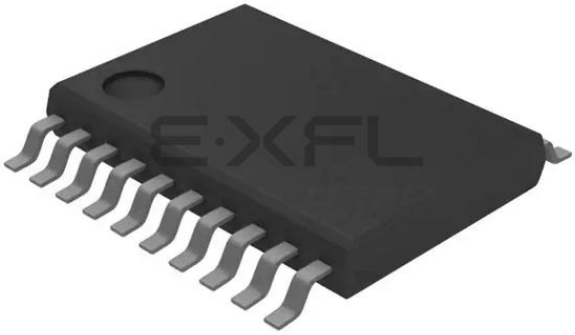


Welcome to E-XFL.COM

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	USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	2KB (1K x 16)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 150°C (TA)
Mounting Type	Surface Mount
Package / Case	20-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	20-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny261-15xd

The fast-access register file contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle arithmetic logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the register file, the operation is executed, and the result is stored back in the register file – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for data space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in flash program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the status register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR® instructions have a single 16-bit word format. Most AVR instructions are 16-bit wide. There are also 32-bit instructions.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The stack pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the status register. All interrupts have a separate interrupt vector in the interrupt vector table. The interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as control registers, SPI, and other I/O functions. The I/O memory can be accessed directly, or as the data space locations following those of the register file, 0x20 - 0x5F.

5.2 ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

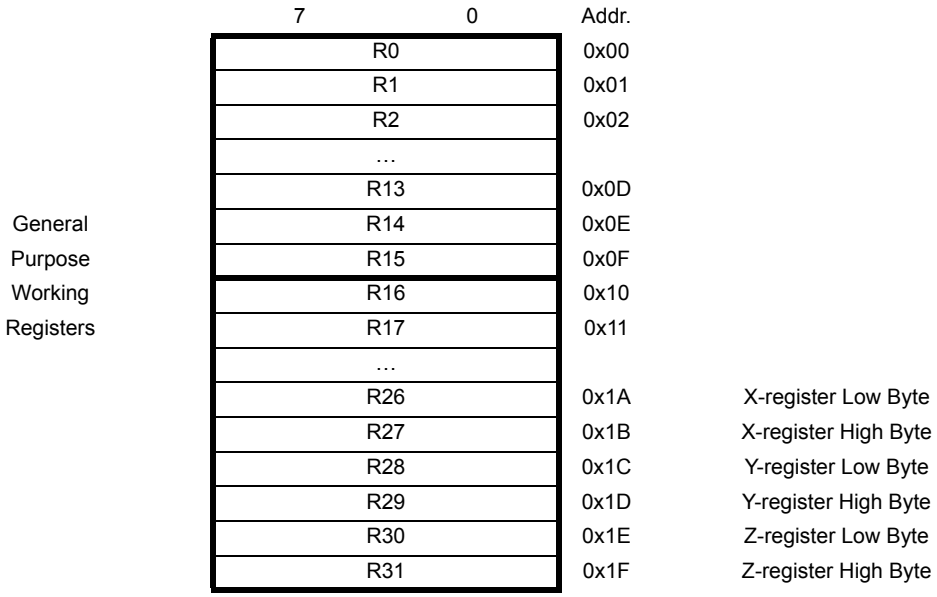
5.4 General Purpose Register File

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

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

Figure 5-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 5-2. AVR CPU General Purpose Working Registers



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

As shown in Figure 5-2, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user data space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y- and Z-pointer registers can be set to index any register in the file.

The CKDIV8 fuse determines the initial value of the CLKPS bits. If CKDIV8 is unprogrammed, the CLKPS bits will be reset to “0000”. If CKDIV8 is programmed, CLKPS bits are reset to “0011”, giving a division factor of eight at start up. This feature should be used if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. Note that any value can be written to the CLKPS bits regardless of the CKDIV8 fuse setting. The application software must ensure that a sufficient division factor is chosen if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. The device is shipped with the CKDIV8 fuse programmed.

Table 7-12. Clock Prescaler Select

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock Division Factor
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256
1	0	0	1	Reserved
1	0	1	0	Reserved
1	0	1	1	Reserved
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

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 Atmel® ATtiny261/461/861 is:

```
Address      Labels Code                Comments
0x0000              rjmp   RESET        ; Reset Handler
0x0001              rjmp   EXT_INT0     ; IRQ0 Handler
0x0002              rjmp   PCINT        ; PCINT Handler
0x0003              rjmp   TIM1_COMPA   ; Timer1 CompareA Handler
0x0004              rjmp   TIM1_COMPB   ; Timer1 CompareB Handler
0x0005              rjmp   TIM1_OVF     ; Timer1 Overflow Handler
0x0006              rjmp   TIM0_OVF     ; Timer0 Overflow Handler
0x0007              rjmp   USI_START    ; USI Start Handler
0x0008              rjmp   USI_OVF      ; USI Overflow Handler
0x0009              rjmp   EE_RDY       ; EEPROM Ready Handler
0x000A              rjmp   ANA_COMP     ; Analog Comparator Handler
0x000B              rjmp   ADC          ; ADC Conversion Handler
0x000C              rjmp   WDT          ; WDT Interrupt Handler
0x000D              rjmp   EXT_INT1     ; IRQ1 Handler
0x000E              rjmp   TIM0_COMPA   ; Timer0 CompareA Handler
0x000F              rjmp   TIM0_COMPB   ; Timer0 CompareB Handler
0x0010              rjmp   TIM0_CAPT    ; Timer0 Capture Event Handler
0x0011              rjmp   TIM1_COMPD   ; Timer1 CompareD Handler
0x0012              rjmp   FAULT_PROTECTION; Timer1 Fault Protection
0x0013       RESET: ldi    r16, low(RAMEND); Main program start
0x0014              ldi    r17, high(RAMEND); Tiny861 have also SPH
0x0015              out    SPL, r16     ; Set Stack Pointer to top of RAM
0x0016              out    SPH, r17     ; Tiny861 have also SPH
0x0017              sei                 ; Enable interrupts
0x0018       <instr> xxx
...    ...    ...    ...
```

- **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 bits in the Atmel® ATtiny261/461/861 and will always read as zero.

11.1.4 PCMSK0 – Pin Change Mask Register A

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	**PCINT7**	**PCINT6**	**PCINT5**	**PCINT4**	**PCINT3**	**PCINT2**	**PCINT1**	**PCINT0**	**PCMSK0**
Read/Write	R/W	R/W	R/W	R/w	R/W	R/W	R/W	R/W	
Initial Value	1	1	0	0	1	0	0	0	

- **Bits 7:0 – PCINT7:0: Pin Change Enable Mask 7..0**

Each PCINT7:0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7:0 is set and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

11.1.5 PCMSK1 – Pin Change Mask Register B

Bit	7	6	5	4	3	2	1	0	
0x22 (0x42)	**PCINT15**	**PCINT14**	**PCINT13**	**PCINT12**	**PCINT11**	**PCINT10**	**PCINT9**	**PCINT8**	**PCMSK1**
Read/Write	R/W	R/W	R/W	R/w	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

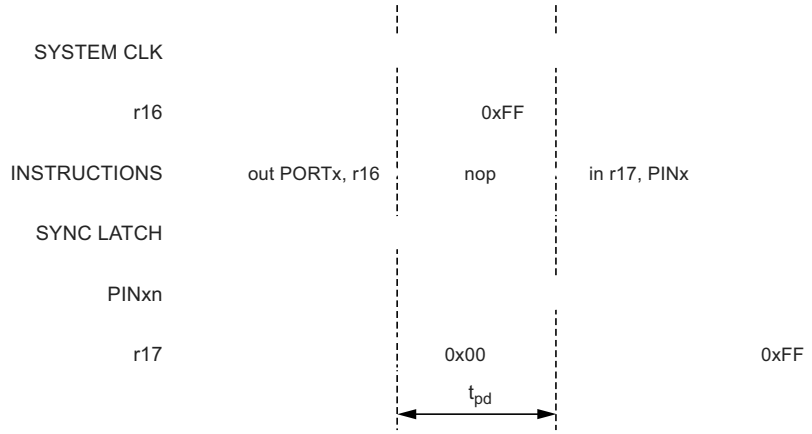
- **Bits 7:0 – PCINT15:8: Pin Change Enable Mask 15..8**

Each PCINT15:8 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT11:8 is set and the PCIE0 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin, and if PCINT15:12 is set and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT15:8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

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 12-4. 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 one system clock period.

Figure 12-4. Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 5 as input with a pull-up assigned to port pin 4. The resulting pin values are read back again, but as previously discussed, a nop instruction is included to be able to read back the value recently assigned to some of the pins.

Assembly Code Example[1]

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi    r16,(1<<PB4)|(1<<PB1)|(1<<PB0)
ldi    r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out    PORTB,r16
out    DDRB,r17
; Insert nop for synchronization
nop
; Read port pins
in     r16,PINB
...
```

C Code Example

```
unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTB = (1<<PB4)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization*/
_NOP();
/* Read port pins */
i = PINB;
...
```

Note: 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1 and 4, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

12.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 12-2 on page 53, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU sleep controller in power-down mode, power-save mode, and standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in Section 12.3 “Alternate Port Functions” on page 57.

If a logic high level (“one”) is present on an asynchronous external interrupt pin configured as “interrupt on rising edge, falling edge, or any logic change on pin” while the external interrupt is not enabled, the corresponding external interrupt flag will be set when resuming from the above mentioned Sleep mode, as the clamping in these sleep mode produces the requested logic change.

12.2.6 Unconnected Pins

If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (reset, active mode and idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

Table 12-8. Overriding Signals for Alternate Functions in PA3..PA0

Signal Name	PA3/AREF/ PCINT3	PA2/ADC2/INT1/ USCK/SCL/PCINT2	PA1/ADC1/DO/ PCINT1	PA0/ADC0/DI/SDA/ PCINT0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	USI_TWO_WIRE × USIPOS	0	USI_TWO_WIRE × USIPOS
DDOV	0	(USI_SCL_HOLD + $\overline{\text{PORTB2}}$) × DDB2 × USIPOS	0	($\overline{\text{SDA}}$ + $\overline{\text{PORTB0}}$) × DDRB0 × USIPOS
PVOE	0	USI_TWO_WIRE × DDRB2	USI_THREE_WIRE × USIPOS	USI_TWO_WIRE × DDRB0 × USIPOS
PVOV	0	0	DO × USIPOS	0
PTOE	0	USI_PTOE × USIPOS	0	0
DIEOE	PCINT3 × PCIE	PCINT2 × PCIE + INT1 + ADC2D + USISIE × USIPOS	PCINT1 × PCIE + ADC1D	PCINT0 × PCIE + ADC0D + USISIE × USIPOS
DIEOV	0	ADC2D	ADC1D	ADC0D
DI	PCINT3	USCK/SCL/INT1/ PCINT2	PCINT1	DI/SDA/PCINT0
AIO	AREF	ADC2	ADC1	ADC0

12.4 Register Description

12.4.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	**-**	**PUD**	**SE**	**SM1**	**SM0**	**-**	**ISC01**	**ISC00**	**MCUCR**
Read/Write	R	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – 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 12.2.1 “Configuring the Pin” on page 53 for more details about this feature.

12.4.2 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	**PORTA7**	**PORTA6**	**PORTA5**	**PORTA4**	**PORTA3**	**PORTA2**	**PORTA1**	**PORTA0**	**PORTA**
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	

12.4.3 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x1A (0x3A)	**DDA7**	**DDA6**	**DDA5**	**DDA4**	**DDA3**	**DDA2**	**DDA1**	**DDA0**	**DDRA**
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	

12.4.4 PINA – Port A Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x19 (0x39)	**PINA7**	**PINA6**	**PINA5**	**PINA4**	**PINA3**	**PINA2**	**PINA1**	**PINA0**	**PINA**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

12.4.5 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x18 (0x38)	**PORTB7**	**PORTB6**	**PORTB5**	**PORTB4**	**PORTB3**	**PORTB2**	**PORTB1**	**PORTB0**	**PORTB**
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	

12.4.6 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	**DDB7**	**DDB6**	**DDB5**	**DDB4**	**DDB3**	**DDB2**	**DDB1**	**DDB0**	**DDRB**
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	

12.4.7 PINB – Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	**PINB7**	**PINB6**	**PINB5**	**PINB4**	**PINB3**	**PINB2**	**PINB1**	**PINB0**	**PINB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

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 OCR0A/B registers.

Assembly Code Example

```
...
; Set TCNT0 to 0x01FF
ldi    r17,0x01
ldi    r16,0xFF
out    TCNT0H,r17
out    TCNT0L,r16
; Read TCNT0 into r17:r16
in     r16,TCNT0L
in     r17,TCNT0H
...
```

C Code Example

```
unsigned int i;
...
/* Set TCNT0 to 0x01FF */
TCNT0H = 0x01;
TCNT0L = 0xff;

/* Read TCNT0 into i */
i = TCNT0L;
i |= ((unsigned int)TCNT0H << 8);
...
```

Note: 1. The example code assumes that the part specific header file is included.
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

The assembly code example returns the TCNT0H/L 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 main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

14.10.6 TIMSK – Timer/Counter0 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	**OCIE1D**	**OCIE1A**	**OCIE1B**	**OCIE0A**	**OCIE0B**	**TOIE1**	**TOIE0**	**TICIE0**	**TIMSK**
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 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 compare match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/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 Section 10. “Interrupts” on page 47) is executed when the ICF0 flag, located in TIFR, is set.

Figure 16-9. Compare Match Output Unit, Schematic

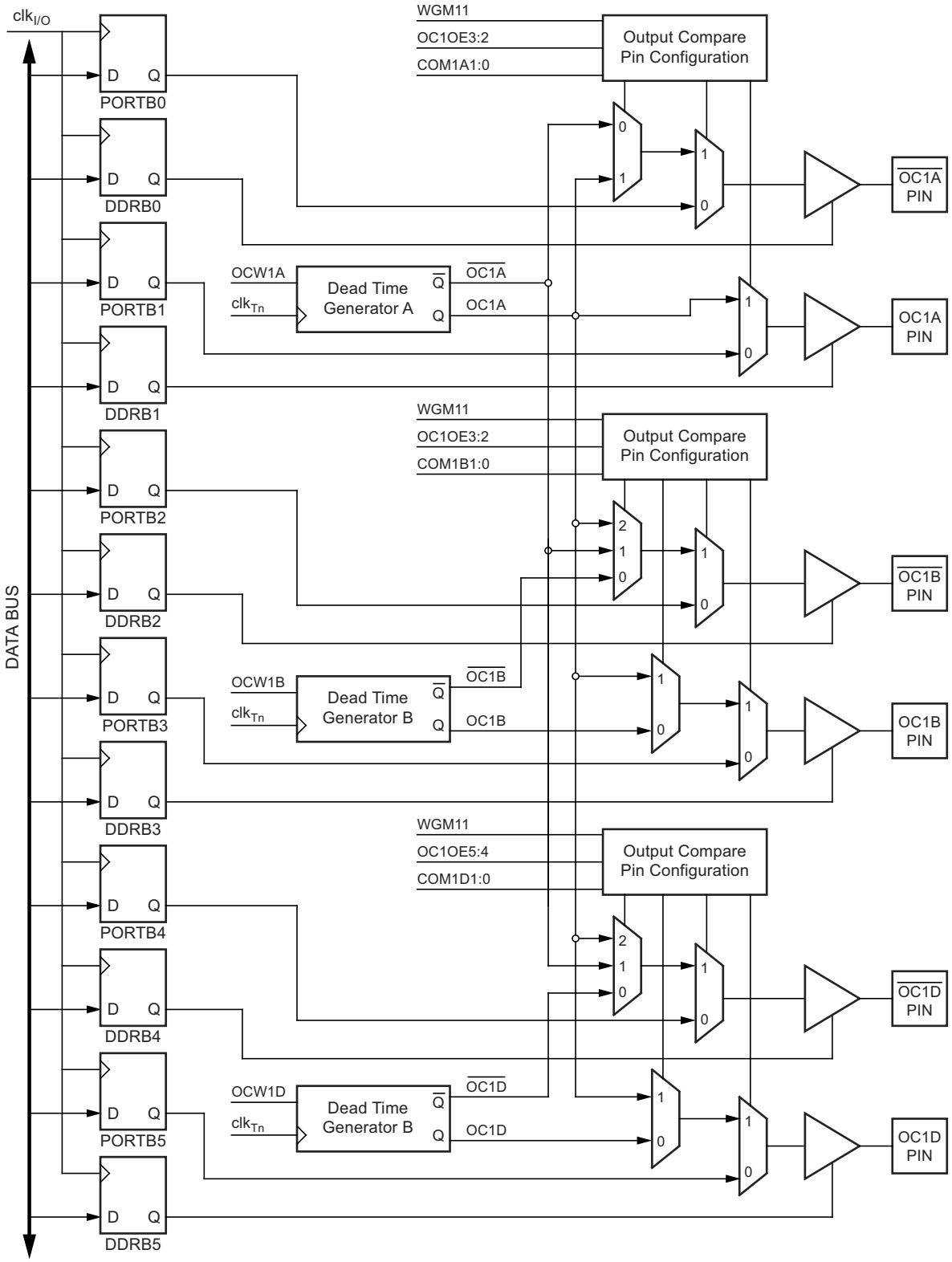


Table 16-8 shows the COM1A1:0 bit functionality when the PWM1A, WGM10 and WGM11 bits are set to Phase and Frequency Correct PWM Mode.

Table 16-8. Compare Output Mode, Phase and Frequency Correct PWM Mode

COM1A1..0	OCW1A Behavior	OC1A Pin	$\overline{\text{OC1A}}$ Pin
00	Normal port operation.	Disconnected	Disconnected
01	Cleared on compare match when up-counting Set on compare match when down-counting	Connected	Connected
10	Cleared on compare match when up-counting Set on compare match when down-counting	Connected	Disconnected
11	Set on compare match when up-counting Cleared on compare match when down-counting	Connected	Disconnected

Table 16-9 shows the COM1A1:0 bit functionality when the PWM1A, WGM10 and WGM11 bits are set to single-slope PWM6 Mode. In the PWM6 Mode the same waveform output (OCW1A) is used for generating all waveforms and the output compare values OC1A and $\overline{\text{OC1A}}$ are connected on the all OC1x and $\overline{\text{OC1x}}$ pins as described below.

Table 16-9. Compare Output Mode, Single-Slope PWM6 Mode

COM1A1..0	OCW1A Behavior	OC1x Pin	$\overline{\text{OC1x}}$ Pin
00	Normal port operation	Disconnected	Disconnected
01	Cleared on compare match Set when TCNT1 = 0x000	OC1A	OC1A
10	Cleared on compare match Set when TCNT1 = 0x000	OC1A	OC1A
11	Set on compare match Cleared when TCNT1 = 0x000	OC1A	OC1A

Table 16-10 shows the COM1A1:0 bit functionality when the PWM1A, WGM10 and WGM11 bits are set to dual-slope PWM6 Mode.

Table 16-10. Compare Output Mode, Dual-Slope PWM6 Mode

COM1A1..0	OCW1A Behavior	OC1x Pin	$\overline{\text{OC1x}}$ Pin
00	Normal port operation	Disconnected	Disconnected
01	Cleared on compare match when up-counting Set on compare match when down-counting	OC1A	OC1A
10	Cleared on compare match when up-counting Set on compare match when down-counting	OC1A	OC1A
11	Set on compare match when up-counting Cleared on compare match when down-counting	OC1A	OC1A

- **Bits 5,4 - COM1B1, COM1B0: Comparator B Output Mode, Bits 1 and 0**

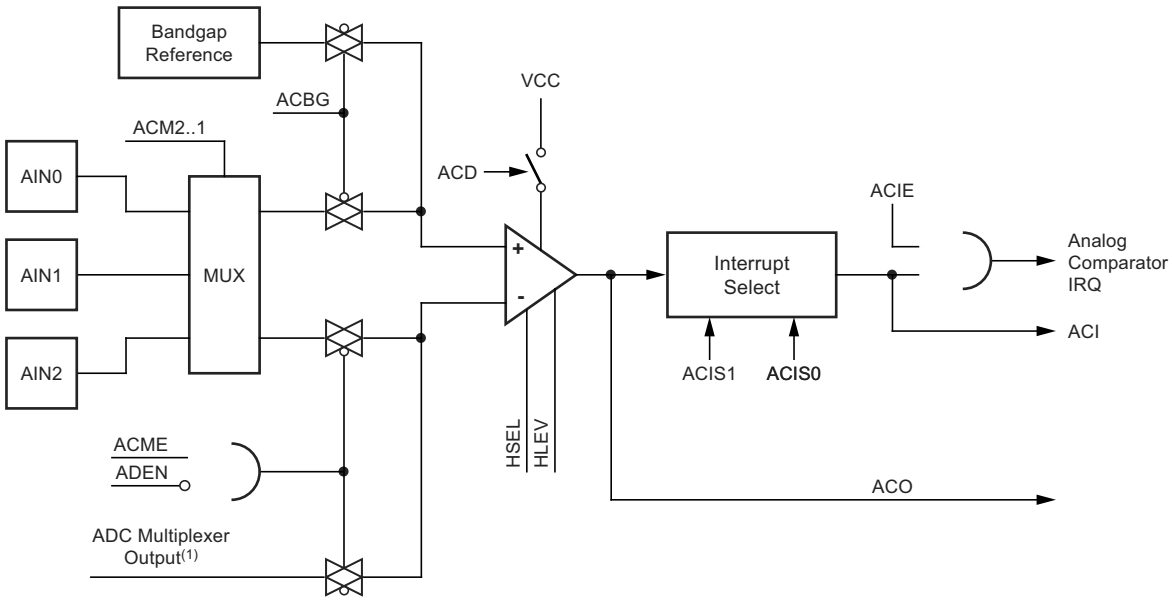
These bits control the behavior of the waveform output (OCW1B) and the connection of the output compare pin (OC1B). If one or both of the COM1B1:0 bits are set, the OC1B output overrides the normal port functionality of the I/O pin it is connected to. The complementary $\overline{\text{OC1B}}$ output is connected only in PWM modes when the COM1B1:0 bits are set to “01”.

Note that the data direction register (DDR) bit corresponding to the OC1B pin must be set in order to enable the output driver.

18. AC – Analog Comparator

The analog comparator compares the input values on the selectable positive pin (AIN0, AIN1 or AIN2) and selectable negative pin (AIN0, AIN1 or AIN2). When the voltage on the positive pin is higher than the voltage on the negative pin, the analog comparator output, ACO, is set. The comparator can trigger a separate interrupt, exclusive to the analog comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 18-1.

Figure 18-1. Analog Comparator Block Diagram(2)



Notes: 1. See Table 18-2 on page 130.

2. Refer to Figure 1-1 on page 3 and Table 12.3.2 on page 62 for analog comparator pin placement.

18.1 Register Description

18.1.1 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. 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 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 ADMUX register. 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**

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

Table 18-2. Analog Comparator Multiplexed Input (Continued)

ACME	ADEN	MUX5..0	ACM2..0	Positive Input	Negative Input
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

18.2.1 ACSRB – Analog Comparator Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	**HSEL**	**HLEV**	**-**	**-**	**-**	**ACM2**	**ACM1**	**ACM0**	**ACSRB**
Read/Write	R/W	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

- **Bit 7 – HSEL: Hysteresis Select**

When this bit is written logic one, the hysteresis of the analog comparator is switched on. The hysteresis level is selected by the HLEV bit.

- **Bit 6 – HLEV: Hysteresis Level**

When the hysteresis is enabled by the HSEL bit, the hysteresis level, HLEV, bit selects the hysteresis level that is either 20mV (HLEV=0) or 50mV (HLEV=1).

- **Bit 2:0 – ACM2:ACM0: Analog Comparator Multiplexer**

The analog comparator multiplexer bits select the positive and negative input pins of the analog comparator. The different settings are shown in Table 18-2.

19.3 Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on V_{CC}, the voltage on the AREF pin or an internal 1.1V/2.56V voltage reference.

The voltage reference for the ADC may be selected by writing to the REFS2..0 bits in ADMUX. The VCC supply, the AREF pin or an internal 1.1V / 2.56V voltage reference may be selected as the ADC voltage reference. Optionally the internal 1.1V/2.56V voltage reference may be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel and differential gain are selected by writing to the MUX5..0 bits in ADMUX. Any of the 11 ADC input pins ADC10..0 can be selected as single ended inputs to the ADC. The positive and negative inputs to the differential gain amplifier are described in Table 19-5 on page 145.

If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input pair by the selected gain factor, 1x, 8x, 20x or 32x, according to the setting of the MUX5..0 bits in ADMUX and the GSEL bit in ADCSRB. This amplified value then becomes the analog input to the ADC. If single ended channels are used, the gain amplifier is bypassed altogether.

If the same ADC input pin is selected as both the positive and negative input to the differential gain amplifier, the remaining offset in the gain stage and conversion circuitry can be measured directly as the result of the conversion. This figure can be subtracted from subsequent conversions with the same gain setting to reduce offset error to below 1 LSW.

The on-chip temperature sensor is selected by writing the code “111111” to the MUX5..0 bits in ADMUX register when the ADC11 channel is used as an ADC input.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC data registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the data registers belongs to the same conversion. Once ADCL is read, ADC access to data registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL registers is re-enabled.

The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the data registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

19.4 Starting a Conversion

A single conversion is started by writing a logical one to the ADC start conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto triggering is enabled by setting the ADC auto trigger enable bit, ADATE in ADCSRA. The trigger source is selected by setting the ADC trigger select bits, ADTS in ADCSRB (see description of the ADTS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an interrupt flag will be set even if the specific interrupt is disabled or the global interrupt enable bit in SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the interrupt flag must be cleared in order to trigger a new conversion at the next interrupt event.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 14.5 ADC clock cycles after the start of an first conversion. When a conversion is complete, the result is written to the ADC data registers, and ADIF is set. In single conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

When auto triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

In free running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see Table 19-1 on page 137.

Figure 19-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)

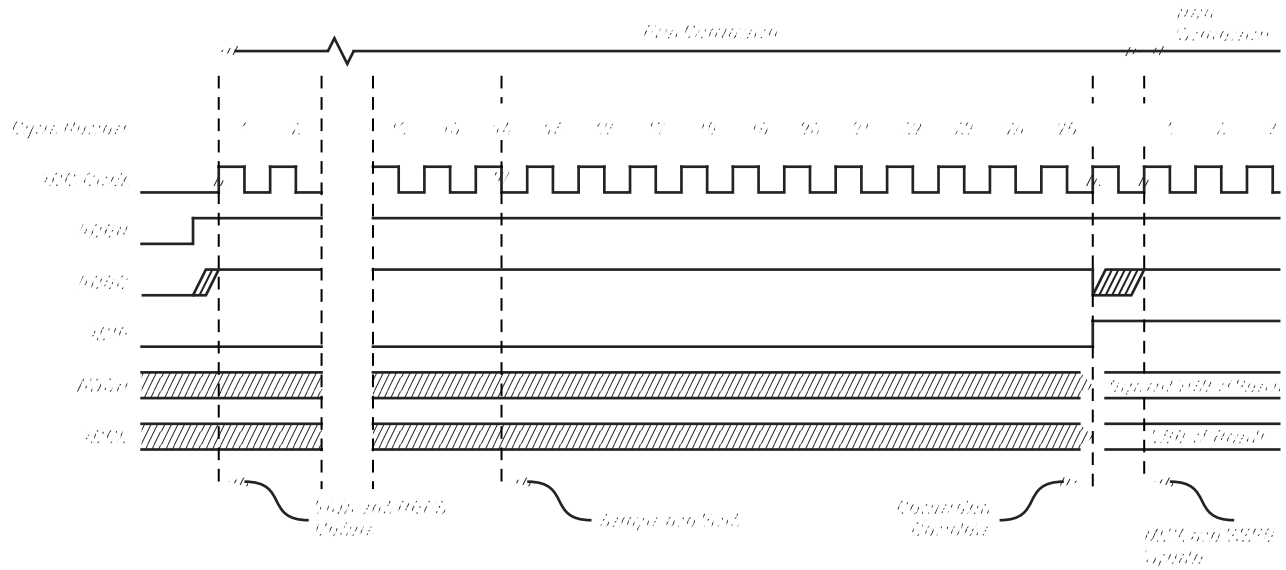


Figure 19-5. ADC Timing Diagram, Single Conversion

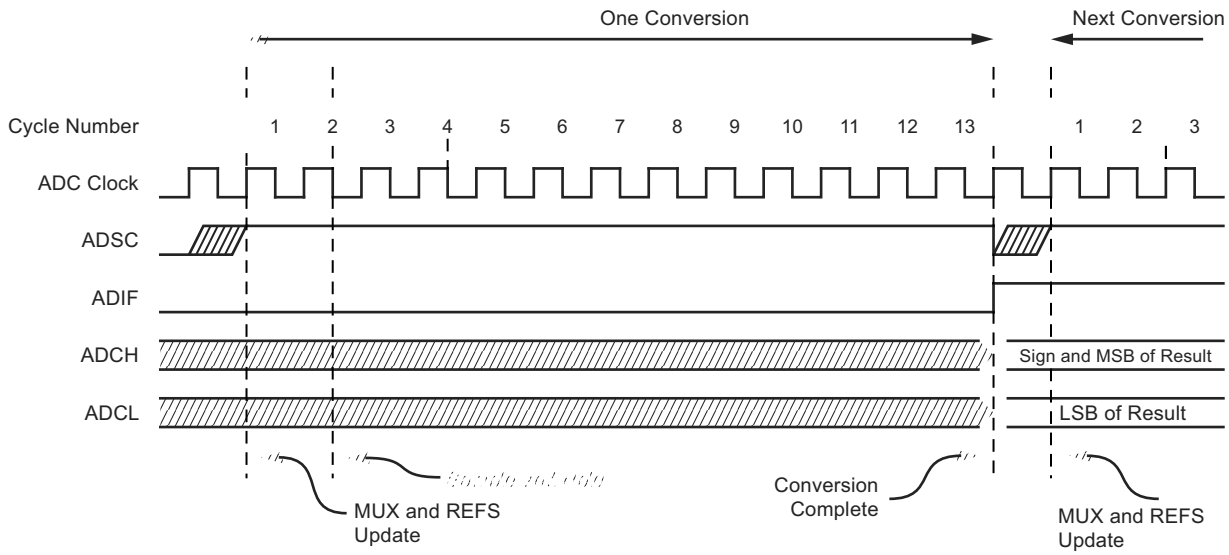


Table 19-5. Input Channel Selections (Continued)

MUX5..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
101000	N/A	ADC2(PA2)	ADC0(PA0)	20x/32x
101001		ADC2(PA2)	ADC0(PA0)	1x/8x
101010		ADC0(PA0)	ADC2(PA2)	20x/32x
101011		ADC0(PA0)	ADC2(PA2)	1x/8x
101100	N/A	ADC4(PA5)	ADC5(PA6)	20x/32x
101101		ADC4(PA5)	ADC5(PA6)	1x/8x
101110		ADC5(PA6)	ADC4(PA5)	20x/32x
101111		ADC5(PA6)	ADC4(PA5)	1x/8x
110000	N/A	ADC5(PA6)	ADC6(PA7)	20x/32x
110001		ADC5(PA6)	ADC6(PA7)	1x/8x
110010		ADC6(PA7)	ADC5(PA6)	20x/32x
110011		ADC6(PA7)	ADC5(PA6)	1x/8x
110100	N/A	ADC6(PA7)	ADC4(PA5)	20x/32x
110101		ADC6(PA7)	ADC4(PA5)	1x/8x
110110		ADC4(PA5)	ADC6(PA7)	20x/32x
110111		ADC4(PA5)	ADC6(PA7)	1x/8x
111000	N/A	ADC0(PA0)	ADC0(PA0)	20x/32x
111001		ADC0(PA0)	ADC0(PA0)	1x/8x
111010		ADC1(PA1)	ADC1(PA1)	20x/32x
111011		ADC2(PA2)	ADC2(PA2)	20x/32x
111100	N/A	ADC4(PA5)	ADC4(PA5)	20x/32x
111101		ADC5(PA6)	ADC5(PA6)	20x/32x
111110		ADC6(PA7)	ADC6(PA7)	20x/32x
111111	ADC11	N/A	N/A	N/A

Note: 1. For temperature sensor

19.10.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	**ADEN**	**ADSC**	**ADATE**	**ADIF**	**ADIE**	**ADPS2**	**ADPS1**	**ADPS0**	**ADCSRA**
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	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In single conversion mode, write this bit to one to start each conversion. In free running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

22.2 Fuse Bytes

The ATtiny261/461/861 has three fuse bytes. Table 22-3, Table 22-4 and Table 22-5 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the fuses are read as logical zero, “0”, if they are programmed.

Table 22-3. Fuse Extended Byte

Fuse High Byte	Bit No	Description	Default Value
	7	-	1 (unprogrammed)
	6	-	1 (unprogrammed)
	5	-	1 (unprogrammed)
	4	-	1 (unprogrammed)
	3	-	1 (unprogrammed)
	2	-	1 (unprogrammed)
	1	-	1 (unprogrammed)
SELFPRGEN	0	Self-programming enable	1 (unprogrammed)

Table 22-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
RSTDISBL(1)	7	External reset disable	1 (unprogrammed)
DWEN(2)	6	DebugWIRE enable	1 (unprogrammed)
SPIEN(3)	6	Enable serial program and data downloading	0 (programmed, SPI prog. enabled)
WDTON(4)	4	Watchdog timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the chip erase	1 (unprogrammed, EEPROM not preserved)
BODLEVEL2(5)	2	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL1(5)	1	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL0(5)	0	Brown-out detector trigger level	1 (unprogrammed)

Notes:
1. See Section 12.3.1 “Alternate Functions of Port B” on page 59 for description of RSTDISBL and DWEN fuses.
2. DWEN must be unprogrammed when lock bit security is required. See Section 22.1 “Program And Data Memory Lock Bits” on page 156.
3. The SPIEN fuse is not accessible in SPI programming mode.
4. See Section 9.10.2 “WDTCR – Watchdog Timer Control Register” on page 44 for details.
5. See Table 23-4 on page 174 for BODLEVEL fuse decoding.
6. When programming the RSTDISBL fuse, high-voltage serial programming has to be used to change fuses to perform further programming.