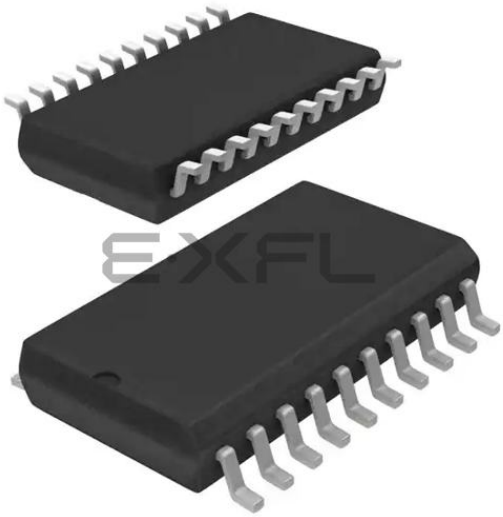


Welcome to E-XFL.COM

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"

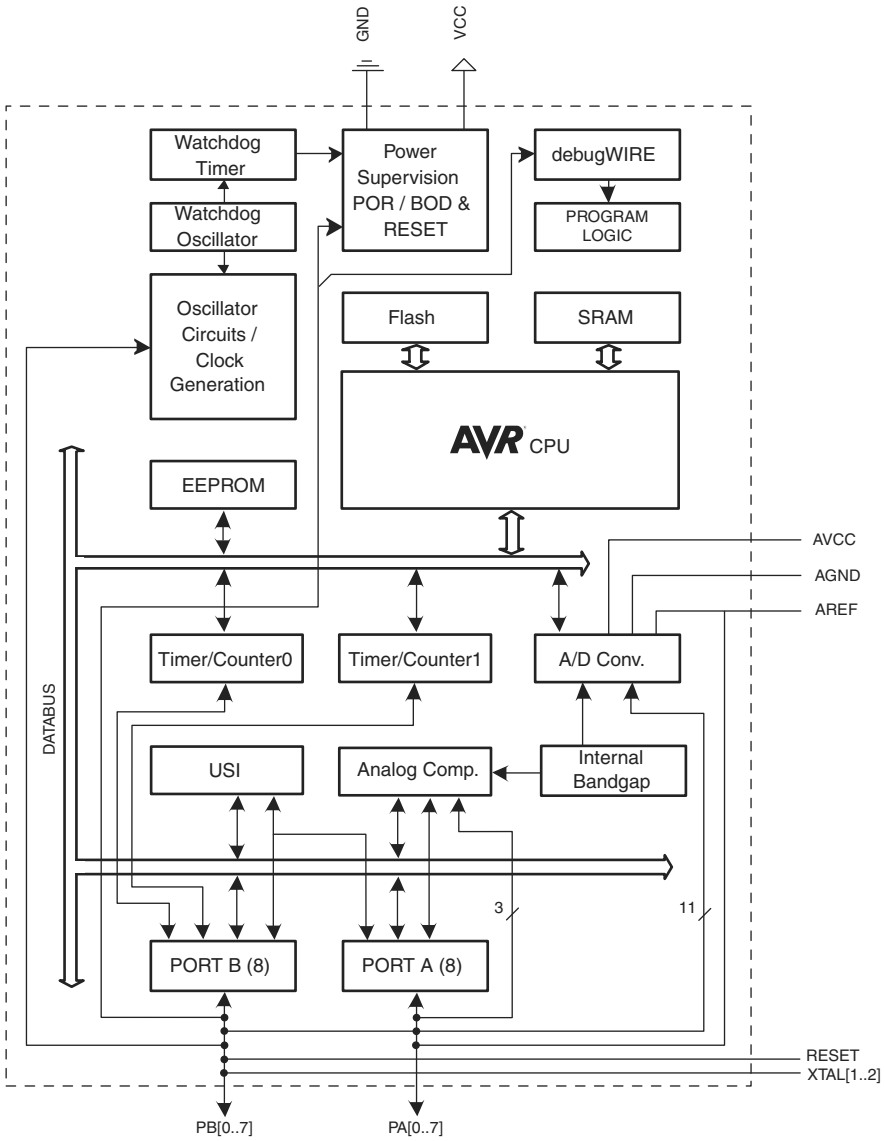
Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	20MHz
Connectivity	USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data 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/attiny861-20su

2. Overview

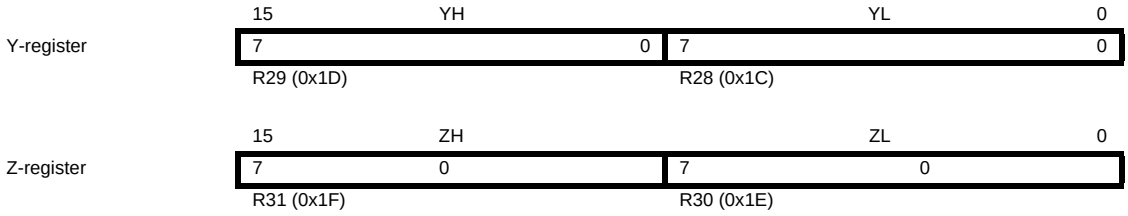
ATtiny261/461/861 are low-power CMOS 8-bit microcontrollers based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATtiny261/461/861 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.



In different addressing modes these address registers function as automatic increment and automatic decrement (see the instruction set reference for details).

4.5 Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above 0x60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present

4.5.1 SPH and SPL — Stack Pointer Register

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	**SP15**	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
0x3D (0x5D)	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	
	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	

4.6 Instruction Execution Timing

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

Figure 4-4 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

old value and program the new value) or to split the Erase and Write operations in two different operations. The Programming times for the different modes are shown in Table 5-1.

Table 5-1. EEPROM Mode Bits

EEPM1	EEPM0	Programming Time	Operation
0	0	3.4 ms	Erase and Write in one operation (Atomic Operation)
0	1	1.8 ms	Erase Only
1	0	1.8 ms	Write Only
1	1	–	Reserved for future use

When EEPE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

Writing EERIE to one enables the EEPROM Ready Interrupt if the I-bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready Interrupt generates a constant interrupt when Non-volatile memory is ready for programming.

- **Bit 2 – EEMPE: EEPROM Master Program Enable**

The EEMPE bit determines whether writing EEPE to one will have effect or not.

When EEMPE is set, setting EEPE within four clock cycles will program the EEPROM at the selected address. If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to one by software, hardware clears the bit to zero after four clock cycles.

- **Bit 1 – EEPE: EEPROM Program Enable**

The EEPROM Program Enable Signal EEPE is the programming enable signal to the EEPROM. When EEPE is written, the EEPROM will be programmed according to the EEPMn bits setting. The EEMPE bit must be written to one before a logical one is written to EEPE, otherwise no EEPROM write takes place. When the write access time has elapsed, the EEPE bit is cleared by hardware. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM Read Enable Signal – EERE – is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed. The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

5.5.5 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x0C (0x2C)	MSB							LSB	GPIOR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

7.1.2 ADC Noise Reduction Mode

When the SM1:0 bits are written to 01, the SLEEP instruction makes the MCU enter ADC Noise Reduction mode, stopping the CPU but allowing the ADC, the external interrupts, and the Watchdog to continue operating (if enabled). This sleep mode halts $clk_{I/O}$, clk_{CPU}, and clk_{FLASH}, while allowing the other clocks to run.

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

7.1.3 Power-Down Mode

When the SM1:0 bits are written to 10, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the Oscillator is stopped, while the external interrupts, and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, an external level interrupt on INT0, or a pin change interrupt can wake up the MCU. This sleep mode halts all generated clocks, allowing operation of asynchronous modules, only.

7.1.4 Standby Mode

When the SM1:0 bits are written to 11 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

7.2 Power Reduction Register

The Power Reduction Register (PRR), see “PRR – Power Reduction Register” on page 39, provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the bit in PRR, puts the module in the same state as before shutdown.

Module shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. In all other sleep modes, the clock is already stopped. See “Supply Current of I/O modules” on page 197 for examples. In all other sleep modes, the clock is already stopped.

7.3 Minimizing Power Consumption

There are several issues to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device’s functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

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
```

```
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>
...            ...
```

9.2 External Interrupts

The External Interrupts are triggered by the INT0 or INT1 pin or any of the PCINT15:0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0, INT1 or PCINT15:0 pins are configured as outputs. This feature provides a way of generating a software interrupt. Pin change interrupts PCI will trigger if any enabled PCINT15:0 pin toggles. The PCMSK Register control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT15:0 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

The INT0 and INT1 interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU Control Register – MCUCR. When the INT0 interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 or INT1 requires the presence of an I/O clock, described in “Clock Subsystems” on page 24.

9.2.1 Low Level Interrupt

A low level interrupt on INT0 is detected asynchronously. This means that the interrupt source can be used for waking the part also from sleep modes other than Idle (the I/O clock is halted in all sleep modes except Idle).

Note that if a level triggered interrupt is used for wake-up from Power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no inter-

Table 10-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 10-5 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 10-2. Generic Description of Overriding Signals for Alternate Functions

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

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

10.2.2 Alternate Functions of Port B

The Port B pins with alternate function are shown in Table 10-6.

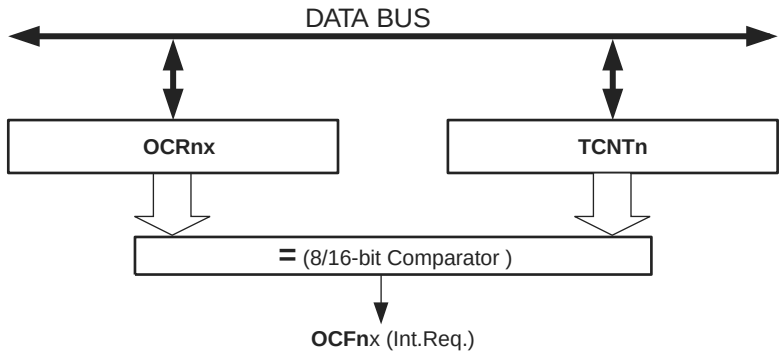
Table 10-6. Port B Pins Alternate Functions

Port Pin	Alternate Function
PB7	$\overline{\text{RESET}}$: Reset pin dW: debugWire I/O ADC10: ADC Input Channel 10 PCINT15:Pin Change Interrupt 0, Source 15
PB6	ADC9: ADC Input Channel 9 T0: Timer/Counter0 Clock Source INT0: External Interrupt 0 Input PCINT14:Pin Change Interrupt 0, Source 14
PB5	XTAL2: Crystal Oscillator Output CLKO: System Clock Output OC1D: Timer/Counter1 Compare Match D Output ADC8: ADC Input Channel 8 PCINT13:Pin Change Interrupt 0, Source 13
PB4	XTAL1: Crystal Oscillator Input CLKI: External Clock Input $\overline{\text{OC1D}}$: Inverted Timer/Counter1 Compare Match D Output ADC7: ADC Input Channel 7 PCINT12:Pin Change Interrupt 0, Source 12
PB3	OC1B: Timer/Counter1 Compare Match B Output PCINT11:Pin Change Interrupt 0, Source 11
PB2	USCK: USI Clock (Three Wire Mode) SCL : USI Clock (Two Wire Mode) $\overline{\text{OC1B}}$: Inverted Timer/Counter1 Compare Match B Output PCINT10:Pin Change Interrupt 0, Source 10
PB1	DO: USI Data Output (Three Wire Mode) OC1A: Timer/Counter1 Compare Match A Output PCINT9: Pin Change Interrupt 1, Source 9
PB0	DI: USI Data Input (Three Wire Mode) SDA: USI Data Input (Two Wire Mode) $\overline{\text{OC1A}}$: Inverted Timer/Counter1 Compare Match A Output PCINT8: Pin Change Interrupt 1, Source 8

- **Port B, Bit 7 – $\overline{\text{RESET}}$/ dW/ ADC10/ PCINT15**
  - RESET, Reset pin: When the RSTDISBL Fuse is programmed, this pin functions as a normal I/O pin, and the part will have to rely on Power-on Reset and Brown-out Reset as its reset sources. When the RSTDISBL Fuse is unprogrammed, the reset circuitry is connected to the pin, and the pin can not be used as an I/O pin.
  - If PB7 is used as a reset pin, DDB7, PORTB7 and PINB7 will all read 0.
  - dW: When the debugWIRE Enable (DWEN) Fuse is programmed and Lock bits are unprogrammed, the RESET port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

OCF0B, but in 16-bit mode the match can set only the Output Compare Flag OCF0A as there is only one Output Compare Unit. 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 11-5 shows a block diagram of the Output Compare unit.

Figure 11-5. Output Compare Unit, Block Diagram



11.6.1 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0H/L Register will block any Compare Match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0A/B to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

11.6.2 Using the Output Compare Unit

Since writing TCNT0H/L will block all Compare Matches for one timer clock cycle, there are risks involved when changing TCNT0H/L when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0H/L equals the OCR0A/B value, the Compare Match will be missed.

11.7 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the Timer/Counter Width (TCW0), Input Capture Enable (ICEN0) and Wave Generation Mode (CTC0) bits. See “TCCR0A – Timer/Counter0 Control Register A” on page 84.

Table 11-3 summarises the different modes of operation.

Table 11-3. Modes of operation

Mode	ICEN0	TCW0	CTC0	Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on
0	0	0	0	Normal, 8-bit Mode	0xFF	Immediate	MAX (0xFF)
1	0	0	1	CTC Mode, 8-bit	OCR0A	Immediate	MAX (0xFF)
2	0	1	X	Normal, 16-bit Mode	0xFFFF	Immediate	MAX (0xFFFF)
3	1	0	X	Input Capture Mode, 8-bit	0xFF	Immediate	MAX (0xFF)
4	1	1	X	Input Capture Mode, 16-bit	0xFFFF	Immediate	MAX (0xFFFF)

11.10.2 TCCR0B – Timer/Counter0 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	-	-	-	TSM	PSR0	CS02	CS01	CS01	TCCR0B
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 4 – TSM: Timer/Counter Synchronization Mode**

Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSR0 bit is kept, hence keeping the Prescaler Reset signal asserted. This ensures that the Timer/Counter is halted and can be configured without the risk of advancing during configuration. When the TSM bit is written to zero, the PSR0 bit is cleared by hardware, and the Timer/Counter start counting.

- **Bit 3 – PSR0: Prescaler Reset Timer/Counter0**

When this bit is one, the Timer/Counter0 prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set.

- **Bits 2:0 – CS02, CS01, CS00: Clock Select0, Bit 2, 1, and 0**

The Clock Select0 bits 2, 1, and 0 define the prescaling source of Timer0.

Table 11-4. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(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 T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

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

11.10.3 TCNT0L – Timer/Counter0 Register Low Byte

Bit	7	6	5	4	3	2	1	0	
0x32 (0x52)	TCNT0L[7:0]								TCNT0L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter0 Register Low Byte, TCNT0L, gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0L Register blocks (disables) the Compare Match on the following timer clock. Modifying the counter (TCNT0L) while the counter is running, introduces a risk of missing a Compare Match between TCNT0L and the OCR0x Registers. In 16-bit mode the TCNT0L register contains the lower part of the 16-bit Timer/Counter0 Register.

The counter value (TCNT1) that is shown as a histogram in Figure 12-11 is incremented until the counter value matches the TOP value. The counter is then cleared at the following clock cycle The diagram includes the Waveform Output (OCW1x) in toggle Compare Mode. The small horizontal line marks on the TCNT1 slopes represent Compare Matches between OCR1x and TCNT1.

The Timer/Counter Overflow Flag (TOV1) is set in the same clock cycle as the TCNT1 becomes zero. The TOV1 Flag in this case behaves like a 11th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt, that automatically clears the TOV1 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

The Output Compare Unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time. For generating a waveform, the OCW1x output can be set to toggle its logical level on each Compare Match by setting the Compare Output mode bits to toggle mode (COM1x1:0 = 1). The OC1x value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC1x} = f_{clkT1}/4$ when OCR1C is set to zero. The waveform frequency is defined by the following equation:

$$f_{OC1x} = \frac{f_{clkT1}}{2 \cdot (1 + OCR1C)}$$

Resolution, R_{PWM}, shows how many bit is required to express the value in the OCR1C register and it can be calculated using the following equation:

$$R_{PWM} = \log_2(OCR1C + 1)$$

The Output Compare Pin configurations in Normal Mode are described in Table 12-2.

Table 12-2. Output Compare Pin Configurations in Normal Mode

COM1x1	COM1x0	$\overline{OC1x}$ Pin	OC1x Pin
0	0	Disconnected	Disconnected
0	1	Disconnected	OC1x
1	0	Disconnected	OC1x
1	1	Disconnected	OC1x

12.8.2 Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (PWM1A/PWM1B = 1 and WGM11:10 = 00) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP (defined as OCR1C) then restarts from BOTTOM. In non-inverting Compare Output mode the Waveform Output (OCW1x) is cleared on the Compare Match between TCNT1 and OCR1x and set at BOTTOM. In inverting Compare Output mode, the Waveform Output is set on Compare Match and cleared at BOTTOM. In complementary Compare Output mode the Waveform Output is cleared on the Compare Match and set at BOTTOM.

Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the Phase and Frequency Correct PWM mode that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and

The dedicated Dead Time prescaler in front of the Dead Time Generator can divide the Timer/Counter1 clock (PCK or CK) by 1, 2, 4 or 8 providing a large range of dead times that can be generated. The Dead Time prescaler is controlled by two bits DTPS11 and DTPS10 from the Dead Time Prescaler register. These bits define the division factor of the Dead Time prescaler. The division factors are given in Table 12-16.

Table 12-16. Division factors of the Dead Time prescaler

DTPS11	DTPS10	Prescaler divides the T/C1 clock by
0	0	1x (no division)
0	1	2x
1	0	4x
1	1	8x

- **Bits 3:0 – CS13, CS12, CS11, CS10: Clock Select Bits 3, 2, 1, and 0**

The Clock Select bits 3, 2, 1, and 0 define the prescaling source of Timer/Counter1.

Table 12-17. Timer/Counter1 Prescaler Select

CS13	CS12	CS11	CS10	Asynchronous Clocking Mode	Synchronous Clocking Mode
0	0	0	0	T/C1 stopped	T/C1 stopped
0	0	0	1	PCK	CK
0	0	1	0	PCK/2	CK/2
0	0	1	1	PCK/4	CK/4
0	1	0	0	PCK/8	CK/8
0	1	0	1	PCK/16	CK/16
0	1	1	0	PCK/32	CK/32
0	1	1	1	PCK/64	CK/64
1	0	0	0	PCK/128	CK/128
1	0	0	1	PCK/256	CK/256
1	0	1	0	PCK/512	CK/512
1	0	1	1	PCK/1024	CK/1024
1	1	0	0	PCK/2048	CK/2048
1	1	0	1	PCK/4096	CK/4096
1	1	1	0	PCK/8192	CK/8192
1	1	1	1	PCK/16384	CK/16384

The Stop condition provides a Timer Enable/Disable function.

```
SPITransfer_loop:
    sts    USICR,r16
    lds    r16, USISR
    sbrs   r16, USIOIF
    rjmp   SPITransfer_loop
    lds    r16,USIDR
    ret
```

The code is size optimized using only eight instructions (plus return). The code example assumes that the DO and USCK pins have been enabled as outputs in DDRA. The value stored in register r16 prior to the function is called is transferred to the slave device, and when the transfer is completed the data received from the slave is stored back into the register r16.

The second and third instructions clear the USI Counter Overflow Flag and the USI counter value. The fourth and fifth instructions set three-wire mode, positive edge clock, count at USITC strobe, and toggle USCK. The loop is repeated 16 times.

The following code demonstrates how to use the USI as an SPI master with maximum speed ($f_{SCK} = f_{CK}/2$):

```
SPITransfer_Fast:
    out    USIDR,r16
    ldi    r16,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)
    ldi    r17,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)|(1<<USICLK)

    out    USICR,r16 ; MSB
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16
    out    USICR,r17
    out    USICR,r16 ; LSB
    out    USICR,r17

    in     r16,USIDR
ret
```

Table 13-1. Relationship between USIWM1:0 and USI Operation (Continued)

USIWM1	USIWM0	Description
1	0	Two-wire mode. Uses SDA (DI) and SCL (USCK) pins (1). The *Serial Data* (SDA) and the *Serial Clock* (SCL) pins are bi-directional and use open-collector output drives. The output drivers are enabled by setting the corresponding bit for SDA and SCL in the DDRA register. When the output driver is enabled for the SDA pin, the output driver will force the line SDA low if the output of the USI Data Register or the corresponding bit in the PORTA register is zero. Otherwise, the SDA line will not be driven (i.e., it is released). When the SCL pin output driver is enabled the SCL line will be forced low if the corresponding bit in the PORTA register is zero, or by the start detector. Otherwise the SCL line will not be driven. The SCL line is held low when a start detector detects a start condition and the output is enabled. Clearing the Start Condition Flag (USISIF) releases the line. The SDA and SCL pin inputs is not affected by enabling this mode. Pull-ups on the SDA and SCL port pin are disabled in Two-wire mode.
1	1	Two-wire mode. Uses SDA and SCL pins. Same operation as in two-wire mode above, except that the SCL line is also held low when a counter overflow occurs, and until the Counter Overflow Flag (USIOIF) is cleared.

Note: 1. The DI and USCK pins are renamed to *Serial Data* (SDA) and *Serial Clock* (SCL) respectively to avoid confusion between the modes of operation.

- **Bits 3:2 – USICS1:0: Clock Source Select**

These bits set the clock source for the USI Data Registerr and counter. The data output latch ensures that the output is changed at the opposite edge of the sampling of the data input (DI/SDA) when using external clock source (USCK/SCL). When software strobe or Timer/Counter0 Compare Match clock option is selected, the output latch is transparent and therefore the output is changed immediately. Clearing the USICS1:0 bits enables software strobe option. When using this option, writing a one to the USICLK bit clocks both the USI Data Register and the counter. For external clock source (USICS1 = 1), the USICLK bit is no longer used as a strobe, but selects between external clocking and software clocking by the USITC strobe bit.

Table 13-2 on page 135 shows the relationship between the USICS1:0 and USICLK setting and clock source used for the USI Data Register and the 4-bit counter.

Table 13-2. Relations between the USICS1:0 and USICLK Setting

USICS1	USICS0	USICLK	USI Data Register Clock Source	4-bit Counter Clock Source
0	0	0	No Clock	No Clock
0	0	1	Software clock strobe (USICLK)	Software clock strobe (USICLK)
0	1	X	Timer/Counter0 Compare Match	Timer/Counter0 Compare Match
1	0	0	External, positive edge	External, both edges

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 15-4.

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.

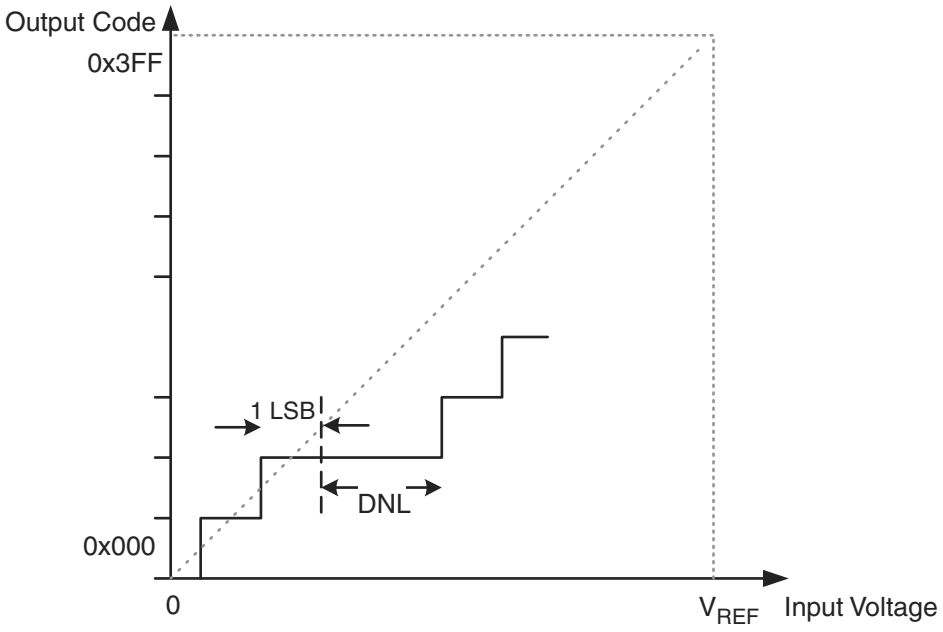
15.4 Starting a Conversion

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

- Differential Non-linearity (DNL): The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSB). Ideal value: 0 LSB.

Figure 15-12. Differential Non-linearity (DNL)



- Quantization Error: Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSB wide) will code to the same value. Always ± 0.5 LSB.

- Absolute Accuracy: The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of offset, gain error, differential error, non-linearity, and quantization error. Ideal value: ± 0.5 LSB.

15.11 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH). The form of the conversion result depends on the type of the conversio as there are three types of conversions: single ended conversion, unipolar differential conversion and bipolar differential conversion.

15.11.1 Single Ended Conversion

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 15-3 on page 155 and Table 15-4 on page 157). 0x000 represents analog ground, and

assuming calibration at room temperature. Better accuracies are achieved by using two temperature points for calibration.

Table 15-2. Temperature vs. Sensor Output Voltage (Typical Case)

Temperature	-40 °C	+25 °C	+85 °C
ADC	230 LSB	300 LSB	370 LSB

The values described in Table 15-2 are typical values. However, due to process variation the temperature sensor output voltage varies from one chip to another. To be capable of achieving more accurate results the temperature measurement can be calibrated in the application software. The sofware calibration can be done using the formula:

$$T = k * [(ADCH << 8) \mid ADCL] + T_{OS}$$

where ADCH and ADCL are the ADC data registers, k is the fixed slope coefficient and T_{OS} is the temperature sensor offset. Typically, k is very close to 1.0 and in single-point calibration the coefficient may be omitted. Where higher accuracy is required the slope coefficient should be evaluated based on measurements at two temperatures.

15.13 Register Description

15.13.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
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 7:6 – REFS1:REFS0: Voltage Reference Selection Bits**

These bits together with the REFS2 bit from the ADC Control and Status Register B (ADCSRB) select the voltage reference for the ADC, as shown in Table 15-3.

Table 15-3. Voltage Reference Selections for ADC

REFS2	REFS1	REFS0	Voltage Reference Selection
X	0	0	V_{CC} used as voltage reference, disconnected from AREF
X	0	1	External voltage reference at AREF pin, internal voltage reference turned off
0	1	0	Internal 1.1V voltage reference
0	1	1	Reserved
1	1	0	Internal 2.56V voltage reference, without external bypass capacitor, disconnected from AREF
1	1	1	Internal 2.56V voltage reference, with external bypass capacitor at AREF pin

If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSR is set). Also note, that when these bits are changed, the next conversion will take 25 ADC clock cycles.

Table 15-5. ADC Prescaler Selections (Continued)

ADPS2	ADPS1	ADPS0	Division Factor
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

15.13.3 ADCL and ADCH – The ADC Data Register

15.13.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
0x05 (0x25)	–	–	–	–	–	–	ADC9	ADC8	ADCH
0x04 (0x24)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

15.13.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
0x05 (0x25)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
0x04 (0x24)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, 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.

The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in “ADC Conversion Result” on page 153.

22. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1

Package Type	
32M1-A	32-pad, 5 x 5 x 1.0 mm Body, Lead Pitch 0.50 mm, Micro Lead Frame Package (MLF)
20P3	20-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
20S2	20-lead, 0.300" Wide, Plastic Gull Wing Smal Outline Package (SOIC)