Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

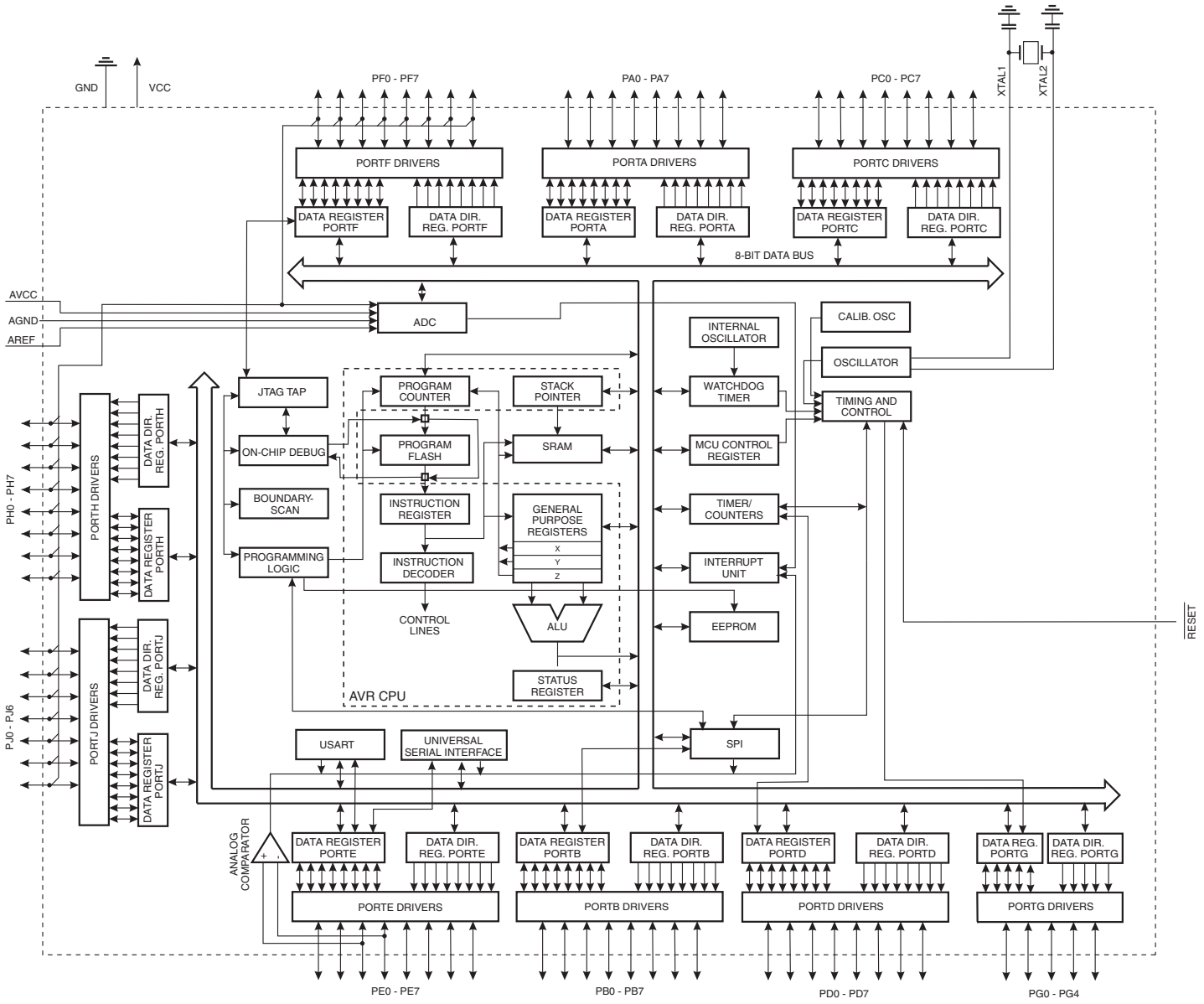
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	69
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)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega3250p-20aur

2.1 Block Diagram

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 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.

The write access time for the EEPROM is given in Table 7-1. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See Section “7.4.3” on page 22. for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

7.4.2 EEPROM Write During Power-down Sleep Mode

When entering Power-down sleep mode while an EEPROM write operation is active, the EEPROM write operation will continue, and will complete before the Write Access time has passed. However, when the write operation is completed, the clock continues running, and as a consequence, the device does not enter Power-down entirely. It is therefore recommended to verify that the EEPROM write operation is completed before entering Power-down.

7.4.3 Preventing EEPROM Corruption

During periods of low $V_{CC,}$ the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low V_{CC} reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

7.5 I/O Memory

The I/O space definition of the ATmega325P/3250P is shown in ”Register Summary” on page 342.

All ATmega325P/3250P I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega325P/3250P is a complex microcontroller with more peripheral units than can be supported within the 64 location

```
0x0032     RESET:    ldi      r16, high(RAMEND)     ; Main program start
0x0033               out      SPH,r16               ; Set Stack Pointer to top of RAM
0x0034               ldi      r16, low(RAMEND)
0x0035               out      SPL,r16
0x0036               sei                            ; Enable interrupts
0x0037               <inst    xxx
                     r>
           ...       ...      ...
```

When the BOOTRST Fuse is unprogrammed, the Boot section size set to 4K 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:

```
Address  Labels Code                 Comments
0x0000   RESET: 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
0x0004          sei                    ; Enable interrupts
0x0005          <instr> xxx
;
.org 0x3802/0x7802
0x3804/0x7804   jmp    EXT_INT0        ; IRQ0 Handler
0x3806/0x7806   jmp    PCINT0          ; PCINT0 Handler
...             ...    ...             ;
0x1C2C          jmp    SPM_RDY         ; Store Program Memory Ready Handler
```

When the BOOTRST Fuse is programmed and the Boot section size set to 4K bytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address  Labels Code             Comments
.org 0x0002
0x0002          jmp    EXT_INT0        ; IRQ0 Handler
0x0004          jmp    PCINT0          ; PCINT0 Handler
...             ...    ...             ;
0x002C          jmp    SPM_RDY         ; Store Program Memory Ready Handler
;
.org 0x3800/0x7800
0x3800/0x7801RESET:ldir16,high(RAMEND); Main program start
0x3801/0x7801   out    SPH,r16       ; Set Stack Pointer to top of RAM
0x3802/0x7802   ldi    r16,low(RAMEND)
0x3803/0x7803   out    SPL,r16
0x3804/0x7804   sei                    ; Enable interrupts
0x3805/0x7805   <instr>  xxx
```

When the BOOTRST Fuse is programmed, the Boot section size set to 4K 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:

```
Address  Labels Code                  Comments
;
.org 0x3800/0x7800
0x3800/0x7800  jmp    RESET           ; Reset handler
0x3802/0x7802  jmp    EXT_INT0        ; IRQ0 Handler
0x3804/0x7804  jmp    PCINT0          ; PCINT0 Handler
...            ...    ...             ;
0x382C/0x782C  jmp    SPM_RDY    ; Store Program Memory Ready Handler
;
0x382E/0x782ERESET:ldir16,high(RAMEND); Main program start
0x382F/0x782F  out    SPH,r16       ; Set Stack Pointer to top of RAM
0x3830/0x7830  ldi    r16,low(RAMEND)
0x3831/0x7831  out    SPL,r16
0x3832/0x7832  sei                    ; Enable interrupts
0x3833/0x7833  <instr>  xxx
```

11.2.1 Moving Interrupts Between Application and Boot Space

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

11.3 Register Description

11.3.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	JTD	BODS	BODSE	PUD	–	–	IVSEL	IVCE	MCUCR
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 256 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 256 for details on Boot Lock bits.

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0A pin. Setting the COM0A1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM0A1:0 to three (See Table 14-4 on page 103). The actual OC0A value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0A Register at the compare match between OCR0A and TCNT0, and clearing (or setting) the OC0A Register at the timer clock cycle the counter is cleared (changes from MAX to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

The *N* variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM0A1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0A to toggle its logical level on each compare match (COM0A1:0 = 1). The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is set to zero. This feature is similar to the OC0A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

14.7.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM01:0 = 1) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to MAX and then from MAX to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0A) is cleared on the compare match between TCNT0 and OCR0A while counting up, and set on the compare match while counting down. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode is fixed to eight bits. In phase correct PWM mode the counter is incremented until the counter value matches MAX. When the counter reaches MAX, it changes the count direction. The TCNT0 value will be equal to MAX for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 14-7. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0A and TCNT0.

16.9 Asynchronous Operation of Timer/Counter2

When Timer/Counter2 operates asynchronously, some considerations must be taken.

- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter2, the Timer Registers TCNT2, OCR2A, and TCCR2A might be corrupted. A safe procedure for switching clock source is:

1. Disable the Timer/Counter2 interrupts by clearing OCIE2A and TOIE2.
2. Select clock source by setting AS2 as appropriate.
3. Write new values to TCNT2, OCR2A, and TCCR2A.
4. To switch to asynchronous operation: Wait for TCN2UB, OCR2UB, and TCR2UB.
5. Clear the Timer/Counter2 Interrupt Flags.
6. Enable interrupts, if needed.

- The CPU main clock frequency must be more than four times the Oscillator frequency.
- When writing to one of the registers TCNT2, OCR2A, or TCCR2A, the value is transferred to a temporary register, and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the three mentioned registers have their individual temporary register, which means that e.g. writing to TCNT2 does not disturb an OCR2A write in progress. To detect that a transfer to the destination register has taken place, the Asynchronous Status Register – ASSR has been implemented.
- When entering Power-save or ADC Noise Reduction mode after having written to TCNT2, OCR2A, or TCCR2A, the user must wait until the written register has been updated if Timer/Counter2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if the Output Compare2 interrupt is used to wake up the device, since the Output Compare function is disabled during writing to OCR2A or TCNT2. If the write cycle is not finished, and the MCU enters sleep mode before the OCR2UB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
- If Timer/Counter2 is used to wake the device up from Power-save or ADC Noise Reduction mode, precautions must be taken if the user wants to re-enter one of these modes: The interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and reentering sleep mode is less than one TOSC1 cycle, correct interrupt handling is not guaranteed. If the user is in doubt whether the time before re-entering Power-save or ADC Noise Reduction mode is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:

1. Write a value to TCCR2A, TCNT2, or OCR2A.
2. Wait until the corresponding Update Busy Flag in ASSR returns to zero.
3. Enter Power-save or ADC Noise Reduction mode.

- When the asynchronous operation is selected, the 32.768 kHz Oscillator for Timer/Counter2 is always running, except in Power-down and Standby modes. After a Power-up Reset or wake-up from Power-down or Standby mode, the user should be aware of the fact that this Oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter2 after power-up or wake-up from Power-down or Standby mode. The contents of all Timer/Counter2 Registers must be considered lost after a wake-up from Power-down or Standby mode due to unstable clock signal upon start-up, no matter whether the Oscillator is in use or a clock signal is applied to the TOSC1 pin.

- **Bit 6 – SPE: SPI Enable**

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD: Data Order**

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

- **Bit 4 – MSTR: Master/Slave Select**

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If $\overline{SS}$ is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

- **Bit 3 – CPOL: Clock Polarity**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 17-3 on page 160 and Figure 17-4 on page 160 for an example. The CPOL functionality is summarized below:

Table 17-2. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

- **Bit 2 – CPHA: Clock Phase**

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 17-3 on page 160 and Figure 17-4 on page 160 for an example. The CPOL functionality is summarized below:

Table 17-3. CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

- **Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency f_{osc} is shown in the following table:

Table 17-4. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$

bits of the data read from the UDRn will be masked to zero. The USART has to be initialized before the function can be used.

Assembly Code Example[1]

```
USART_Receive:
  ; Wait for data to be received
  sbis UCSR0A, RXC0
  rjmp USART_Receive
  ; Get and return received data from buffer
  in   r16, UDR0
  ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSR0A & (1<<RXC0)) )
       ;
  /* Get and return received data from buffer */
  return UDR0;
}
```

Note: 1. See Section “5.” on page 10.

The function simply waits for data to be present in the receive buffer by checking the RXCn Flag, before reading the buffer and returning the value.

18.7.2 Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZ=7), the ninth bit must be read from the RXB8n bit in UCSRnB before reading the low bits from the UDRn. This rule applies to the FEn, DORn and UPEn Status Flags as well. Read status from UCSRnA, then data from UDRn. Reading the UDRn I/O location will change the state of the receive buffer FIFO and consequently the TXB8n, FEn, DORn and UPEn bits, which all are stored in the FIFO, will change.

18.10 Register Description

18.10.1 UDRn – USART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	
	RXBn[7:0]								UDRn (Read)
	TXBn[7:0]								UDRn (Write)
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 USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDRn. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDRn Register location. Reading the UDRn Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDREn Flag in the UCSRnA Register is set. Data written to UDRn when the UDREn Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

18.10.2 UCSRnA – USART Control and Status Register n A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXCn: USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXCn bit will become zero. The RXCn Flag can be used to generate a Receive Complete interrupt (see description of the RXCIEn bit).

- **Bit 6 – TXCn: USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDRn). The TXCn Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXCn Flag can generate a Transmit Complete interrupt (see description of the TXCIEn bit).

Figure 18-13. UCPOLn Bit Settings

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

18.10.5 UBRRnL and UBRRnH – USART Baud Rate Registers n

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	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	

- **Bit 15:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRnH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

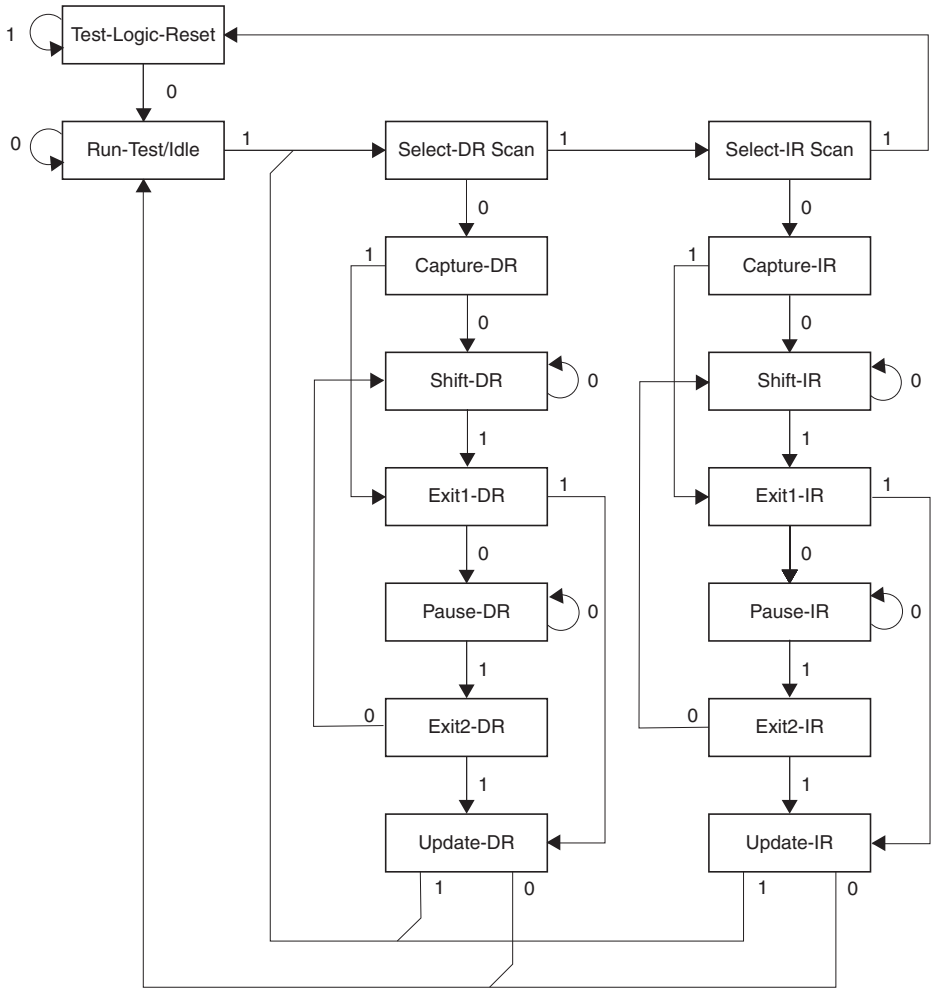
This is a 12-bit register which contains the USART baud rate. The UBRRnH contains the four most significant bits, and the UBRRnL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRnL will trigger an immediate update of the baud rate prescaler.

18.11 Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRRn settings in Table 18-3. UBRRn values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the Receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see ”Asynchronous Operational Range” on page 179). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

Figure 22-2. TAP Controller State Diagram



22.4 TAP Controller

The TAP controller is a 16-state finite state machine that controls the operation of the Boundary-scan circuitry, JTAG programming circuitry, or On-chip Debug system. The state transitions depicted in Figure 22-2 depend on the signal present on TMS (shown adjacent to each state transition) at the time of the rising edge at TCK. The initial state after a Power-on Reset is Test-Logic-Reset.

As a definition in this document, the LSB is shifted in and out first for all Shift Registers.

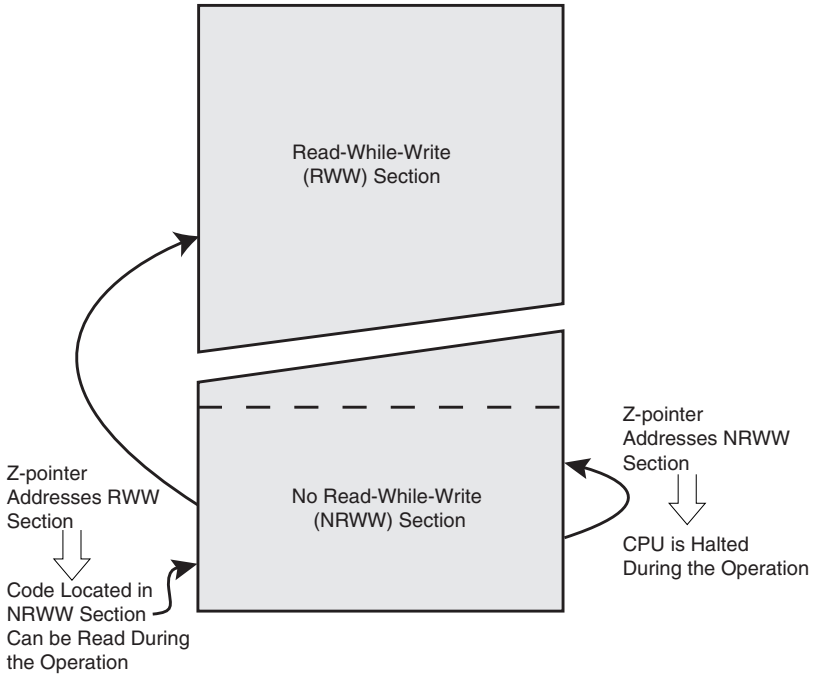
Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG interface is:

- At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the Shift Instruction Register – Shift-IR state. While in this state, shift the four bits of the JTAG instructions into the JTAG Instruction Register from the TDI input at the rising edge of TCK. The TMS input must be held low during input of the 3 LSBs in order to remain in the Shift-IR state. The MSB of the instruction is shifted in when this state is left by setting TMS high. While the instruction is shifted in from the TDI pin, the captured IR-state 0x01 is shifted out on the TDO pin. The JTAG Instruction selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.

Table 23-7. ATmega3250P Boundary-scan Order, 100-pin (Continued)

Bit Number	Signal Name	Module
135	EXTCLK (XTAL1)	Clock input and Oscillators for the main clock (Observe-only)
134	OSCCK	
133	RCCK	
132	OSC32CK	
131	PJ2.Data	Port J
130	PJ2.Control	
129	PJ2.Pull-up_Enable	
128	PJ3.Data	
127	PJ3.Control	
126	PJ3.Pull-up_Enable	
125	PJ4.Data	
124	PJ4.Control	
123	PJ4.Pull-up_Enable	
122	PJ5.Data	
121	PJ5.Control	
120	PJ5.Pull-up_Enable	
119	PJ6.Data	
118	PJ6.Control	
117	PJ6.Pull-up_Enable	
116	PD0.Data	Port D
115	PD0.Control	
114	PD0.Pull-up_Enable	
113	PD1.Data	
112	PD1.Control	
111	PD1.Pull-up_Enable	
110	PD2.Data	
109	PD2.Control	
108	PD2.Pull-up_Enable	
107	PD3.Data	
106	PD3.Control	
105	PD3.Pull-up_Enable	
104	PD4.Data	
103	PD4.Control	
102	PD4.Pull-up_Enable	
101	PD5.Data	
100	PD5.Control	

Figure 24-1. Read-While-Write vs. No Read-While-Write



```
  sbiw loophi:looplo, 1          ;use subi for PAGESIZEB<=256
  brne Rdloop

  ; return to RWW section
  ; verify that RWW section is safe to read
Return:
  in   temp1, SPMCSR
  sbrs temp1, RWWSB     ; If RWWSB is set, the RWW section is not ready yet
  ret
  ; re-enable the RWW section
  ldi  spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  call Do_spm
  rjmp Return

Do_spm:
  ; check for previous SPM complete
Wait_spm:
  in   temp1, SPMCSR
  sbrc temp1, SPMEN
  rjmp Wait_spm
  ; input: spmcrval determines SPM action
  ; disable interrupts if enabled, store status
  in   temp2, SREG
  cli
  ; check that no EEPROM write access is present
Wait_ee:
  sbic EECR, EEWE
  rjmp Wait_ee
  ; SPM timed sequence
  out  SPMCSR, spmcrval
  spm
  ; restore SREG (to enable interrupts if originally enabled)
  out  SREG, temp2
  ret
```

Table 25-2. Lock Bit Protection Modes[1][2] (Continued)

Memory Lock Bits			Protection Type
BLB1 Mode	**BLB12**	**BLB11**	
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If Interrupt Vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

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

25.2 Fuse Bits

The ATmega325P/3250P has three Fuse bytes. Table 25-3 - Table 25-5 describe briefly the functionality of all the fuses and how they are mapped into the Fuse bytes. Note that the fuses are read as logical zero, "0", if they are programmed.

Table 25-3. Extended Fuse Byte

Extended Fuse Byte	Bit No	Description	Default Value
–	7	–	1
–	6	–	1
–	5	–	1
–	4	–	1
–	5	–	1
BODLEVEL1[1]	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0[1]	1	Brown-out Detector trigger level	1 (unprogrammed)
RSTDISBL[2]	0	External Reset Disable	1 (unprogrammed)

Notes: 1. See "System and Reset Characterizations" on page 308 for BODLEVEL Fuse decoding.
2. Port G, PG5 is input only. Pull-up is always on. See "Alternate Functions of Port G" on page 79

Table 25-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
OCDEN[4]	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN[5]	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN[1]	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
WDTON[3]	4	Watchdog Timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 25-7 on page 275 for details)	0 (programmed)[2]
BOOTSZ0	1	Select Boot Size (see Table 25-7 on page 275 for details)	0 (programmed)[2]
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

Notes:
1. The SPIEN Fuse is not accessible in serial programming mode.
2. The default value of BOOTSZ1..0 results in maximum Boot Size. See Table 24-6 on page 268 for details.
3. See "WDTCR – Watchdog Timer Control Register" on page 51 for details.
4. Never ship a product with the OCDEN Fuse programmed regardless of the setting of Lock bits and JTAGEN Fuse. A programmed OCDEN Fuse enables some parts of the clock system to be running in all sleep modes. This may increase the power consumption.
5. If the JTAG interface is left unconnected, the JTAGEN fuse should if possible be disabled. This to avoid static current at the TDO pin in the JTAG interface.

Table 25-5. Fuse Low Byte

Fuse Low Byte	Bit No	Description	Default Value
CKDIV8[4]	7	Divide clock by 8	0 (programmed)
CKOUT[3]	6	Clock output	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)[1]
SUT0	4	Select start-up time	0 (programmed)[1]
CKSEL3	3	Select Clock source	0 (programmed)[2]
CKSEL2	2	Select Clock source	0 (programmed)[2]
CKSEL1	1	Select Clock source	1 (unprogrammed)[2]
CKSEL0	0	Select Clock source	0 (programmed)[2]

Notes:
1. The default value of SUT1..0 results in maximum start-up time for the default clock source. See "System and Reset Characterizations" on page 308 for details.
2. The default setting of CKSEL3..0 results in internal RC Oscillator @ 8 MHz. See Table 8-6 on page 33 for details.
3. The CKOUT Fuse allow the system clock to be output on PORTE7. See "Clock Output Buffer" on page 35 for details.
4. See "System Clock Prescaler" on page 35 for details.

The status of the Fuse bits is not affected by Chip Erase. Note that the Fuse bits are locked if Lock bit1 (LB1) is programmed. Program the Fuse bits before programming the Lock bits.

Table 26-5. Reset, Brown-out and Internal Voltage Reference Characteristics[1], T_A = -40°C to 85°C

Symbol	Parameter	Min	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)	1.1	1.4	1.6	V
	Power-on Reset Threshold Voltage (falling)[2]	0.6	1.3	1.6	V
SR_{ON}	Power-on Slope Rate	0.01		10	V/ms

Notes: 1. Values are guidelines only. Actual values are TBD.
2. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling)

26.8 Brown-out Detection

Table 26-6. BODLEVEL Fuse Coding[1]

BODLEVEL 1:0 Fuses	Min V_{BOT}	Typ V_{BOT}	Max V_{BOT}	Units
11	BOD Disabled			
10	1.7	1.8	2	V
01	2.5	2.7	2.9	
00	4.1	4.3	4.5	

Note: 1. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the production test. This guarantees that a Brown-Out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 10 for ATmega325P/3250PV and BODLEVEL = 01 for ATmega325P/3250PL.

Figure 27-14. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 2.7 V)

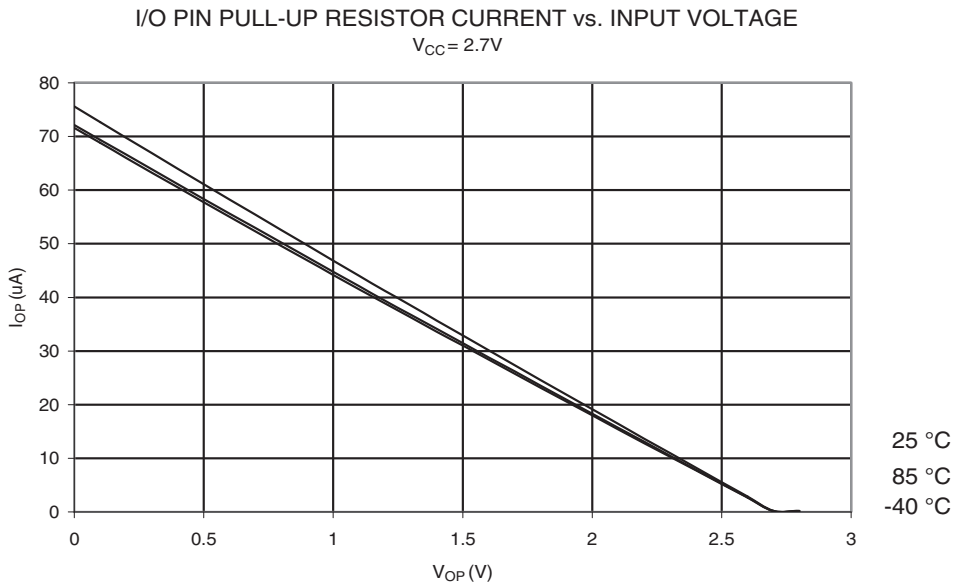


Figure 27-15. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 5 V)

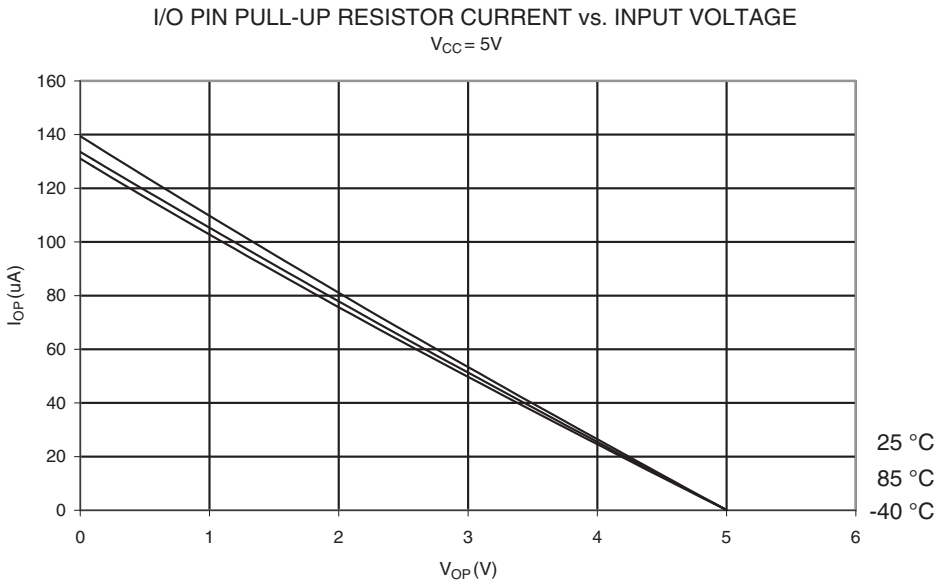


Figure 27-26. I/O Pin Source Current vs. Output Voltage, Port B (V_{CC} = 2.7 V)

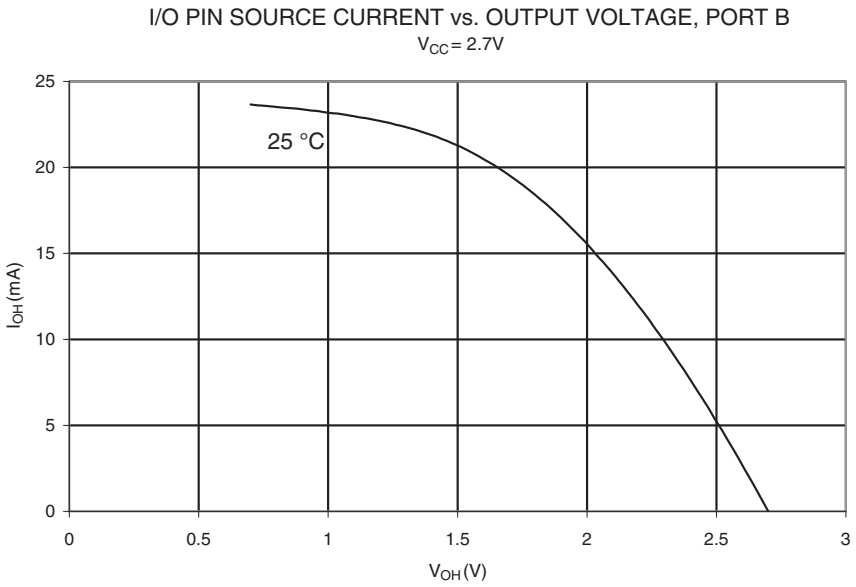
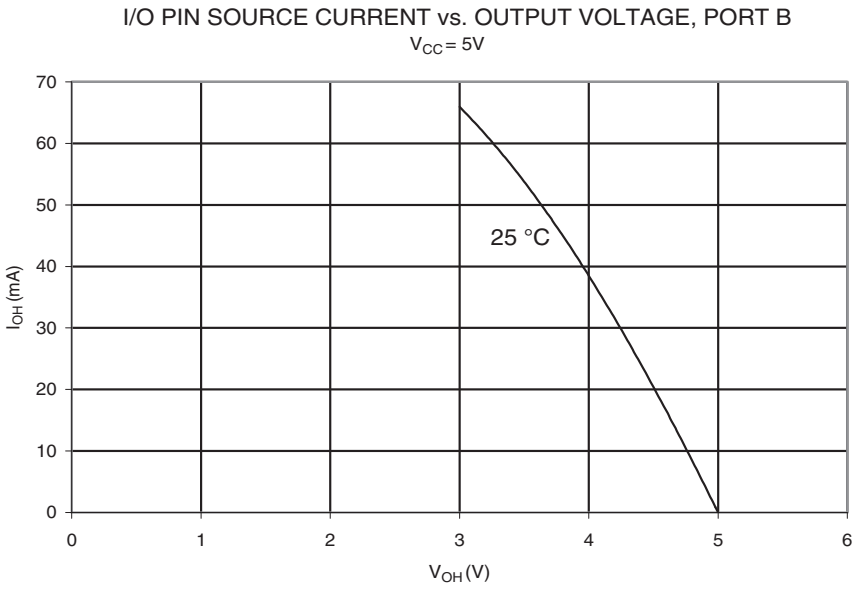


Figure 27-27. I/O Pin Source Current vs. Output Voltage, Port B (V_{CC} = 5 V)



32.4 ATmega3250P rev. A

- **Interrupts may be lost when writing the timer registers in the asynchronous timer.**
- **Using BOD disable will make the chip reset.**

1. **Interrupts may be lost when writing the timer registers in the asynchronous timer.**

   The interrupt will be lost if a timer register that is synchronous timer clock is written when the asynchronous Timer/Counter register (TCNTx) is 0x00.

   Problem Fix/Workaround

   Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register (TCCRx), asynchronous Timer Counter Register (TCNTx), or asynchronous Output Compare Register (OCRx).

2. **Using BOD disable will make the chip reset.**

   If the part enters sleep with the BOD turned off with the BOD disable option enabled, a BOD reset will be generated at wakeup and the chip will reset.

   Problem Fix/Workaround

   Do not use BOD disable

32.5 ATmega3250P rev. B

- **Interrupts may be lost when writing the timer registers in the asynchronous timer.**

1. **Interrupts may be lost when writing the timer registers in the asynchronous timer.**

   The interrupt will be lost if a timer register that is synchronous timer clock is written when the asynchronous Timer/Counter register (TCNTx) is 0x00.

   Problem Fix/Workaround

   Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register (TCCRx), asynchronous Timer Counter Register (TCNTx), or asynchronous Output Compare Register (OCRx).

32.6 ATmega3250P rev. C

- **Interrupts may be lost when writing the timer registers in the asynchronous timer.**

1. **Interrupts may be lost when writing the timer registers in the asynchronous timer.**

   The interrupt will be lost if a timer register that is synchronous timer clock is written when the asynchronous Timer/Counter register (TCNTx) is 0x00.

   Problem Fix/Workaround

   Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register (TCCRx), asynchronous Timer Counter Register (TCNTx), or asynchronous Output Compare Register (OCRx).