR1
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	

7.5.3 GPIOR0 – General Purpose I/O Register 0

Bit	7	6	5	4	3	2	1	0	
0x1E (0x3E)	MSB							LSB	GPIOR0
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	

7.6 EEPROM Register Description

7.6.1 EEARH and EEARL – EEPROM Address Register

Bit	15	14	13	12	11	10	9	8	
0x22 (0x42)	–	–	–	–	–	–	–	EEAR8	EEARH
0x21 (0x41)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	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	X	
	X	X	X	X	X	X	X	X	

- **Bits 15:9 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bits 8:0 – EEAR8:0: EEPROM Address**

The EEPROM Address Registers – EEARH and EEARL specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

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 clk_{CPU} and clk_{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 $clk_{I/O}$, clk_{CPU}, and clk_{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 form 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.

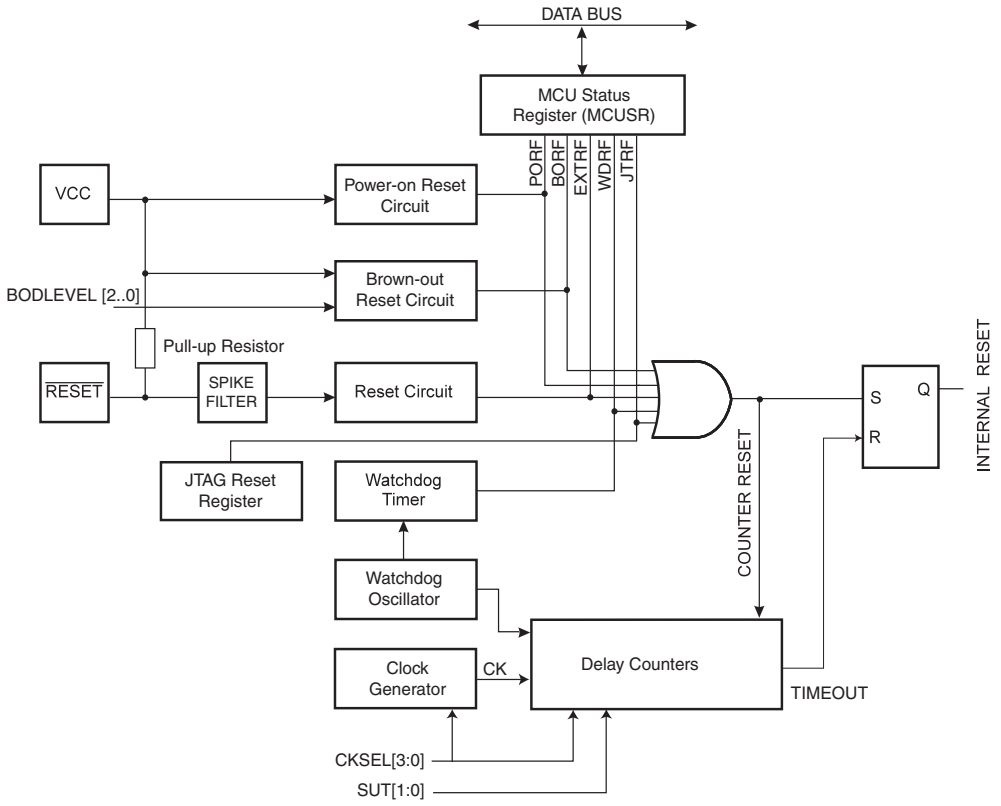
9.4 Power-down Mode

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.

Figure 10-1. Reset Logic

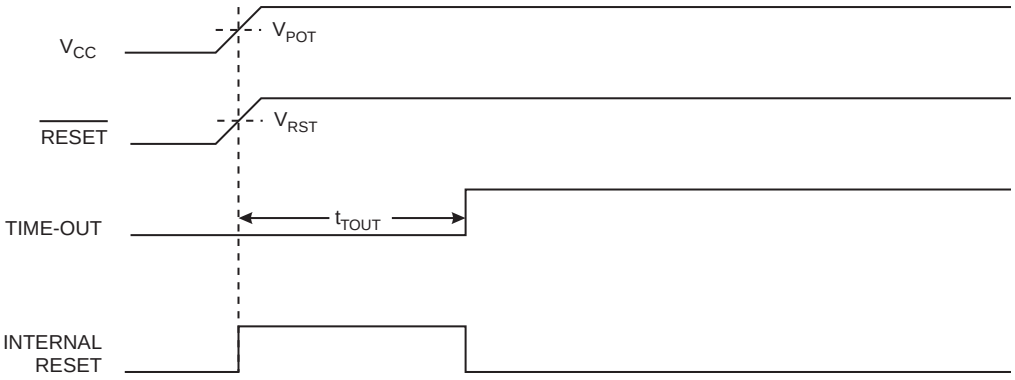


10.2.1 Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in "System and Reset Characteristics" on page 333. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the start-up Reset, as well as to detect a failure in supply voltage.

A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after V_{CC} rise. The RESET signal is activated again, without any delay, when V_{CC} decreases below the detection level.

Figure 10-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}



low. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB0. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB0 bit.

PCINT8, Pin Change Interrupt Source 8: The PB0 pin can serve as an external interrupt source.

Table 13-7 and Table 13-8 on page 77 relate the alternate functions of Port B to the overriding signals shown in Figure 13-5 on page 71. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

Table 13-7. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/OC2A/ PCINT15	PB6/OC1B/ PCINT14	PB5/OC1A/ PCINT13	PB4/OC0A/ PCINT12
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC2A ENABLE	OC1B ENABLE	OC1A ENABLE	OC0A ENABLE
PVOV	OC2A	OC1B	OC1A	OC0A
PTOE	–	–	–	–
DIEOE	PCINT15 • PCIE1	PCINT14 • PCIE1	PCINT13 • PCIE1	PCINT12 • PCIE1
DIEOV	1	1	1	1
DI	PCINT15 INPUT	PCINT14 INPUT	PCINT13 INPUT	PCINT12 INPUT
AIO	–	–	–	–

- **SEG4 – Port G, Bit 2**

SEG4, LCD front plane 4

- **SEG13 – Port G, Bit 1**

SEG13, Segment driver 13

- **SEG14 – Port G, Bit 0**

SEG14, LCD front plane 14

Table 13-21 on page 85 and Table 13-22 relates the alternate functions of Port G to the overriding signals shown in Figure 13-5 on page 71.

Table 13-22. Overriding Signals for Alternate Functions in PG4

Signal Name				PG4/T0/SEG23
PUOE				LCDEN • (LCDPM>5)
PUOV				0
DDOE				LCDEN • (LCDPM>5)
DDOV				1
PVOE				0
PVOV				0
PTOE	–	–	–	–
DIEOE				LCDEN • (LCDPM>5)
DIEOV				0
DI				T0 INPUT
AIO				SEG23

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). See ”Output Compare Unit” on page 93. for details. The compare match event will also set the Compare Flag (OCF0A) which can be used to generate an Output Compare interrupt request.

14.2.2 Definitions

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

The definitions in Table 14-1 are also used extensively throughout the document.

Table 14-1. Timer/Counter Definitions

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

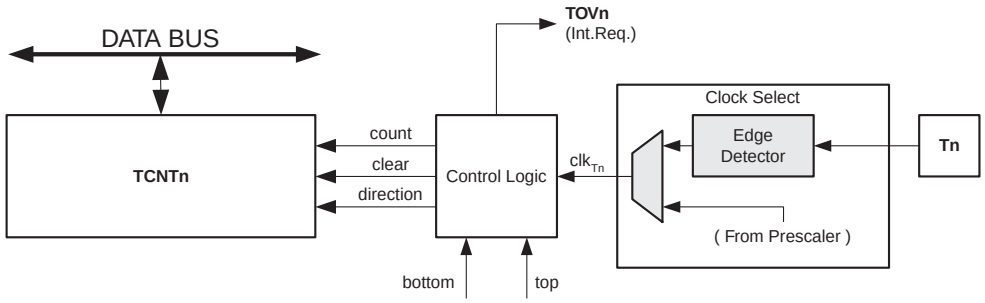
14.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS02:0) bits located in the Timer/Counter Control Register (TCCR0A). For details on clock sources and prescaler, see ”Timer/Counter0 and Timer/Counter1 Prescalers” on page 135.

14.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 14-2 shows a block diagram of the counter and its surroundings.

Figure 14-2. Counter Unit Block Diagram



When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM01:0 bit setting. Table 14-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to a normal or CTC mode (non-PWM).

Table 14-3. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on compare match
1	0	Clear OC0A on compare match
1	1	Set OC0A on compare match

Table 14-4 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 14-4. Compare Output Mode, Fast PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Reserved
1	0	Clear OC0A on compare match, set OC0A at BOTTOM (non-inverting mode)
1	1	Set OC0A on compare match, clear OC0A at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See ”Fast PWM Mode” on page 97 for more details.

Table 14-5 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to phase correct PWM mode.

Table 14-5. Compare Output Mode, Phase Correct PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Reserved
1	0	Clear OC0A on compare match when up-counting. Set OC0A on compare match when down counting.
1	1	Set OC0A on compare match when up-counting. Clear OC0A on compare match when down counting.

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See ”Phase Correct PWM Mode” on page 99 for more details.

- **Bit 2:0 – CS02:0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter.

15.3 Accessing 16-bit Registers

The TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

Not all 16-bit accesses uses the temporary register for the high byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

To do a 16-bit write, the high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

The following code examples show how to access the 16-bit Timer Registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR1A/B and ICR1 Registers. Note that when using “C”, the compiler handles the 16-bit access.

Assembly Code Examples[1]

```
...
; Set TCNT1 to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNT1H,r17
out TCNT1L,r16
; Read TCNT1 into r17:r16
in  r16,TCNT1L
in  r17,TCNT1H
...
```

C Code Examples[1]

```
unsigned int i;
...
/* Set TCNT1 to 0x01FF */
TCNT1 = 0x1FF;
/* Read TCNT1 into i */
i = TCNT1;
...
```

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

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit Timer Registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both

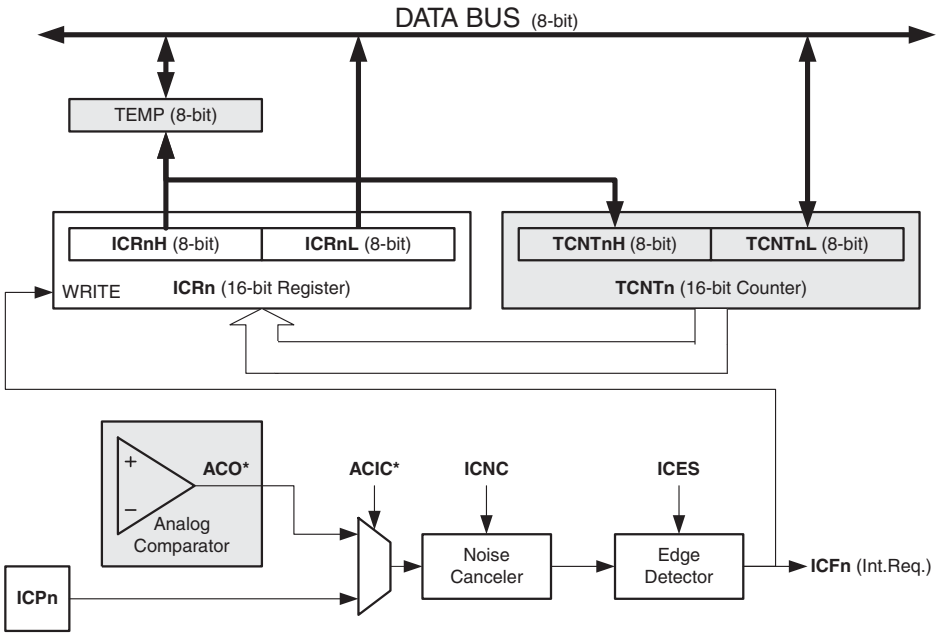
The Timer/Counter Overflow Flag (TOV1) is set according to the mode of operation selected by the WGM13:0 bits. TOV1 can be used for generating a CPU interrupt.

15.6 Input Capture Unit

The Timer/Counter incorporates an Input Capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP1 pin or alternatively, via the analog-comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

The Input Capture unit is illustrated by the block diagram shown in Figure 15-3. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded. The small “n” in register and bit names indicates the Timer/Counter number.

Figure 15-3. Input Capture Unit Block Diagram



When a change of the logic level (an event) occurs on the *Input Capture pin* (ICP1), alternatively on the *Analog Comparator output* (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNT1) is written to the *Input Capture Register* (ICR1). The *Input Capture Flag* (ICF1) is set at the same system clock as the TCNT1 value is copied into ICR1 Register. If enabled (ICIE1 = 1), the Input Capture Flag generates an Input Capture interrupt. The ICF1 Flag is automatically cleared when the interrupt is executed. Alternatively the ICF1 Flag can be cleared by software by writing a logical one to its I/O bit location.

Reading the 16-bit value in the *Input Capture Register* (ICR1) is done by first reading the low byte (ICR1L) and then the high byte (ICR1H). When the low byte is read the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR1H I/O location it will access the TEMP Register.

The ICR1 Register can only be written when using a Waveform Generation mode that utilizes the ICR1 Register for defining the counter’s TOP value. In these cases the *Waveform Genera-*

17. 8-bit Timer/Counter2 with PWM and Asynchronous Operation

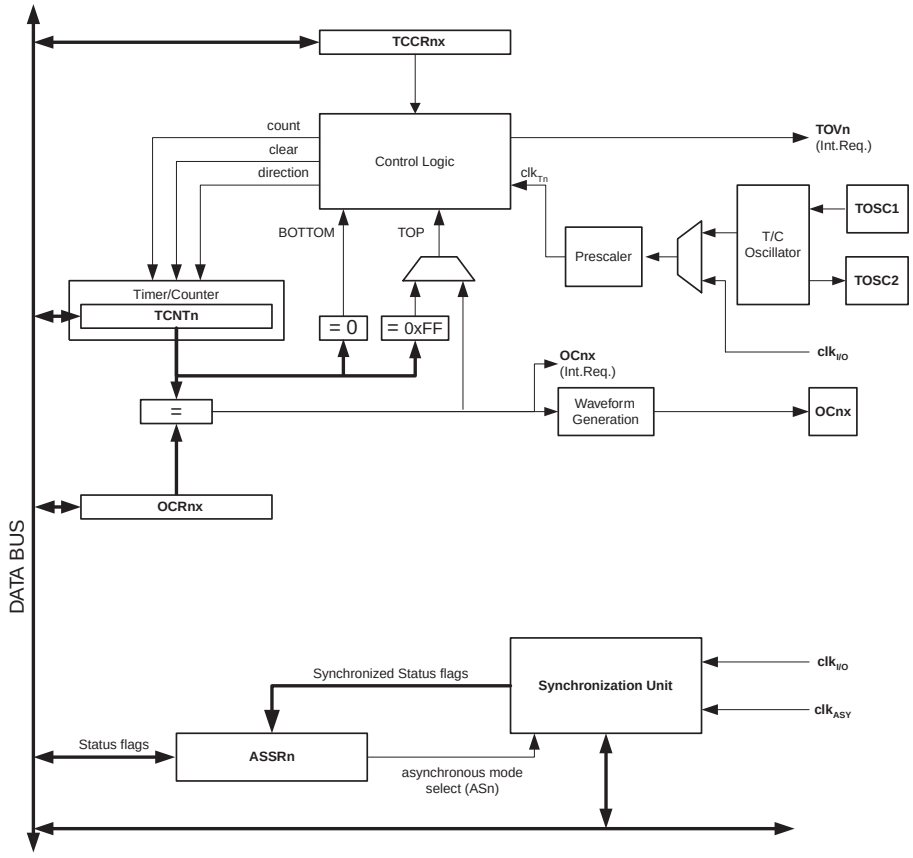
Timer/Counter2 is a general purpose, single compare unit, 8-bit Timer/Counter module. The main features are:

- **Single Compare Unit Counter**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Frequency Generator**
- **10-bit Clock Prescaler**
- **Overflow and Compare Match Interrupt Sources (TOV2 and OCF2A)**
- **Allows Clocking from External 32 kHz Watch Crystal Independent of the I/O Clock**

17.1 Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 17-1. For the actual placement of I/O pins, refer to Figure 1-1 on page 2. 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 17.10 ”8-bit Timer/Counter Register Description” on page 153.

Figure 17-1. 8-bit Timer/Counter Block Diagram



17.1.1 Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2A) are 8-bit registers. Interrupt request (shorten as Int.Req.) signals are all visible in the Timer Interrupt Flag Register

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

Assembly Code Example[1]

```
USART_Init:
  ; Set baud rate
  sts  UBRRH0, r17
  sts  UBRRL0, 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[1]

```
#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);
  UBRRL0 = (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.

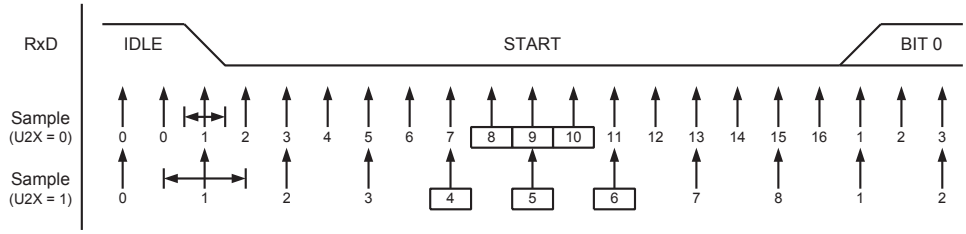
19.8 Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxD pin. The data recovery logic samples and low pass filters each incoming bit, thereby improving the noise immunity of the Receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

19.8.1 Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. Figure 19-5 illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16 times the baud rate for Normal mode, and eight times the baud rate for Double Speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the Double Speed mode (U2Xn = 1) of operation. Samples denoted zero are samples done when the RxD line is idle (that is, no communication activity).

Figure 19-5. Start Bit Sampling



When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and 6 for Double Speed mode (indicated with sample numbers inside boxes on the figure), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the Receiver starts looking for the next high to low-transition. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

19.8.2 Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in Normal mode and eight states for each bit in Double Speed mode. Figure 19-6 on page 186 shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

Table 22-4. Input Channel Selections

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input
00000	ADC0	N/A	
00001	ADC1		
00010	ADC2		
00011	ADC3		
00100	ADC4		
00101	ADC5		
00110	ADC6		
00111	ADC7		
01000	N/A		
01001			
01010			
01011			
01100			
01101			
01110			
01111			
10000		ADC0	ADC1
10001		ADC1	ADC1
10010		ADC2	ADC1
10011		ADC3	ADC1
10100		ADC4	ADC1
10101		ADC5	ADC1
10110		ADC6	ADC1
10111		ADC7	ADC1
11000		ADC0	ADC2
11001		ADC1	ADC2
11010		ADC2	ADC2
11011		ADC3	ADC2
11100		ADC4	ADC2
11101		ADC5	ADC2
11110	1.1V (V_{BG})	N/A	
11111	0V (GND)		

- **Bit 0 – LCDBL: LCD Blanking**

When this bit is written to one, the display will be blanked after completion of a frame. All segment and common pins will be driven to ground.

23.5.2 LCDCRB – LCD Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0xE5)	LCDCS	LCD2B	LCDMUX1	LCDMUX0	–	LCDPM2	LCDPM1	LCDPM0	LCDCRB
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – LCDCS: LCD Clock Select**

When this bit is written to zero, the system clock is used. When this bit is written to one, the external asynchronous clock source is used. The asynchronous clock source is either Timer/Counter Oscillator or external clock, depending on EXCLK in ASSR. See ”Asynchronous operation of the Timer/Counter” on page 150 for further details.

- **Bit 6 – LCD2B: LCD 1/2 Bias Select**

When this bit is written to zero, 1/3 bias is used. When this bit is written to one, ½ bias is used. Refer to the LCD Manufacture for recommended bias selection.

- **Bit 5:4 – LCDMUX1:0: LCD Mux Select**

The LCDMUX1:0 bits determine the duty cycle. Common pins that are not used are ordinary port pins. The different duty selections are shown in Table 23-2.

Table 23-2. LCD Duty Select

LCDMUX1	LCDMUX0	Duty	Bias	COM Pin	I/O Port Pin
0	0	Static	Static	COM0	COM1:3
0	1	1/2	1/2 or 1/3(1)	COM0:1	COM2:3
1	0	1/3	1/2 or 1/3(1)	COM0:2	COM3
1	1	1/4	1/2 or 1/3(1)	COM0:3	None

Note: 1. 1/2 bias when LCD2B is written to one and 1/3 otherwise.

- **Bit 3 – Res: Reserved Bit**

This bit is reserved and will always read as zero.

- **Bits 2:0 – LCDPM2:0: LCD Port Mask**

The LCDPM2:0 bits determine the number of port pins to be used as segment drivers. The different selections are shown in Table 23-3. Unused pins can be used as ordinary port pins.

Table 23-3. LCD Port Mask

LCDPM2	LCDPM1	LCDPM0	I/O Port in Use as Segment Driver	Maximum Number of Segments
0	0	0	SEG0:12	13
0	0	1	SEG0:14	15
0	1	0	SEG0:16	17
0	1	1	SEG0:18	19

- 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”).

26.7 Addressing the Flash During Self-Programming

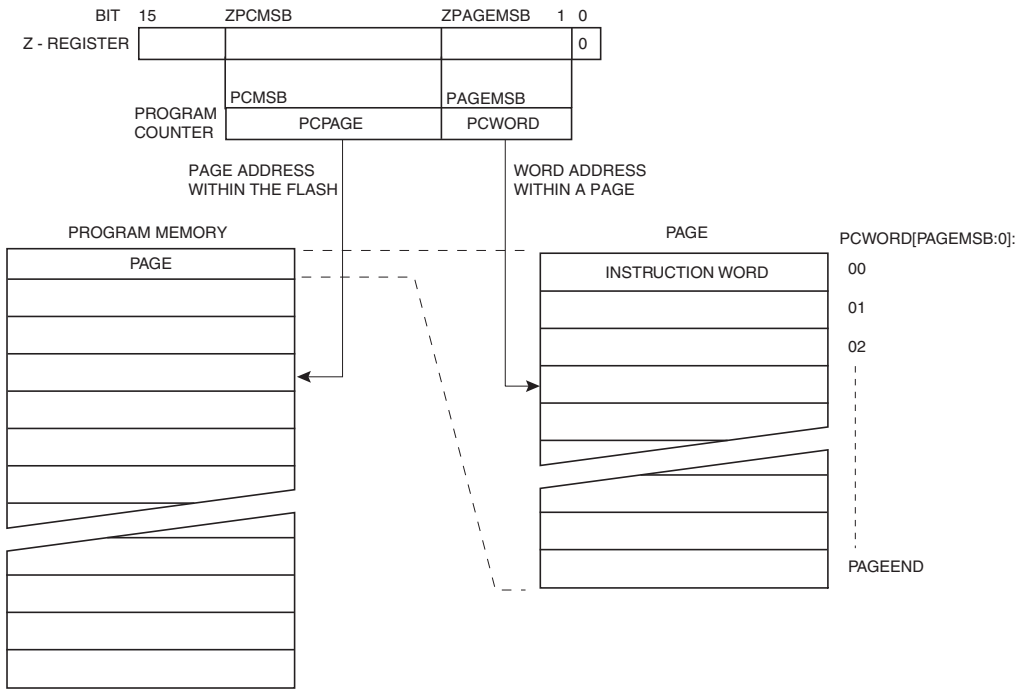
The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	Z15	Z14	Z13	Z12	Z11	Z10	Z9	Z8
ZL (R30)	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0
	7	6	5	4	3	2	1	0

Since the Flash is organized in pages (see Table 27-7 on page 299), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 26-3. Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the Boot Loader software addresses the same page in both the Page Erase and Page Write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

The only SPM operation that does not use the Z-pointer is Setting the Boot Loader Lock bits. The content of the Z-pointer is ignored and will have no effect on the operation. The LPM instruction does also use the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

Figure 26-3. Addressing the Flash During SPM(1)



Note: 1. The different variables used in Figure 26-3 are listed in Table 26-8 on page 293.
2. PCPAGE and PCWORD are listed in Table 27-7 on page 299.

Table 26-7. Read-While-Write Limit[1]

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

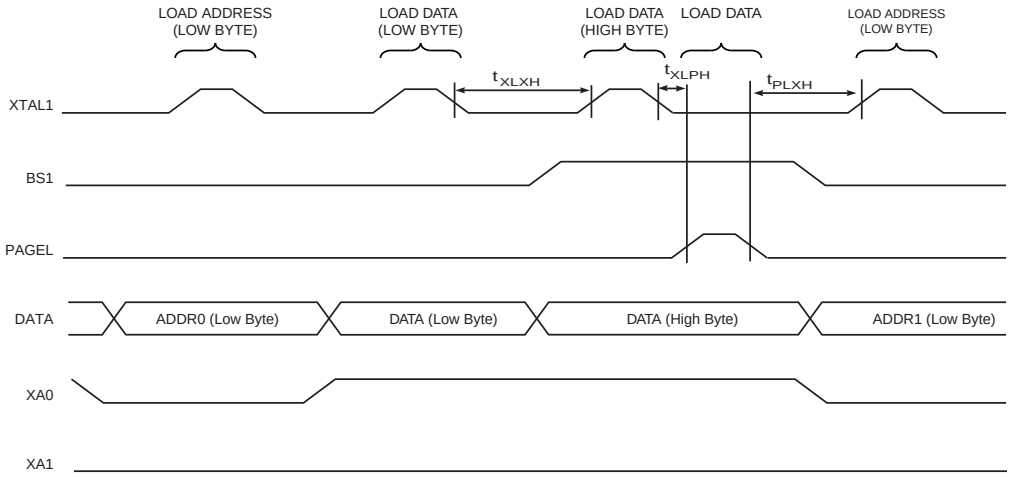
Note: 1. For details about these two section, see "NRWW – No Read-While-Write Section" on page 281 and "RWW – Read-While-Write Section" on page 281.

Table 26-8. Explanation of different variables used in Figure 26-3 on page 286 and the mapping to the Z-pointer[1]

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)

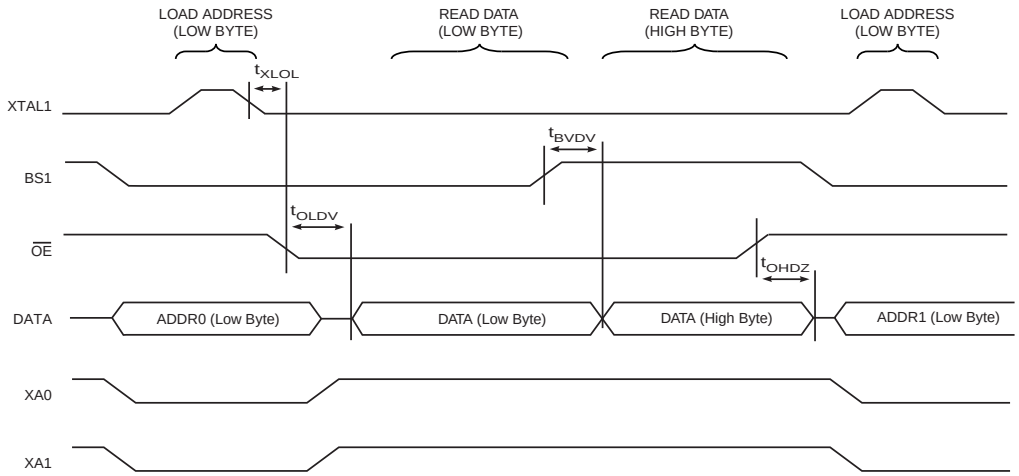
Note: 1. Z15:Z14: always ignored
Z0: should be zero for all SPM commands, byte select for the LPM instruction.
See "Addressing the Flash During Self-Programming" on page 286 for details about the use of Z-pointer during Self-Programming.

Figure 27-8. Parallel Programming Timing, Loading Sequence with Timing Requirements[1]



Note: 1. The timing requirements shown in Figure 27-7 on page 308 (that is, t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 27-9. Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements[1]



Note: 1. The timing requirements shown in Figure 27-7 on page 308 (that is, t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

Table 27-13. Parallel Programming Characteristics, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA

27.9.7 Data Registers

The Data Registers are selected by the JTAG instruction registers described in section ”Programming Specific JTAG Instructions” on page 316. The Data Registers relevant for programming operations are:

- Reset Register
- Programming Enable Register
- Programming Command Register
- Flash Data Byte Register

27.9.8 Reset Register

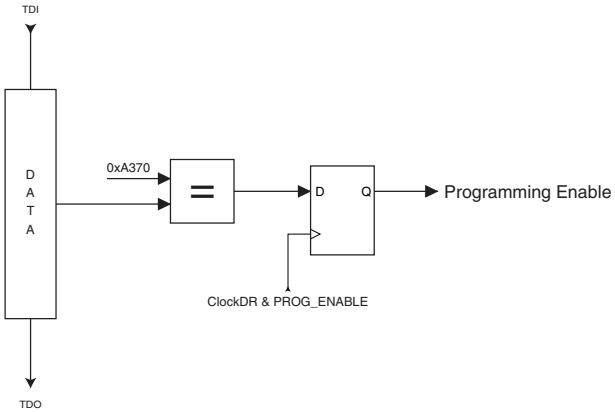
The Reset Register is a Test Data Register used to reset the part during programming. It is required to reset the part before entering Programming mode.

A high value in the Reset Register corresponds to pulling the external reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the Fuse settings for the clock options, the part will remain reset for a Reset Time-out period (refer to ”Clock Sources” on page 31) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in Figure 25-2 on page 261.

27.9.9 Programming Enable Register

The Programming Enable Register is a 16-bit register. The contents of this register is compared to the programming enable signature, binary code 0b1010_0011_0111_0000. When the contents of the register is equal to the programming enable signature, programming via the JTAG port is enabled. The register is reset to 0 on Power-on Reset, and should always be reset when leaving Programming mode.

Figure 27-14. Programming Enable Register



27.9.10 Programming Command Register

The Programming Command Register is a 15-bit register. This register is used to serially shift in programming commands, and to serially shift out the result of the previous command, if any. The JTAG Programming Instruction Set is shown in Table 27-17 on page 321. The state sequence when shifting in the programming commands is illustrated in Figure 27-16 on page 324.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1B (0x3B)	Reserved	–	–	–	–	–	–	–	–	
0x1A (0x3A)	Reserved	–	–	–	–	–	–	–	–	
0x19 (0x39)	Reserved	–	–	–	–	–	–	–	–	
0x18 (0x38)	Reserved	–	–	–	–	–	–	–	–	
0x17 (0x37)	TIFR2	–	–	–	–	–	–	OCF2A	TOV2	156
0x16 (0x36)	TIFR1	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	134
0x15 (0x35)	TIFR0	–	–	–	–	–	–	OCF0A	TOV0	105
0x14 (0x34)	PORTG	–	–	PORTG5	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0	90
0x13 (0x33)	DDRG	–	–	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	90
0x12 (0x32)	PING	–	–	PING5	PING4	PING3	PING2	PING1	PING0	90
0x11 (0x31)	PORTF	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0	90
0x10 (0x30)	DDRF	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0	90
0x0F (0x2F)	PINF	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0	90
0x0E (0x2E)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	89
0x0D (0x2D)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	89
0x0C (0x2C)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	90
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	89
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	89
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	89
0x08 (0x28)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	89
0x07 (0x27)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	89
0x06 (0x26)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	89
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	88
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	88
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	88
0x02 (0x22)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	88
0x01 (0x21)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	88
0x00 (0x20)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	88

Note:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega169P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.