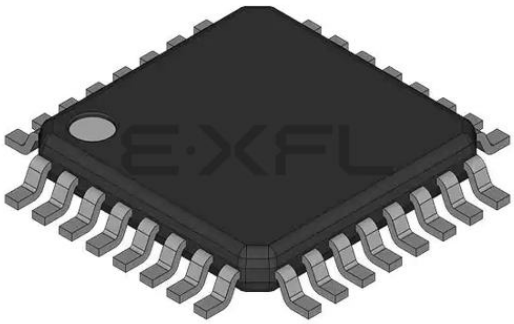


Welcome to E-XFL.COM

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	16MHz
Connectivity	I²C, SPI
Peripherals	Brown-out Detect/Reset, POR, WDT
Number of I/O	28
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	64 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	32-TQFP
Supplier Device Package	32-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/atmel/attiny88-15az

When this clock source is selected, start-up times are determined by the SUT fuses as shown in Table 6-8.

Table 6-8. Start-up Times for the External Clock Selection

SUT1..0	Power Conditions	Start-up Time from Power-down	Additional Delay from Reset (V_{CC} = 5.0V)
00	BOD enabled	6CK	14CK
01	Fast rising power	6CK	14CK + 4ms
10	Slowly rising power	6CK	14CK + 64ms
11	Reserved		

When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the MCU. A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. If changes of more than 2% is required, ensure that the MCU is kept in reset during the changes.

Note that the system clock prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation. Refer to Section 6.7 “System Clock Prescaler” on page 29 for details.

6.6 Clock Output Buffer

The device can output the system clock on the CLKO pin. To enable the output, the CKOUT fuse has to be programmed. This mode is suitable when the chip clock is used to drive other circuits on the system. The clock also will be output during reset, and the normal operation of I/O pin will be overridden when the fuse is programmed. Any clock source, including the internal oscillator, can be selected when the clock is output on CLKO. If the system clock prescaler is used, it is the divided system clock that is output.

6.7 System Clock Prescaler

The Atmel® ATtiny88 has a system clock prescaler, and the system clock can be divided by setting the Section 6.8.2 “CLKPR – Clock Prescale Register” on page 30. This feature can be used to decrease the system clock frequency and the power consumption when the requirement for processing power is low. This can be used with all clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, clk_{ADC}, clk_{CPU}, and clk_{FLASH} are divided by a factor as shown in Table 6-9 on page 31.

When switching between prescaler settings, the system clock prescaler ensures that no glitches occur in the clock system. It also ensures that no intermediate frequency is higher than neither the clock frequency corresponding to the previous setting, nor the clock frequency corresponding to the new setting. The ripple counter that implements the prescaler runs at the frequency of the undivided clock, which may be faster than the CPU's clock frequency. Hence, it is not possible to determine the state of the prescaler – even if it were readable, and the exact time it takes to switch from one clock division to the other cannot be exactly predicted. From the time the CLKPS values are written, it takes between T1 + T2 and T1 + 2 × T2 before the new clock frequency is active. In this interval, 2 active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

To avoid unintentional changes of clock frequency, a special write procedure must befollowed to change the CLKPS bits:

1. Write the clock prescaler change enable (CLKPCE) bit to one and all other bitsin CLKPR to zero.
2. Within four cycles, write the desired value to CLKPS while writing a zero to CLKPCE.

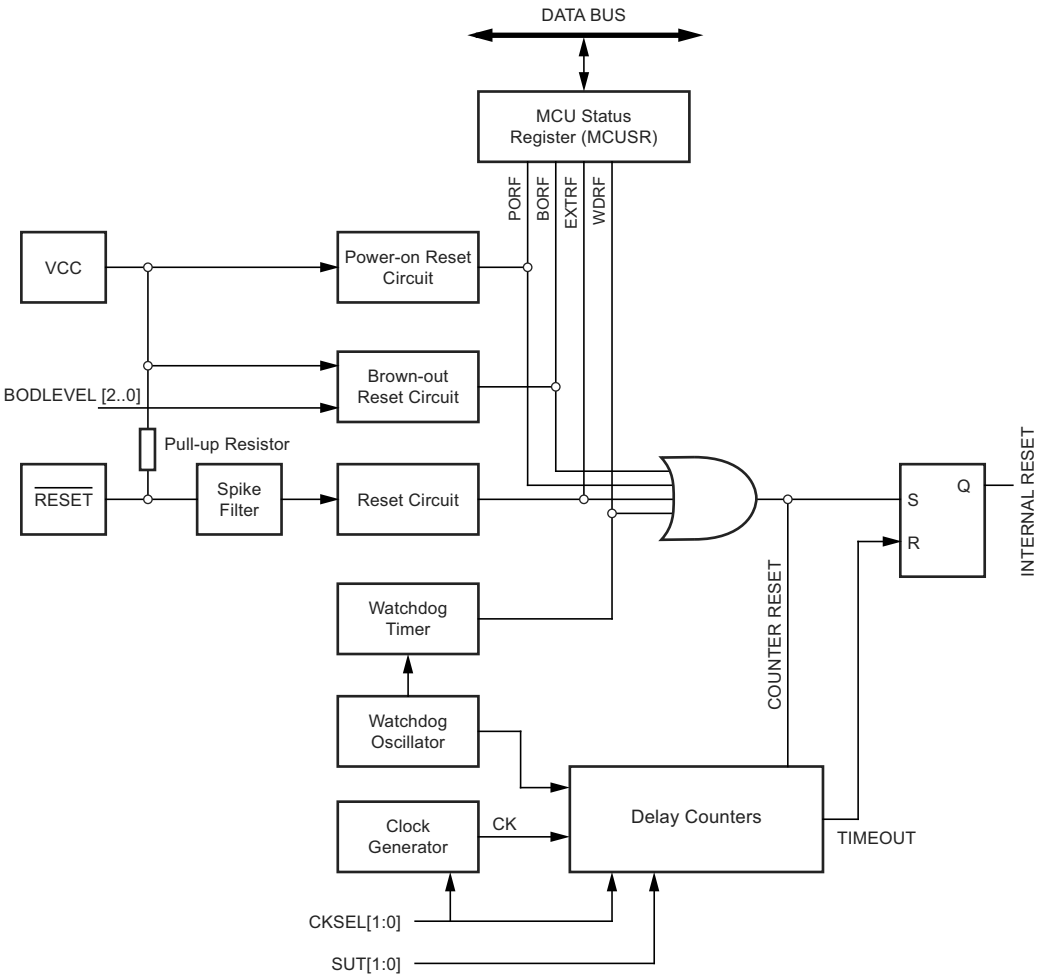
Interrupts must be disabled when changing prescaler setting to make sure the write procedure is not interrupted.

8. System Control and Reset

8.1 Resetting the AVR

During reset, all I/O registers are set to their initial values, and the program starts execution from the reset vector. The instruction placed at the reset vector must be an RJMP – relative jump – instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 8-1 shows the reset circuit. Table 21-4 on page 186 shows the electrical parameters of the reset circuitry.

Figure 8-1. Reset Logic



The I/O ports of the AVR® are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

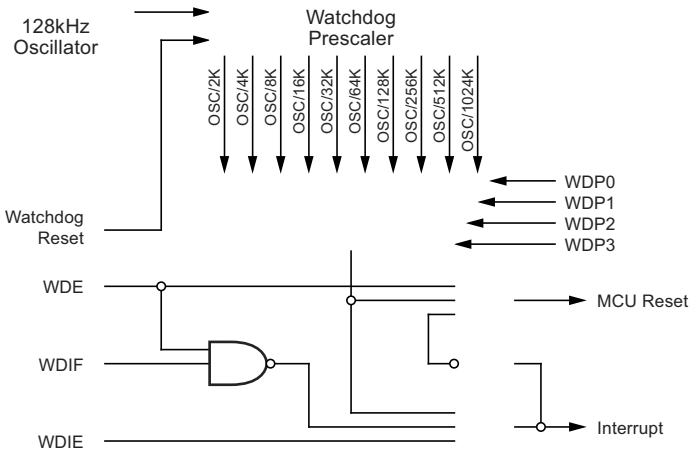
After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL fuses. The different selections for the delay period are presented in Section 6.2 “Clock Sources” on page 26.

To prevent unintentional disabling of the watchdog or unintentional change of time-out period, two different safety levels are selected by the fuse WDTON as shown in Table 8-1 See Section 8.4.1 “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 41 for details.

Table 8-1. WDT Configuration as a Function of the WDTON Fuse Setting

WDTON	Safety Level	WDT Initial State	Disable WDT	Change Time-out
Unprogrammed	1	Disabled	Timed sequence	No limitations
Programmed	2	Enabled	Always enabled	Timed sequence

Figure 8-7. Watchdog Timer



8.4.1 Timed Sequences for Changing the Configuration of the Watchdog Timer

The sequence for changing configuration differs slightly between the two safety levels. Separate procedures are described for each level.

8.4.2 Safety Level 1

In this mode, the watchdog timer is initially disabled, but can be enabled by writing the WDE bit to one without any restriction. A timed sequence is needed when disabling an enabled watchdog timer. To disable an enabled watchdog timer, the following procedure must be followed:

1. In the same operation, write a logic one to WDCE and WDE. A logic one must be written to WDE regardless of the previous value of the WDE bit.
2. Within the next four clock cycles, in the same operation, write the WDE and WDP bits as desired, but with the WDCE bit cleared.

8.4.3 Safety Level 2

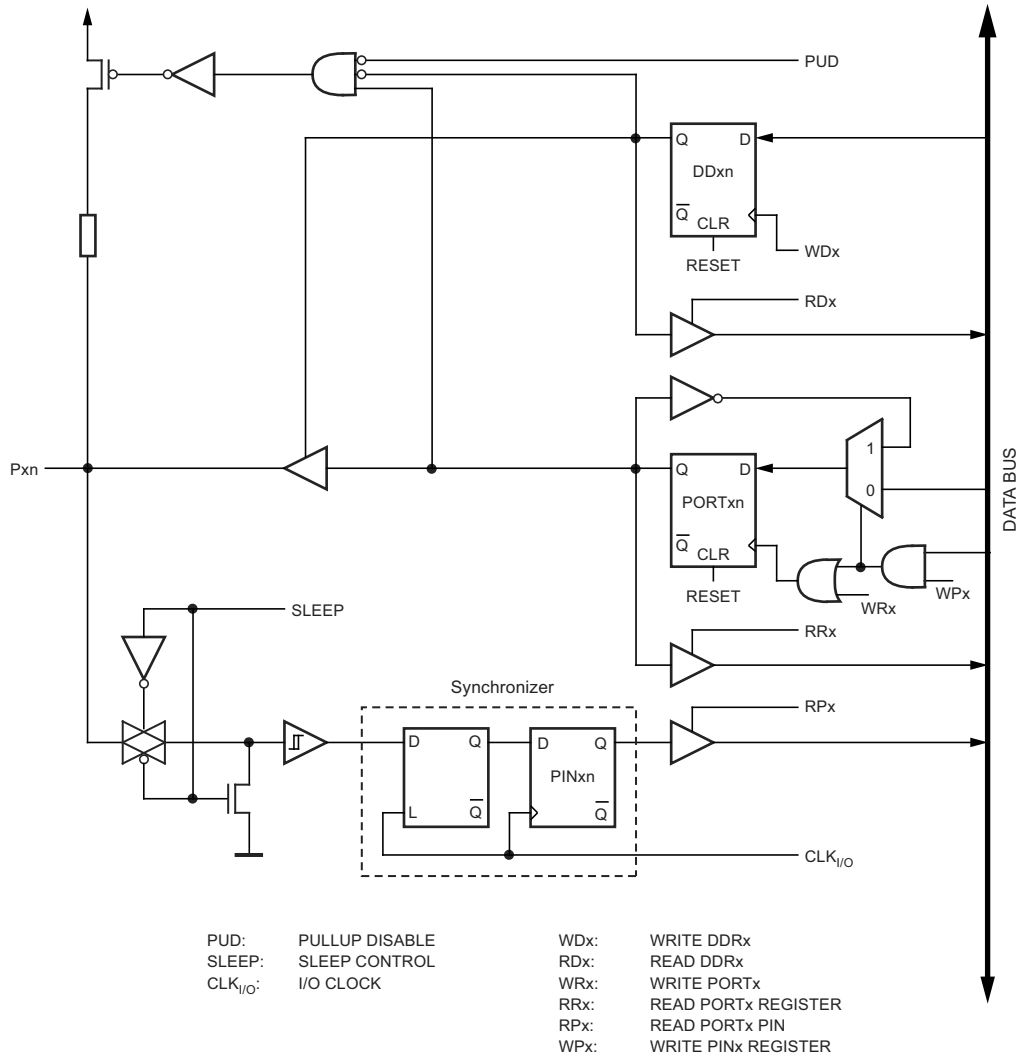
In this mode, the watchdog timer is always enabled, and the WDE bit will always read as one. A timed sequence is needed when changing the watchdog time-out period. To change the watchdog time-out, the following procedure must be followed:

1. In the same operation, write a logical one to WDCE and WDE. Even though the WDE always is set, the WDE must be written to one to start the timed sequence.
2. Within the next four clock cycles, in the same operation, write the WDP bits as desired, but with the WDCE bit cleare d. The value written to the WDE bit is irrelevant.

10.2 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 10-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 10-2. General Digital I/O(1)



Note: 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.

10.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in Section 10.4 “Register Description” on page 65, 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.

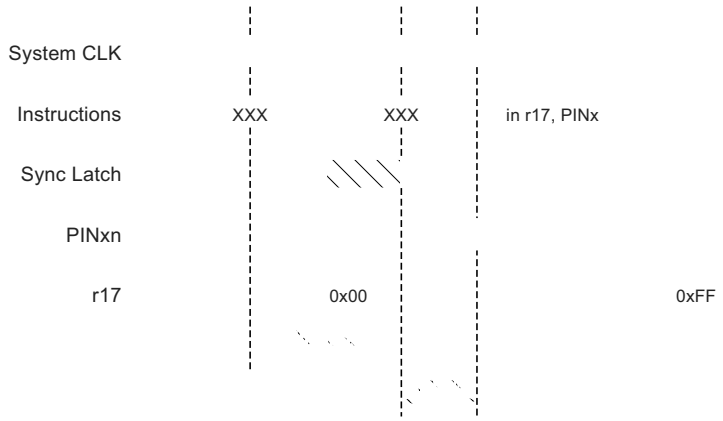
If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when reset condition becomes active, even 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).

10.2.5 Reading the Pin Value

Independent of the setting of data direction bit DDxn, the port pin can be read through the PINxn register bit. As shown in Figure 10-2 on page 52, the PINxn register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 10-4 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

Figure 10-4. Synchronization when Reading an Externally Applied Pin Value



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

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 10-5. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is 1 system clock period.

Figure 10-5. Synchronization when Reading a Software Assigned Pin Value

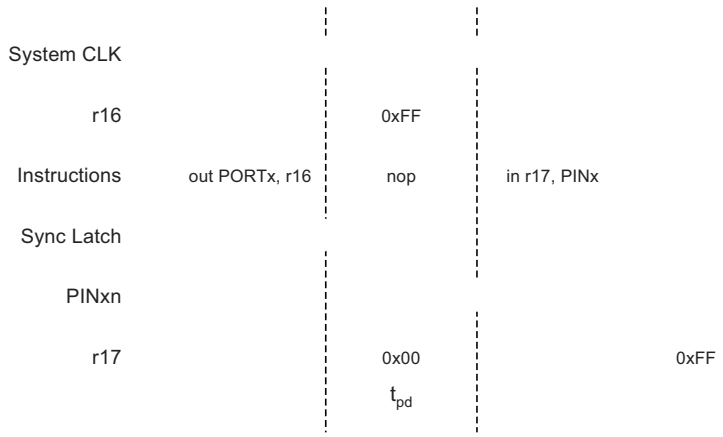


Table 10-13. Overriding Signals for Alternate Functions in PD3..PD0

Signal Name	PD3/INT1/PCINT19	PD2/INT0/PCINT18	PD1/PCINT17	PD0/PCINT16
PUOE	0	0	0	0
PUO	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	INT1 ENABLE + PCINT19 × PCIE2	INT0 ENABLE + PCINT18 × PCIE1	PCINT17 × PCIE2	PCINT16 × PCIE2
DIEOV	1	1	1	1
DI	PCINT19 INPUT INT1 INPUT	PCINT18 INPUT INT0 INPUT	PCINT17 INPUT	PCINT16 INPUT
AIO	–	–	–	–

10.4 Register Description

10.4.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
	–	BPDS	BPDSE	PUD	–	–	–	–	MCUCR
Read/Write	R	R/W	R/W	R/W	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 4 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See Section 10.2.1 “Configuring the Pin” on page 52 for more details about this feature.

10.4.2 PORTCR – Port Control Register

Bit	7	6	5	4	3	2	1	0	
	BBMD	**BBMC**	**BBMB**	**BBMA**	PUDD	PUDC	PUDB	PUDA	PORTR
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..4 – BBMx: Break-Before-Make Mode Enable**

When these bits are written to one, the port-wise break-before-make mode is activated. The intermediate tri-state cycle is then inserted when writing DDRxn to make an output. For further information, see
Section 10.2.3 “Break-Before-Make Switching” on page 53.

- **Bits 3..0 – PUDx: Port-Wise Pull-up Disable**

When these bits are written to one, the port-wise pull-ups in the defined I/O ports are disabled even if the DDxn and PORTxn registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). The port-wise pull-up disable bits are ORed with the global pull-up disable bit (PUD) from the MCUCR register. See Section 10.2.1 “Configuring the Pin” on page 52 for more details about this feature.

Signal description (internal signals):

count	Increment or decrement TCNT0 by 1.
clear	Clear TCNT0 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T0} in the following.
top	Signalize that TCNT0 has reached maximum value.

Depending of the mode of operation used, the counter is cleared or incremented at each timer clock (clk_{T0}). clk_{T0} 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_{T0} 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 clear timer on compare match bit (CTC0) located in the Timer/Counter control register (TCCR0A). For more details about advanced counting sequences, see Section 11.6 “Modes of Operation” on page 71.

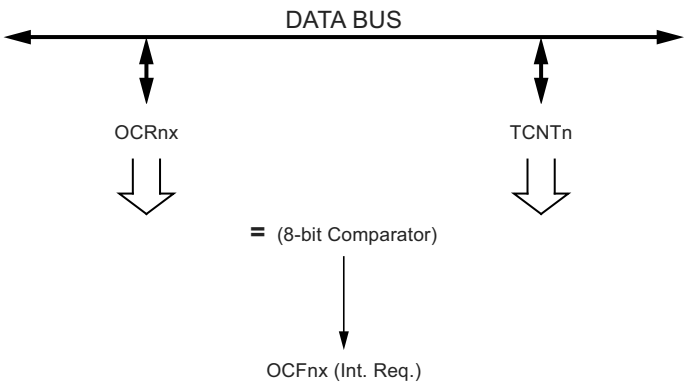
The Timer/Counter overflow flag (TOV0) is set according to the mode of operation selected by the CTC0 bit. TOV0 can be used for generating a CPU interrupt.

11.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the output compare registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the output compare flag (OCF0A or OCF0B) at the next timer clock cycle. If the corresponding interrupt is enabled, the output compare flag generates an output compare interrupt. The output compare flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a logical one to its I/O bit location.

Figure 11-3 shows a block diagram of the output compare unit.

Figure 11-3. Output Compare Unit, Block Diagram



11.5.1 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0x to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

11.5.2 Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the output compare unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the compare match will be missed, resulting in incorrect waveform generation.

12.2.1 Registers

The Timer/Counter (TCNT1), output compare registers (OCR1A/B), and input capture register (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section Section 12.3 “Accessing 16-bit Registers” on page 78. The Timer/Counter control registers (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the figure) signals are all visible in the timer interrupt flag register (TIFR1). All interrupts are individually masked with the timer interrupt mask register (TIMSK1). TIFR1 and TIMSK1 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T1 pin. The clock select logic block controls which clock source and edge 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_{T1}).

The double buffered output compare registers (OCR1A/B) are compared with the Timer/Counter value at all time. 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 (OC1A/B). See Section 12.7 “Output Compare Units” on page 84. The compare match event will also set the compare match flag (OCF1A/B) which can be used to generate an output compare interrupt request.

The input capture register can capture the Timer/Counter value at a given external (edge triggered) event on either the input capture pin (ICP1) or on the analog comparator pins (See Section 16. “Analog Comparator” on page 142). The input capture unit includes a digital filtering unit (noise canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR1A register, the ICR1 register, or by a set of fixed values. When using OCR1A as TOP value in a PWM mode, the OCR1A register can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICR1 register can be used as an alternative, freeing the OCR1A to be used as PWM output.

12.2.2 Definitions

The following definitions are used extensively throughout the section:

Table 12-1. Definitions

Parameter	Definition
BOTTOM	The counter reaches the BOTTOM when it becomes 0x0000.
MAX	The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65535).
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 one of the fixed values: 0x00FF, 0x01FF, or 0x03FF, or to the value stored in the OCR1A or ICR1 register. The assignment is dependent of the mode of operation.

12.3 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.

When a capture is triggered according to the ICES1 setting, the counter value is copied into the input capture register (ICR1). The event will also set the input capture flag (ICF1), and this can be used to cause an input capture interrupt, if this interrupt is enabled.

When the ICR1 is used as TOP value (see description of the WGM13:0 bits located in the TCCR1A and the TCCR1B register), the ICP1 is disconnected and consequently the input capture function is disabled.

- **Bit 5 – Reserved Bit**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero when TCCR1B is written.

- **Bit 4:3 – WGM13:2: Waveform Generation Mode**

See TCCR1A register description.

- **Bit 2:0 – CS12:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter, see Figure 12-10 on page 93 and Figure 12-11 on page 94.

Table 12-6. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

If external pin modes are used for the Timer/Counter1, transitions on the T1 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

12.11.3 TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
	FOC1A	**FOC1B**	–	–	–	–	–	–	**TCCR1C**
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A: Force Output Compare for Channel A**

- **Bit 6 – FOC1B: Force Output Compare for Channel B**

The FOC1A/FOC1B bits are only active when the WGM13:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written when operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the waveform generation unit. The OC1A/OC1B output is changed according to its COM1x1:0 bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x1:0 bits that determine the effect of the forced compare.

15. 2-Wire Serial Interface

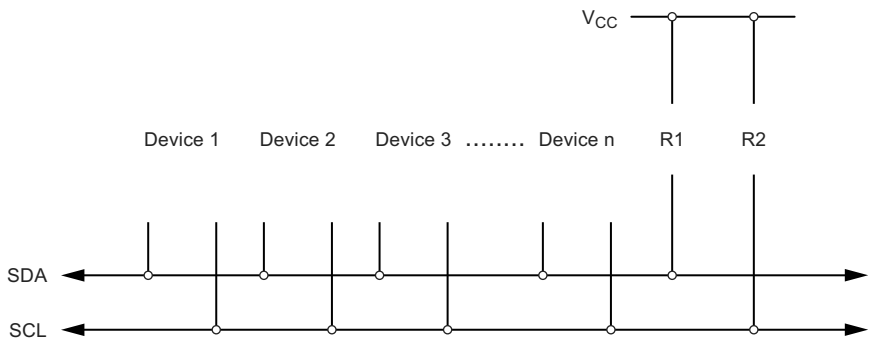
15.1 Features

- Simple yet powerful and flexible communication interface, only two bus lines needed
- Both master and slave operation supported
- Device can operate as transmitter or receiver
- 7-bit address space allows up to 128 different slave addresses
- Multi-master arbitration support
- Data transfer speed up to 400kHz in slave mode
- Slew-rate limited output drivers
- Noise suppression circuitry rejects spikes on bus lines
- Fully programmable slave address with general call support
- Address recognition causes wake-up when AVR® is in Sleep mode
- Compatible with philips I^2C protocol

15.2 2-wire Serial Interface Bus Definition

The 2-wire serial interface (TWI) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.

Figure 15-1. TWI Bus Interconnection



15.2.1 TWI Terminology

The following definitions are frequently encountered in this section.

Table 15-1. TWI Terminology

Term	Description
Master	The device that initiates and terminates a transmission and generates the SCL clock.
Slave	The device addressed by a master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

The PRTWI bit in Section 7.4.3 “PRR – Power Reduction Register” on page 36 must be written to zero to enable the 2-wire serial interface.

Table 15-5. Status Codes for Slave Receiver Mode (Continued)

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
0x98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
			1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
			1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

16.2 Register Description

16.2.1 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – ACME: Analog Comparator Multiplexer Enable**

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the analog comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the analog comparator. For a detailed description of this bit, see
Section 16.1 “Analog Comparator Multiplexed Input” on page 142.

16.2.2 ACSR – Analog Comparator Control and Status Register

Bit	7	6	5	4	3	2	1	0	
	ACD	**ACBG**	**ACO**	**ACI**	**ACIE**	**ACIC**	**ACIS1**	**ACIS0**	**ACSR**
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

- **Bit 7 – ACD: Analog Comparator Disable**

When this bit is written logic one, the power to the analog comparator is switched off. This bit can be set at any time to turn off the analog comparator. This will reduce power consumption in active and idle mode. When changing the ACD bit, the analog comparator interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – ACBG: Analog Comparator Bandgap Select**

When this bit is set, a fixed internal bandgap reference voltage replaces the positive input to the analog comparator. When this bit is cleared, AIN0 is applied to the positive input of the analog comparator. See
Section 8.3 “Internal Voltage Reference” on page 40

- **Bit 5 – ACO: Analog Comparator Output**

The output of the analog comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1 – 2 clock cycles.

- **Bit 4 – ACI: Analog Comparator Interrupt Flag**

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The analog comparator interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

- **Bit 3 – ACIE: Analog Comparator Interrupt Enable**

When the ACIE bit is written logic one and the I-bit in the status register is set, the analog comparator interrupt is activated. When written logic zero, the interrupt is disabled.

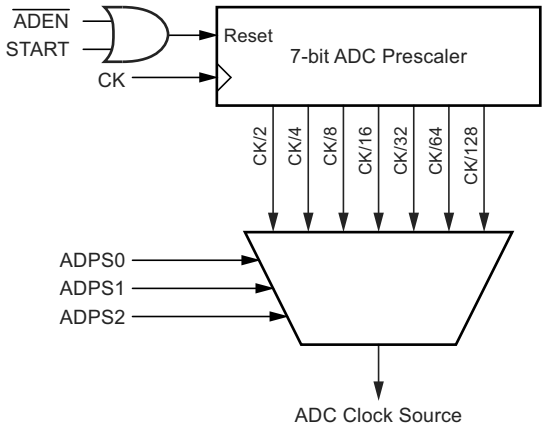
- **Bit 2 – ACIC: Analog Comparator Input Capture Enable**

When written logic one, this bit enables the input capture function in Timer/Counter1 to be triggered by the analog comparator. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 input capture interrupt. When written logic zero, no connection between the analog comparator and the input capture function exists. To make the comparator trigger the Timer/Counter1 input capture interrupt, the ICIE1 bit in the timer interrupt mask register (TIMSK1) must be set.

17.5 Prescaling and Conversion Timing

The successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution.

Figure 17-3. ADC Prescaler

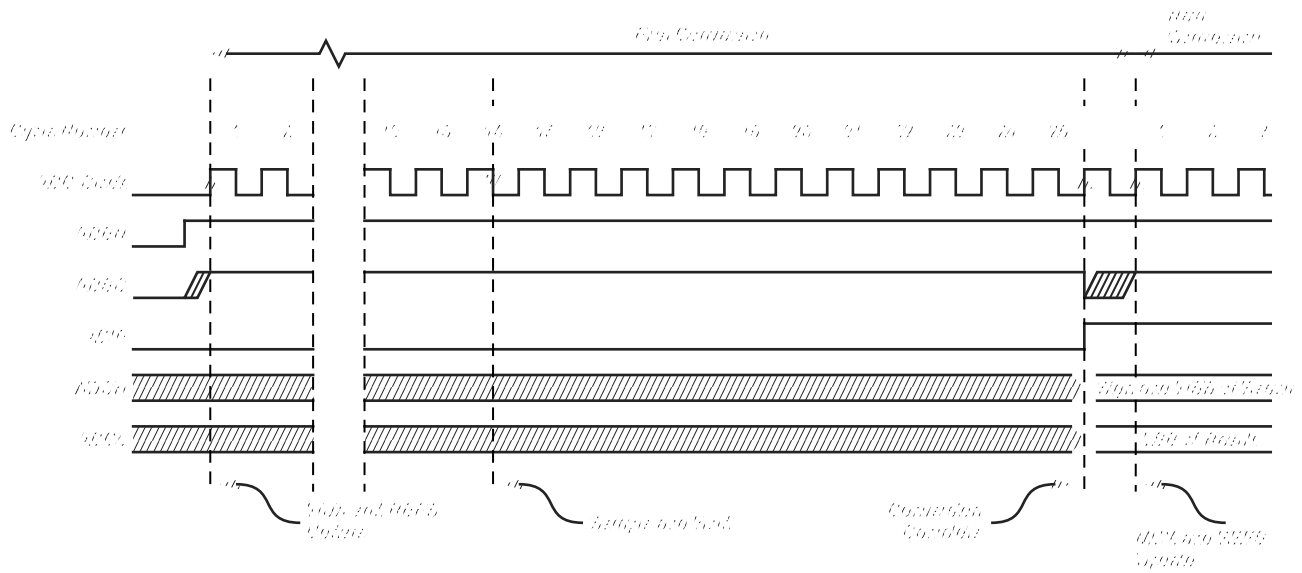


The ADC module contains a prescaler, as illustrated in Figure 17-3, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle.

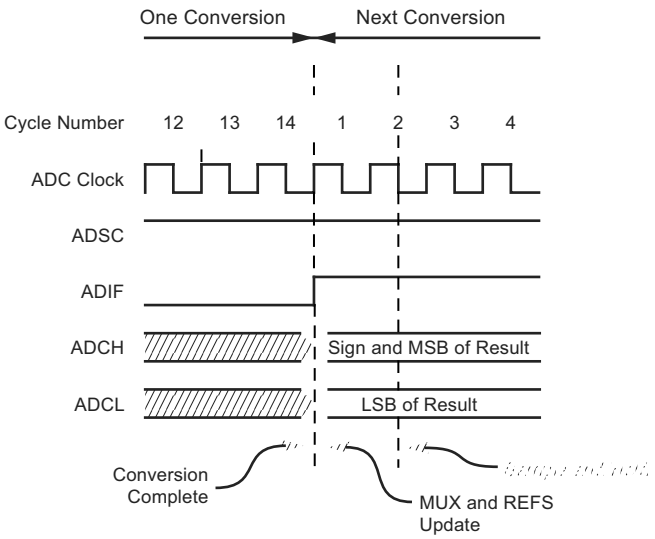
A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry, as shown in Figure 17-4 below.

Figure 17-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)



In free running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. See Figure 17-7.

Figure 17-7. ADC Timing Diagram, Free Running Conversion



For a summary of conversion times, see Table 17-1.

Table 17-1. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto triggered conversions	2	13.5
Free running conversions	2.5	14

17.6 Changing Channel or Reference Selection

Bits MUXn and REFS0 in the ADMUX register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA is set). Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or reference selection values to ADMUX until one ADC clock cycle after ADSC is written.

If auto triggering is used, the exact time of the triggering event can be indeterministic. Special care must be taken when updating the ADMUX register, in order to control which conversion will be affected by the new settings.

If both ADATE and ADEN is written to one, an interrupt event can occur at any time. If the ADMUX register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

- When ADATE or ADEN is cleared.
- During conversion, minimum one ADC clock cycle after the trigger event.
- After a conversion, before the interrupt flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next ADC conversion.

19.4.4 Programming Time for Flash when Using SPM

The calibrated oscillator is used to time flash accesses. Table 19-1 shows the typical programming time for flash accesses from the CPU.

Table 19-1. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash write (page erase, page write, and write lock bits by SPM)	3.7ms	4.5ms

19.4.5 Simple Assembly Code Example for a Boot Loader

Note that the RWWSB bit will always be read as zero in Atmel® ATtiny88. Nevertheless, to ensure compatibility with devices supporting read-while-write it is recommended to check this bit as shown in the code example.

```
;-the routine writes one page of data from RAM to Flash
; the first data location in RAM is pointed to by the Y pointer
; the first data location in Flash is pointed to by the Z-pointer
;-error handling is not included
;-the routine must be placed inside the Boot space
; (at least the Do_spm sub routine). Only code inside NRWW section can
; be read during Self-Programming (Page Erase and Page Write).
;-registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
; loophi (r25), spmcrval (r20)
; storing and restoring of registers is not included in the routine
; register usage can be optimized at the expense of code size
;-It is assumed that either the interrupt table is moved to the Boot
; loader section or that the interrupts are disabled.
.equ PAGESIZEB = PAGESIZE*2      ;PAGESIZEB is page size in BYTES, not words
.org SMALLBOOTSTART
Write_page:
   ;   Page Erase
   ldi        spmcrval, (1<<PGERS) | (1<<SELFPRGEN)
   rcall      Do_spm

   ;   re-enable the RWW section
   ldi        spmcrval, (1<<CTPB) | (1<<SELFPRGEN)
   rcall      Do_spm

   ;   transfer data from RAM to Flash page buffer
   ldi        looplo, low(PAGESIZEB)     ;init loop variable
   ldi        loophi, high(PAGESIZEB)    ;not required for PAGESIZEB<=256
Wrloop:
   ld         r0, Y+
   ld         r1, Y+
   ldi        spmcrval, (1<<SELFPRGEN)
   rcall      Do_spm
   adiw       ZH:ZL, 2
   sbiw       loophi:looplo, 2           ;use subi for PAGESIZEB<=256
   brne       Wrloop

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

   ;   re-enable the RWW section
```

```
   ldi        spmcrval, (1<<CTPB) | (1<<SELFPRGEN)
   rcall      Do_spm

   ;   read back and check, optional
   ldi        looplo, low(PAGESIZEB)     ;init loop variable
   ldi        loophi, high(PAGESIZEB)    ;not required for PAGESIZEB<=256
   subi       YL, low(PAGESIZEB)         ;restore pointer
   sbci       YH, high(PAGESIZEB)
Rdloop:
   lpm        r0, Z+
   ld         r1, Y+
   cpse       r0, r1
   rjmp       Error
   sbiw       loophi:looplo, 1           ;use subi for PAGESIZEB<=256
   brne       Rdloop

   ;   return to RWW section
   ;   verify that RWW section is safe to read
Return:
   in         temp1, SPMCSR
   sbrs       temp1, RWWSB               ; If RWWSB is set, the RWW section is not
                                         ready yet
   ret
   ;   re-enable the RWW section
   ldi        spmcrval, (1<<CTPB) | (1<<SELFPRGEN)
   rcall      Do_spm
   rjmp       Return

Do_spm:
   ;   check for previous SPM complete
Wait_spm:
   in         temp1, SPMCSR
   sbrc       temp1, SELFPRGEN
   rjmp       Wait_spm
   ;   input: spmcrval determines SPM action
   ;   disable interrupts if enabled, store status
   in  temp2, SREG
   cli
   ;   check that no EEPROM write access is present
Wait_ee:
   sbic       EECR, EEPE
   rjmp       Wait_ee
   ;   SPM timed sequence
   out        SPMCSR, spmcrval
   spm
   ;   restore SREG (to enable interrupts if originally enabled)
   out        SREG, temp2
   ret
```

The XA1/XA0 pins determine the action executed when the CLKI pin is given a positive pulse. The bit coding is shown in Table 20-11.

Table 20-11. XA1 and XA0 Coding

XA1	XA0	Action when CLKI is Pulsed
0	0	Load flash or EEPROM address (high or low address byte determined by BS1).
0	1	Load data (high or low data byte for flash determined by BS1).
1	0	Load command
1	1	No action, idle

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different commands are shown in Table 20-12.

Table 20-12. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip erase
0100 0000	Write fuse bits
0010 0000	Write lock bits
0001 0000	Write flash
0001 0001	Write EEPROM
0000 1000	Read signature bytes and calibration byte
0000 0100	Read fuse and lock bits
0000 0010	Read flash
0000 0011	Read EEPROM

20.7 Parallel Programming

20.7.1 Enter Programming Mode

The following algorithm puts the device in parallel (high-voltage) programming mode:

1. Set prog_enable pins listed in Table 20-10 on page 171 to “0000”, RESET pin to 0V and V_{CC} to 0V.
2. Apply 4.5 – 5.5V between V_{CC} and GND.

Ensure that V_{CC} reaches at least 2.7V within the next 20µs.

3. Wait 20 – 60µs, and apply 11.5 – 12.5V to RESET.
4. Keep the prog_enable pins unchanged for at least 10µs after the high-voltage has been applied to ensure the prog_enable signature has been latched.
5. Wait at least 300µs before giving any parallel programming commands.
6. Exit programming mode by power the device down or by bringing RESET pin to 0V.

If the rise time of the V_{CC} is unable to fulfill the requirements listed above, the following alternative algorithm can be used.

1. Set prog_enable pins listed in Table 20-10 on page 171 to “0000”, RESET pin to 0V and V_{CC} to 0V.
2. Apply 4.5 – 5.5V between V_{CC} and GND.
3. Monitor V_{CC}, and as soon as V_{CC} reaches 0.9 – 1.1V, apply 11.5 – 12.5V to RESET.
4. Keep the prog_enable pins unchanged for at least 10µs after the high-voltage has been applied to ensure the prog_enable signature has been latched.
5. Wait until V_{CC} actually reaches 4.5 -5.5V before giving any parallel programming commands.
6. Exit programming mode by power the device down or by bringing RESET pin to 0V.

20.8.3 Serial Programming Instruction set

Table 20-15 and Figure 20-8 on page 182 describes the Instruction set.

Table 20-15. Serial Programming Instruction Set (Hexadecimal values)

Instruction/Operation	Instruction Format			
	Byte 1	Byte 2	Byte 3	Byte4
Programming enable	$AC	$53	$00	$00
Chip erase (program memory/EEPROM)	$AC	$80	$00	$00
Poll RDY/$\overline{\text{BSY}}$	$F0	$00	$00	data byte out
Load instructions				
Load extended address byte(1)	$4D	$00	Extended adr	$00
Load program memory page, high byte	$48	$00	adr LSB	high data byte in
Load program memory page, low byte	$40	$00	adr LSB	low data byte in
Load EEPROM memory page (page access)	$C1	$00	0000 000aa	data byte in
Read instructions				
Read program memory, high byte	$28	adr MSB	adr LSB	high data byte out
Read program memory, low byte	$20	adr MSB	adr LSB	low data byte out
Read EEPROM memory	$A0	0000 00aa	aaaa aaaa	data byte out
Read lock bits	$58	$00	$00	data byte out
Read signature byte	$30	$00	0000 000aa	data byte out
Read fuse bits	$50	$00	$00	data byte out
Read fuse high bits	$58	$08	$00	data byte out
Read fuse extended bits	$50	$08	$00	data byte out
Read calibration byte	$38	$00	$00	data byte out
Write instructions				
Write program memory page	$4C	adr MSB	adr LSB	$00
Write EEPROM memory	$C0	0000 00aa	aaaa aaaa	data byte in
Write EEPROM memory page (page access)	$C2	0000 00aa	aaaa aa00	$00
Write lock bits	$AC	$E0	$00	data byte in
Write fuse bits	$AC	$A0	$00	data byte in
Write fuse high bits	$AC	$A8	$00	data byte in
Write fuse extended bits	$AC	$A4	$00	data byte in

Notes:
1. Not all instructions are applicable for all parts.
2. a = address.
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused fuses and lock bits should be unprogrammed (‘1’).
5. Refer to the corresponding section for fuse and lock bits, calibration and signature bytes and page size.
6. See htt://www.atmel.com/avr for application notes regarding programming and programmers.

Table 21-7. 2-wire Serial Bus Requirements (Continued)

Parameter	Condition	Symbol	Min	Max	Unit
Setup time for STOP condition	$f_{SCL} \leq 100kHz$	$t_{SU;STO}$	4.0	–	µs
	$f_{SCL} > 100kHz$		0.6	–	µs
Bus free time between a STOP and START condition	$f_{SCL} \leq 100kHz$	t_{BUF}	4.7	–	µs
	$f_{SCL} > 100kHz$		1.3	–	µs

Notes:
1. This parameter is characterized, only, and not fully tested.
2. Required only for $f_{SCL} > 100kHz$.
3. C_b = capacitance of one bus line in pF.
4. f_{CK} = CPU clock frequency
5. This requirement applies to all 2-wire serial interface operation in ATtiny88. Other devices connected to the 2-wire serial bus need only obey the general f_{SCL} requirement.
6. The actual low period generated by the 2-wire serial interface of ATtiny88 is $(1/f_{SCL} – 2/f_{CK})$, thus f_{CK} must be greater than 6MHz for the low time requirement to be strictly met at f_{SCL} = 100kHz.
7. The actual low period generated by the 2-wire serial interface of ATtiny88 is $(1/f_{SCL} – 2/f_{CK})$, thus the low time requirement will not be strictly met for $f_{SCL} > 308kHz$ when f_{CK} = 8MHz. Still, ATtiny88 devices connected to the bus may communicate at full speed (400kHz) with other ATtiny88 devices, as well as any other device with a proper t_{LOW} acceptance margin.

Figure 21-3. 2-wire Serial Bus Timing