



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 "<u>Embedded -</u> <u>Microcontrollers</u>"

Details

Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	CANbus, I ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	53
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at90can64-15az

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong





• Bits 7..5 - Reserved Bits

These bits are reserved bits for future use.

• Bit 4 – WDCE: Watchdog Change Enable

This bit must be set when the WDE bit is written to logic zero. Otherwise, the Watchdog will not be disabled. Once written to one, hardware will clear this bit after four clock cycles. Refer to the description of the WDE bit for a Watchdog disable procedure. This bit must also be set when changing the prescaler bits. See "Timed Sequences for Changing the Configuration of the Watchdog Timer" on page 59.

Bit 3 – WDE: Watchdog Enable

When the WDE is written to logic one, the Watchdog Timer is enabled, and if the WDE is written to logic zero, the Watchdog Timer function is disabled. WDE can only be cleared if the WDCE bit has logic level one. To disable an enabled Watchdog Timer, the following procedure must be followed:

- 1. In the same operation, write a logic one to WDCE and WDE. A logic one must be written to WDE even though it is set to one before the disable operation starts.
- 2. Within the next four clock cycles, write a logic 0 to WDE. This disables the Watchdog.

In safety level 2, it is not possible to disable the Watchdog Timer, even with the algorithm described above. See "Timed Sequences for Changing the Configuration of the Watchdog Timer" on page 59.

• Bits 2..0 – WDP2, WDP1, WDP0: Watchdog Timer Prescaler 2, 1, and 0

The WDP2, WDP1, and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 7-6.

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V _{CC} = 3.0V	Typical Time-out at V _{CC} = 5.0V
0	0	0	16K cycles	17.1 ms	16.3 ms
0	0	1	32K cycles	34.3 ms	32.5 ms
0	1	0	64K cycles	68.5 ms	65 ms
0	1	1	32/64K cycles	0.14 s	0.13 s
1	0	0	256K cycles	0.27 s	0.26 s
1	0	1	512K cycles	0.55 s	0.52 s
1	1	0	1,024K cycles	1.1 s	1.0 s
1	1	1	2,048K cycles	2.2 s	2.1 s

 Table 7-6.
 Watchdog Timer Prescale Select



0x0012	jmp	TIM2_COMP ;	Timer2 Compare Handler
0x0014	jmp	TIM2_OVF ;	Timer2 Overflow Handler
0x0016	jmp	TIM1_CAPT ;	Timer1 Capture Handler
0x0018	jmp	TIM1_COMPA;	Timer1 CompareA Handler
0x001A	jmp	TIM1_COMPB;	Timer1 CompareB Handler
0x001C	jmp	TIM1_OVF ;	Timer1 CompareC Handler
0x001E	jmp	TIM1_OVF ;	Timer1 Overflow Handler
0x0020	jmp	TIMO_COMP ;	Timer0 Compare Handler
0x0022	jmp	TIMO_OVF ;	Timer0 Overflow Handler
0x0024	jmp	CAN_IT ;	CAN Handler
0x0026	jmp	CTIM_OVF ;	CAN Timer Overflow Handler
0x0028	jmp	SPI_STC ;	SPI Transfer Complete Handler
0x002A	jmp	USART0_RXC;	USARTO RX Complete Handler
0x002C	jmp	USARTO_DRE;	USART0,UDR Empty Handler
0x002E	jmp	USART0_TXC;	USARTO TX Complete Handler
0x0030	jmp	ANA_COMP ;	Analog Comparator Handler
0x0032	jmp	ADC ;	ADC Conversion Complete Handler
0x0034	jmp	EE_RDY ;	EEPROM Ready Handler
0x0036	jmp	TIM3_CAPT ;	Timer3 Capture Handler
0x0038	jmp	TIM3_COMPA;	Timer3 CompareA Handler
0x003A	jmp	TIM3_COMPB;	Timer3 CompareB Handler
0x003C	jmp	TIM3_COMPC;	Timer3 CompareC Handler
0x003E	jmp	TIM3_OVF ;	Timer3 Overflow Handler
0x0040	jmp	USART1_RXC;	USART1 RX Complete Handler
0x0042	jmp	USART1_DRE;	USART1,UDR Empty Handler
0x0044	jmp	USART1_TXC;	USART1 TX Complete Handler
0x0046	jmp	TWI ;	TWI Interrupt Handler
0x0048	jmp	SPM_RDY ;	SPM Ready Handler
;			
0x004A	RESET: ldi	r16, high(RAMEND) ; Main program start
0x004B	out	SPH,r16	;Set Stack Pointer to top of RAM
0x004C	ldi	r16, low(F	RAMEND)
0x004D	out	SPL,r16	
0x004E	sei		; Enable interrupts
0x004F	<inst< td=""><td>tr> xxx</td><td></td></inst<>	tr> xxx	

When the BOOTRST Fuse is unprogrammed, the Boot section size set to 8K bytes and the IVSEL bit in the MCUCR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

;AddressLab	els	Code	Comments
0x0000 RES	ET: ldi	r16,high(RAMEND) ;	Main program start
0x0001	out	SPH,r16 ;	Set Stack Pointer to top of RAM
0x0002	ldi	r16,low(RAMEND)	
0x0003	out	SPL,r16	



8.2 Moving Interrupts Between Application and Boot Space

The General Interrupt Control Register controls the placement of the Interrupt Vector table.

8.2.1 MCU Control Register – MCUCR

Bit	7	6	5	4	3	2	1	0	_
	JTD	-	-	PUD	-	-	IVSEL	IVCE	MCUCF
Read/Write	R/W	R	R	R/W	R	R	R/W	R/W	-
Initial Value	0	0	0	0	0	0	0	0	

Bit 1 – IVSEL: Interrupt Vector Select

When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. Refer to the section "Boot Loader Support – Read-While-Write Self-Programming" on page 320 for details. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

- 1. Write the Interrupt Vector Change Enable (IVCE) bit to one.
- 2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the Status Register is unaffected by the automatic disabling.

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section. Refer to the section "Boot Loader Support – Read-While-Write Self-Programming" on page 320 for details on Boot Lock bits.



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 7 as input with pull-ups assigned to port pins 6 and 7. 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<sup>(1)</sup>
```

```
. . .
     ; Define pull-ups and set outputs high
     ; Define directions for port pins
     ldi
            r16, (1<<PB7) | (1<<PB6) | (1<<PB1) | (1<<PB0)
            r17, (1<<DDB3) | (1<<DDB2) | (1<<DDB1) | (1<<DDB0)
     ldi
            PORTB, r16
     out
            DDRB, r17
     out
     ; Insert nop for synchronization
     nop
     ; Read port pins
     in
             r16, PINB
      . . .
C Code Example<sup>(1)</sup>
   unsigned char i;
     /* Define pull-ups and set outputs high */
     /* Define directions for port pins */
     PORTB = (1<<PB7) | (1<<PB6) | (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 pullups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

9.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 9-2, 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 "Alternate Port Functions" on page 71.

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







Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. clk_{I/O}, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

Table 9-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 9-5 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

MOSI, SPI Master Data output, Slave Data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB2. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB2. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB2 bit.

• SCK – Port B, Bit 1

SCK, Master Clock output, Slave Clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB1. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB1. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB1 bit.

• SS – Port B, Bit 0

 \overline{SS} , Slave Port Select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB0. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB0. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB0 bit.

Table 9-7 and Table 9-8 relate the alternate functions of Port B to the overriding signals shown in Figure 9-5 on page 72. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

Table 9-7 and Table 9-8 relates the alternate functions of Port B to the overriding signals shown in Figure 9-5 on page 72.

Signal Name	PB7/OC0A/OC1C	PB6/OC1B	PB5/OC1A	PB4/OC2A
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC0A/OC1C ENABLE ⁽¹⁾	OC1B ENABLE	OC1A ENABLE	OC2A ENABLE
PVOV	OC0A/OC1C ⁽¹⁾	OC1B	OC1A	OC2A
PTOE	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	_
AIO	-	_	-	-

 Table 9-7.
 Overriding Signals for Alternate Functions in PB7..PB4

Note: 1. See "Output Compare Modulator - OCM" on page 164 for details.



A14 – Port C, Bit 6

A14, External memory interface address 14.

• A13 – Port C, Bit 5

A13, External memory interface address 13.

• A12 – Port C, Bit 4

A12, External memory interface address 12.

• A11 – Port C, Bit 3 A11, External memory interface address 11.

• A10 – Port C, Bit 2

A10, External memory interface address 10.

• A9 – Port C, Bit 1

A9, External memory interface address 9.

• A8 – Port C, Bit 0

A8, External memory interface address 8.

Table 9-10 and Table 9-11 relate the alternate functions of Port C to the overriding signals shown in Figure 9-5 on page 72.

Signal Name	PC7/A15	PC6/A14	PC5/A13	PC4/A12
PUOE	SRE • (XMM<1)	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
PUOV	0	0	0	0
DDOE	CKOUT ⁽¹⁾ + (SRE • (XMM<1))	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
DDOV	1	1	1	1
PVOE	CKOUT ⁽¹⁾ + (SRE • (XMM<1))	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
PVOV	(A15 • CKOUT ⁽¹⁾) + (CLKO • CKOUT ⁽¹⁾)	A14	A13	A12
PTOE	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	_
AIO	-	-	_	_

 Table 9-10.
 Overriding Signals for Alternate Functions in PC7..PC4

Note: 1. CKOUT is one if the CKOUT Fuse is programmed



13. 16-bit Timer/Counter (Timer/Counter1 and Timer/Counter3)

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement. The main features are:

13.1 Features

- True 16-bit Design (i.e., Allows 16-bit PWM)
- Three independent Output Compare Units
- Double Buffered Output Compare Registers
- One Input Capture Unit
- Input Capture Noise Canceler
- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Variable PWM Period
- Frequency Generator
- External Event Counter
- Four independent interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1 for Timer/Counter1 TOV3, OCF3A, OCF3B, and ICF3 for Timer/Counter3)

13.2 Overview

Many register and bit references in this section are written in general form.

- A lower case "n" replaces the Timer/Counter number, in this case 1 or 3. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT1 for accessing Timer/Counter1 counter value and so on.
- A lower case "x" replaces the Output Compare unit channel, in this case A, B or C. However, when using the register or bit defines in a program, the precise form must be used, i.e., OCRnA for accessing Timer/Countern output compare channel A value and so on.

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 13-1. For the actual placement of I/O pins, refer to "Pinout AT90CAN32/64/128 - TQFP" on page 5. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the "16-bit Timer/Counter Register Description" on page 135.



AT90CAN32/64/128

For detailed timing information refer to "Timer/Counter Timing Diagrams" on page 154.

14.7.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM21:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV2) will be set in the same timer clock cycle as the TCNT2 becomes zero. The TOV2 flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV2 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.

14.7.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM21:0 = 2), the OCR2A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT2) matches the OCR2A. The OCR2A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 14-6. The counter value (TCNT2) increases until a compare match occurs between TCNT2 and OCR2A, and then counter (TCNT2) is cleared.



Figure 14-6. CTC Mode, Timing Diagram

An interrupt can be generated each time the counter value reaches the TOP value by using the OCF2A flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR2A is lower than the current value of TCNT2, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the compare match can occur.



units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the UMSELn, U2Xn and DDR_XCKn bits.

Table 17-1 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRRn value for each mode of operation using an internally generated clock source.

Operating Mode	Equation for Calculating Baud Rate $^{(1)}$	Equation for Calculating UBRRn Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{CLKio}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{CLKio}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{CLKio}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{CLKio}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{CLKio}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{CLKio}}{2BAUD} - 1$

Table 17-1. Equations for Calculating Baud Rate Register Setting

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD	Baud rate (in bits per second, bps).
fclk _{io}	System I/O Clock frequency.
UBRRn	Contents of the UBRRnH and UBRRnL Registers, (0-4095).

Some examples of UBRRn values for some system clock frequencies are found in Table 17-9 (see page 199).

17.4.2 Double Speed Operation (U2X)

The transfer rate can be doubled by setting the U2Xn bit in UCSRnA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the Receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the Transmitter, there are no downsides.

17.4.3 External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to Figure 17-2 for details.

External clock input from the XCKn pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the Transmitter and Receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCKn clock frequency is limited by the following equation:

 $f_{XCKn} < \frac{f_{CLKio}}{4}$





17.11.6 USART1 Control and Status Register B – UCSR1B

Bit	7	6	5	4	3	2	1	0	_
	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	UCSR1B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	-
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – RXCIEn: RX Complete Interrupt Enable

Writing this bit to one enables interrupt on the RXCn flag. A USARTn Receive Complete interrupt will be generated only if the RXCIEn bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXCn bit in UCSRnA is set.

Bit 6 – TXCIEn: TX Complete Interrupt Enable

Writing this bit to one enables interrupt on the TXCn flag. A USARTn Transmit Complete interrupt will be generated only if the TXCIEn bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXCn bit in UCSRnA is set.

Bit 5 – UDRIEn: USARTn Data Register Empty Interrupt Enable

Writing this bit to one enables interrupt on the UDREn flag. A Data Register Empty interrupt will be generated only if the UDRIEn bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDREn bit in UCSRnA is set.

Bit 4 – RXENn: Receiver Enable

Writing this bit to one enables the USARTn Receiver. The Receiver will override normal port operation for the RxDn pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FEn, DORn, and UPEn Flags.

• Bit 3 – TXENn: Transmitter Enable

Writing this bit to one enables the USARTn Transmitter. The Transmitter will override normal port operation for the TxDn pin when enabled. The disabling of the Transmitter (writing TXENn to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxDn port.

Bit 2 – UCSZn2: Character Size

The UCSZn2 bits combined with the UCSZn1:0 bit in UCSRnC sets the number of data bits (Character SiZe) in a frame the Receiver and Transmitter use.

Bit 1 – RXB8n: Receive Data Bit 8

RXB8n is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDRn.

Bit 0 – TXB8n: Transmit Data Bit 8

TXB8n is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDRn.

17.11.7 USART0 Control and Status Register C – UCSR0C

Bit	7	6	5	4	3	2	1	0	
	-	UMSEL0	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	UCSR0C
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

196 AT90CAN32/64/128 I



• Bit 2:1 – UCSZn1:0: Character Size

The UCSZn1:0 bits combined with the UCSZn2 bit in UCSRnB sets the number of data bits (Character SiZe) in a frame the Receiver and Transmitter use.

Table 17-7.	UCSZn Bits Settings
-------------	---------------------

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

• Bit 0 – UCPOLn: Clock Polarity

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOLn bit sets the relationship between data output change and data input sample, and the synchronous clock (XCKn).

Table 17-8.UCPOLn Bit Settings

UCPOLn	Transmitted Data Changed (Output of TxDn Pin)	Received Data Sampled (Input on RxDn Pin)
0	Rising XCK Edge	Falling XCK Edge
1	Falling XCK Edge	Rising XCK Edge

17.11.9 USART0 Baud Rate Registers – UBRR0L and UBRR0H

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-		UBRR	0[11:8]		UBRR0H
				UBRF	R0[7:0]				UBRR0L
	7	6	5	4	3	2	1	0	-
Read/Write	R	R	R	R	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	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

17.11.10 USART1 Baud Rate Registers - UBRR1L and UBRR1H

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-		UBRR	1[11:8]		UBRR1H
				UBRF	R1[7:0]				UBRR1L
	7	6	5	4	3	2	1	0	-
Read/Write	R	R	R	R	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	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	



• Bits 1, 0 – ACIS1, ACIS0: Analog Comparator Interrupt Mode Select

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 20-1.

Table 20-1.	ACIS1/ACIS0 Settings
-------------	----------------------

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle.
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge.
1	1	Comparator Interrupt on Rising Output Edge.

When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

20.3 Analog Comparator Multiplexed Input

It is possible to select any of the ADC7..0 pins to replace the negative input to the Analog Comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX2..0 in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 20-2. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the Analog Comparator.

ACME	ADEN	MUX20	Analog Comparator Negative Input
0	х	XXX	AIN1
1	1	ххх	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

Table 20-2. Analog Comparator Multiplexed Input

AT90CAN32/64/128





Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA. In this mode the ADC will perform successive conversions independently of whether the ADC Interrupt Flag, ADIF is cleared or not.

If Auto Triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

21.4 Prescaling and Conversion Timing

Figure 21-3. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200 kHz to get a higher sample rate.

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





When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle. See "Differential Channels" on page 277 for details on differential conversion timing.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 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 21-1.



Figure 21-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)





AT90CAN32/64/128



• 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.





- 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.

21.7 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).



AT90CAN32/64/128

21.8.5 Digital Input Disable Register 0 – DIDR0



Bit 7:0 – ADC7D..ADC0D: ADC7:0 Digital Input Disable

When this bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC7..0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.









23.6.3 Scanning the RESET Pin

The RESET pin accepts 3V or 5V active low logic for standard reset operation, and 12V active high logic for High Voltage Parallel programming. An observe-only cell as shown in Figure 23-6 is inserted both for the 3V or 5V reset signal - RSTT, and the 12V reset signal - RSTHV.





23.6.4 Scanning the Clock Pins

The AVR devices have many clock options selectable by fuses. These are: Internal RC Oscillator, External Clock, (High Frequency) Crystal Oscillator, Low-frequency Crystal Oscillator, and Ceramic Resonator.

Figure 23-7 shows how each oscillator with external connection is supported in the scan chain. The Enable signal is supported with a general Boundary-scan cell, while the Oscillator/clock output is attached to an observe-only cell. In addition to the main clock, the Timer2 Oscillator is scanned in the same way. The output from the internal RC Oscillator is not scanned, as this oscillator does not have external connections.

sections that are configurable by the BOOTSZ Fuses as described above, the Flash is also divided into two fixed sections, the Read-While-Write (RWW) section and the No Read-While-Write (NRWW) section. The limit between the RWW- and NRWW sections is given in Table 24-7 on page 333 and Figure 24-2 on page 323. The main difference between the two sections is:

- When erasing or writing a page located inside the RWW section, the NRWW section can be read during the operation.
- When erasing or writing a page located inside the NRWW section, the CPU is halted during the entire operation.

Note that the user software can never read any code that is located inside the RWW section during a Boot Loader software operation. The syntax "Read-While-Write section" refers to which section that is being programmed (erased or written), not which section that actually is being read during a Boot Loader software update.

24.3.1 RWW – Read-While-Write Section

If a Boot Loader software update is programming a page inside the RWW section, it is possible to read code from the Flash, but only code that is located in the NRWW section. During an ongoing programming, the software must ensure that the RWW section never is being read. If the user software is trying to read code that is located inside the RWW section (i.e., by a call/jmp/lpm or an interrupt) during programming, the software might end up in an unknown state. To avoid this, the interrupts should either be disabled or moved to the Boot Loader section. The Boot Loader section is always located in the NRWW section. The RWW Section Busy bit (RWWSB) in the Store Program Memory Control and Status Register (SPMCSR) will be read as logical one as long as the RWW section is blocked for reading. After a programming is completed, the RWWSB must be cleared by software before reading code located in the RWW section. See "Store Program Memory Control and Status Register – SPMCSR" on page 325. for details on how to clear RWWSB.

24.3.2 NRWW – No Read-While-Write Section

The code located in the NRWW section can be read when the Boot Loader software is updating a page in the RWW section. When the Boot Loader code updates the NRWW section, the CPU is halted during the entire Page Erase or Page Write operation.

Which Section does the Z-pointer Address During the Programming?	Which Section Can be Read During Programming?	Is the CPU Halted?	Read-While-Write Supported?	
RWW Section	NRWW Section	No	Yes	
NRWW Section	None	Yes	No	

Table 24-1. Read-While-Write Features





Table 27-1.	External Clock Drive

Symbol	Parameter	V _{CC} = 2	.7 - 5.5V	V _{CC} = 4	Unito	
Symbol	Farameter	Min.	Max.	Min.	Max.	Units
1/t _{CLCL}	Oscillator Frequency	0	8	0	16	MHz
t _{CLCL}	Clock Period	125		62.5		ns
t _{CHCX}	High Time	50		25		ns
t _{CLCX}	Low Time	50		25		ns
t _{CLCH}	Rise Time		1.6		0.5	μs
t _{CHCL}	Fall Time		1.6		0.5	μs
Δt_{CLCL}	Change in period from one clock cycle to the next		2		2	%

Maximum Speed vs. V_{CC} 27.4

Maximum frequency is depending on V_{CC} . As shown in Figure 27-2., the Maximum Frequency vs. V_{CC} curve is linear between 1.8V < V_{CC} < 4.5V. To calculate the maximum frequency at a given voltage in this interval, use this equation:

 $Frequency = a \bullet (V - Vx) + Fy$

To calculate required voltage for a given frequency, use this equation:

$$Voltage = b \bullet (F - Fy) + Vx$$

Table 27-2. Constants used to calculate maximum speed vs. V_{CC}

Voltage and Frequency range	а	b	Vx	Fy
2.7 < VCC < 4.5 or 8 < Frequency < 16	8/1.8	1.8/8	2.7	8

At 3 Volt, this gives: $Frequency = \frac{8}{1.8} \cdot (3 - 2.7) + 8 = 9.33$

Thus, when $V_{CC} = 3V$, maximum frequency will be 9.33 MHz.

 $Voltage = \frac{1.8}{8} \bullet (8-8) + 2.7 = 2.7$ At 8 MHz this gives:

Thus, a maximum frequency of 8 MHz requires V_{CC} = 2.7V.