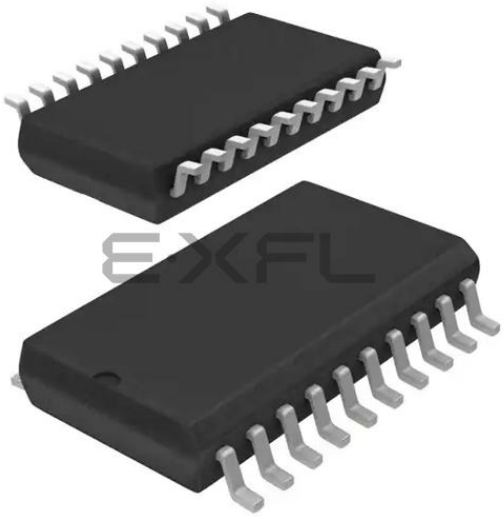
Microchip Technology - ATTINY461-20SUR Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

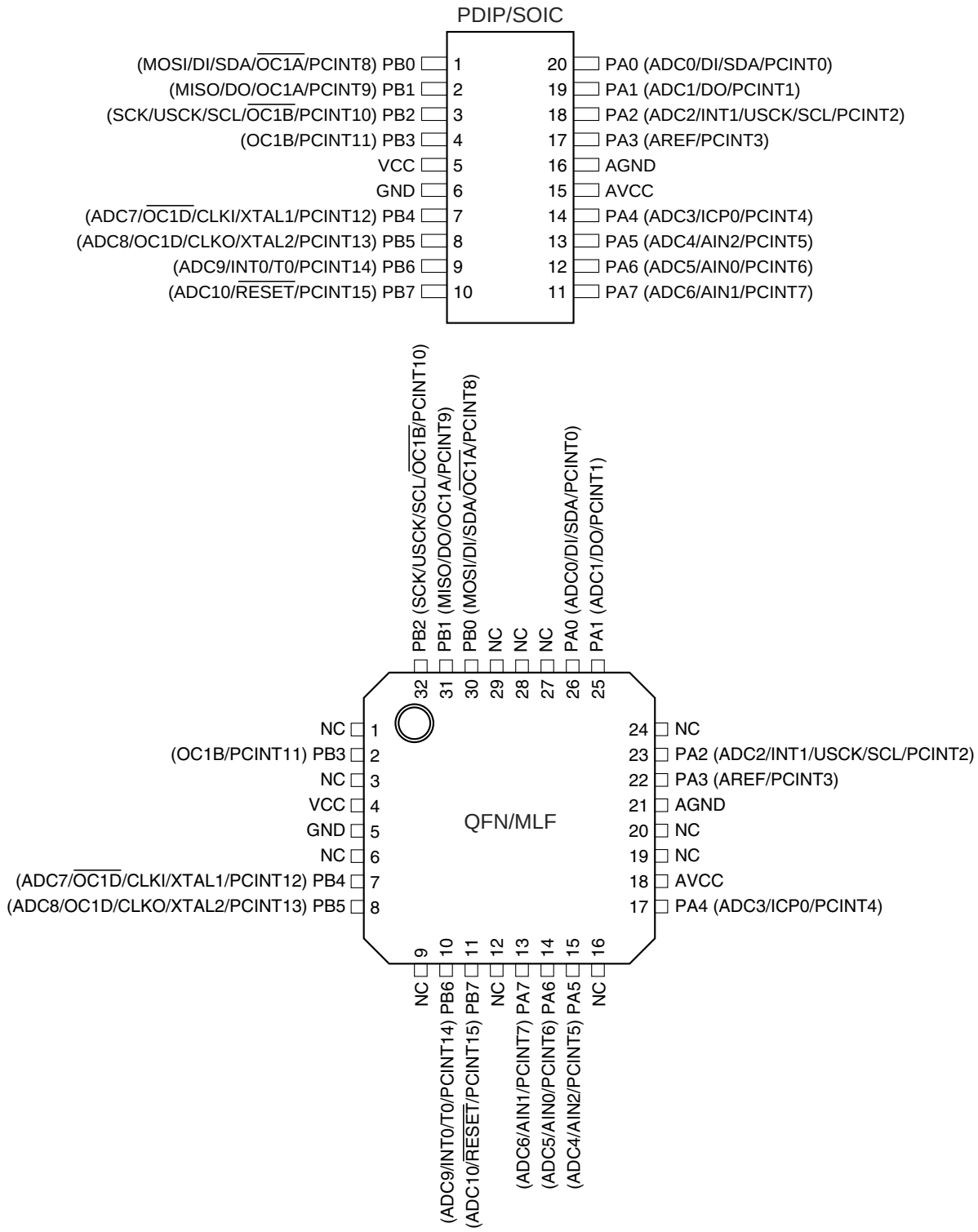
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	20MHz
Connectivity	USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	4KB (2K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny461-20sur

1. Pin Configurations

Figure 1-1. Pinout ATtiny261/461/861 and ATtiny261V/461V/861V



Note: To ensure mechanical stability the center pad underneath the QFN/MLF package should be soldered to ground on the board.

Figure 4-4. The Parallel Instruction Fetches and Instruction Executions

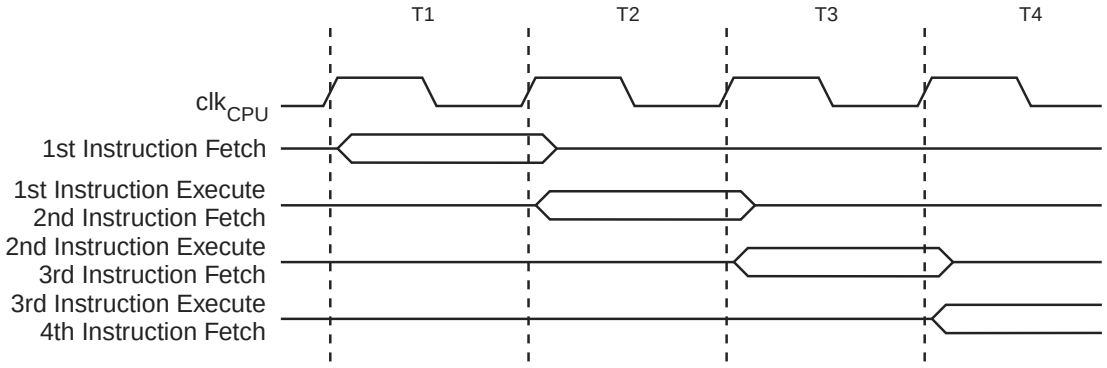
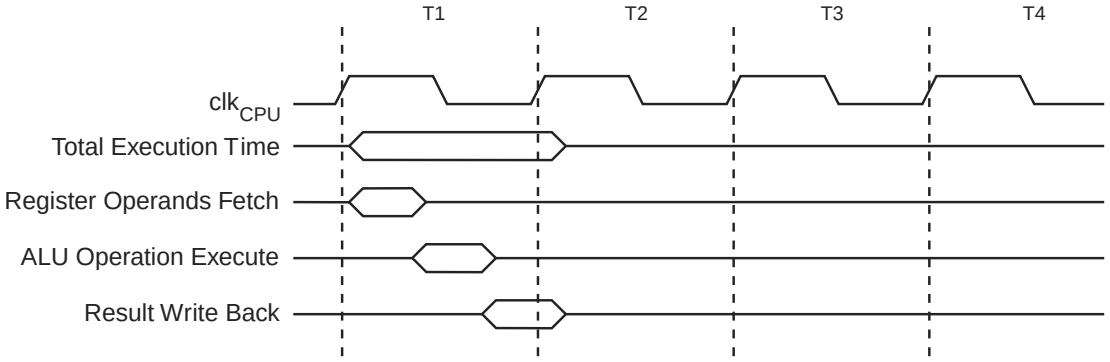


Figure 4-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 4-5. Single Cycle ALU Operation



4.7 Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate Program Vector in the Program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt.

The lowest addresses in the Program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in “Interrupts” on page 50. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the Interrupt Flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding Interrupt Flag. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s)

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 cleared. The value written to the WDE bit is irrelevant.

8.3.2 Code Examples

The following code example shows one assembly and one C function for turning off the WDT. The example assumes that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
WDT_off:
  wdr
  ; Clear WDRF in MCUSR
  ldi  r16, (0<<WDRF)
  out  MCUSR, r16
  ; Write logical one to WDCE and WDE
  ; Keep old prescaler setting to prevent unintentional Watchdog Reset
  in   r16, WDTCSR
  ori  r16, (1<<WDCE)|(1<<WDE)
  out  WDTCSR, r16
  ; Turn off WDT
  ldi  r16, (0<<WDE)
  out  WDTCSR, r16
  ret
```

C Code Example

```
void WDT_off(void)
{
  _WDR();
  /* Clear WDRF in MCUSR */
  MCUSR = 0x00
  /* Write logical one to WDCE and WDE */
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Turn off WDT */
  WDTCSR = 0x00;
}
```

Note: See “Code Examples” on page 6.

9. Interrupts

This section describes the specifics of the interrupt handling as performed in ATtiny261/461/861. For a general explanation of the AVR interrupt handling, refer to “Reset and Interrupt Handling” on page 12.

9.1 Interrupt Vectors

Interrupt vectors of ATtiny261/461/861 are described in Table 9-1 below.

Table 9-1. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset
2	0x0001	INT0	External Interrupt Request 0
3	0x0002	PCINT	Pin Change Interrupt Request
4	0x0003	TIMER1_COMPA	Timer/Counter1 Compare Match A
5	0x0004	TIMER1_COMPB	Timer/Counter1 Compare Match B
6	0x0005	TIMER1_OVF	Timer/Counter1 Overflow
7	0x0006	TIMER0_OVF	Timer/Counter0 Overflow
8	0x0007	USI_START	USI Start
9	0x0008	USI_OVF	USI Overflow
10	0x0009	EE_RDY	EEPROM Ready
11	0x000A	ANA_COMP	Analog Comparator
12	0x000B	ADC	ADC Conversion Complete
13	0x000C	WDT	Watchdog Time-out
14	0x000D	INT1	External Interrupt Request 1
15	0x000E	TIMER0_COMPA	Timer/Counter0 Compare Match A
16	0x000F	TIMER0_COMPB	Timer/Counter0 Compare Match B
17	0x0010	TIMER0_CAPT	Timer/Counter1 Capture Event
18	0x0011	TIMER1_COMPD	Timer/Counter1 Compare Match D
19	0x0012	FAULT_PROTECTION	Timer/Counter1 Fault Protection

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 most typical and general program setup for the Reset and Interrupt Vector Addresses in ATtiny261/461/861 is shown in the program example below.

```
Address  Labels Code                 Comments

0x0000          rjmp   RESET          ; Reset Handler
0x0001          rjmp   EXT_INT0       ; IRQ0 Handler
0x0002          rjmp   PCINT          ; PCINT Handler
```

Control Register (MCUCR) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 Interrupt Vector.

- **Bit 5 – PCIE1: Pin Change Interrupt Enable**

When the PCIE1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt is enabled. Any change on any enabled PCINT7:0 or PCINT15:12 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI Interrupt Vector. PCINT7:0 and PCINT15:12 pins are enabled individually by the PCMSK0 and PCMSK1 Register.

- **Bit 4 – PCIE0: Pin Change Interrupt Enable**

When the PCIE0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt is enabled. Any change on any enabled PCINT11:8 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI Interrupt Vector. PCINT11:8 pins are enabled individually by the PCMSK1 Register.

- **Bits 3:0 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

9.3.3 GIFR – General Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x3A (0x5A)	**INT1**	**INTF0**	**PCIF**	–	–	–	–	–	**GIFR**
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – INTF1: External Interrupt Flag 1**

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT1 is configured as a level interrupt.

- **Bit 6 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in GIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

- **Bit 5 – PCIF: Pin Change Interrupt Flag**

When a logic change on any PCINT15 pin triggers an interrupt request, PCIF becomes set (one). If the I-bit in SREG and the PCIE bit in GIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bits 4:0 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

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).

The pull-up resistor is activated, if the PUExn is written logic one. To switch the pull-up resistor off, PUExn has to be written logic zero.

10.1.2 Toggling the Pin

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn. Note that the SBI instruction can be used to toggle one single bit in a port.

10.1.3 Switching Between Input and Output

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled {DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the MCUCR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b10) as an intermediate step.

Table 10-1 summarizes the control signals for the pin value.

Table 10-1. Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

10.1.4 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, 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 10-3 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.

The following code examples show how to do an atomic write of the TCNT0H/L register contents. Writing any of the OCR0A/B registers can be done by using the same principle.

Assembly Code Example

```
TIM0_WriteTCNT0:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Set TCNT0 to r17:r16
  out TCNT0H,r17
  out TCNT0L,r16
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

C Code Example

```
void TIM0_WriteTCNT0( unsigned int i )
{
  unsigned char sreg;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  _CLI();
  /* Set TCNT0 to i */
  TCNT0H = (i >> 8);
  TCNT0L = (unsigned char)i;
  /* Restore global interrupt flag */
  SREG = sreg;
}
```

Note: See “Code Examples” on page 6.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT0H/L.

11.9.1 Reusing the temporary high byte register

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

if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

- **Bit 3 – OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable**

When the OCIE0B bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter Compare Match B interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter occurs, i.e., when the OCF0B bit is set in the Timer/Counter Interrupt Flag Register – TIFR0.

- **Bit 1 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

- **Bit 0 – TICIE0: Timer/Counter0, Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Input Capture interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 50.) is executed when the ICF0 flag, located in TIFR, is set.

11.10.8 TIFR – Timer/Counter0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x38 (0x58)	OCF1D	OCF1A	OCF1B	OCF0A	OCF0B	TOV1	TOV0	ICF0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 4 – OCF0A: Output Compare Flag 0 A**

The OCF0A bit is set when a Compare Match occurs between the Timer/Counter0 and the data in OCR0A – Output Compare Register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 Compare Match Interrupt Enable), and OCF0A are set, the Timer/Counter0 Compare Match Interrupt is executed.

The OCF0A is also set in 16-bit mode when a Compare Match occurs between the Timer/Counter and 16-bit data in OCR0B/A. The OCF0A is not set in Input Capture mode when the Output Compare Register OCR0A is used as an Input Capture Register.

- **Bit 3 – OCF0B: Output Compare Flag 0 B**

The OCF0B bit is set when a Compare Match occurs between the Timer/Counter and the data in OCR0B – Output Compare Register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter Compare B Match Interrupt Enable), and OCF0B are set, the Timer/Counter Compare Match Interrupt is executed.

The OCF0B is not set in 16-bit Output Compare mode when the Output Compare Register OCR0B is used as the high byte of the 16-bit Output Compare Register or in 16-bit Input Capture mode when the Output Compare Register OCR0B is used as the high byte of the Input Capture Register.

12.3.1.1 Prescaler Reset

Setting the PSR1 bit in TCCR1B register resets the prescaler. It is possible to use the Prescaler Reset for synchronizing the Timer/Counter to program execution.

12.3.1.2 Prescaler Initialization for Asynchronous Mode

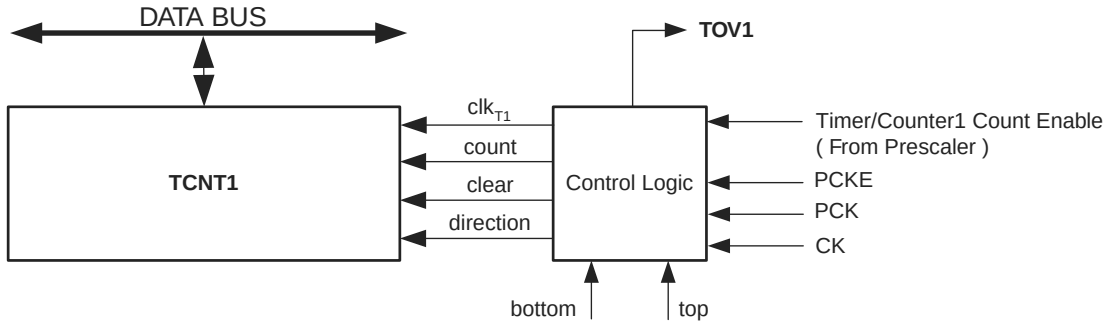
To change Timer/Counter1 to the asynchronous mode follow the procedure below:

1. Enable PLL.
2. Wait 100 µs for PLL to stabilize.
3. Poll the PLOCK bit until it is set.
4. Set the PCKE bit in the PLLCSR register which enables the asynchronous mode.

12.4 Counter Unit

The main part of the Timer/Counter1 is the programmable bi-directional counter unit. Figure 12-4 shows a block diagram of the counter and its surroundings.

Figure 12-4. Counter Unit Block Diagram



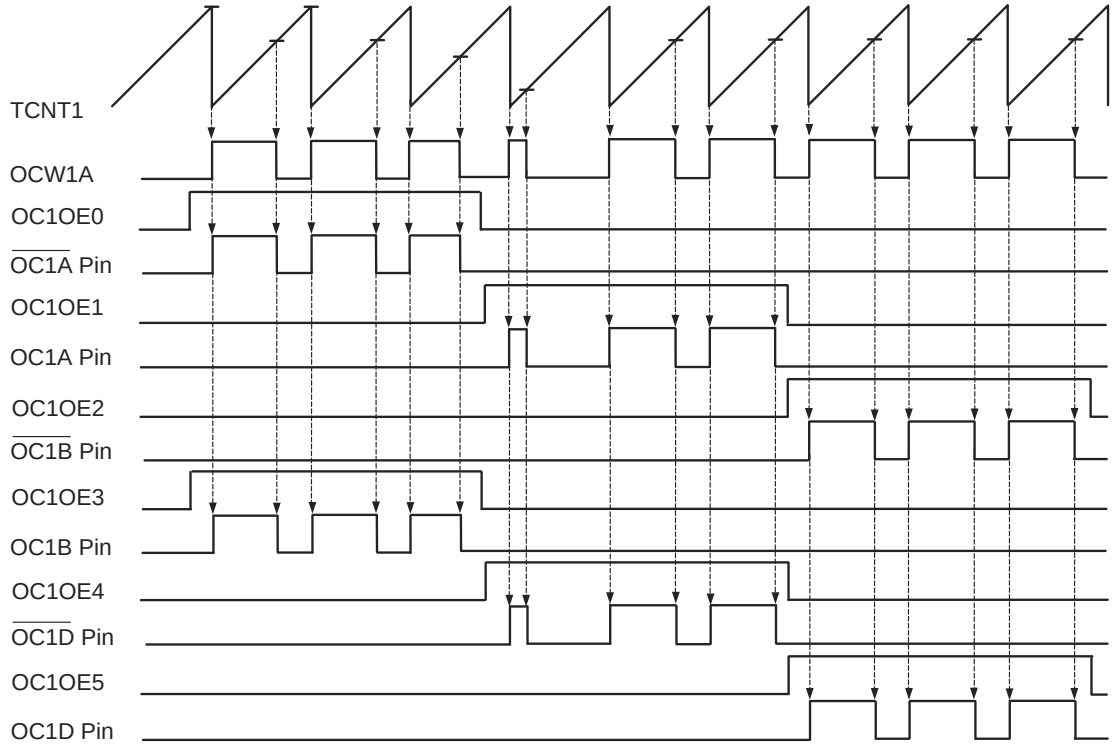
Signal description (internal signals):

count	TCNT1 increment or decrement enable.
direction	Select between increment and decrement.
clear	Clear TCNT1 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T1} in the following.
top	Signalize that TCNT1 has reached maximum value.
bottom	Signalize that TCNT1 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T1}). The timer clock is generated from an synchronous system clock or an asynchronous PLL clock using the Clock Select bits (CS13:0) and the PCK Enable bit (PCKE). When no clock source is selected (CS13:0 = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, regardless of whether clk_{T1} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence of the Timer/Counter1 is determined by bits WGM11:10, PWM1A and PWM1B, located in the Timer/Counter1 Control Registers (TCCR1A, TCCR1C and TCCR1D). For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 99. The Timer/Counter Overflow Flag (TOV1) is set according to the mode of operation and can be used for generating a CPU interrupt.

Figure 12-14. PWM6 Mode, Single-slope Operation, Timing Diagram



The general I/O port function is overridden by the Output Compare value (OC1x / $\overline{OC1x}$) from the Dead Time Generator if either of the COM1x1:0 bits are set. The Output Compare pins can also be overriden by the Output Compare Override Enable bits OC1OE5:OC1OE0. If an Override Enable bit is cleared, the actual value from the port register will be visible on the port pin and, if the Override Enable bit is set, the Output Compare pin is allowed to be connected on the port pin. The Output Compare Pin configurations are described in Table 12-5, Table 12-6 and Table 12-7.

Table 12-5. Configuration of Output Compare Pins OC1A and $\overline{OC1A}$ in PWM6 Mode

COM1A1	COM1A0	$\overline{OC1A}$ Pin (PB0)	OC1A Pin (PB1)
0	0	Disconnected	Disconnected
0	1	OC1A • OC1OE0	OC1A • OC1OE1
1	0	OC1A • OC1OE0	OC1A • OC1OE1
1	1	OC1A • OC1OE0	OC1A • OC1OE1

Table 12-6. Configuration of Output Compare Pins OC1B and $\overline{OC1B}$ in PWM6 Mode

COM1B1	COM1B0	$\overline{OC1B}$ Pin (PB2)	OC1B Pin (PB3)
0	0	Disconnected	Disconnected
0	1	OC1A • OC1OE2	OC1A • OC1OE3
1	0	OC1A • OC1OE2	OC1A • OC1OE3
1	1	OC1A • OC1OE2	OC1A • OC1OE3

Table 12-7. Configuration of Output Compare Pins OC1D and $\overline{\text{OC1D}}$ in PWM6 Mode

COM1D1	COM1D0	$\overline{\text{OC1D}}$ Pin (PB4)	OC1D Pin (PB5)
0	0	Disconnected	Disconnected
0	1	OC1A • OC1OE4	OC1A • OC1OE5
1	0	OC1A • OC1OE4	OC1A • OC1OE5
1	1	OC1A • OC1OE4	OC1A • OC1OE5

12.9 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T1}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set.

Figure 12-15 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than Phase and Frequency Correct PWM Mode.

Figure 12-15. Timer/Counter Timing Diagram, no Prescaling

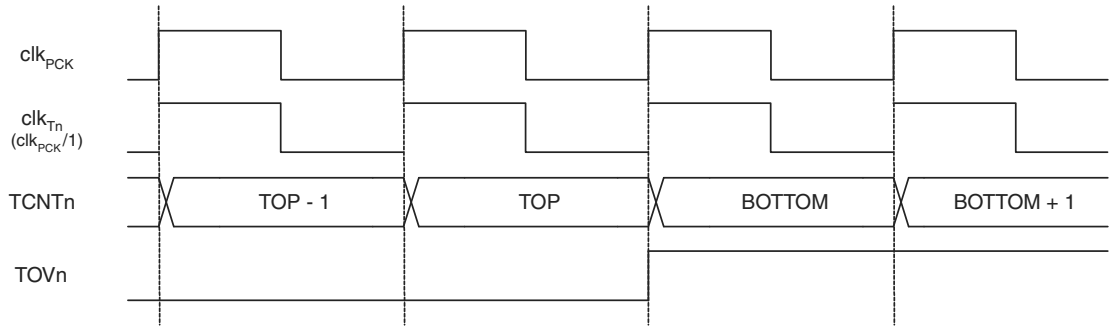


Figure 12-16 shows the same timing data, but with the prescaler enabled, in all modes other than Phase and Frequency Correct PWM Mode.

Figure 12-16. Timer/Counter Timing Diagram, with Prescaler ($f_{clkT1}/8$)

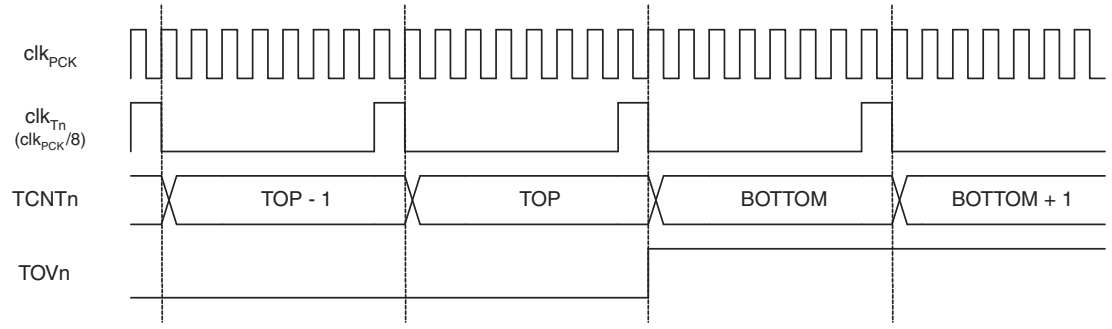


Figure 12-17 shows the setting of OCF1A, OCF1B and OCF1D in all modes.

12.12 Register Description

12.12.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	PWM1A	PWM1B	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7,6 – COM1A1, COM1A0: Comparator A Output Mode, Bits 1 and 0**

These bits control the behaviour of the Waveform Output (OCW1A) and the connection of the Output Compare pin (OC1A). If one or both of the COM1A1:0 bits are set, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. The complementary $\overline{\text{OC1B}}$ output is connected only in PWM modes when the COM1A1:0 bits are set to “01”. Note that the Data Direction Register (DDR) bit corresponding to the OC1A and $\overline{\text{OC1A}}$ pins must be set in order to enable the output driver.

The function of the COM1A1:0 bits depends on the PWM1A, WGM10 and WGM11 bit settings. Table 12-8 shows the COM1A1:0 bit functionality when the PWM1A bit is set to Normal Mode (non-PWM).

Table 12-8. Compare Output Mode, Normal Mode (non-PWM)

COM1A1:0	OCW1A Behaviour	OC1A Pin	$\overline{\text{OC1A}}$ Pin
00	Normal port operation.	Disconnected	Disconnected
01	Toggle on Compare Match.	Connected	Disconnected
10	Clear on Compare Match.	Connected	Disconnected
11	Set on Compare Match.	Connected	Disconnected

Table 12-9 shows the COM1A1:0 bit functionality when the PWM1A, WGM10 and WGM11 bits are set to fast PWM mode.

Table 12-9. Compare Output Mode, Fast PWM Mode

COM1A1:0	OCW1A Behaviour	OC1A	OC1A
00	Normal port operation.	Disconnected	Disconnected
01	Cleared on Compare Match. Set when TCNT1 = 0x000.	Connected	Connected
10	Cleared on Compare Match. Set when TCNT1 = 0x000.	Connected	Disconnected
11	Set on Compare Match. Cleared when TCNT1 = 0x000.	Connected	Disconnected

Table 14-1. Analog Comparator Multiplexed Input (Continued)

ACME	ADEN	MUX5:0	ACM2:0	Positive Input	Negative Input
0	x	xxxxxx	100	AIN2	AIN0
0	x	xxxxxx	101,110,111	AIN2	AIN1
1	1	xxxxxx	000	AIN0	AIN1
1	0	000000	000	AIN0	ADC0
1	0	000000	01x	AIN1	ADC0
1	0	000000	1xx	AIN2	ADC0
1	0	000001	000	AIN0	ADC1
1	0	000001	01x	AIN1	ADC1
1	0	000001	1xx	AIN2	ADC1
1	0	000010	000	AIN0	ADC2
1	0	000010	01x	AIN1	ADC2
1	0	000010	1xx	AIN2	ADC2
1	0	000011	000	AIN0	ADC3
1	0	000011	01x	AIN1	ADC3
1	0	000011	1xx	AIN2	ADC3
1	0	000100	000	AIN0	ADC4
1	0	000100	01x	AIN1	ADC4
1	0	000100	1xx	AIN2	ADC4
1	0	000101	000	AIN0	ADC5
1	0	000101	01x	AIN1	ADC5
1	0	000101	1xx	AIN2	ADC5
1	0	000110	000	AIN0	ADC6
1	0	000110	01x	AIN1	ADC6
1	0	000110	1xx	AIN2	ADC6
1	0	000111	000	AIN0	ADC7
1	0	000111	01x	AIN1	ADC7
1	0	000111	1xx	AIN2	ADC7
1	0	001000	000	AIN0	ADC8
1	0	001000	01x	AIN1	ADC8
1	0	001000	1xx	AIN2	ADC8
1	0	001001	000	AIN0	ADC9
1	0	001001	01x	AIN1	ADC9
1	0	001001	1xx	AIN2	ADC9
1	0	001010	000	AIN0	ADC10
1	0	001010	01x	AIN1	ADC10
1	0	001010	1xx	AIN2	ADC10

14.2 Register Description

14.2.1 ACSRA – Analog Comparator Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	ACD	ACBG	ACO	ACI	ACIE	ACME	ACIS1	ACIS0	ACSRA
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, thus reducing 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 ACSRA. Otherwise an interrupt can occur when the bit is changed.

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

When this bit is set an internal 1.1V reference voltage replaces the positive input to the analog comparator. The selection of the internal voltage reference is done by writing the REFS2:0 bits in ADCSRB and ADMUX registers. When this bit is cleared, AIN0, AIN1 or AIN2 depending on the ACM2:0 bits is applied to the positive input of the analog comparator.

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

Enables output of analog comparator. 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 – 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 Table 14-1 on page 137.

Special care should be taken when changing differential channels. Once a differential channel has been selected the input stage may take a while to stabilize. It is therefore recommended to force the ADC to perform a long conversion when changing multiplexer or voltage reference settings. This can be done by first turning off the ADC, then changing reference settings and then turn on the ADC. Alternatively, the first conversion results after changing reference settings should be discarded.

It is not recommended to use an external AREF higher than (V_{CC} - 1V) for channels with differential gain, as this will affect ADC accuracy.

Internal voltage reference options may not be used if an external voltage is being applied to the AREF pin.

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a comple te description of this bit, see “ADCL and ADCH – The ADC Data Register” on page 160.

- **Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits**

These bits and the MUX5 bit from the ADC Control and Status Register B (ADCSRB) select which combination of analog inputs are connected to the ADC. In case of differential input, gain selection is also made with these bits. Selecting the same pin as both inputs to the differential

15.13.4 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	**BIN**	**GSEL**	**-**	**REFS2**	**MUX5**	**ADTS2**	**ADTS1**	**ADTS0**	ADCSRB
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – BIN: Bipolar Input Mode**

The gain stage is working in the unipolar mode as default, but the bipolar mode can be selected by writing the BIN bit in the ADCSRB register. In the unipolar mode only one-sided conversions are supported and the voltage on the positive input must always be larger than the voltage on the negative input. Otherwise the result is saturated to the voltage reference. In the bipolar mode two-sided conversions are supported and the result is represented in the two's complement form. In the unipolar mode the resolution is 10 bits and the bipolar mode the resolution is 9 bits + 1 sign bit.

- **Bit 6 – GSEL: Gain Select**

The Gain Select bit selects the 32x gain instead of the 20x gain and the 8x gain instead of the 1x gain when the Gain Select bit is written to one.

- **Bit 5 – Res: Reserved Bit**

This bit is a reserv ed bit in the ATtiny261/461/861 and will always read as zero.

- **Bit 4 – REFS2: Reference Selection Bit**

These bit selects either the voltage reference of 1.1 V or 2.56 V for the ADC, as shown in Table 15-3. If active channels are used, using AVCC or an external AREF higher than (AVCC - 1V) is not recommended, as this will affect ADC accuracy. The internal voltage reference options may not be used if an external voltage is being applied to the AREF pin.

- **Bit 3 – MUX5: Analog Channel and Gain Selection Bit 5**

The MUX5 bit is the MSB of the Analog Channel and Gain Selection bits. Refer to Table 15-4 for details. If this bit is changed during a conversion, the change will not go into effect until this conversion is complete (ADIF in ADCSRA is set).

- **Bits 2:0 – ADTS2:0: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 15-6. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow

When designing a system where debugWIRE will be used, the following must be observed:

- Pull-Up resistor on the dW/(RESET) line must be in the range of 10k to 20 kΩ. However, the pull-up resistor is optional.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors inserted on the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

16.4 Software Break Points

debugWIRE supports Program memory Break Points by the AVR Break instruction. Setting a Break Point in AVR Studio® will insert a BREAK instruction in the Program memory. The instruction replaced by the BREAK instruction will be stored. When program execution is continued, the stored instruction will be executed before continuing from the Program memory. A break can be inserted manually by putting the BREAK instruction in the program.

The Flash must be re-programmed each time a Break Point is changed. This is automatically handled by AVR Studio through the debugWIRE interface. The use of Break Points will therefore reduce the Falsh Data retention. Devices used for debugging purposes should not be shipped to end customers.

16.5 Limitations of debugWIRE

The debugWIRE communication pin (dW) is physically located on the same pin as External Reset (RESET). An External Reset source is therefore not supported when the debugWIRE is enabled.

The debugWIRE system accurately emulates all I/O functions when running at full speed, i.e., when the program in the CPU is running. When the CPU is stopped, care must be taken while accessing some of the I/O Registers via the debugger (AVR Studio). See the debugWIRE documentation for detailed description of the limitations.

The debugWIRE interface is asynchronous, which means that the debugger needs to synchronize to the system clock. If the system clock is changed by software (e.g. by writing CLKPS bits) communication via debugWIRE may fail. Also, clock frequencies below 100 kHz may cause communication problems.

A programmed DWEN Fuse enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption while in sleep. Thus, the DWEN Fuse should be disabled when debugWire is not used.

16.6 Register Description

The following section describes the registers used with the debugWire.

16.6.1 DWDR – debugWire Data Register

Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	DWDR[7:0]								DWDR
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 DWDR Register provides a communication channel from the running program in the MCU to the debugger. This register is only accessible by the debugWIRE and can therefore not be used as a general purpose register in the normal operations.

If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded will be lost.

17.3 Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write “00000101” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

Note: The CPU is halted during the Page Write operation.

17.4 Addressing the Flash During Self-Programming

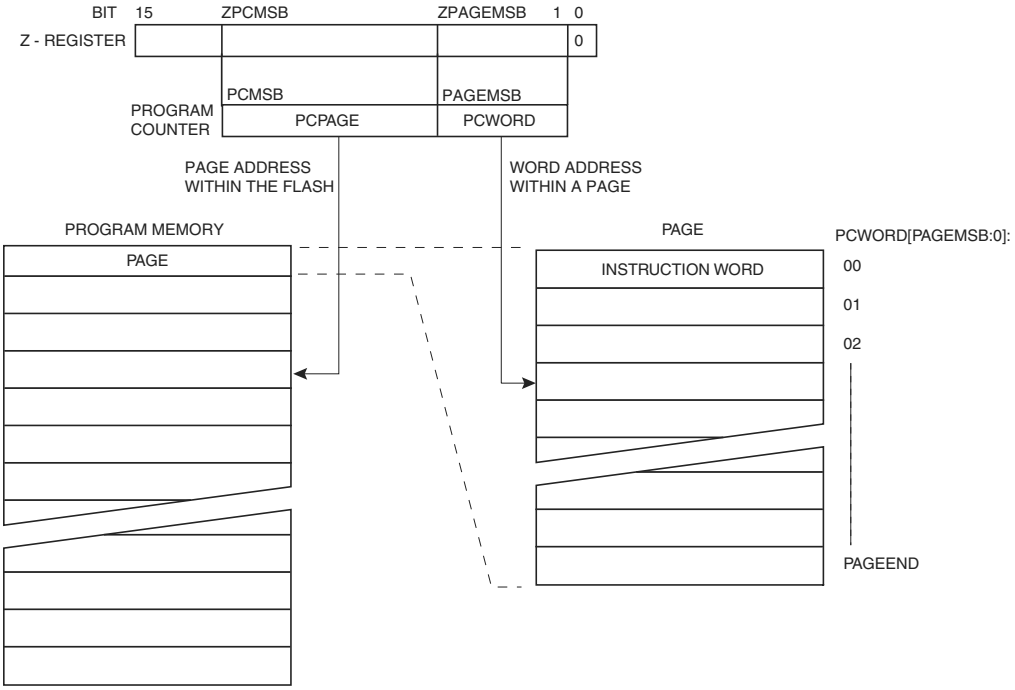
The Z-pointer is used to address the SPM commands.

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 18-7 on page 173), 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 17-1. Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the software addresses the same page in both the Page Erase and Page Write operation.

The LPM instruction uses 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 17-1. Addressing the Flash During SPM



Note: The different variables used in Figure 17-1 are listed in Table 18-7 on page 173.

18.7.8 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to “Programming the Flash” on page 180 for details on Command and Address loading):

1. A: Load Command “0000 0011”.
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. Set $\overline{OE}$ to “0”, and BS1 to “0”. The EEPROM Data byte can now be read at DATA.
5. Set $\overline{OE}$ to “1”.

18.7.9 Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to “Programming the Flash” on page 180 for details on Command and Data loading):

1. A: Load Command “0100 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

18.7.10 Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to “Programming the Flash” on page 180 for details on Command and Data loading):

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

18.7.11 Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (refer to “Programming the Flash” on page 180 for details on Command and Data loading):

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