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	8KB (4K 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/atmega88pb-an

8. Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

13.6.5. GPIOR2 – General Purpose I/O Register 2

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: GPIOR2
Offset: 0x4B
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2B

Bit	7	6	5	4	3	2	1	0
	GPIOR2[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 – GPIOR2[7:0]: General Purpose I/O

- CL - is the load capacitance for a 32.768kHz crystal specified by the crystal vendor
- C_S - is the total stray capacitance for one TOSC pin.

Crystals specifying a load capacitance (CL) higher than 6pF require external capacitors applied as described in Crystal Oscillator Connections.

The Low-frequency Crystal Oscillator must be selected by setting the CKSEL Fuses to '0110' or '0111',and Start-Up times are determined by the SUT Fuses, as shown in the following two tables.

Table 14-7. Start-Up Times for the Low-frequency Crystal Oscillator Clock Selection - SUT Fuses

SUT[1:0]	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	19CK	Fast rising power or BOD enabled
01	19CK + 4.1ms	Slowly rising power
10	19CK + 65ms	Stable frequency at start-up
11	Reserved	

Table 14-8. Start-Up Times for the Low-frequency Crystal Oscillator Clock Selection - CKSEL Fuses

CKSEL[3:0]	Start-Up Time from Power-down and Power-Save	Recommended Usage
0100(1)	1K CK	
0101	32K CK	Stable frequency at start-up

Note:

1. This option should only be used if frequency stability at start-up is not important for the application.

14.5. Calibrated Internal RC Oscillator

By default, the Internal RC Oscillator provides an 8.0MHz clock. Though voltage and temperature dependent, this clock can be very accurately calibrated by the user. The device is shipped with the CKDIV8 Fuse programmed.

This clock may be selected as the system clock by programming the CKSEL Fuses as shown in the following Table. If selected, it will operate with no external components. During reset, hardware loads the pre-programmed calibration value into the OSCCAL Register and thereby automatically calibrates the RC Oscillator.

Table 14-9. Internal Calibrated RC Oscillator Operating Modes

Frequency Range(1) [MHz]	CKSEL[3:0]
7.3 - 8.1	0010(2)

Note:

1. If 8MHz frequency exceeds the specification of the device (depends on V_{CC}), the CKDIV8 Fuse can be programmed in order to divide the internal frequency by 8.
2. The device is shipped with this option selected.

When this Oscillator is selected, start-up times are determined by the SUT Fuses:

14.11.1. Oscillator Calibration Register

Name: OSCCAL
Offset: 0x66
Reset: Device Specific Calibration Value
Property: -

Bit	7	6	5	4	3	2	1	0
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – CALn: Oscillator Calibration Value [n = 7:0]
The Oscillator Calibration Register is used to trim the Calibrated Internal RC Oscillator to remove process variations from the oscillator frequency. A pre-programmed calibration value is automatically written to this register during chip reset, giving the Factory calibrated frequency as specified in the *Clock Characteristics* section of *Electrical Characteristics* chapter.. The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies as specified in the *Clock Characteristics* section of *Electrical Characteristics* chapter.. Calibration outside that range is not guaranteed.

Note that this oscillator is used to time EEPROM and Flash write accesses, and these write times will be affected accordingly. If the EEPROM or Flash are written, do not calibrate to more than 8.8MHz. Otherwise, the EEPROM or Flash write may fail.

The CAL7 bit determines the range of operation for the oscillator. Setting this bit to 0 gives the lowest frequency range, setting this bit to 1 gives the highest frequency range. The two frequency ranges are overlapping, in other words a setting of OSCCAL=0x7F gives a higher frequency than OSCCAL=0x80.

The CAL[6:0] bits are used to tune the frequency within the selected range. A setting of 0x00 gives the lowest frequency in that range, and a setting of 0x7F gives the highest frequency in the range.

Address	Labels		Code Comments
0x0033	RESET: ldi	r16, high(RAMEND)	; Main program start
0x0034	out	SPH,r16	; Set Stack Pointer to top of RAM
0x0035	ldi	r16, low(RAMEND)	
0x0036	out	SPL,r16	
0x0037	sei		; Enable interrupts
0x0038	<instr>	xxx	
...	...		...

When the BOOTRST Fuse is unprogrammed, the Boot section size set to 2Kbytes 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 in ATmega168PB 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	0x1C02		
0x1C02	jmp	EXT_INT0	; IRQ0 Handler
0x1C04	jmp	EXT_INT1	; IRQ1 Handler
...	...	...	
;			
0x1C32	jmp	SPM_RDY	; Store Program Memory Ready Handler

When the BOOTRST Fuse is programmed and the Boot section size set to 2Kbytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega168PB is:

Address	Labels		Code Comments
.org	0x0002		
0x0002	jmp	EXT_INT0	; IRQ0 Handler
0x0004	jmp	EXT_INT1	; IRQ1 Handler
...	...	...	

18.2.1. External Interrupt Control Register A

The External Interrupt Control Register A contains control bits for interrupt sense control.

Name: EICRA
Offset: 0x69
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					ISC11	ISC10	ISC01	ISC00
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:2 – ISC1n: Interrupt Sense Control 1 [n = 1:0]

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT1 pin that activate the interrupt are defined in the table below. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

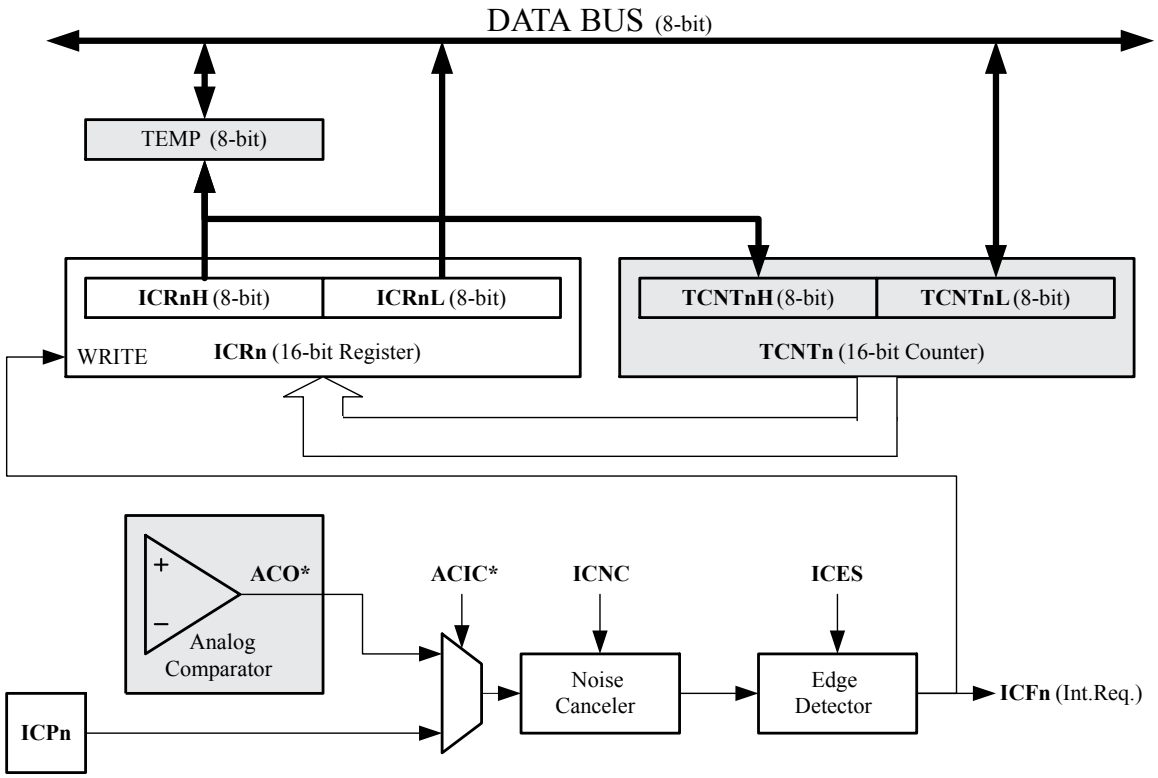
Value	Description
00	The low level of INT1 generates an interrupt request.
01	Any logical change on INT1 generates an interrupt request.
10	The falling edge of INT1 generates an interrupt request.
11	The rising edge of INT1 generates an interrupt request.

Bits 1:0 – ISC0n: Interrupt Sense Control 0 [n = 1:0]

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in table below. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Value	Description
00	The low level of INT0 generates an interrupt request.
01	Any logical change on INT0 generates an interrupt request.
10	The falling edge of INT0 generates an interrupt request.
11	The rising edge of INT0 generates an interrupt request.

Figure 21-3. Input Capture Unit Block Diagram for TC1



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).

When a change of the logic level (an event) occurs on the Input Capture pin (ICP1), or alternatively on the Analog Comparator output (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered: the 16-bit value of the counter (TCNT1) is written to the Input Capture Register (ICR1). The Input Capture Flag (ICF) is set at the same system clock cycle as the TCNT1 value is copied into the ICR1 Register. If enabled (TIMSK1.ICIE=1), the Input Capture Flag generates an Input Capture interrupt. The ICF1 Flag is automatically cleared when the interrupt is executed. Alternatively the ICF Flag can be cleared by software by writing '1' to its I/O bit location.

Reading the 16-bit value in the Input Capture Register (ICR1) is done by first reading the low byte (ICR1L) and then the high byte (ICR1H). When the low byte is read form ICR1L, the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR1H I/O location it will access the TEMP Register.

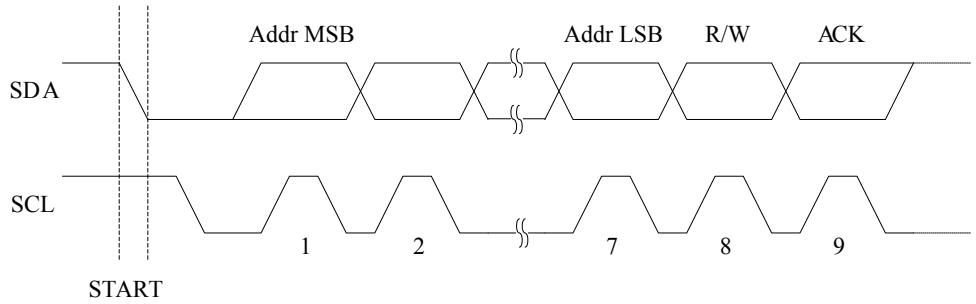
The ICR1 Register can only be written when using a Waveform Generation mode that utilizes the ICR1 Register for defining the counter’s TOP value. In these cases the Waveform Generation mode bits (WGM1[3:0]) must be set before the TOP value can be written to the ICR1 Register. When writing the ICR1 Register, the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

See also Accessing 16-bit Registers.

21.7.1. Input Capture Trigger Source

The main trigger source for the Input Capture unit is the Input Capture pin (ICP1). Timer/Counter1 can alternatively use the Analog Comparator output as trigger source for the Input Capture unit. The Analog Comparator is selected as trigger source by setting the Analog Comparator Input Capture (ACIC) bit in

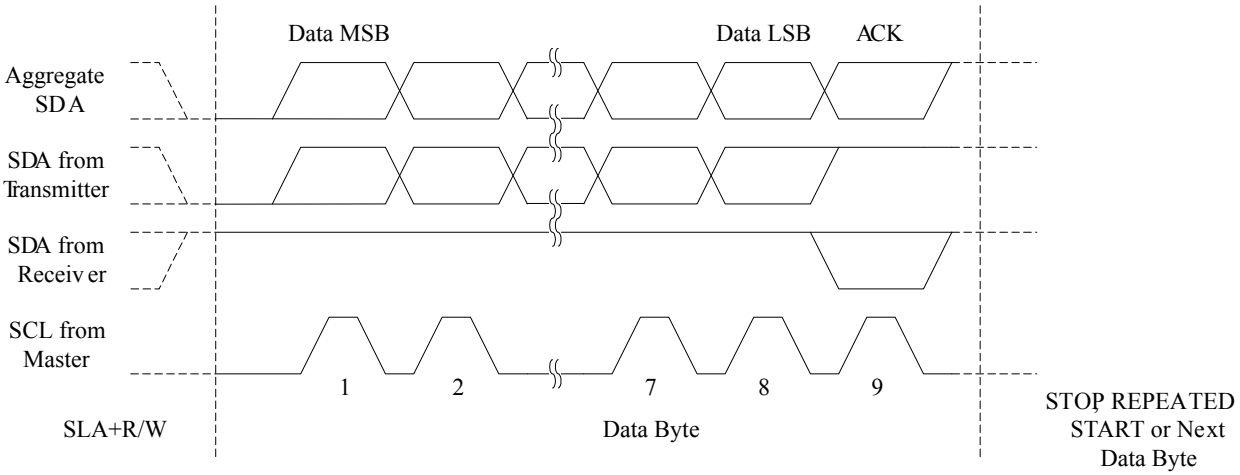
Figure 27-4. Address Packet Format



27.3.4. Data Packet Format

All data packets transmitted on the TWI bus are nine bits long, consisting of one data byte and an acknowledge bit. During a data transfer, the Master generates the clock and the START and STOP conditions, while the Receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signalled by the Receiver pulling the SDA line low during the ninth SCL cycle. If the Receiver leaves the SDA line high, a NACK is signalled. When the Receiver has received the last byte, or for some reason cannot receive any more bytes, it should inform the Transmitter by sending a NACK after the final byte. The MSB of the data byte is transmitted first.

Figure 27-5. Data Packet Format

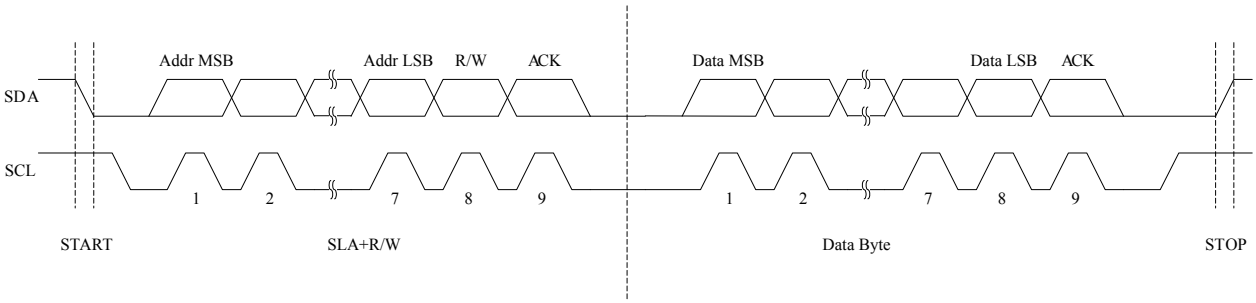


27.3.5. Combining Address and Data Packets into a Transmission

A transmission basically consists of a START condition, a SLA+R/W, one or more data packets and a STOP condition. An empty message, consisting of a START followed by a STOP condition, is illegal. Note that the "Wired-ANDing" of the SCL line can be used to implement handshaking between the Master and the Slave. The Slave can extend the SCL low period by pulling the SCL line low. This is useful if the clock speed set up by the Master is too fast for the Slave, or the Slave needs extra time for processing between the data transmissions. The Slave extending the SCL low period will not affect the SCL high period, which is determined by the Master. As a consequence, the Slave can reduce the TWI data transfer speed by prolonging the SCL duty cycle.

The following figure depicts a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP condition, depending on the software protocol implemented by the application software.

Figure 27-6. Typical Data Transmission



27.4. Multi-master Bus Systems, Arbitration and Synchronization

The TWI protocol allows bus systems with several masters. Special concerns have been taken in order to ensure that transmissions will proceed as normal, even if two or more masters initiate a transmission at the same time. Two problems arise in multi-master systems:

- An algorithm must be implemented allowing only one of the masters to complete the transmission. All other masters should cease transmission when they discover that they have lost the selection process. This selection process is called arbitration. When a contending master discovers that it has lost the arbitration process, it should immediately switch to Slave mode to check whether it is being addressed by the winning master. The fact that multiple masters have started transmission at the same time should not be detectable to the slaves, i.e. the data being transferred on the bus must not be corrupted.
- Different masters may use different SCL frequencies. A scheme must be devised to synchronize the serial clocks from all masters, in order to let the transmission proceed in a lockstep fashion. This will facilitate the arbitration process.

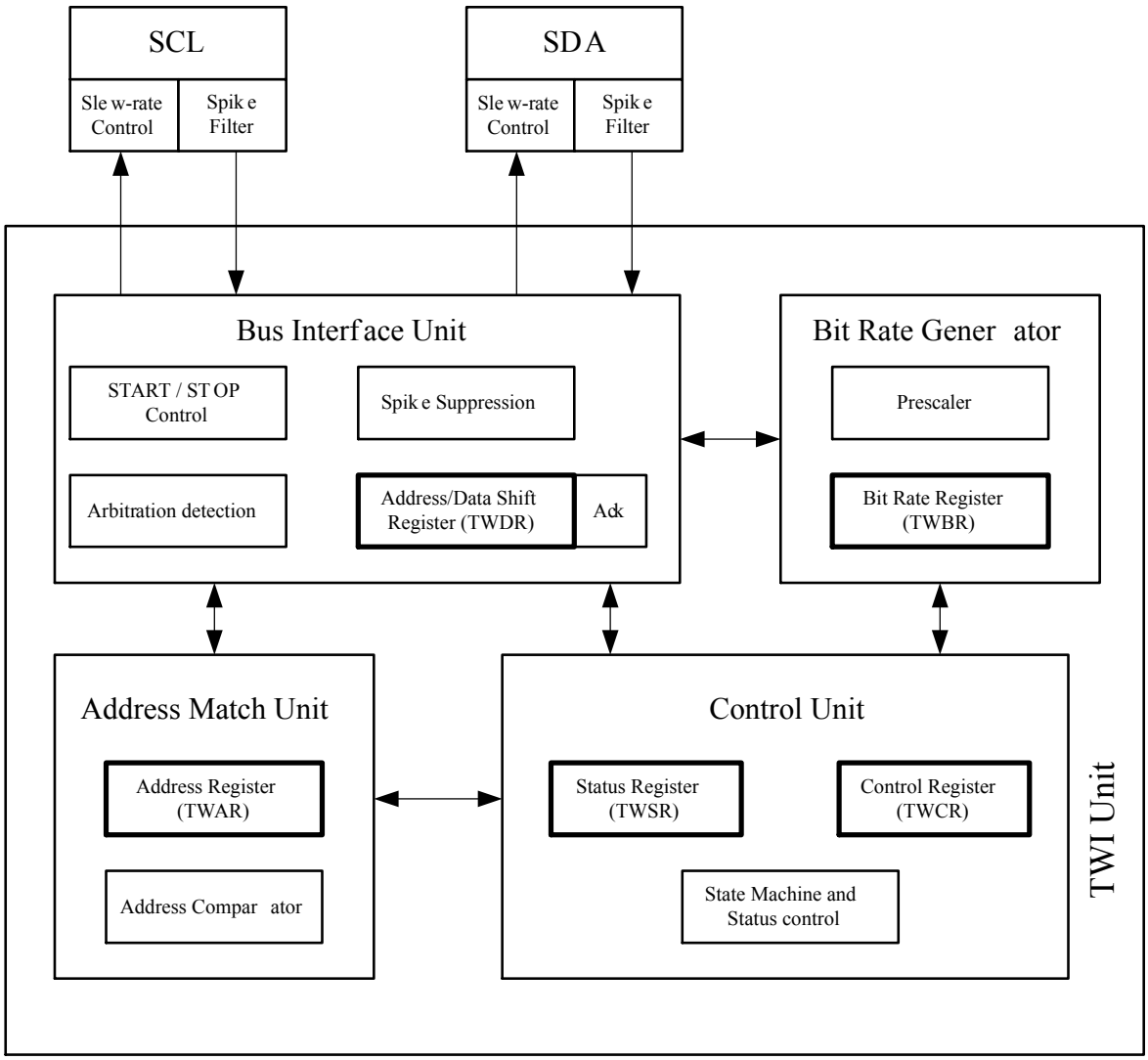
The wired-ANDing of the bus lines is used to solve both these problems. The serial clocks from all masters will be wired-ANDed, yielding a combined clock with a high period equal to the one from the Master with the shortest high period. The low period of the combined clock is equal to the low period of the Master with the longest low period. Note that all masters listen to the SCL line, effectively starting to count their SCL high and low time-out periods when the combined SCL line goes high or low, respectively.

data packets. In other words; All transmissions must contain the same number of data packets, otherwise the result of the arbitration is undefined.

27.5. Overview of the TWI Module

The TWI module is comprised of several submodules, as shown in the following figure. The registers drawn in a thick line are accessible through the AVR data bus.

Figure 27-9. Overview of the TWI Module



27.5.1. SCL and SDA Pins

These pins interface the AVR TWI with the rest of the MCU system. The output drivers contain a slew-rate limiter in order to conform to the TWI specification. The input stages contain a spike suppression unit removing spikes shorter than 50ns. Note that the internal pull-ups in the AVR pads can be enabled by setting the PORT bits corresponding to the SCL and SDA pins, as explained in the I/O Port section. The internal pull-ups can in some systems eliminate the need for external ones.

27.5.2. Bit Rate Generator Unit

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBRn) and the Prescaler bits in the TWI Status Register

- After the TWI has transmitted SLA+R/W
- After the TWI has transmitted an address byte
- After the TWI has lost arbitration
- After the TWI has been addressed by own slave address or general call
- After the TWI has received a data byte
- After a STOP or REPEATED START has been received while still addressed as a Slave
- When a bus error has occurred due to an illegal START or STOP condition

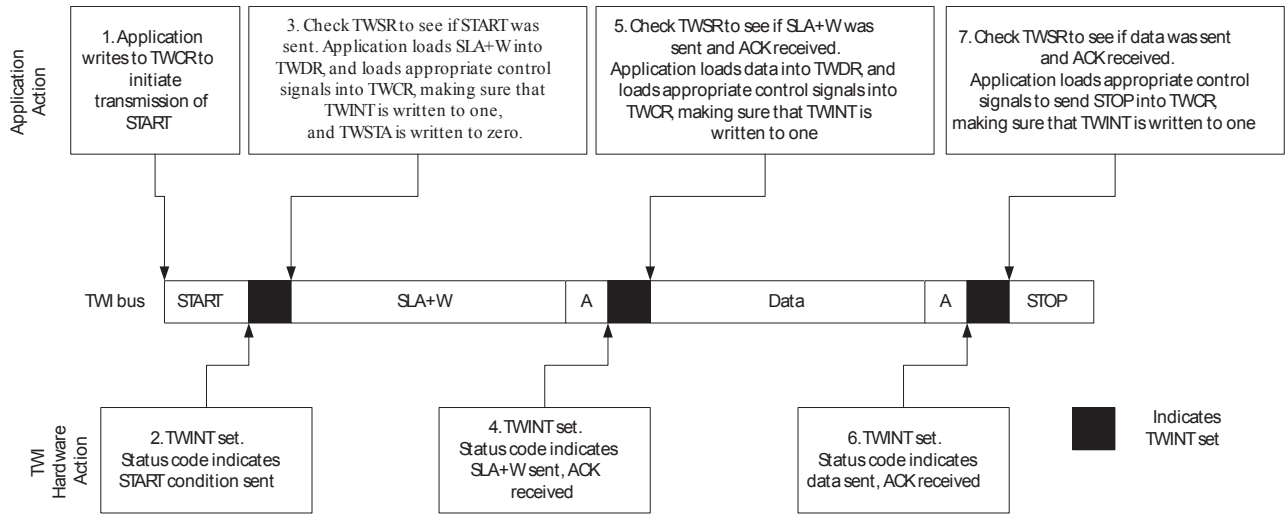
27.6. Using the TWI

The AVR TWI is byte-oriented and interrupt based. Interrupts are issued after all bus events, like reception of a byte or transmission of a START condition. Because the TWI is interrupt-based, the application software is free to carry on other operations during a TWI byte transfer. Note that the TWI Interrupt Enable (TWIE) bit in TWCRn together with the Global Interrupt Enable bit in SREG allow the application to decide whether or not assertion of the TWINT Flag should generate an interrupt request. If the TWIE bit is cleared, the application must poll the TWINT Flag in order to detect actions on the TWI bus.

When the TWINT Flag is asserted, the TWI has finished an operation and awaits application response. In this case, the TWI Status Register (TWSRn) contains a value indicating the current state of the TWI bus. The application software can then decide how the TWI should behave in the next TWI bus cycle by manipulating the TWCRn and TWDRn Registers.

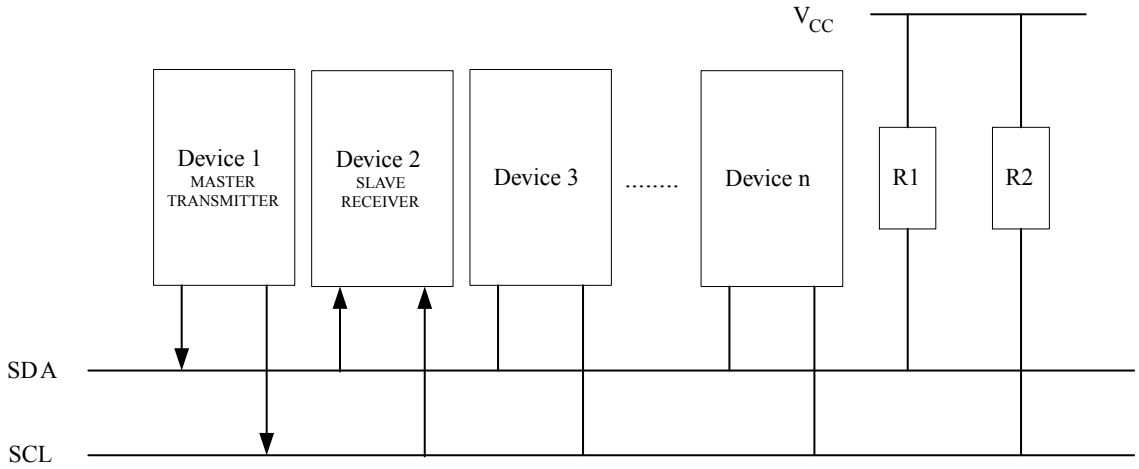
The following figure illustrates a simple example of how the application can interface to the TWI hardware. In this example, a Master wishes to transmit a single data byte to a Slave. A more detailed explanation follows later in this section. Simple code examples are presented in the table below.

Figure 27-10. Interfacing the Application to the TWI in a Typical Transmission



1. The first step in a TWI transmission is to transmit a START condition. This is done by writing a specific value into TWCRn, instructing the TWI n hardware to transmit a START condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI n will not start any operation as long as the TWINT bit in TWCRn is set. Immediately after the application has cleared TWINT, the TWI n will initiate transmission of the START condition.
2. When the START condition has been transmitted, the TWINT Flag in TWCRn is set, and TWSRn is updated with a status code indicating that the START condition has successfully been sent.

Figure 27-11. Data Transfer in Master Transmitter Mode



A START condition is sent by writing a value to the TWI Control Register n (TWCRn) of the type TWCRn=1x10x10x:

- The TWI Enable bit (TWCRn.TWEN) must be written to '1' to enable the 2-wire Serial Interface
- The TWI Start Condition bit (TWCRn.TWSTA) must be written to '1' to transmit a START condition
- The TWI Interrupt Flag (TWCRn.TWINT) must be written to '1' to clear the flag.

The TWI n will then test the 2-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSRn will be 0x08 (see Status Code table below). In order to enter MT mode, SLA +W must be transmitted. This is done by writing SLA+W to the TWI Data Register (TWDRn). Thereafter, the TWCRn.TWINT Flag should be cleared (by writing a '1' to it) to continue the transfer. This is accomplished by writing a value to TWRC of the type TWCR=1x00x10x.

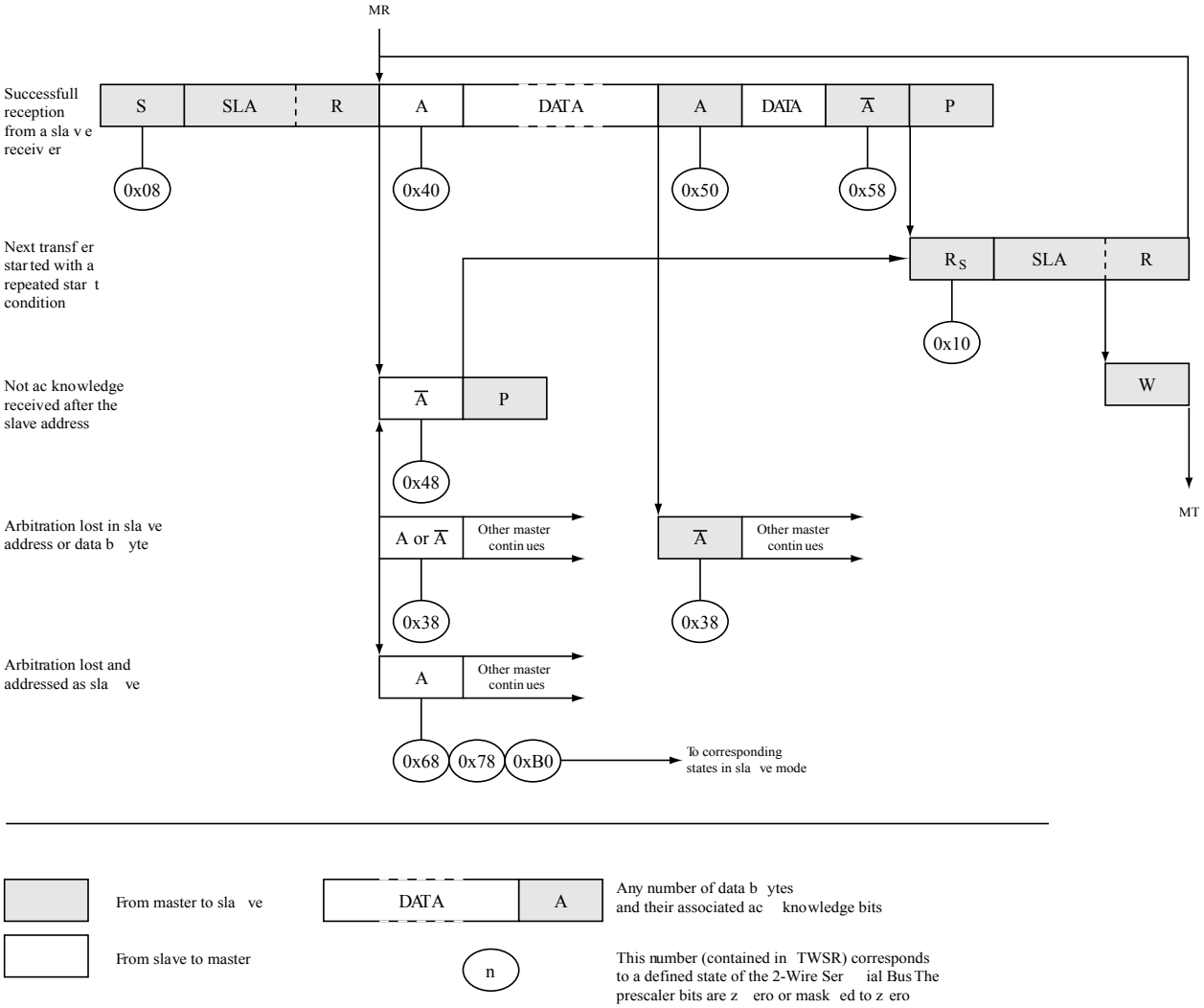
When SLA+W have been transmitted and an acknowledgment bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are 0x18, 0x20, or 0x38. The appropriate action to be taken for each of these status codes is detailed in the Status Code table below.

When SLA+W has been successfully transmitted, a data packet should be transmitted. This is done by writing the data byte to TWDR. TWDR must only be written when TWINT is high. If not, the access will be discarded, and the Write Collision bit (TWWC) will be set in the TWCRn Register. After updating TWDRn, the TWINT bit should be cleared (by writing '1' to it) to continue the transfer. This is accomplished by writing again a value to TWCRn of the type TWCRn=1x00x10x.

This scheme is repeated until the last byte has been sent and the transfer is ended, either by generating a STOP condition or a by a repeated START condition. A repeated START condition is accomplished by writing a regular START value TWCRn=1x10x10x. A STOP condition is generated by writing a value of the type TWCRn=1x01x10x.

After a repeated START condition (status code 0x10), the 2-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control of the bus.

Figure 27-14. Formats and States in the Master Receiver Mode



27.7.3. Slave Receiver Mode

In the Slave Receiver (SR) mode, a number of data bytes are received from a Master Transmitter (see figure below). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

29.9.4. ADC Data Register High (ADLAR=0)

Name: ADCH
Offset: 0x79
Reset: 0x00
Property: ADLAR = 0

Bit	7	6	5	4	3	2	1	0
							ADC9	ADC8
Access							R	R
Reset							0	0

Bit 1 – ADC9: ADC Conversion Result
Refer to ADCL.

Bit 0 – ADC8: ADC Conversion Result

29.9.6. ADC Data Register High (ADLAR=1)

Name: ADCH
Offset: 0x79
Reset: 0x00
Property: ADLAR = 1

Bit	7	6	5	4	3	2	1	0
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 7 – ADC9: ADC Conversion Result 9
Refer to ADCL.

Bit 6 – ADC8: ADC Conversion Result 8
Refer to ADCL.

Bit 5 – ADC7: ADC Conversion Result 7
Refer to ADCL.

Bit 4 – ADC6: ADC Conversion Result 6
Refer to ADCL.

Bit 3 – ADC5: ADC Conversion Result 5
Refer to ADCL.

Bit 2 – ADC4: ADC Conversion Result 4
Refer to ADCL.

Bit 1 – ADC3: ADC Conversion Result 3
Refer to ADCL.

Bit 0 – ADC2: ADC Conversion Result 2
Refer to ADCL.

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.

35.1.8. Pin Driver Strength

Figure 35-21. ATmega48PB/88PB: I/O Pin Output Voltage vs. Sink Current (V_{CC} = 3V)

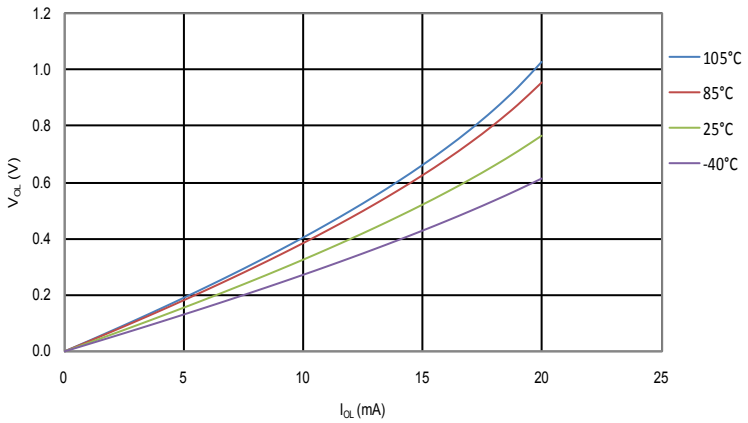


Figure 35-22. ATmega48PB/88PB: I/O Pin Output Voltage vs. Sink Current (V_{CC} = 5V)

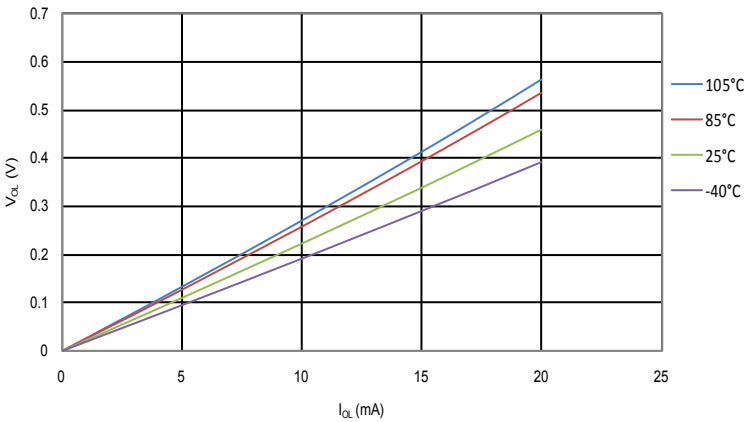


Figure 35-23. ATmega48PB/88PB: I/O Pin Output Voltage vs. Source Current (V_{CC} = 3V)

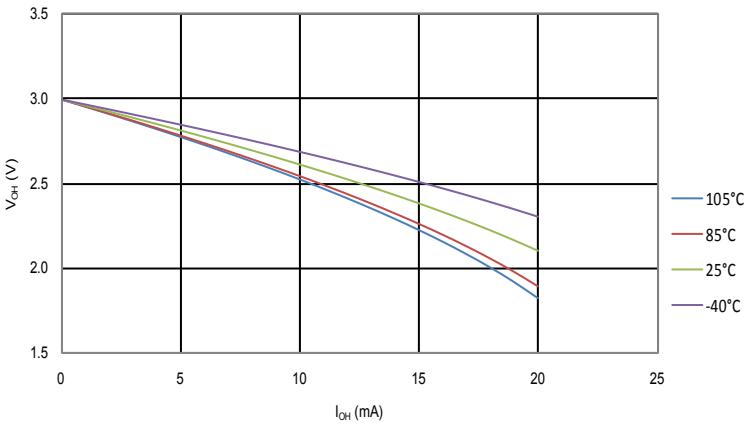


Figure 35-39. ATmega48PB_88PB: Calibrated 8MHz RC Oscillator Frequency vs. Temperature

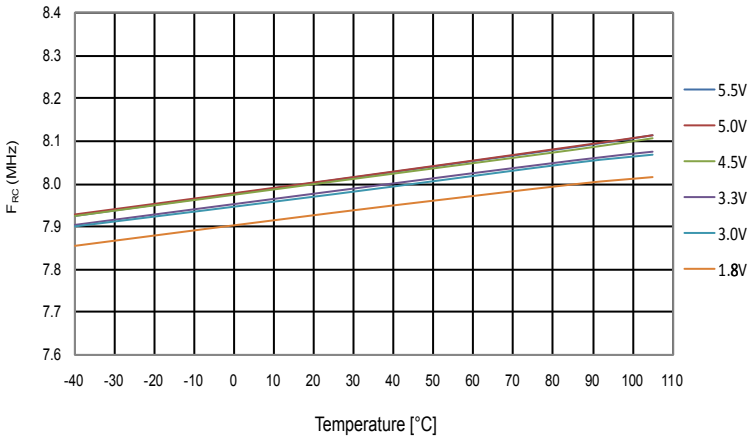
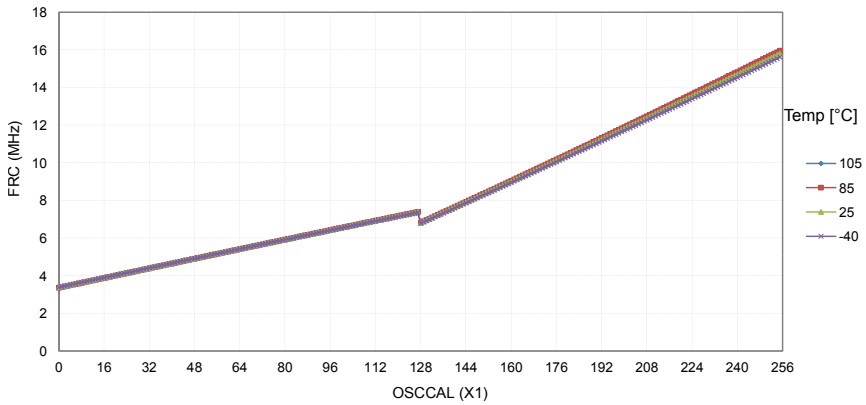
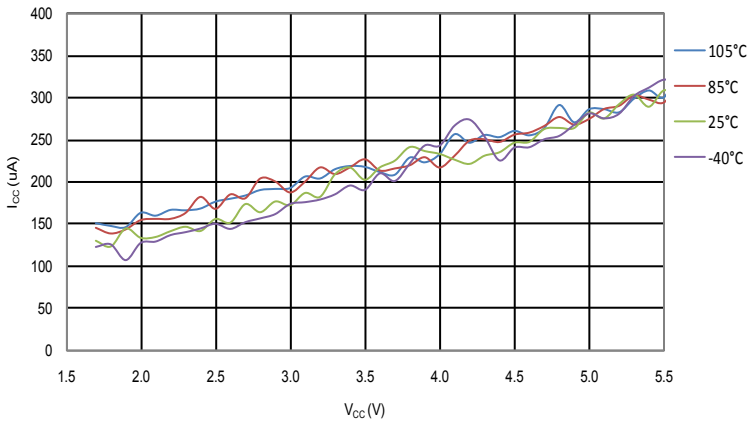


Figure 35-40. ATmega48PB_88PB: Calibrated 8MHz RC Oscillator Frequency vs. OSCCAL Value



35.1.12. Current Consumption of Peripheral Units

Figure 35-41. ATmega48PB_88PB: ADC Current vs. V_{CC} ($AREF = AV_{CC}$)



BRANCH INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL(1)	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd - Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd - K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIS	A, b	Skip if Bit in I/O Register is Set	if (I/O(A,b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2

BIT AND BIT-TEST INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**