Microchip Technology - ATMEGA163L-4AC Datasheet

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	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	4MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	32
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)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega163l-4ac

cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega163 provides the following features: 16K bytes of In-System Self-Programmable Flash, 512 bytes EEPROM, 1024 bytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC, a programmable Watchdog Timer with internal Oscillator, a programmable serial UART, an SPI serial port, and four software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous Timer Oscillator continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions.

The On-chip ISP Flash can be programmed through an SPI serial interface or a conventional programmer. By installing a Self-Programming Boot Loader, the microcontroller can be updated within the application without any external components. The Boot Program can use any interface to download the application program in the Application Flash memory. By combining an 8-bit CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega163 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega163 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators, and evaluation kits.

Pin Descriptions

VCC Digital supply voltage.

GND Digital ground.

Port A (PA7..PA0) Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers can sink 20mA and can drive LED displays directly. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tristated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0) Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers can sink 20 mA. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. Port B also serves the functions of various special features of the ATmega83/163 as listed on page 117. The Port B pins are tristated when a reset condition becomes active, even if the clock is not running.

Port C (PC7..PC0) Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers can sink 20 mA. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tristated when a reset condition becomes active, even if the clock is not running.

Table 3. Reset and Interrupt Vectors (Continued)

Vector No.	Program Address	Source	Interrupt Definition
13	$018	UART, UDRE	UART Data Register Empty
14	$01A	UART, TXC	UART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface

Note: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see “Boot Loader Support” on page 134.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega163 is:

```
Address   Labels  Code            Comments
$000              jmp     RESET      ; Reset Handler
$002              jmp     EXT_INT0   ; IRQ0 Handler
$004              jmp     EXT_INT1   ; IRQ1 Handler
$006              jmp     TIM2_COMP  ; Timer2 Compare Handler
$008              jmp     TIM2_OVF   ; Timer2 Overflow Handler
$00a              jmp     TIM1_CAPT  ; Timer1 Capture Handler
$00c              jmp     TIM1_COMPA ; Timer1 Compare A Handler
$00e              jmp     TIM1_COMPB ; Timer1 Compare B Handler
$010              jmp     TIM1_OVF   ; Timer1 Overflow Handler
$012              jmp     TIM0_OVF   ; Timer0 Overflow Handler
$014              jmp     SPI_STC    ; SPI Transfer Complete Handler
$016              jmp     UART_RXC   ; UART RX Complete Handler
$018              jmp     UART_DRE   ; UDR Empty Handler
$01a              jmp     UART_TXC   ; UART TX Complete Handler
$01c              jmp     ADC     ; ADC Conversion Complete Interrupt Handler
$01e              jmp     EE_RDY     ; EEPROM Ready Handler
$020              jmp     ANA_COMP   ; Analog Comparator Handler
$022              jmp     TWI     ; Two-wire Serial Interface Interrupt Handler
;
$024    MAIN:     ldi     r16,high(RAMEND) ; Main program start
$025              out     SPH,r16    ; Set stack pointer to top of RAM
$026              ldi     r16,low(RAMEND)
$027              out     SPL,r16
...     ...       ...
```

Timer/Counter2 Overflow Interrupt is executed. In up/down PWM mode, this bit is set when Timer/Counter2 changes counting direction at $00.

- **Bit 5 – ICF1: Input Capture Flag1**

The ICF1 bit is set (one) to Flag an Input Capture Event, indicating that the Timer/Counter1 value has been transferred to the Input Capture Register – ICR1. ICF1 is cleared by hardware when executing the corresponding Interrupt Handling Vector. Alternatively, ICF1 is cleared by writing a logic one to the flag.

- **Bit 4 – OCF1A: Output Compare Flag 1A**

The OCF1A bit is set (one) when a Compare Match occurs between the Timer/Counter1 and the data in OCR1A – Output Compare Register 1A. OCF1A is cleared by hardware when executing the corresponding Interrupt Handling Vector. Alternatively, OCF1A is cleared by writing a logic one to the flag. When the I-bit in SREG, and OCIE1A (Timer/Counter1 Compare Match Interrupt A Enable), and the OCF1A are set (one), the Timer/Counter1A Compare Match Interrupt is executed.

- **Bit 3 – OCF1B: Output Compare Flag 1B**

The OCF1B bit is set (one) when a Compare Match occurs between the Timer/Counter1 and the data in OCR1B – Output Compare Register 1B. OCF1B is cleared by hardware when executing the corresponding Interrupt Handling Vector. Alternatively, OCF1B is cleared by writing a logic one to the flag. When the I-bit in SREG, and OCIE1B (Timer/Counter1 Compare Match Interrupt B Enable), and the OCF1B are set (one), the Timer/Counter1B Compare Match Interrupt is executed.

- **Bit 2 – TOV1: Timer/Counter1 Overflow Flag**

The TOV1 is set (one) when an overflow occurs in Timer/Counter1. TOV1 is cleared by hardware when executing the corresponding Interrupt Handling Vector. Alternatively, TOV1 is cleared by writing a logic one to the flag. When the I-bit in SREG, and TOIE1 (Timer/Counter1 Overflow Interrupt Enable), and TOV1 are set (one), the Timer/Counter1 Overflow Interrupt is executed. In up/down PWM mode, this bit is set when Timer/Counter1 changes counting direction at $0000.

- **Bit 1 – Res: Reserved Bit**

This bit is a reserved bit in the ATmega163 and the read value is undefined.

- **Bit 0 – TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding Interrupt Handling Vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, and TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed.

External Interrupts

The external interrupts are triggered by the INT0 and INT1 pins. Observe that, if enabled, the interrupts will trigger even if the INT0/INT1 pins are configured as outputs. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU Control Register – MCUCR. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low.

set), the 9th bit is one for an address byte and zero for a data byte, whereas the stop bit is always high.

The following procedure should be used to exchange data in Multi-Processor Communication mode:

1. All Slave MCUs are in Multi-Processor Communication mode (MPCM in UCSRA is set).
2. The Master MCU sends an address byte, and all slaves receive and read this byte. In the Slave MCUs, the RXC Flag in UCSRA will be set as normal.
3. Each Slave MCU reads the UDR Register and determines if it has been selected. If so, it clears the MPCM bit in UCSRA, otherwise it waits for the next address byte.
4. For each received data byte, the receiving MCU will set the Receive Complete Flag (RXC in UCSRA). In 8-bit mode, the receiving MCU will also generate a Framing Error (FE in UCSRA set), since the stop bit is zero. The other slave MCUs, which still have the MPCM bit set, will ignore the data byte. In this case, the UDR Register and the RXC or FE Flags will not be affected.
5. After the last byte has been transferred, the process repeats from step 2.

UART Control

UART I/O Data Register – UDR

Bit	7	6	5	4	3	2	1	0	
$0C ($2C)	**MSB**							**LSB**	**UDR**
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 UDR Register is actually two physically separate registers sharing the same I/O address. When writing to the register, the UART Transmit Data Register is written. When reading from UDR, the UART Receive Data Register is read.

UART Control and Status Register A – UCSRA

Bit	7	6	5	4	3	2	1	0	
$0B ($2B)	**RXC**	**TXC**	**UDRE**	**FE**	**OR**	**–**	**U2X**	**MPCM**	**UCSRA**
Read/Write	r	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RXC: UART Receive Complete**

This bit is set (one) when a received character is transferred from the Receiver Shift Register to UDR. The bit is set regardless of any detected framing errors. When the RXCIE bit in UCR is set, the UART Receive Complete interrupt will be executed when RXC is set(one). RXC is cleared by reading UDR. When interrupt-driven data reception is used, the UART Receive Complete Interrupt routine must read UDR in order to clear RXC, otherwise a new interrupt will occur once the interrupt routine terminates.

- **Bit 6 – TXC: UART Transmit Complete**

This bit is set (one) when the entire character (including the stop bit) in the Transmit Shift Register has been shifted out and no new data has been written to UDR. This Flag is especially useful in half-duplex communications interfaces, where a transmitting application must enter receive mode and free the communications bus immediately after completing the transmission.

number of samples are reduced, and the system clock might have some variance (this applies especially when using resonators), it is recommended that the baud rate error is less than 0.5%.

Table 28. UBR Settings at Various Crystal Frequencies in Double Speed Mode

1.0000 MHz	% Error	**1.8432 MHz**	% Error	**2.0000 MHz**	% Error
UBR = **51**	**0.2**	UBR = **95**	**0.0**	UBR = **103**	**0.2**
UBR = **25**	**0.2**	UBR = **47**	**0.0**	UBR = **51**	**0.2**
UBR = **12**	**0.2**	UBR = **23**	**0.0**	UBR = **25**	**0.2**
UBR = 8	3.7	UBR = **15**	**0.0**	UBR = 16	2.1
UBR = 6	7.5	UBR = **11**	**0.0**	UBR = **12**	**0.2**
UBR = 3	7.8	UBR = **7**	**0.0**	UBR = 8	3.7
UBR = 2	7.8	UBR = **5**	**0.0**	UBR = 6	7.5
UBR = 1	7.8	UBR = **3**	**0.0**	UBR = 3	7.8
UBR = 1	22.9	UBR = **2**	**0.0**	UBR = 2	7.8
UBR = 0	84.3	UBR = **1**	**0.0**	UBR = 1	7.8
-	-	UBR = **0**	**0.0**	-	-

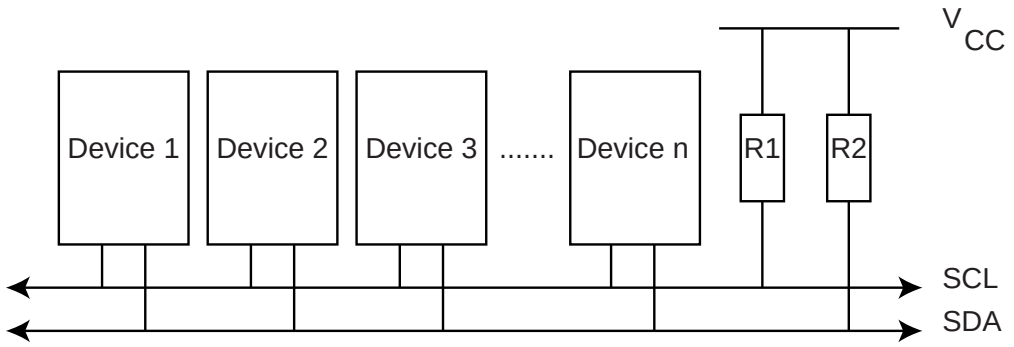
3.2768 MHz	% Error	**3.6864 MHz**	% Error	**4.0000 MHz**	% Error
UBR = **170**	**0.2**	UBR = **191**	**0.0**	UBR = **207**	**0.2**
UBR = **84**	**0.4**	UBR = **95**	**0.0**	UBR = **103**	**0.2**
UBR = **42**	**0.8**	UBR = **47**	**0.0**	UBR = **51**	**0.2**
UBR = 27	1.6	UBR = **31**	**0.0**	UBR = **34**	**0.8**
UBR = 20	1.6	UBR = **23**	**0.0**	UBR = **25**	**0.2**
UBR = 13	1.6	UBR = **15**	**0.0**	UBR = 16	2.1
UBR = 10	3.1	UBR = **11**	**0.0**	UBR = **12**	**0.2**
UBR = 6	1.6	UBR = **7**	**0.0**	UBR = 8	3.7
UBR = 4	6.2	UBR = **5**	**0.0**	UBR = 6	7.5
UBR = 3	12.5	UBR = **3**	**0.0**	UBR = 3	7.8
UBR = 1	12.5	UBR = **1**	**0.0**	UBR = 1	7.8
UBR = 0	12.5	UBR = **0**	**0.0**	UBR = 0	7.8

7.3728 MHz	% Error	**8.0000 MHz**	% Error
UBR = **383**	**0.0**	UBR = **416**	**0.1**
UBR = **191**	**0.0**	UBR = **207**	**0.2**
UBR = **95**	**0.0**	UBR = **103**	**0.2**
UBR = **63**	**0.0**	UBR = **68**	**0.6**
UBR = **47**	**0.0**	UBR = **51**	**0.2**
UBR = **31**	**0.0**	UBR = **34**	**0.8**
UBR = **23**	**0.0**	UBR = **25**	**0.2**
UBR = 15	**0.0**	UBR = 16	2.1
UBR = **11**	**0.0**	UBR = 12	**0.2**
UBR = **7**	**0.0**	UBR = 8	3.7
UBR = **3**	**0.0**	UBR = 3	7.8
UBR = **1**	**0.0**	UBR = 1	7.8
UBR = **0**	**0.0**	UBR = 0	7.8

Two-wire Serial Interface (Byte Oriented)

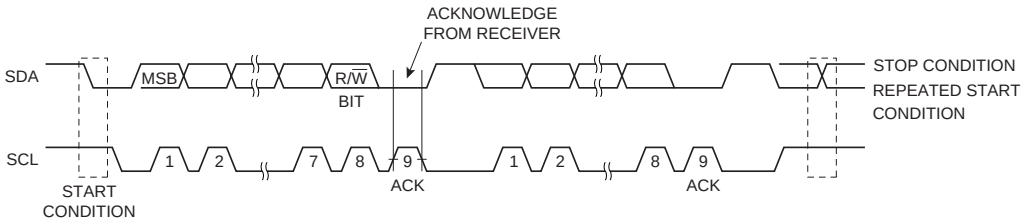
The Two-wire Serial Interface supports bi-directional serial communication. It is designed primarily for simple but efficient integrated circuit (IC) control. The system is comprised of two lines, SCL (Serial Clock) and SDA (Serial Data) that carry information between the ICs connected to them. Various communication configurations can be designed using this bus. Figure 49 shows a typical Two-wire Serial Bus configuration. Any device connected to the bus can be master or slave. Note that all AVR devices connected to the bus must be powered to allow any bus operation.

Figure 49. Two-wire Serial Bus Configuration



The Two-wire Serial Interface supports Master/Slave and Transmitter/Receiver operation at up to 217 kHz bus clock rate. The Two-wire Serial Interface has hardware support for 7-bit addressing, but is easily extended to, e.g., a 10-bit addressing format in software. When the Two-wire Serial Interface is enabled (TWEN in TWCR is set), a glitch filter is enabled for the input signals from the pins PC0 (SCL) and PC1 (SDA), and the output from these pins is slew-rate controlled. The Two-wire Serial Interface is byte oriented. The operation of the Two-wire Serial Bus is shown as a pulse diagram in Figure 50, including the START and STOP conditions and generation of ACK signal by the bus receiver.

Figure 50. Two-wire Serial Bus Timing Diagram



The block diagram of the Two-wire Serial Interface is shown in Figure 51.

- **Bit 0 – TWIE: Two-wire Serial Interface Interrupt Enable**

When this bit is enabled, and the I-bit in SREG is set, the Two-wire Serial Interface interrupt will be activated for as long as the TWINT Flag is high.

The TWCR is used to control the operation of the Two-wire Serial Interface. It is used to enable the Two-wire Serial Interface, to initiate a Master access by applying a START condition to the bus, to generate a receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are written to the TWDR. It also indicates a write collision if data is attempted written to TWDR while the register is inaccessible.

The Two-wire Serial Interface Status Register – TWSR

Bit	7	6	5	4	3	2	1	0	
$01 ($21)	**TWS7**	**TWS6**	**TWS5**	**TWS4**	**TWS3**	**–**	**–**	**–**	**TWSR**
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	1	1	1	1	1	0	0	0	

- **Bits 7..3 – TWS: Two-wire Serial Interface Status**

These five bits reflect the status of the Two-wire Serial Interface logic and the Two-wire Serial Bus.

- **Bits 2..0 – Res: Reserved bits**

These bits are reserved in ATmega163 and will always read as zero

The TWSR is read only. It contains a status code which reflects the status of the Two-wire Serial Interface logic and the Two-wire Serial Bus. There are 26 possible status codes. When TWSR contains $F8, no relevant state information is available and no Two-wire Serial Interface interrupt is requested. A valid status code is available in TWSR one CPU clock cycle after the Two-wire Serial Interface Interrupt Flag (TWINT) is set by hardware and is valid until one CPU clock cycle after TWINT is cleared by software. Table 32 to Table 36 give the status information for the various modes.

The Two-wire Serial Interface Data Register – TWDR

Bit	7	6	5	4	3	2	1	0	
$03 ($23)	**TWD7**	**TWD6**	**TWD5**	**TWD4**	**TWD3**	**TWD2**	**TWD1**	**TWD0**	**TWDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

- **Bits 7..0 – TWD: Two-wire Serial Interface Data Register**

These eight bits constitute the next data byte to be transmitted, or the latest data byte received on the Two-wire Serial Bus.

In Transmit mode, TWDR contains the next byte to be transmitted. In Receive mode, the TWDR contains the last byte received. It is writeable while the Two-wire Serial Interface is not in the process of shifting a byte. This occurs when the Two-wire Serial Interface Interrupt Flag (TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remain stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from ADC Noise Reduction mode, Power-down mode, or Power-save mode by the Two-wire Serial Interface interrupt. For example, in the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK Flag is controlled automatically by the Two-wire Serial Interface logic, the CPU cannot access the ACK bit directly.

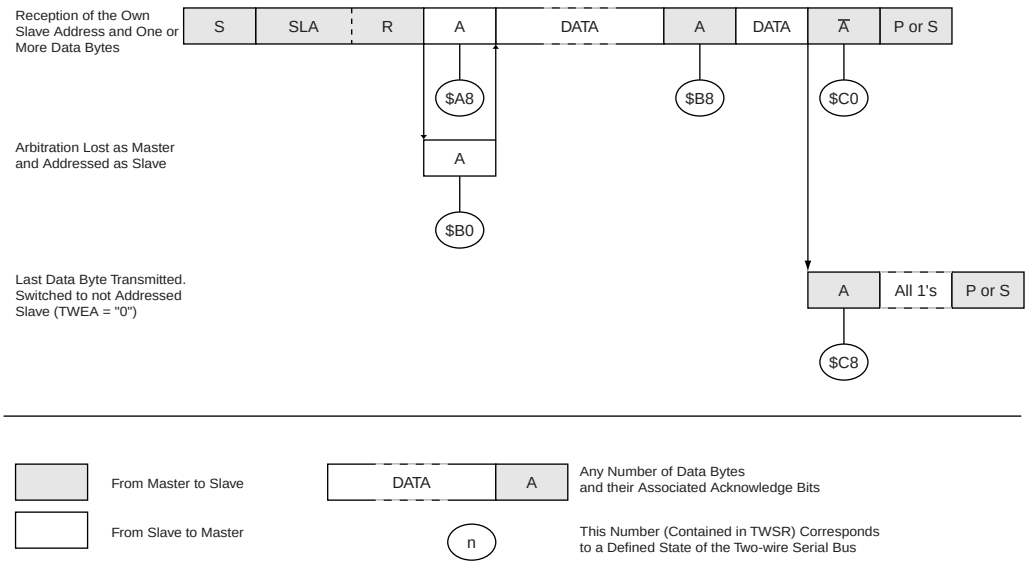
Assembly Code Example – Master Receiver Mode

```
;Part specific include file and TWI include file must be included.
; <Initialize registers TWAR and TWBR>

      ldi     r16, (1<<TWINT) | (1<<TWSTA) | (1<<TWEN)
      out     TWCR, r16     ;Send START condition

wait5:in      r16,TWCR      ; Wait for TWINT flag set. This indicates that
      sbrs    r16, TWINT    ; the START condition has been transmitted
      rjmp    wait5

      in      r16, TWSR     ; Check value of TWI Status Register. If status
      cpi     r16, START    ; different from START, go to ERROR
      brne    ERROR

      ldi     r16, 0xc9     ; Load SLA+R into TWDR Register
      out     TWDR, r16
      ldi     r16, (1<<TWINT) | (1<<TWEN)
      out     TWCR, r16     ; Clear TWINT bit in TWCR to start transmission of
                            ; SLA+R

wait6:in    r16,TWCR        ; Wait for TWINT flag set. This indicates that
      sbrs    r16, TWINT    ; SLA+R has been transmitted, and ACK/NACK has
      rjmp    wait6         ; been received

      in      r16, TWSR       ; Check value of TWI Status Register. If status
      cpi     r16, MR_SLA_ACK; different from MR_SLA_ACK, go to ERROR
      brne    ERROR

      ldi     r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out     TWCR, r16       ; Clear TWINT bit in TWCR to start reception of
                              ; data.
                              ; Setting TWEA causes ACK to be returned after
                              ; reception of data byte
wait7:in      r16,TWCR        ; Wait for TWINT flag set. This indicates that
      sbrs    r16, TWINT      ; data has been received and ACK returned
      rjmp    wait7

      in      r16, TWSR       ; Check value of TWI Status Register. If status
      cpi     r16, MR_DATA_ACK ; different from MR_DATA_ACK, go to ERROR
      brne    ERROR

      in      r16, TWDR       ; Input received data from TWDR.
      nop                     ;<do something with received data>
      ldi     r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out     TWCR, r16       ; Clear TWINT bit in TWCR to start reception of
                              ; data. Setting TWEA causes ACK to be returned
                              ; after reception of data byte
  ;<Receive more data bytes if needed>

  ;receive next to last data byte.
wait8:in      r16,TWCR        ; Wait for TWINT flag set. This indicates that
```

Table 35. Status Codes for Slave Transmitter Mode

Status Code (TWSR)	Status of the Two-wire Serial Bus and Two-wire Serial Interface hardware	Application Software Response					Next Action Taken by Two-wire Serial Interface Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$A8	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$B0	Arbitration lost in SLA+R/W as master; own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$B8	Data byte in TWDR has been transmitted; ACK has been received	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$C0	Data byte in TWDR has been transmitted; NOT ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
$C8	Last data byte in TWDR has been transmitted (TWEA = “0”); ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		No TWDR action or	1	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

Figure 55. Formats and States in the Slave Transmitter Mode



```
                              ; be received after data byte Master signalling end
                              ; of transmission)
wait17:in       r16,TWCR      ; Wait for TWINT flag set. This indicates that
      sbrs      r16, TWINT    ; data has been transmitted, and ACK/NACK has
      rjmp      wait17        ; been received

      in        r16, TWSR     ; Check value of TWI Status Register. If status
      cpi       r16, ST_LAST_DATA ; different from ST_LAST_DATA, go to ERROR
      brne      ERROR

      ldi       r16, (1<<TWINT) | (1<<TWEA) | (1<<TWEN)
      out       TWCR, r16     ; Continue address reckognition in Slave
Transmitter mode
```

Table 36. Status Codes for Miscellaneous States

Status Code (TWSR)	Status of the Two-wire Serial Bus and Two-wire Serial Interface hardware	Application Software Response					Next Action Taken by Two-wire Serial Interface Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$F8	No relevant state information available; TWINT = “0”	No TWDR action	No TWCR action				Wait or proceed current transfer
$00	Bus error due to an illegal START or STOP condition	No TWDR action	0	1	1	X	Only the internal hardware is affected, no STOP condition is sent on the bus. In all cases, the bus is released and TWSTO is cleared.

TWI Include File

```
;***** General Master staus codes *****
.equ    START               =$08                ;START has been
transmitted
.equ    REP_START           =$10          ;Repeated START has been
transmitted

;***** Master Transmitter staus codes *****
.equ    MT_SLA_ACK      =$18    ;SLA+W has been tramsmitted and ACK received
.equ    MT_SLA_NACK     =$20    ;SLA+W has been tramsmitted and NACK received
.equ    MT_DATA_ACK     =$28    ;Data byte has been tramsmitted and ACK
                                ;received
.equ    MT_DATA_NACK    =$30    ;Data byte has been tramsmitted and NACK
received
.equ    MT_ARB_LOST     =$38    ;Arbitration lost in SLA+W or data bytes

;***** Master Receiver staus codes *****
.equ    MR_ARB_LOST     =$38    ;Arbitration lost in SLA+R or NACK bit
.equ    MR_SLA_ACK      =$40    ;SLA+R has been tramsmitted and ACK received
.equ    MR_SLA_NACK     =$48    ;SLA+R has been tramsmitted and NACK received
.equ    MR_DATA_ACK     =$50    ;Data byte has been received and ACK returned
.equ    MR_DATA_NACK    =$58    ;Data byte has been received and NACK
                                ; tramsmitted

;***** Slave Transmitter staus codes *****
.equ    ST_SLA_ACK      =$A8    ;Own SLA+R has been received and ACK returned
.equ    ST_ARB_LOST_SLA_ACK=$B0;Arbitration lost in SLA+R/W as Master. Own
                                ; SLA+W has been received and ACK returned
.equ    ST_DATA_ACK     =$B8    ;Data byte has been tramsmitted and ACK
                                ;received
```

Figure 61. ADC Timing Diagram, Free Run Conversion

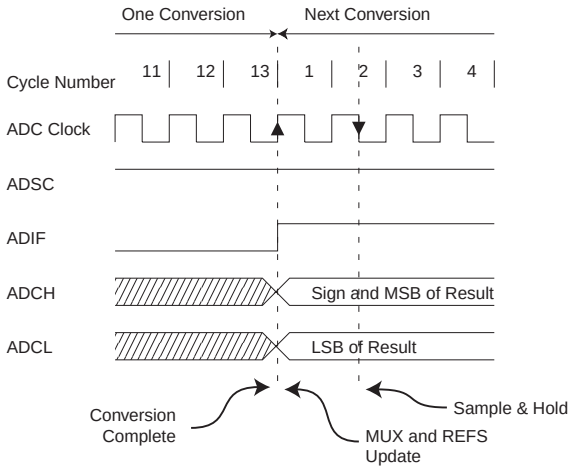


Table 39. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)	Conversion Time (µs)
Extended Conversion	13.5	25	125 - 500
Normal Conversions	1.5	13	65 - 260

ADC Noise Canceler Function

The ADC features a Noise Canceler that enables conversion during ADC Noise Reduction mode (see “Sleep Modes” on page 35) to reduce noise induced from the CPU core and other I/O peripherals. If other I/O peripherals must be active during conversion, this mode works equivalently for Idle mode. To make use of this feature, the following procedure should be used:

1. Make sure that the ADC is enabled and is not busy converting. Single Conversion Mode must be selected and the ADC conversion complete interrupt must be enabled.

   ADEN = 1

   ADSC = 0

   ADFR = 0

   ADIE = 1
2. Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
3. If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC Conversion Complete interrupt routine.

Port B As General Digital I/O

All eight bits in Port B are equal when used as digital I/O pins. PBn, General I/O pin: The DDBn bit in the DDRB Register selects the direction of this pin, if DDBn is set (one), PBn is configured as an output pin. If DDBn is cleared (zero), PBn is configured as an input pin. If PORTBn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, the PORTBn has to be cleared (zero), the pin has to be configured as an output pin, or the PUD bit has to be set. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Table 46. DDBn Effects on Port B Pins[1]

DDBn	PORTBn	PUD	I/O	Pull Up	Comment
0	0	x	Input	No	Tri-state (Hi-Z)
0	1	1	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	PBn will source current if ext. pulled low.
1	0	x	Output	No	Push-pull Zero Output
1	1	x	Output	No	Push-pull One Output

Note: 1. n: 7,6…0, pin number.

Alternate Functions Of PORTB

The alternate pin configuration is as follows:

- **SCK – PORTB, Bit 7**

SCK: Master Clock output, Slave Clock input pin for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB7. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB7 bit. See the description of the SPI port for further details.

- **MISO – PORTB, Bit 6**

MISO: Master Data input, Slave Data output pin for SPI channel. When the SPI is enabled as a Master, this pin is configured as an input regardless of the setting of DDB6. When the SPI is enabled as a Slave, the data direction of this pin is controlled by DDB6. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB6 bit. See the description of the SPI port for further details.

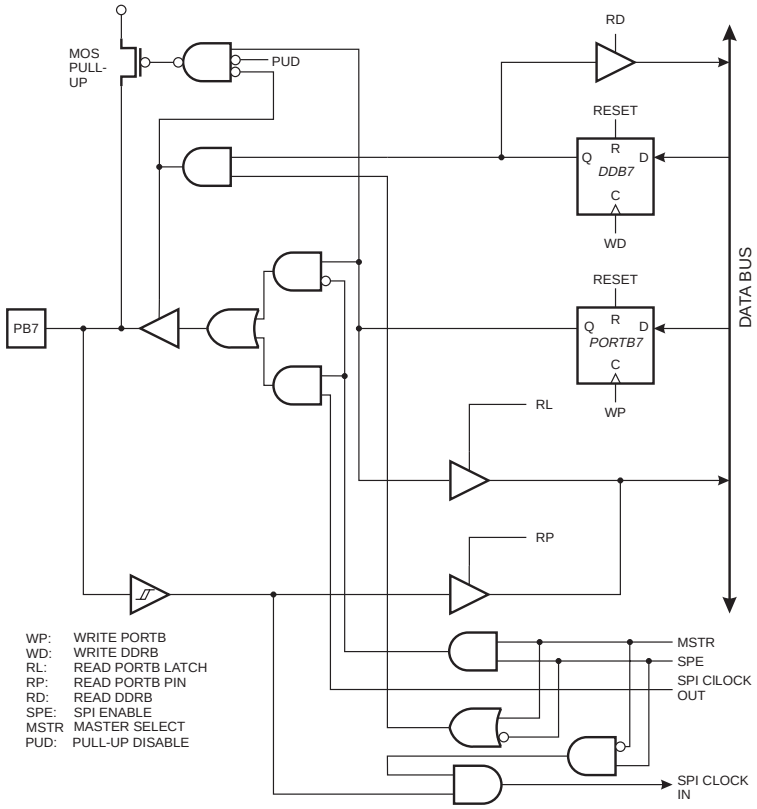
- **MOSI – PORTB, Bit 5**

MOSI: SPI Master Data output, Slave Data input for SPI channel. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB5 bit. See the description of the SPI port for further details.

- **$\overline{SS}$ – PORTB, Bit 4**

$\overline{SS}$: Slave Port Select input. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB4. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB4. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB4 bit. See the description of the SPI port for further details.

Figure 69. PORTB Schematic Diagram (Pin PB7)



Port D

Port D is an 8 bit bi-directional I/O port with internal pull-up resistors.

Three I/O memory address locations are allocated for Port D, one each for the Data Register – PORTD, $12($32), Data Direction Register – DDRD, $11($31) and the Port D Input Pins – PIND, $10($30). The Port D Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. Some Port D pins have alternate functions as shown in Table 49.

Table 49. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD0	RXD (UART Input Pin)
PD1	TXD (UART Output Pin)
PD2	INT0 (External Interrupt 0 Input)
PD3	INT1 (External Interrupt 1 Input)
PD4	OC1B (Timer/Counter1 Output CompareB Match Output)
PD5	OC1A (Timer/Counter1 Output CompareA Match Output)
PD6	ICP (Timer/Counter1 Input Capture Pin)
PD7	OC2 (Timer/Counter2 Output Compare Match Output)

The Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
$12 ($32)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
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 Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
$11 ($31)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
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 Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
$10 ($30)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

The Port D Input Pins Address – PIND – is not a register, and this address enables access to the physical value on each Port D pin. When reading PORTD, the PORTD Data Latch is read, and when reading PIND, the logical values present on the pins are read.

Reading the Signature Bytes

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

1. A: Load Command “0000 1000”.
2. C: Load Address Low Byte ($00 - $02).
3. Set $\overline{OE}$ to “0”, and BS to “0”. The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

Reading the Calibration Byte

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

1. A: Load Command “0000 1000”.
2. C: Load Address Low Byte, $00.
   Set $\overline{OE}$ to “0”, and BS1 to “1”. The Calibaration byte can now be read at DATA.
3. Set $\overline{OE}$ to “1”.

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL	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	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(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

Instruction Set Summary (Continued)

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
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
BIT AND BIT-TEST INSTRUCTIONS					
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
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1

Erratas

ATmega163(L) Errata Rev. F

- **Increased Interrupt Latency**
- **Interrupts Abort TWI Power-down**
- **TWI Master Does not Accept Spikes on Bus Lines**
- **TWCR Write Operations Ignored**
- **PWM not Phase Correct**
- **TWI is Speed Limited in Slave Mode**

6. Increased Interrupt Latency

In this device, some instructions are not interruptable, and will cause the interrupt latency to increase. The only practical problem concerns a loop followed by a two-word instruction while waiting for an interrupt. The loop may consist of a branch instruction or an absolute or relative jump back to itself like this:

```
loop: rjmp loop
<Two-word instruction>
```

In this case, a dead-lock situation arises.

Problem Fix/Workaround

In assembly, insert a nop instruction immediately after a loop to itself. The problem will normally be detected during development. In C, the only construct that will give this problem is an empty “for” loop; “for(;;)”. Use “while(1)” or “do{} while (1)” to avoid the problem.

5. Interrupts Abort TWI Power-down

TWI Power-down operation may be aborted by other interrupts. If an interrupt (e.g., INT0) occurs during TWI Power-down address watch and wakes the CPU up, the TWI aborts operation and returns to its idle state.

Problem Fix/Workaround

Ensure that the TWI Address Match is the only enabled interrupt when entering Power-down.

4. TWI Master Does not Accept Spikes on Bus Lines

When the part operates as Master, and the bus is idle (SDA = 1; SCL = 1), generating a short spike on SDA (SDA = 0 for a short interval), no interrupt is generated, and the status code is still $F8 (idle). But when the software initiates a new start condition and clears TWINT, nothing happens on SDA or SCL, and TWINT is never set again.

Problem Fix/Workaround

Either of the following:

1. Ensure that no spikes occur on SDA or SCL lines.
2. Receiving a valid START condition followed by a STOP condition provokes a bus error reported as a TWI interrupt with status code $00.
3. In a Single Master systems, the user should write the TWSTO bit immediately before writing the TWSTA bit.

3. TWCR Write Operation Ignored

Repeated write to TWCR must be delayed. If a write operation to TWCR is immediately followed by another write operation to TWCR, the first write operation may be ignored.

Problem Fix/Workaround

Ensure at least one instruction (e.g., nop) is executed between two writes to TWCR.

2. PWM not Phase Correct

In Phase-correct PWM mode, a change from OCRx = TOP to anything less than TOP does not change the OCx output. This gives a phase error in the following period.

Problem Fix/Workaround

Make sure this issue is not harmful to the application.

1. TWI is Speed Limited in Slave Mode

When the two-wire Serial Interface operates in Slave mode, frames may be undetected if the CPU frequency is less than 64 times the bus frequency.

Problem Fix/Workaround

Ensure that the CPU frequency is at least 64 times the TWI bus frequency.