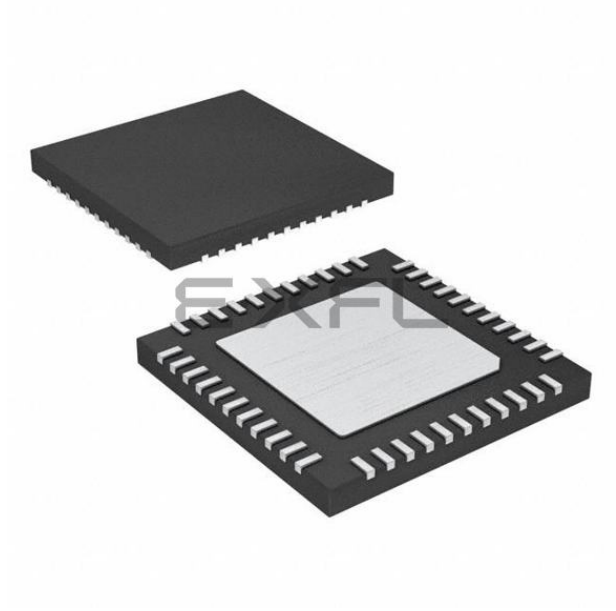


Welcome to E-XFL.COM

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	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	32
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VFQFN Exposed Pad
Supplier Device Package	44-VQFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega324pa-mnr

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

There are basically two types of interrupts:

The first type is triggered by an event that sets the Interrupt Flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding Interrupt Flag. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding Interrupt Flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered. When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

The Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example(1)

```
in r16, SREG ; store SREG value
cli ; disable interrupts during timed sequence
sbi EECR, EEMPE ; start EEPROM write
sbi EECR, EEPE
out SREG, r16 ; restore SREG value (I-bit)
```

C Code Example(1)

```
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMPE); /* start EEPROM write */
EECR |= (1<<EEPE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

1. Refer to *About Code Examples*.

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example(1)

```
sei ; set Global Interrupt Enable
sleep ; enter sleep, waiting for interrupt
; note: will enter sleep before any pending interrupt(s)
```

- Brown-out Reset
- 2-wire Serial Interface address match
- Timer/Counter interrupt
- SPM/EEPROM ready interrupt
- External level interrupt on INT
- Pin change interrupt

Note: 1. Timer/Counter will only keep running in asynchronous mode.

Related Links

8-bit Timer/Counter2 with PWM and Asynchronous Operation on page 191

11.6. Power-Down Mode

When the SM[2:0] bits are written to '010', the SLEEP instruction makes the MCU enter Power-Down mode. In this mode, the external Oscillator is stopped, while the external interrupts, the 2-wire Serial Interface address watch, and the Watchdog continue operating (if enabled).

Only one of these events can wake up the MCU:

- External Reset
- Watchdog System Reset
- Watchdog Interrupt
- Brown-out Reset
- 2-wire Serial Interface address match
- External level interrupt on INT
- Pin change interrupt

This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note: 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 interrupt will be generated. The start-up time is defined by the SUT and CKSEL Fuses.

When waking up from Power-Down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL Fuses that define the Reset Time-out period.

Related Links

Clock Sources on page 46
EXINT - External Interrupts on page 86

11.7. Power-Save Mode

When the SM[2:0] bits are written to 011, the SLEEP instruction makes the MCU enter Power-save mode. This mode is identical to Power-down, except:

If Timer/Counter2 is enabled, it will keep running during sleep. The device can wake up from either Timer Overflow or Output Compare event from Timer/Counter2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK2, and the Global Interrupt Enable bit in SREG is set.

If Timer/Counter2 is not running, Power-down mode is recommended instead of Power-save mode.

Vector No	Program Address(2)	Source	Interrupts definition
21	0x0028	USART_RX	USART Rx Complete
22	0x002A	USART_UDRE	USART Data Register Empty
23	0x002C	USART_TX	USART Tx Complete
24	0x002E	ANALOG_COMP	Analog Comparator
25	0x0030	ADC	ADC Conversion Complete
26	0x0032	EE_READY	EEPROM Ready
27	0x0034	TWI	TWI Transfer complete
28	0x0036	SPM_READY	Store Program Memory Ready
29	0x0038	USART1_RX	USART1 Rx Complete
30	0x003A	USART1_UDRE	USART1, Data Register Empty
31	0x003C	USART1_TX	USART1, Tx Complete

Note:

1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see *Memory programming*
2. When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.

The table below shows reset and Interrupt Vectors placement for the various combinations of BOOTRST and MCUCR.IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.

Table 13-2. Reset and Interrupt Vectors placement

BOOTRST	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x0000	0x0002
1	1	0x0000	Boot Reset Address + 0x0002
0	0	Boot Reset Address	0x0002
0	1	Boot Reset Address	Boot Reset Address + 0x0002

Note: The Boot Reset Address is shown in Table *Boot size configuration* in Boot Loader Parameters. For the BOOTRST Fuse “1” means unprogrammed while “0” means programmed.

```
Address   Labels    Code                        Comments
0x0000              jmp      RESET               ; Reset
0x0002              jmp      INT0                ; IRQ0
0x0004              jmp      INT1                ; IRQ1
0x0006              jmp      INT2                ; IRQ2
0x0008              jmp      PCINT0              ; PCINT0
0x000A              jmp      PCINT1              ; PCINT1
0x000C              jmp      PCINT2              ; PCINT2
0x000E              jmp      PCINT3              ; PCINT3
0x0010              jmp      WDT                 ; Watchdog Timeout
0x0012              jmp      TIM2_COMPA          ; Timer2 CompareA
0x0014              jmp      TIM2_COMPB          ; Timer2 CompareB
0x0016              jmp      TIM2_OVF            ; Timer2 Overflow
0x0018              jmp      TIM1_CAPT           ; Timer1 Capture
```

15.2.2. Toggling the Pin

Writing a '1' to PINxn toggles the value of PORTxn, independent on the value of DDRxn. The SBI instruction can be used to toggle one single bit in a port.

15.2.3. Switching Between Input and Output

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

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b11) as an intermediate step.

The following table summarizes the control signals for the pin value.

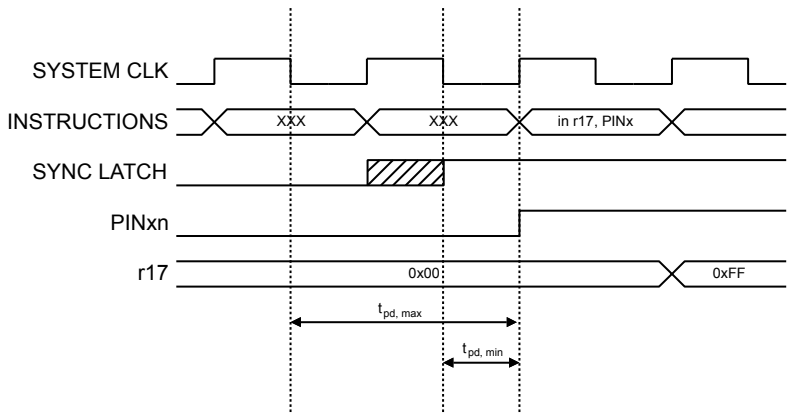
Table 15-1. Port Pin Configurations

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

15.2.4. Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Ports as General Digital I/O, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. The following figure shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

Figure 15-3. Synchronization when Reading an Externally Applied Pin value



Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded

15.4.11. Port D Data Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: PORTD
Offset: 0x2B
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x0B

Bit	7	6	5	4	3	2	1	0
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – PORTDn: Port D Data [n = 7:0]

Table 16-10. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/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.

Related Links

Timer/Counter Timing Diagrams on page 174
Compare Match Output Unit on page 165

17.12.1. Normal Mode

The simplest mode of operation is the Normal mode (TCCR1A.WGM1[3:0]=0x0). 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 16-bit value (MAX=0xFFFF) and then restarts from BOTTOM=0x0000. In normal operation the Timer/Counter Overflow Flag (TIFR1.TOV) will be set in the same timer clock cycle as the TCNT1 becomes zero. In this case, the TOV Flag in behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV 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 Input Capture unit is easy to use in Normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

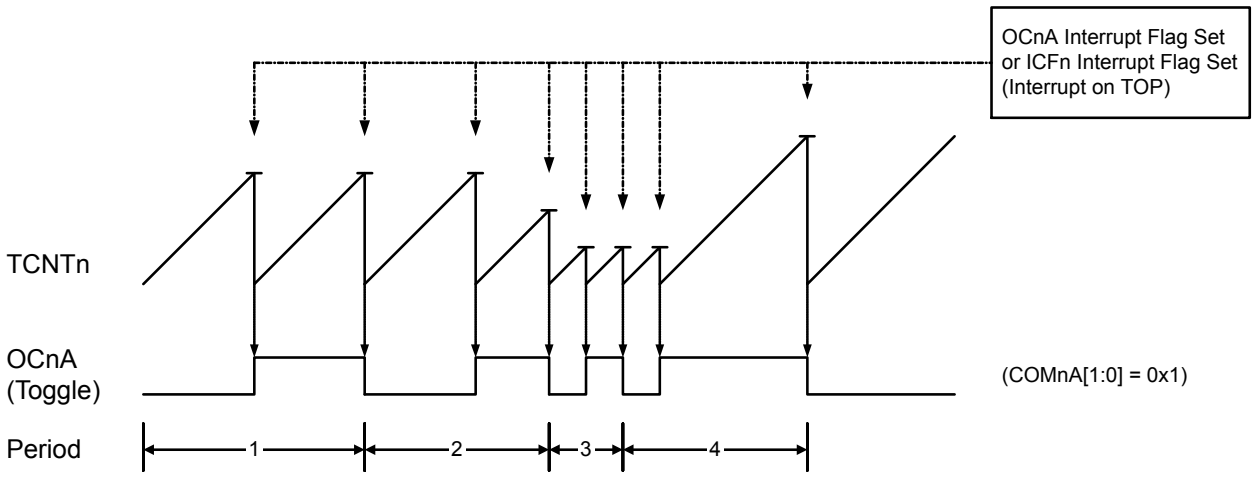
The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

17.12.2. Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC modes (mode 4 or 12, WGM1[3:0]=0x4 or 0xC), the OCR1A or ICR1 registers are used to manipulate the counter resolution: the counter is cleared to ZERO when the counter value (TCNT1) matches either the OCR1A (if WGM1[3:0]=0x4) or the ICR1 (WGM1[3:0]=0xC). The OCR1A or ICR1 define 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 below. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then TCNT1 is cleared.

Figure 17-6. CTC Mode, Timing Diagram



Note: The “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates Output Compare unit (A/B).

```
SPI_MasterTransmit:
   ; Start transmission of data (r16)
   out    SPDR,r16
Wait_Transmit:
   ; Wait for transmission complete
   in     r16, SPSR
   sbrs   r16, SPIF
   rjmp   Wait_Transmit
   ret
```

C Code Example

```
void SPI_MasterInit(void)
{
   /* Set MOSI and SCK output, all others input */
   DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
   /* Enable SPI, Master, set clock rate fck/16 */
   SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
   /* Start transmission */
   SPDR = cData;
   /* Wait for transmission complete */
   while(!(SPSR & (1<<SPIF)))
      ;
}
```

The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

Assembly Code Example

```
SPI_SlaveInit:
   ; Set MISO output, all others input
   ldi    r17,(1<<DD_MISO)
   out    DDR_SPI,r17
   ; Enable SPI
   ldi    r17,(1<<SPE)
   out    SPCR,r17
   ret
SPI_SlaveReceive:
   ; Wait for reception complete
   in     r16, SPSR
   sbrs   r16, SPIF
   rjmp SPI_SlaveReceive
   ; Read received data and return
   in     r16,SPDR
   ret
```

C Code Example

```
void SPI_SlaveInit(void)
{
   /* Set MISO output, all others input */
   DDR_SPI = (1<<DD_MISO);
   /* Enable SPI */
   SPCR = (1<<SPE);
}
char SPI_SlaveReceive(void)
{
   /* Wait for reception complete */
   while(!(SPSR & (1<<SPIF)))
      ;
   /* Return Data Register */
   return SPDR;
}
```

Related Links

23.9.2. TWI Status Register

Name: TWSR
Offset: 0xB9
Reset: 0xF8
Property: -

Bit	7	6	5	4	3	2	1	0
	TWS7	TWS6	TWS5	TWS4	TWS3		TWPS[1:0]	
Access	R	R	R	R	R	R	R/W	R/W
Reset	1	1	1	1	1	0	0	0

Bits 3, 4, 5, 6, 7 – TWS3, TWS4, TWS5, TWS6, TWS7: TWI Status Bit
The TWS[7:3] reflect the status of the TWI logic and the 2-wire Serial Bus. The different status codes are described later in this section. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

Bits 1:0 – TWPS[1:0]: TWI Prescaler
These bits can be read and written, and control the bit rate prescaler.

Table 23-8. TWI Bit Rate Prescaler

TWS[1:0]	Prescaler Value
00	1
01	4
10	16
11	64

To calculate bit rates, refer to Bit Rate Generator Unit. The value of TWPS1...0 is used in the equation.

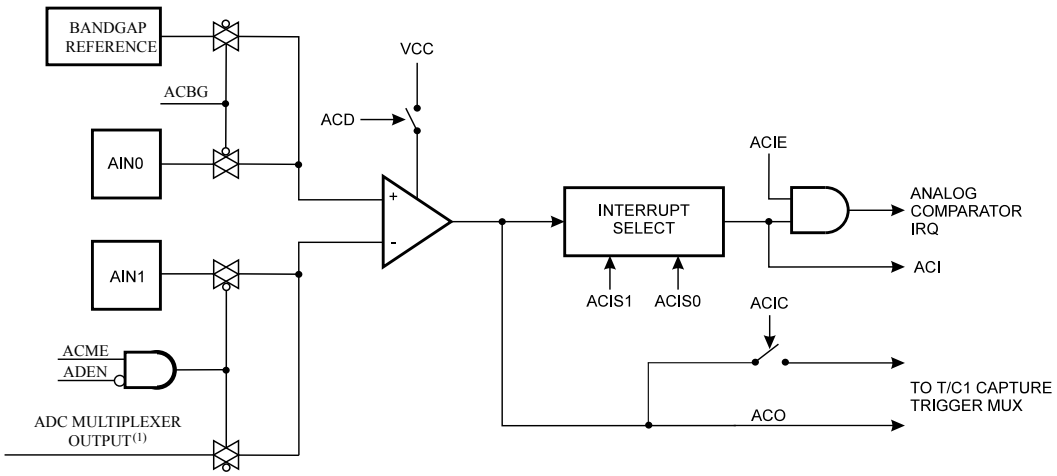
24. AC - Analog Comparator

24.1. Overview

The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator output, ACO, is set. The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, 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 below.

The Power Reduction ADC bit in the Power Reduction Register (0.PRADC) must be written to '0' in order to be able to use the ADC input MUX.

Figure 24-1. Analog Comparator Block Diagram



Note: Refer to the *Pin Configuration* and the I/O Ports description for Analog Comparator pin placement

Related Links

I/O-Ports on page 98

Power Management and Sleep Modes on page 59

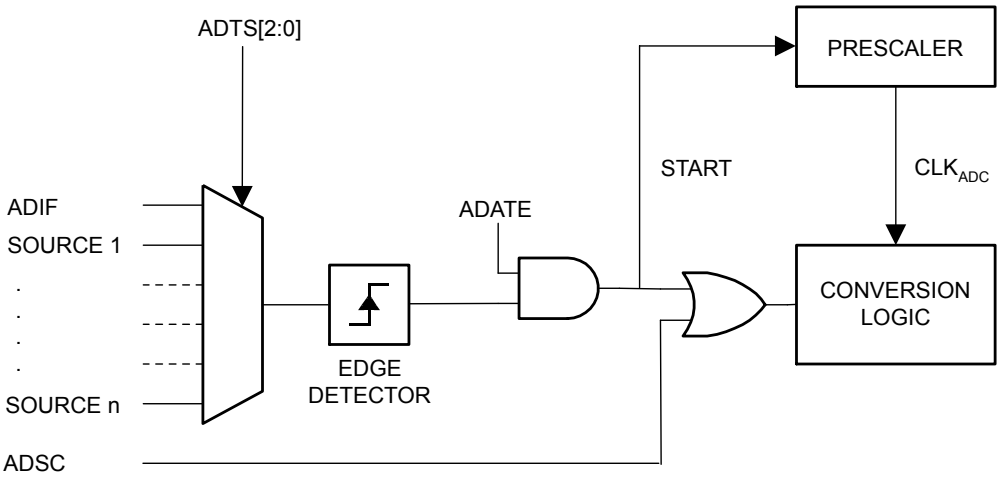
Minimizing Power Consumption on page 62

Pinout on page 15

24.2. Analog Comparator Multiplexed Input

It is possible to select any of the ADC[7:0] 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 in the ADC Control and Status Register B (ADCSRB.ACME) is '1' and the ADC is switched off (ADCSRA.ADEN=0), the three least significant Analog Channel Selection bits in the ADC Multiplexer Selection register (ADMUX.MUX[2:0]) select the input pin to replace the negative input to the Analog Comparator, as shown in the table below. When ADCSRB.ACME=0 or ADCSRA.ADEN=1, AIN1 is applied to the negative input of the Analog Comparator.

Figure 25-2. ADC Auto Trigger Logic

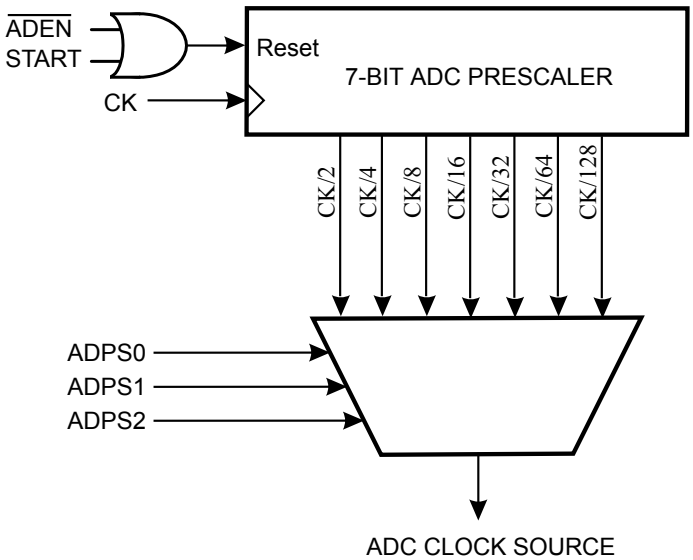


Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a '1' to ADCSRA.ADSC. 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 ADCSRA.ADSC to '1'. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as '1' during a conversion, independently of how the conversion was started.

25.4. Prescaling and Conversion Timing

Figure 25-3. ADC Prescaler



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

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is selected by the ADC Prescaler Select bits in the ADC Control and Status Register A (ADCSRA.ADPS). The prescaler starts counting from the moment the ADC

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

If both the ADC Auto Trigger Enable and ADC Enable bits (ADCRSA.ADATE, ADCRSA.ADEN) are written to '1', an interrupt event can occur at any time. If the ADMUX Register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

1. When ADATE or ADEN is cleared.
    1.1. During conversion, minimum one ADC clock cycle after the trigger event.
    1.2. After a conversion, before the Interrupt Flag used as trigger source is cleared.

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

Special care should be taken when changing differential channels. Once a differential channel has been selected, the gain stage may take as much as 125 µs to stabilize to the new value. Thus conversions should not be started within the first 125 µs after selecting a new differential channel. Alternatively, conversion results obtained within this period should be discarded. The same settling time should be observed for the first differential conversion after changing ADC reference (by changing the REFS[1:0] bits in ADMUX).

25.5.1. ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

- In Single Conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.
- In Free Running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

The user is advised not to write new channel or reference selection values during Free Running mode.

When switching to a differential gain channel, the first conversion result may have a poor accuracy due to the required settling time for the automatic offset cancellation circuitry. The user should preferably disregard the first conversion result.

25.5.2. ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result in codes close to 0x3FF. V_{REF} can be selected as either AV_{CC}, internal 2.56V reference, or external AREF pin.

AV_{CC} is connected to the ADC through a passive switch. The internal 2.56V reference is generated from the internal bandgap reference (V_{BG}) through an internal amplifier. In either case, the external AREF pin is directly connected to the ADC, and the reference voltage can be made more immune to noise by connecting a capacitor between the AREF pin and ground. V_{REF} can also be measured at the AREF pin with a high impedance voltmeter. Note that V_{REF} is a high impedance source, and only a capacitive load should be connected in a system.

If the user has a fixed voltage source connected to the AREF pin, the user may not use the other reference voltage options in the application, as they will be shorted to the external voltage. If no external voltage is applied to the AREF pin, the user may switch between AV_{CC} and 2.56V as reference selection.

25.8.2. ADC Control and Status Register A

Name: ADCSRA
Offset: 0x7A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	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.

Bit 5 – ADATE: ADC Auto Trigger Enable
When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

Bit 4 – ADIF: ADC Interrupt Flag
This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

Bit 3 – ADIE: ADC Interrupt Enable
When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

Bits 2:0 – ADPSn: ADC Prescaler Select [n = 2:0]
These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 25-5. Input Channel Selection

ADPS[2:0]	Division Factor
000	2
001	2

27. BTLDR - Boot Loader Support – Read-While-Write Self-Programming

27.1. Features

- Read-While-Write Self-Programming
- Flexible Boot Memory Size
- High Security (Separate Boot Lock Bits for a Flexible Protection)
- Separate Fuse to Select Reset Vector
- Optimized Page(1) Size
- Code Efficient Algorithm
- Efficient Read-Modify-Write Support

Note: 1. A page is a section in the Flash consisting of several bytes (see Table. No. of Words in a Page and No. of Pages in the Flash in *Page Size*) used during programming. The page organization does not affect normal operation.

Related Links

Page Size on page 369

27.2. Overview

In this device, the Boot Loader Support provides a real Read-While-Write Self-Programming mechanism for downloading and uploading program code by the MCU itself. This feature allows flexible application software updates controlled by the MCU using a Flash-resident Boot Loader program. The Boot Loader program can use any available data interface and associated protocol to read code and write (program) that code into the Flash memory, or read the code from the program memory. The program code within the Boot Loader section has the capability to write into the entire Flash, including the Boot Loader memory. The Boot Loader can thus even modify itself, and it can also erase itself from the code if the feature is not needed anymore. The size of the Boot Loader memory is configurable with fuses and the Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

27.3. Application and Boot Loader Flash Sections

The Flash memory is organized in two main sections, the Application section and the Boot Loader section. The size of the different sections is configured by the BOOTSZ Fuses. These two sections can have different level of protection since they have different sets of Lock bits.

27.3.1. Application Section

The Application section is the section of the Flash that is used for storing the application code. The protection level for the Application section can be selected by the application Boot Lock bits (Boot Lock bits 0). The Application section can never store any Boot Loader code since the SPM instruction is disabled when executed from the Application section.

27.3.2. BLS – Boot Loader Section

While the Application section is used for storing the application code, the Boot Loader software must be located in the BLS since the SPM instruction can initiate a programming when executing from the BLS only. The SPM instruction can access the entire Flash, including the BLS itself. The protection level for the Boot Loader section can be selected by the Boot Loader Lock bits (Boot Lock bits 1).

28. MEMPROG- Memory Programming

28.1. Program And Data Memory Lock Bits

The devices provides Lock bits. These can be left unprogrammed ('1') or can be programmed ('0') to obtain the additional features listed in Table. Lock Bit Protection Modes in this section. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 28-1. Lock Bit Byte(1)

Lock Bit Byte	Bit No.	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
BLB12	5	Boot Lock bit	1 (unprogrammed)
BLB11	4	Boot Lock bit	1 (unprogrammed)
BLB02	3	Boot Lock bit	1 (unprogrammed)
BLB01	2	Boot Lock bit	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note:

1. '1' means unprogrammed, '0' means programmed.

Table 28-2. Lock Bit Protection Modes(1)(2)

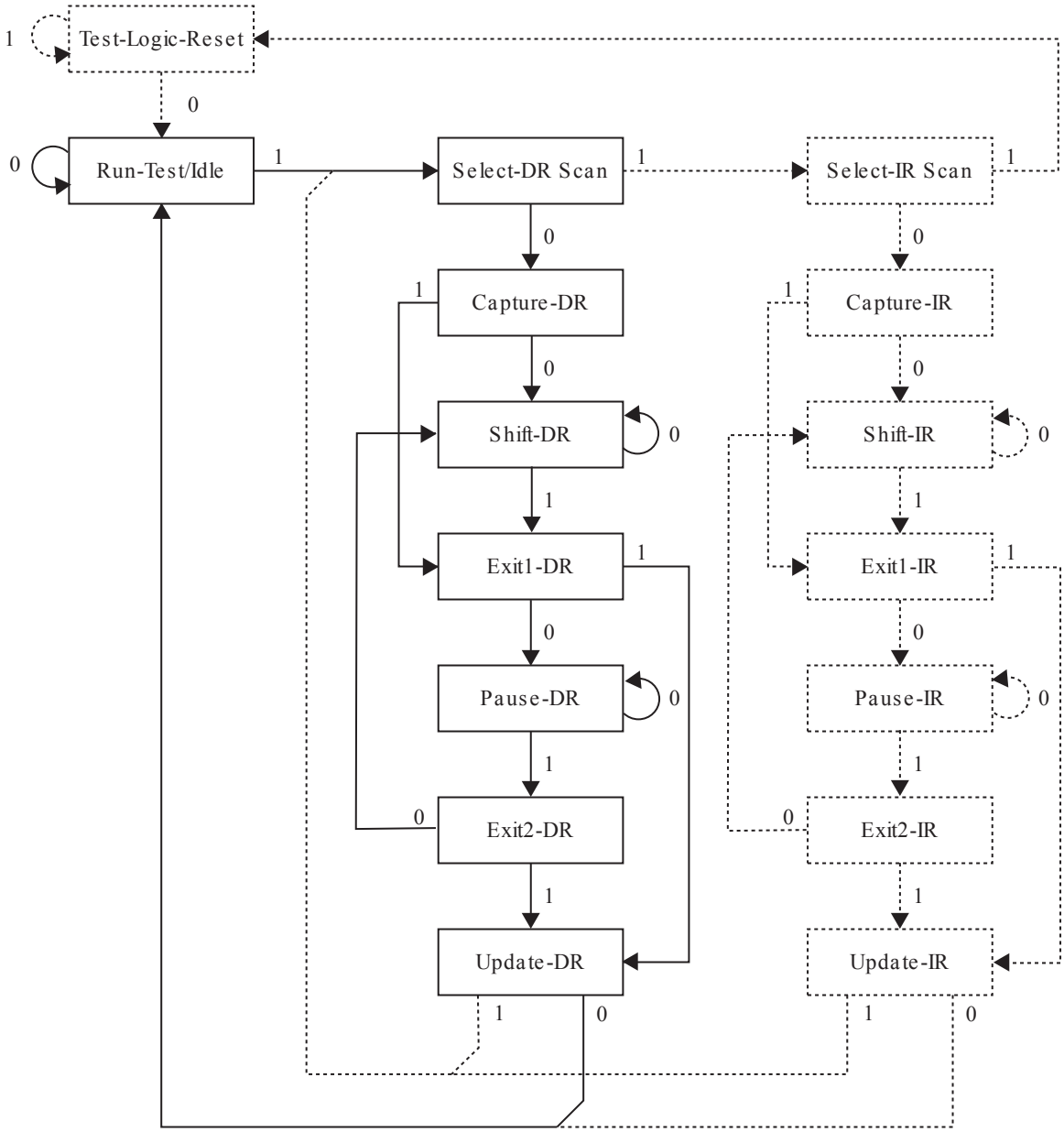
Memory Lock Bits			Protection Type
LB Mode	LB2	LB1	
1	1	1	No memory lock features enabled.
2	1	0	Further programming of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Fuse bits are locked in both Serial and Parallel Programming mode.(1)
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in Parallel and Serial Programming mode. The Boot Lock bits and Fuse bits are locked in both Serial and Parallel Programming mode.(1)

Note:

1. Program the Fuse bits and Boot Lock bits before programming the LB1 and LB2.
2. '1' means unprogrammed, '0' means programmed.

Instruction	TDI sequence	TDO sequence	Notes
2g. Write Flash Page	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2h. Poll for Page Write complete	0110111_00000000	xxxxxox_xxxxxxxx	(2)
3a. Enter Flash Read	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Load Address High Byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
3c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
3d. Read Data Low and High Byte	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo xxxxxxx_oooooooo	low byte high byte
4a. Enter EEPROM Write	0100011_00010001	xxxxxxx_xxxxxxxx	
4b. Load Address High Byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
4c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
4d. Load Data Byte	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
4e. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4f. Write EEPROM Page	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4g. Poll for Page Write complete	0110011_00000000	xxxxxox_xxxxxxxx	(2)
5a. Enter EEPROM Read	0100011_00000011	xxxxxxx_xxxxxxxx	
5b. Load Address High Byte	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
5c. Load Address Low Byte	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
5d. Read Data Byte	0110011_bbbbbbbb 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
6a. Enter Fuse Write	0100011_01000000	xxxxxxx_xxxxxxxx	
6b. Load Data Low Byte(6)	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	(3)

Figure 28-12. State Machine Sequence for Changing/Reading the Data Word



28.10.11. Virtual Flash Page Load Register

The Virtual Flash Page Load Register is a virtual scan chain with length equal to the number of bits in one Flash page. Internally the Shift Register is 8-bit, and the data are automatically transferred to the Flash page buffer byte by byte. Shift in all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page. This provides an efficient way to load the entire Flash page buffer before executing Page Write.

Table 29-4. ATmega324PA DC Characteristics - T_A = -40°C to 105°C, V_{CC} = 1.8V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
I_{CC}	Power Supply Current(1)	Active 1MHz, V_{CC} = 2V	-	-	0.7	mA
		Active 4MHz, V_{CC} = 3V	-	-	3	
		Active 8MHz, V_{CC} = 5V	-	-	11	
		Idle 1MHz, V_{CC} = 2V	-	-	0.17	
		Idle 4MHz, V_{CC} = 3V	-	-	0.85	
		Idle 8MHz, V_{CC} = 5V	-	-	6.0	
	Power-down mode(2)	WDT enabled, V_{CC} = 3V	-	-	15	µA
		WDT disabled, V_{CC} = 3V	-	-	5	

Note:

1. All bits set in the ”PRR – Power Reduction Register ”
2. The current consumption values include input leakage current.

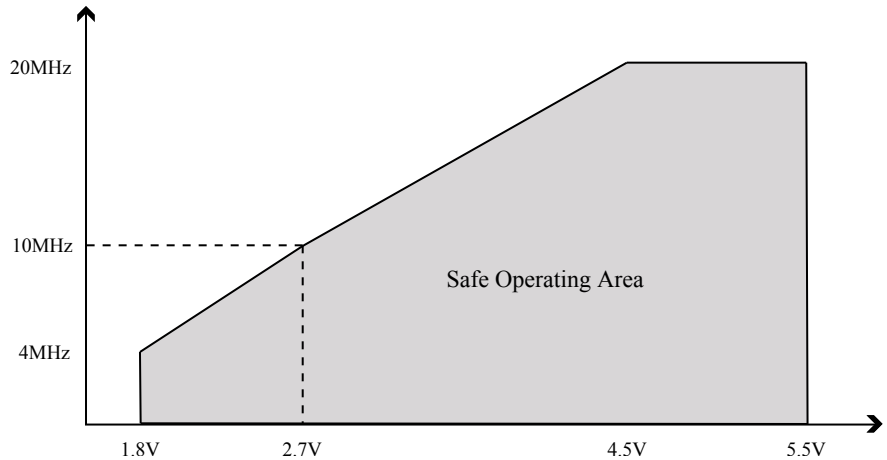
Related Links

PRR0 on page 69

29.3. Speed Grades

Maximum frequency is dependent on V_{CC}. As shown in Figure. Maximum Frequency vs. V_{CC}, the Maximum Frequency vs. V_{CC} curve is linear between 1.8V < V_{CC} < 2.7V and between 2.7V < V_{CC} < 4.5V.

Figure 29-1. Maximum Frequency vs. V_{CC}



29.4. Clock Characteristics

Related Links

Calibrated Internal RC Oscillator on page 51

30.13. Current Consumption in Reset and Reset Pulse Width

Figure 30-46. Reset supply current vs. low frequency (0.1 - 1.0Mhz)

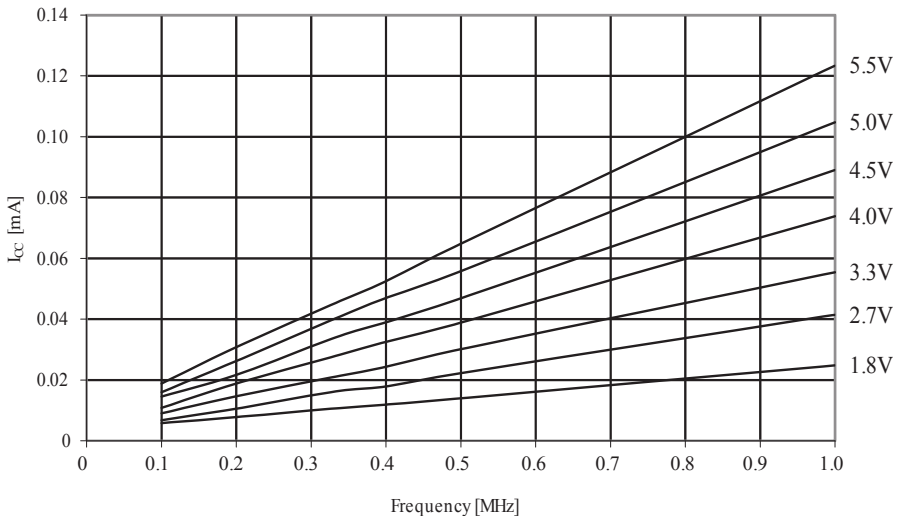
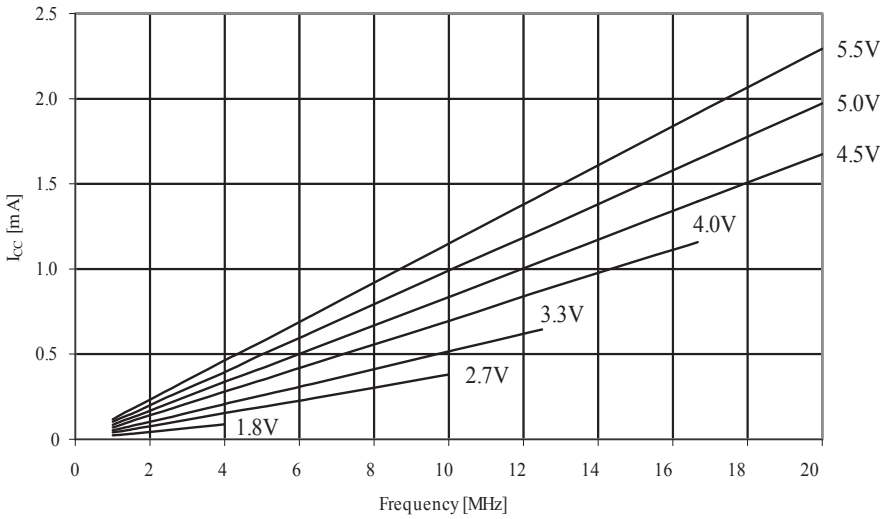


Figure 30-47. Reset supply current vs. frequency (1 - 20Mhz)



Offset	Name	Bit Pos.								
0x82	TCCR1C	7:0	FOC1A	FOC1B						
0x83	Reserved									
0x84	TCNT1L and	7:0	TCNT1[7:0]							
0x85	TCNT1H	15:8	TCNT1[15:8]							
0x86	ICR1L and ICR1H	7:0	ICR1[7:0]							
0x87		15:8	ICR1[15:8]							
0x88	OCR1AL and	7:0	OCR1A[7:0]							
0x89	OCR1AH	15:8	OCR1A[15:8]							
0x8A	OCR1BL and	7:0	OCR1B[7:0]							
0x8B	OCR1BH	15:8	OCR1B[15:8]							
0x8C ... 0xAF	Reserved									
0xB0	TCCR2A	7:0	COM2A1	COM2A0	COM2B1	COM2B0			WGM21	WGM20
0xB1	TCCR2B	7:0	FOC2A	FOC2B			WGM22	CS2[2:0]		
0xB2	TCNT2	7:0	TCNT2[7:0]							
0xB3	OCR2A	7:0	OCR2A[7:0]							
0xB4	OCR2B	7:0	OCR2B[7:0]							
0xB5	Reserved									
0xB6	ASSR	7:0		EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB
0xB7	Reserved									
0xB8	TWBR	7:0	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0
0xB9	TWSR	7:0	TWS7	TWS6	TWS5	TWS4	TWS3		TWPS[1:0]	
0xBA	TWAR	7:0	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
0xBB	TWDR	7:0	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0
0xBC	TWCR	7:0	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN		TWIE
0xBD	TWAMR	7:0	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	
0xBE ... 0xBF	Reserved									
0xC0	UCSR0A	7:0	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
0xC1	UCSR0B	7:0	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
0xC2	UCSR0C	7:0	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
0xC3	Reserved									
0xC4	UBRR0L and	7:0	UBRR[7:0]							
0xC5	UBRR0H	15:8					UBRR[11:8]			
0xC6	UDR0	7:0	TXB / RXB[7:0]							
0xC7	Reserved									
0xC8	UCSR1A	7:0	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
0xC9	UCSR1B	7:0	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
0xCA	UCSR1C	7:0	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
0xCB	Reserved									
0xCC	UBRR1L and	7:0	UBRR[7:0]							
0xCD	UBRR1H	15:8					UBRR[11:8]			
0xCE	UDR1	7:0	TXB / RXB[7:0]							