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	27
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K 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	32-TQFP
Supplier Device Package	32-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega168pb-an

12.3.1. Status 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.

Name: SREG
Offset: 0x5F
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x3F

Bit	7	6	5	4	3	2	1	0
	I	T	H	S	V	N	Z	C
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 – I: Global Interrupt Enable
The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

Bit 6 – T: Copy Storage
The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

Bit 5 – H: Half Carry Flag
The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry Flag is useful in BCD arithmetic. See the *Instruction Set Description* for detailed information.

Bit 4 – S: Sign Flag, S = N ⊕ V

The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V. See the *Instruction Set Description* for detailed information.

Bit 3 – V: Two's Complement Overflow Flag
The Two's Complement Overflow Flag V supports two's complement arithmetic. See the *Instruction Set Description* for detailed information.

Bit 2 – N: Negative Flag
The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the *Instruction Set Description* for detailed information.

Bit 1 – Z: Zero Flag
The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the *Instruction Set Description* for detailed information.

13.6.7. GPIOR0 – General Purpose I/O Register 0

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.

Name: GPIOR0
Offset: 0x3E
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1E

Bit	7	6	5	4	3	2	1	0
	GPIOR0[7:0]							
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 – GPIOR0[7:0]: General Purpose I/O

13.6.8. Unique Device ID

Each individual part has a specific unique device ID. This can be used to identify a specify part while it is in the field. The Unique Device ID consists of nine serial number bytes in which the user can access directly from registers. The register address locations are located at 0xF0 to 0xF8.

13.6.8.1. SNOBRx - Serial Number Byte 8 to 0

	7	6	5	4	3	2	1	0	
(0xF8)	Serial Number Byte 5								SNOBR5
(0xF7)	Serial Number Byte 4								SNOBR4
(0xF6)	Serial Number Byte 3								SNOBR3
(0xF5)	Serial Number Byte 2								SNOBR2
(0xF4)	Serial Number Byte 1								SNOBR1
(0xF3)	Serial Number Byte 0								SNOBR0
(0xF2)	Serial Number Byte 6								SNOBR6
(0xF1)	Serial Number Byte 7								SNOBR7
(0xF0)	Serial Number Byte 8								SNOBR8
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	Serial Number Byte Value								

16.9.2. Watchdog Timer Control Register

Name: WDTCSR
Offset: 0x60
Reset: 0x00 / 0x08
Property: -

Bit	7	6	5	4	3	2	1	0
	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	x	0	0	0

Bit 7 – WDIF: Watchdog Interrupt Flag
This bit is set when a time-out occurs in the Watchdog Timer and the Watchdog Timer is configured for interrupt. WDIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, WDIF is cleared by writing a logic one to the flag. When the I-bit in SREG and WDIE are set, the Watchdog Time-out Interrupt is executed.

Bit 6 – WDIE: Watchdog Interrupt Enable
When this bit is written to one and the I-bit in the Status Register is set, the Watchdog Interrupt is enabled. If WDE is cleared in combination with this setting, the Watchdog Timer is in Interrupt Mode, and the corresponding interrupt is executed if time-out in the Watchdog Timer occurs. If WDE is set, the Watchdog Timer is in Interrupt and System Reset Mode. The first time-out in the Watchdog Timer will set WDIF. Executing the corresponding interrupt vector will clear WDIE and WDIF automatically by hardware (the Watchdog goes to System Reset Mode).

This is useful for keeping the Watchdog Timer security while using the interrupt. To stay in Interrupt and System Reset Mode, WDIE must be set after each interrupt. This should however not be done within the interrupt service routine itself, as this might compromise the safety-function of the Watchdog System Reset mode. If the interrupt is not executed before the next time-out, a System Reset will be applied.

Table 16-1. Watchdog Timer Configuration

WDTON(1)	WDE	WDIE	Mode	Action on Time-out
1	0	0	Stopped	None
1	0	1	Interrupt Mode	Interrupt
1	1	0	System Reset Mode	Reset
1	1	1	Interrupt and System Reset Mode	Interrupt, then go to System Reset Mode
0	X	X	System Reset Mode	Reset

Note:
1. WDTON Fuse set to '0' means programmed and '1' means unprogrammed.

Bit 4 – WDCE: Watchdog Change Enable
This bit is used in timed sequences for changing WDE and prescaler bits. To clear the WDE bit, and/or change the prescaler bits, WDCE must be set.

Once written to '1', hardware will clear WDCE after four clock cycles.

Address	Labels		Code Comments
0x015	rjmp	ADC	; ADC Conversion Complete Handler
0x016	rjmp	EE_RDY	; EEPROM Ready Handler
0x017	rjmp	ANA_COMP	; Analog Comparator Handler
0x018	rjmp	TWI	; 2-wire Serial Interface Handler
0x019	rjmp	SPM_RDY	; Store Program Memory Ready Handler
;			
0x01A	RESET: ldi	r16, high(RAMEND)	; Main program start
0x01B	out	SPH,r16	; Set Stack Pointer to top of RAM
0x01C	ldi	r16, low(RAMEND)	
0x01D	out	SPL, r16	
0x01E	sei		; Enable interrupts
0x01F	<instr>	xxx	
...	...	...	...

17.2. Interrupt Vectors in ATmega88PB

Table 17-2. Reset and Interrupt Vectors in ATmega88PB

Vector No.	Program Address(2)	Source	Interrupt Definition
1	0x000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	PCINT0	Pin Change Interrupt Request 0
5	0x004	PCINT1	Pin Change Interrupt Request 1
6	0x005	PCINT2	Pin Change Interrupt Request 2
7	0x006	WDT	Watchdog Time-out Interrupt
8	0x007	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x008	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x009	TIMER2 OVF	Timer/Counter2 Overflow
11	0x00A	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x00B	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x00C	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x00D	TIMER1 OVF	Timer/Counter1 Overflow
15	0x00E	TIMER0 COMPA	Timer/Counter0 Compare Match A

```
i = PINB;
...
```

19.2.5. Digital Input Enable and Sleep Modes

As shown in the figure of General Digital I/O, 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.

If a logic high level is present on an asynchronous external interrupt pin configured as “Interrupt on Rising Edge, Falling Edge, or Any Logic Change on Pin” while the external interrupt is not enabled, the corresponding External Interrupt Flag will be set when resuming from the above mentioned Sleep mode, as the clamping in these sleep mode produces the requested logic change.

19.2.6. Unconnected Pins

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

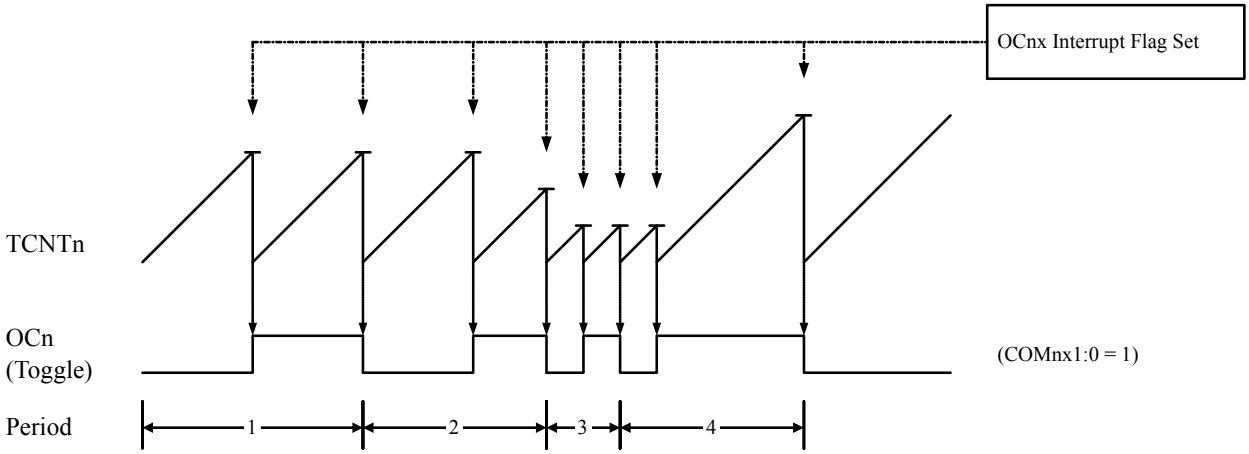
The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to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

19.3. Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. The following figure shows how the port pin control signals from the simplified Figure 19-2 General Digital I/O(1) can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

The timing diagram for the CTC mode is shown below. The counter value (TCNTn) increases until a compare match occurs between TCNTn and OCRnA, and then counter (TCNTn) is cleared.

Figure 20-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by setting the OCFnA Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value.

Note: Changing TOP to a value close to BOTTOM while the counter is running must be done with care, since the CTC mode does not provide double buffering. If the new value written to OCRnA is lower than the current value of TCNTn, the counter will miss the compare match. The counter will then count to its maximum value (0xFF for a 8-bit counter, 0xFFFF for a 16-bit counter) and wrap around starting at 0x00 before the compare match will occur.

For generating a waveform output in CTC mode, the OCnA output can be set to toggle its logical level on each compare match by writing the two least significant Compare Output mode bits in the Timer/Counter Control Register A Control to toggle mode (TCCRnA.COMnA[1:0]=0x1). The OCnA value will only be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OCn} = f_{clk\_I/O}/2$ when OCRnA is written to 0x00. The waveform frequency is defined by the following equation:

$$f_{OCnx} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnx)}$$

N represents the prescaler factor (1, 8, 64, 256, or 1024).

As for the Normal mode of operation, the Timer/Counter Overflow Flag TOVn is set in the same clock cycle that the counter wraps from MAX to 0x00.

20.7.3. Fast PWM Mode

The Fast Pulse Width Modulation or Fast PWM modes (WGM0[2:0]=0x3 or WGM0[2:0]=0x7) provide a high frequency PWM waveform generation option. The Fast PWM modes differ from the other PWM options by their single-slope operation. The counter counts from BOTTOM to TOP, then restarts from BOTTOM. TOP is defined as 0xFF when WGM0[2:0]=0x3. TOP is defined as OCR0A when WGM0[2:0]=0x7.

In non-inverting Compare Output mode, the Output Compare register (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the Fast PWM mode can be twice as high as the phase correct PWM modes, which use dual-slope operation. This high frequency makes the Fast PWM mode well suited for power regulation,

Figure 23-2. Counter Unit Block Diagram

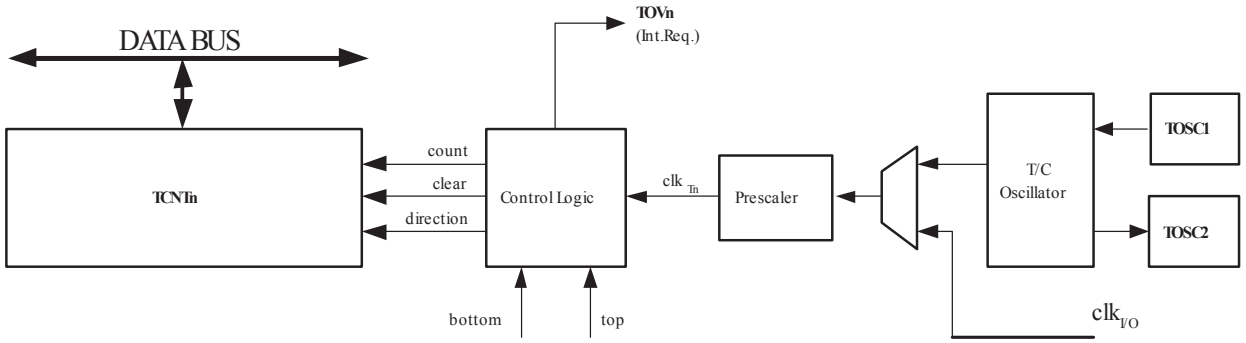


Table 23-2. Signal description (internal signals):

Signal name	Description
count	Increment or decrement TCNT2 by 1.
direction	Selects between increment and decrement.
clear	Clear TCNT2 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T2} in the following.
top	Signalizes that TCNT2 has reached maximum value.
bottom	Signalizes that TCNT2 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T2}). clk_{T2} can be generated from an external or internal clock source, selected by the Clock Select bits (CS2[2:0]). When no clock source is selected (CS2[2:0]=0x0) the timer is stopped. However, the TCNT2 value can be accessed by the CPU, regardless of whether clk_{T2} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM21 and WGM20 bits located in the Timer/Counter Control Register (TCCR2A) and the WGM22 bit located in the Timer/Counter Control Register B (TCCR2B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC2A and OC2B. For more details about advanced counting sequences and waveform generation, see "Modes of Operation".

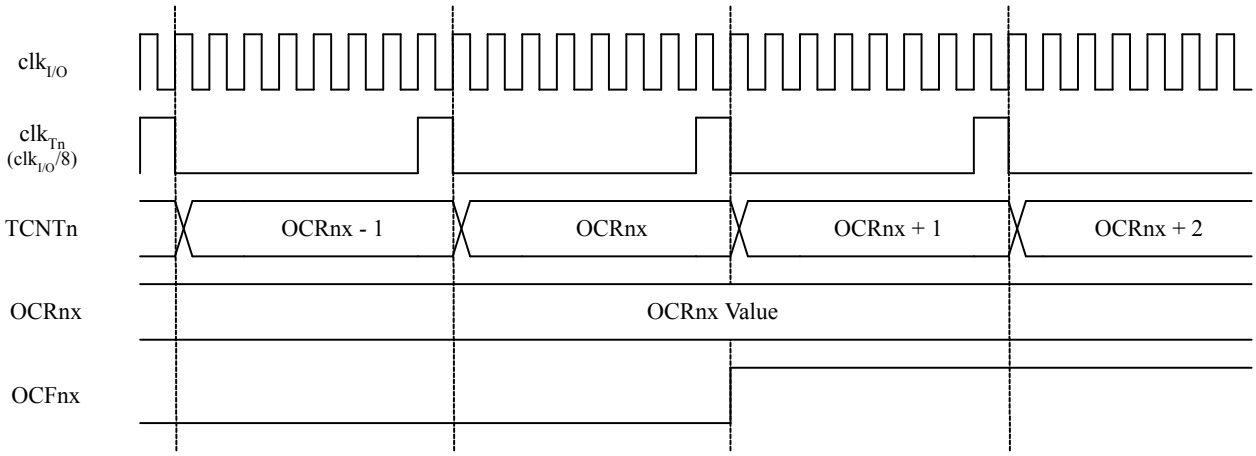
The Timer/Counter Overflow Flag (TOV2) is set according to the mode of operation selected by the TCC2B.WGM2[2:0] bits. TOV2 can be used for generating a CPU interrupt.

23.5. Output Compare Unit

The 8-bit comparator continuously compares TCNT2 with the Output Compare Register (OCR2A and OCR2B). Whenever TCNT2 equals OCR2A or OCR2B, the comparator signals a match. A match will set the Output Compare Flag (OCF2A or OCF2B) at the next timer clock cycle. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the Output Compare Flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the WGM2[2:0] bits and Compare Output mode (COM2x[1:0]) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (See Modes of Operation).

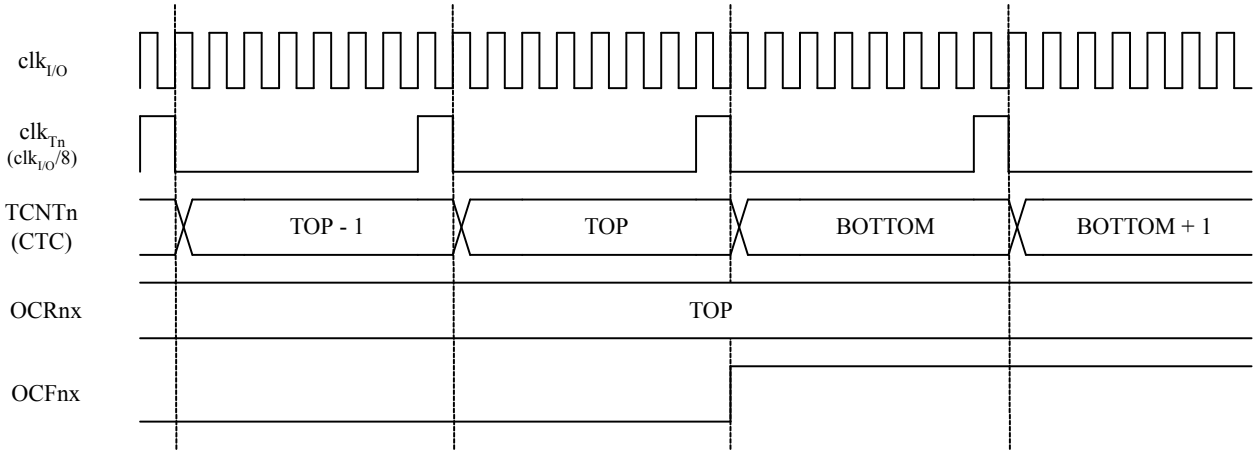
The following figure shows a block diagram of the Output Compare unit.

Figure 23-10. Timer/Counter Timing Diagram, Setting of OCF2A, with Prescaler ($f_{clk\_I/O}/8$)



The following figure shows the setting of OCF2A and the clearing of TCNT2 in CTC mode.

Figure 23-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk\_I/O}/8$)



23.9. Asynchronous Operation of Timer/Counter2

When TC2 operates asynchronously, some considerations must be taken:

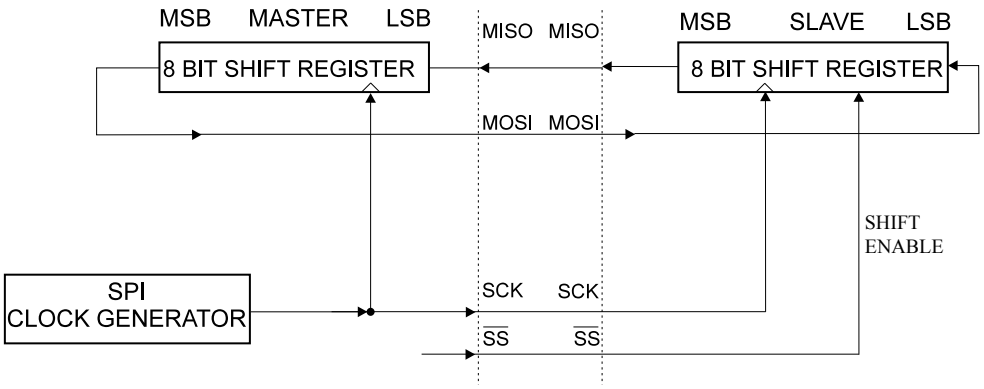
1. When switching between asynchronous and synchronous clocking of TC2, the registers TCNT2, OCR2x, and TCCR2x might be corrupted. A safe procedure for switching clock source is:
   1.1. Disable the TC2 interrupts by clearing OCIE2x and TOIE2.
   1.2. Select clock source by setting AS2 as appropriate.
   1.3. Write new values to TCNT2, OCR2x, and TCCR2x.
   1.4. To switch to asynchronous operation: Wait for TCN2xUB, OCR2xUB, and TCR2xUB.
   1.5. Clear the TC2 Interrupt Flags.
   1.6. Enable interrupts, if needed.
2. The CPU main clock frequency must be more than four times the oscillator frequency.
3. When writing to one of the registers TCNT2, OCR2x, or TCCR2x, 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 five mentioned registers has its individual temporary register, which means that e.g. writing to TCNT2 does not disturb an OCR2x write in progress. The Asynchronous Status Register (ASSR) indicates that a transfer to the destination register has taken place.

4. When entering Power-save or ADC Noise Reduction mode after having written to TCNT2, OCR2x, or TCCR2x, the user must wait until the written register has been updated if TC2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if any of the Output Compare2 interrupts is used to wake up the device, since the Output Compare function is disabled during writing to OCR2x or TCNT2. If the write cycle is not finished, and the MCU enters sleep mode before the corresponding OCR2xUB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
5. If TC2 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: If re-entering sleep mode within the TOSC1 cycle, the interrupt will immediately occur and the device wake up again. The result is multiple interrupts and wake-ups within one TOSC1 cycle from the first interrupt. 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:
    5.1. Write a value to TCCR2x, TCNT2, or OCR2x.
    5.2. Wait until the corresponding Update Busy Flag in ASSR returns to zero.
    5.3. Enter Power-save or ADC Noise Reduction mode.
6. When the asynchronous operation is selected, the 32.768kHz oscillator for TC2 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 TC2 after power-up or wake-up from Power-down or Standby mode. The contents of all TC2 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.
7. Description of wake up from Power-save or ADC Noise Reduction mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.
8. Reading of the TCNT2 Register shortly after wake-up from Power-save may give an incorrect result. Since TCNT2 is clocked on the asynchronous TOSC clock, reading TCNT2 must be done through a register synchronized to the internal I/O clock domain. Synchronization takes place for every rising TOSC1 edge. When waking up from Power-save mode, and the I/O clock ($clk_{I/O}$) again becomes active, TCNT2 will read as the previous value (before entering sleep) until the next rising TOSC1 edge. The phase of the TOSC clock after waking up from Power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading TCNT2 is thus as follows:
    8.1. Wait for the corresponding Update Busy Flag to be cleared.
    8.2. Read TCNT2.

During asynchronous operation, the synchronization of the Interrupt Flags for the asynchronous timer takes 3 processor cycles plus one timer cycle. The timer is therefore advanced by at least one before the processor can read the timer value causing the setting of the Interrupt Flag. The Output Compare pin is changed on the timer clock and is not synchronized to the processor clock.

data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the Buffer Register for later use.

Figure 24-2. SPI Master-slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the minimum low and high periods should be longer than two CPU clock cycles.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to the table below. For more details on automatic port overrides, refer to the IO Port description.

Table 24-1. SPI Pin Overrides

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

Note: 1. See the IO Port description for how to define the SPI pin directions.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. `DDR_SPI` in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. `DD_MOSI`, `DD_MISO` and `DD_SCK` must be replaced by the actual data direction bits for these pins. E.g. if MOSI is placed on pin PB5, replace `DD_MOSI` with `DDB5` and `DDR_SPI` with `DDRB`.

Assembly Code Example

```
SPI_MasterInit:
   ; Set MOSI and SCK output, all others input
   ldi     r17,(1<<DD_MOSI)|(1<<DD_SCK)
   out     DDR_SPI,r17
   ; Enable SPI, Master, set clock rate fck/16
   ldi     r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
   out     SPCR,r17
   ret
```

```
   in    r17, UCSR0A
   sbrs  r17, RXC
   rjmp  USART_Receive
   ; Get and return received data from buffer
   in    r16, UDR0
   ret
```

C Code Example

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

For I/O Registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

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

Related Links

About Code Examples on page 22

25.8.2. Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZn=7) the ninth bit must be read from the RXB8 bit in UCSRnB before reading the low bits from the UDRn. This rule applies to the FE, DOR and UPE 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 TXB8, FE, DOR and UPE bits, which all are stored in the FIFO, will change.

The following code example shows a simple receive function for USART0 that handles both nine bit characters and the status bits. For the assembly code, the received data will be stored in R17:R16 after the code completes.

Assembly Code Example

```
USART_Receive:
   ; Wait for data to be received
   in     r16, UCSR0A
   sbrs   r16, RXC
   rjmp   USART_Receive
   ; Get status and 9th bit, then data from buffer
   in     r18, UCSR0A
   in     r17, UCSR0B
   in     r16, UDR0
   ; If error, return -1
   andi   r18,(1<<FE)|(1<<DOR)|(1<<UPE)
   breq   USART_ReceiveNoError
   ldi    r17, HIGH(-1)
   ldi    r16, LOW(-1)
USART_ReceiveNoError:
   ; Filter the 9th bit, then return
   lsr    r17
   andi   r17, 0x01
   ret
```

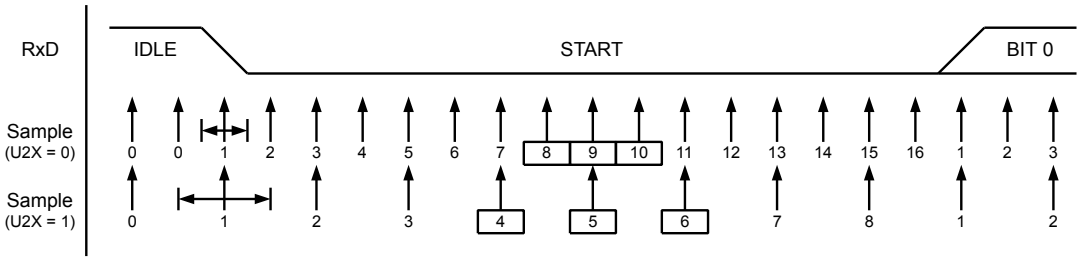
25.9. Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxDn pin. The data recovery logic samples and low pass filters each incoming bit, thereby improving the noise immunity of the Receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

25.9.1. Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. The figure below illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16-times the baud rate for Normal mode, and 8 times the baud rate for Double Speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the Double Speed mode (UCSRnA.U2Xn=1) of operation. Samples denoted '0' are samples taken while the RxDn line is idle (i.e., no communication activity).

Figure 25-5. Start Bit Sampling

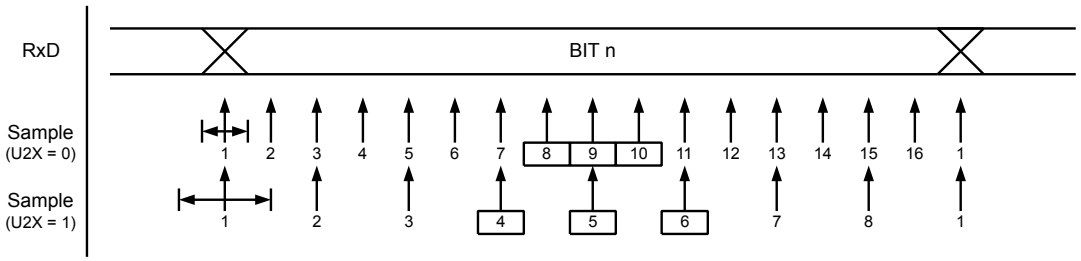


When the clock recovery logic detects a high (idle) to low (start) transition on the RxDn line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and 6 for Double Speed mode (indicated with sample numbers inside boxes on the figure), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the Receiver starts looking for the next high to low-transition on RxDn. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

25.9.2. Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in Normal mode and eight states for each bit in Double Speed mode. The figure below shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

Figure 25-6. Sampling of Data and Parity Bit



The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit: If two or all three center samples (those marked by

31.3.1. Store Program Memory Control and Status Register

The Store Program Memory Control and Status Register contains the control bits needed to control the Program memory operations.

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.

Name: SPMCSR
Offset: 0x57
Reset: 0x00
Property: When addressing I/O Registers as data space the offset address is 0x37

Bit	7	6	5	4	3	2	1	0
	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN
Access	R/W	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – SPMIE: SPM Interrupt Enable
When the SPMIE bit is written to '1', and the I-bit in the Status Register is set ('1'), the SPM ready interrupt will be enabled. The SPM ready Interrupt will be executed as long as the SPMEN bit in the SPMCSR Register is cleared (SPMCSR.SPMEN). The interrupt will not be generated during EEPROM write or SPM.

Bit 6 – RWWSB: Read-While-Write Section Busy
This bit is for compatibility with devices supporting Read-While-Write. It will always read as zero in ATmega48PB.

Bit 5 – SIGRD: Signature Row Read
If this bit is written to one at the same time as SPMEN, the next LPM instruction within three clock cycles will read a byte from the signature row into the destination register. Please refer to *Reading the Signature Row from Software*. An SPM instruction within four cycles after SIGRD and SPMEN are set will have no effect. This operation is reserved for future use and should not be used.

Bit 4 – RWWSRE: Read-While-Write Section Read Enable
The functionality of this bit in ATmega48PB is a subset of the functionality in this device. If the RWWSRE bit is written while filling the temporary page buffer, the temporary page buffer will be cleared and the data will be lost.

Bit 3 – BLBSET: Boot Lock Bit Set
The functionality of this bit in ATmega48PB is a subset of the functionality in this device. An LPM instruction within three cycles after BLBSET and SPMEN are set in the SPMCSR Register, will read either the Lock bits or the Fuse bits (depending on Z0 in the Z-pointer) into the destination register. Please refer to Reading the Fuse and Lock Bits from Software

Bit 2 – PGWRT: Page Write
If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Write, with the data stored in the temporary buffer. The page address is taken from the high part of the Zpointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon

System Clock Prescaler on page 56

33.2.1. Latching of Fuses

The fuse values are latched when the device enters programming mode and changes of the fuse values will have no effect until the part leaves Programming mode. This does not apply to the EESAVE Fuse which will take effect once it is programmed. The fuses are also latched on Power-up in Normal mode.

33.3. Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode, also when the device is locked. The three bytes reside in a separate address space. For the device the signature bytes are given in the following table.

Table 33-9. Device ID

Part	Signature Bytes Address		
	0x000	0x001	0x002
ATmega48PB	0x1E	0x92	0x10
ATmega88PB	0x1E	0x93	0x16
ATmega168PB	0x1E	0x94	0x15

33.4. Calibration Byte

The device has a byte calibration value for the Internal RC Oscillator. This byte resides in the high byte of address 0x000 in the signature address space. During reset, this byte is automatically written into the OSCCAL Register to ensure correct frequency of the calibrated RC Oscillator.

33.5. Page Size

Table 33-10. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATmega48PB	2K words (4Kbytes)	32 words	PC[4:0]	64	PC[10:5]	10
ATmega88PB	4K words (8Kbytes)	32 words	PC[4:0]	128	PC[11:5]	11
ATmega168PB	8K words (16Kbytes)	64 words	PC[5:0]	128	PC[12:6]	12

Step F. Repeat B Through E Until the Entire Buffer Is Filled or Until All Data Within the Page Is Loaded

While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in the following figure, Addressing the Flash Which is Organized in Pages, in this section. Note that if less than eight bits are required to address words in the page (pagesize < 256), the most significant bit(s) in the address low byte are used to address the page when performing a Page Write.

Step G. Load Address High Byte

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “1”. This selects high address.
3. Set DATA = Address high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address high byte.

Step H. Program Page

1. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.
2. Wait until RDY/$\overline{BSY}$ goes high (Please refer to the figure, Programming the Flash Waveforms, in this section for signal waveforms).

Step I. Repeat B Through H Until the Entire Flash Is Programmed or Until All Data Has Been Programmed

Step J. End Page Programming

1. 1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

33.7.14. Reading the Calibration Byte

The algorithm for reading the Calibration byte is as follows (Please refer to Programming the Flash for details on Command and Address loading):

1. Step A: Load Command “0000 1000”.
2. Step B: Load Address Low Byte, 0x00.
3. Set $\overline{OE}$ to “0”, and BS1 to “1”. The Calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

33.7.15. Parallel Programming Characteristics

For characteristics of the Parallel Programming, please refer to *Parallel Programming Characteristics*.

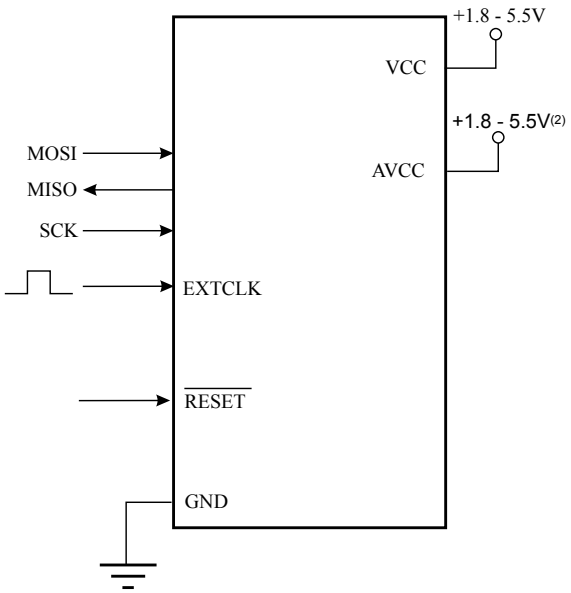
Related Links

Parallel Programming Characteristics on page 403

33.8. Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{RESET}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

Figure 33-6. Serial Programming and Verify



Note:

1. If the device is clocked by the internal Oscillator, it is no need to connect a clock source to the XTAL1 pin.
2. V_{CC} - 0.3V < AVCC < V_{CC} + 0.3V, however, AVCC should always be within 1.8 - 5.5V

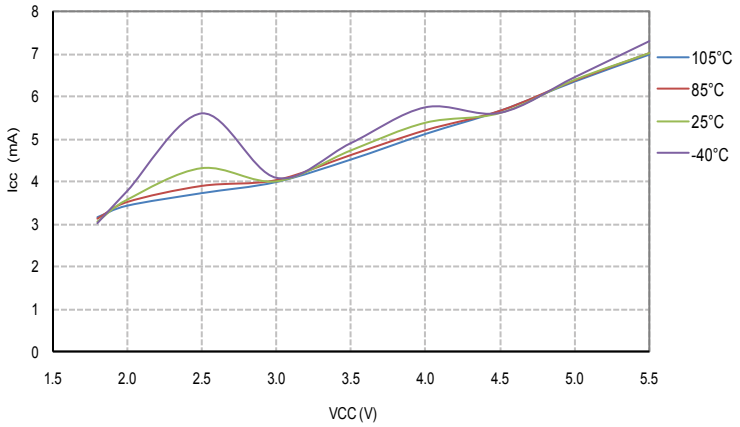
When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the Serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into 0xFF.

Depending on CKSEL Fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Table 34-10. Two-wire Serial Bus Requirements

Symbol	Parameter	Condition	Min.	Max	Units
V_{IL}	Input Low-voltage		-0.5	0.3 V_{CC}	V
V_{IH}	Input High-voltage		0.7 V_{CC}	V_{CC} + 0.5	V
V_{hys}(1)	Hysteresis of Schmitt Trigger Inputs		0.05 V_{CC}(2)	–	V
V_{OL}(1)	Output Low-voltage	3mA sink current	0	0.4	V
t_r(1)	Rise Time for both SDA and SCL		20 + 0.1C_b(3)(2)	300	ns
t_{of}(1)	Output Fall Time from V_{IHmin} to V_{ILmax}	10pF < C_b < 400pF(3)	20 + 0.1C_b(3)(2)	250	ns
t_{SP}(1)	Spikes Suppressed by Input Filter		0	50(2)	ns
I_i	Input Current each I/O Pin	0.1V_{CC} < V_i < 0.9V_{CC}	-10	10	µA
C_i(1)	Capacitance for each I/O Pin		–	10	pF
f_{SCL}	SCL Clock Frequency	f_{CK}(4) > max(16f_{SCL}, 250kHz)(5)	0	400	kHz
Rp	Value of Pull-up resistor	f_{SCL} ≤ 100kHz	$\frac{V_{CC} + -0}{3\text{mA}}$	$\frac{1000\text{ns}}{C_b}$	Ω
		f_{SCL} > 100kHz	$\frac{V_{CC} + -0}{3\text{mA}}$	$\frac{300\text{ns}}{C_b}$	Ω
$t_{HD;STA}$	Hold Time (repeated) START Condition	f_{SCL} ≤ 100kHz	4.0	–	µs
		f_{SCL} > 100kHz	0.6	–	µs
t_{LOW}	Low Period of the SCL Clock	f_{SCL} ≤ 100kHz	4.7	–	µs
		f_{SCL} > 100kHz	1.3	–	µs
t_{HIGH}	High period of the SCL clock	f_{SCL} ≤ 100kHz	4.0	–	µs
		f_{SCL} > 100kHz	0.6	–	µs
$t_{SU;STA}$	Set-up time for a repeated START condition	f_{SCL} ≤ 100kHz	4.7	–	µs
		f_{SCL} > 100kHz	0.6	–	µs
$t_{HD;DAT}$	Data hold time	f_{SCL} ≤ 100kHz	0	3.45	µs
		f_{SCL} > 100kHz	0	0.9	µs
$t_{SU;DAT}$	Data setup time	f_{SCL} ≤ 100kHz	250	–	ns
		f_{SCL} > 100kHz	100	–	ns
$t_{SU;STO}$	Setup time for STOP condition	f_{SCL} ≤ 100kHz	4.0	–	µs
		f_{SCL} > 100kHz	0.6	–	µs

Figure 35-45. ATmega48PB_88PB: Programming Current vs. V_{CC}



35.1.13. Current Consumption in Reset and Reset Pulse width

Figure 35-46. ATmega48PB_88PB: Reset Supply Current vs. Low Frequency (0.1MHz - 1.0MHz)

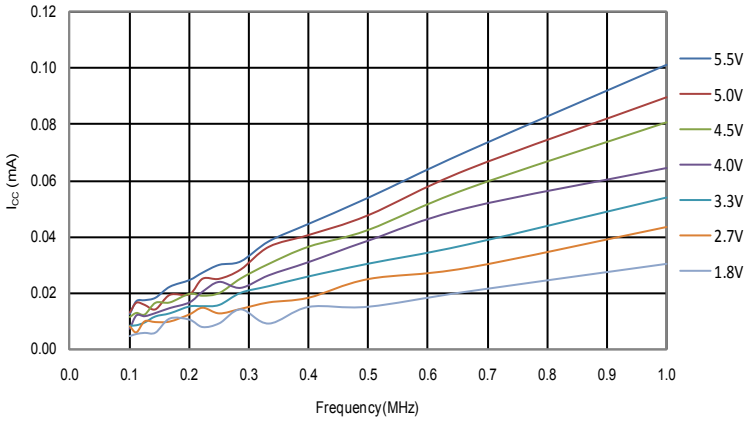
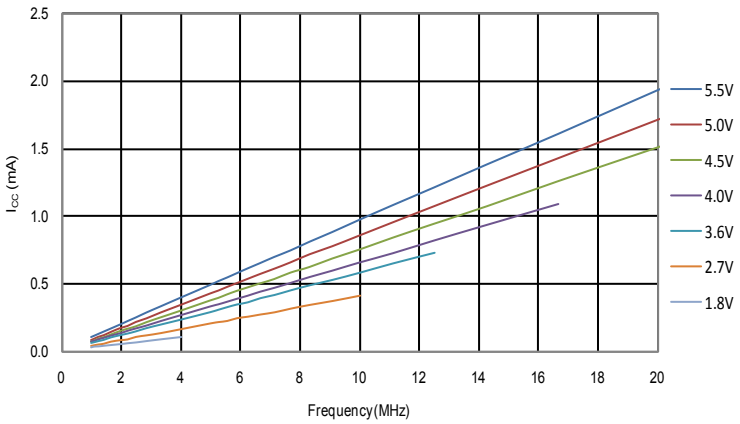


Figure 35-47. ATmega48PB_88PB: Reset Supply Current vs. Frequency (1MHz - 20MHz)



Offset	Name	Bit Pos.								
0x52	Reserved									
0x53	SMCR	7:0					SM2	SM1	SM0	SE
0x54	MCUSR	7:0					WDRT	BORF	EXTRF	PORF
0x55	MCUCR	7:0		BODS	BODSE	PUD			IVSEL	IVCE
0x56	Reserved									
0x57	SPMCSR	7:0	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN
0x58 ... 0x5C	Reserved									
0x5D	SPL	7:0	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
0x5E	SPH	7:0	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
0x5F	SREG	7:0	I	T	H	S	V	N	Z	C
0x60	WDTCSR	7:0	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0
0x61	CLKPR	7:0	CLKPCE				CLKPS3	CLKPS2	CLKPS1	CLKPS0
0x62 ... 0x63	Reserved									
0x64	PRR	7:0	PRTWI	PRTIM2	PRTIM0		PRTIM1	PRSPI	PRUSART0	PRADC
0x65	Reserved									
0x66	OSCCAL	7:0	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0
0x67	Reserved									
0x68	PCICR	7:0						PCIE2	PCIE1	PCIE0
0x69	EICRA	7:0					ISC11	ISC10	ISC01	ISC00
0x6A	Reserved									
0x6B	PCMSK0	7:0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
0x6C	PCMSK1	7:0		PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
0x6D	PCMSK2	7:0	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
0x6E	TIMSK0	7:0						OCIE0B	OCIE0A	TOIE0
0x6F	TIMSK1	7:0			ICIE1			OCIE1B	OCIE1A	TOIE1
0x70	TIMSK2	7:0						OCIE2B	OCIE2A	TOIE2
0x71 ... 0x77	Reserved									
0x78	ADCL	7:0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
0x79	ADCH	7:0							ADC9	ADC8
0x7A	ADCSRA	7:0	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
0x7B	ADCSRB	7:0		ACME				ADTS2	ADTS1	ADTS0
0x7C	ADMUX	7:0	REFS1	REFS0	ADLAR		MUX3	MUX2	MUX1	MUX0
0x7D	Reserved									
0x7E	DIDR0	7:0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
0x7F	DIDR1	7:0							AIN1D	AIN0D
0x80	TCCR1A	7:0	COM1A1	COM1A0	COM1B1	COM1B0			WGM11	WGM10
0x81	TCCR1B	7:0	ICNC1	ICES1		WGM13	WGM12	CS12	CS11	CS10
0x82	TCCR1C	7:0	FOC1A	FOC1B						
0x83	Reserved									
0x84	TCNT1L	7:0	TCNT1L[7:0]							
0x85	TCNT1H	7:0	TCNT1H[7:0]							