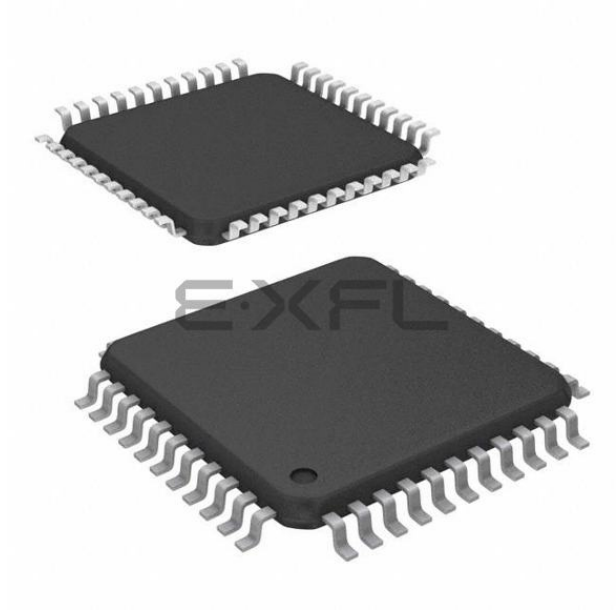
Microchip Technology - ATMEGA163-8AI Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

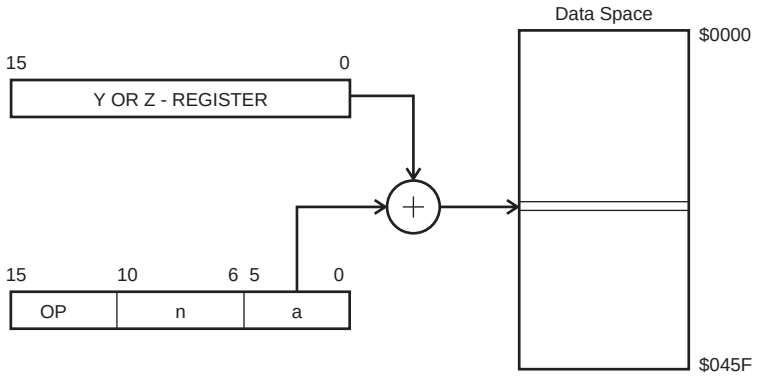
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	8MHz
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)	4V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°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/atmega163-8ai

Data Indirect with Displacement

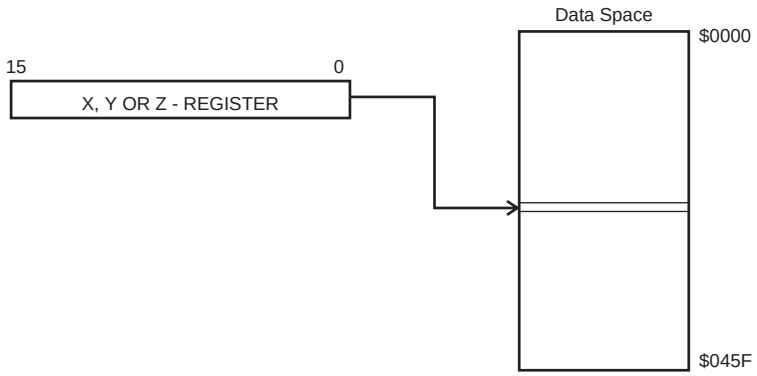
Figure 14. Data Indirect with Displacement



Operand address is the result of the Y- or Z-register contents added to the address contained in 6 bits of the instruction word.

Data Indirect

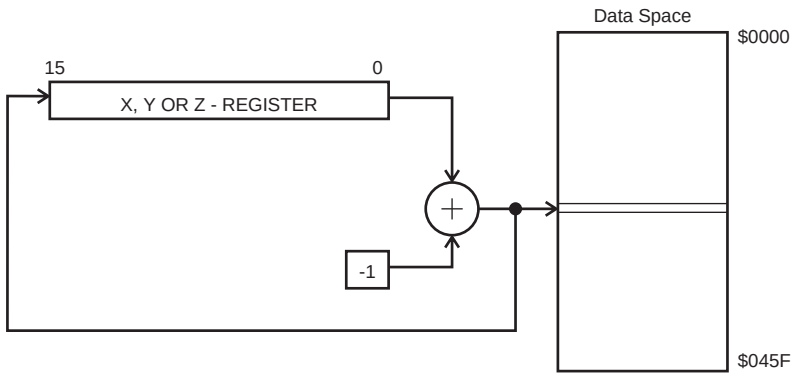
Figure 15. Data Indirect Addressing



Operand address is the contents of the X-, Y-, or the Z-register.

Data Indirect with Pre-decrement

Figure 16. Data Indirect Addressing with Pre-decrement



The X-, Y-, or the Z-register is decremented before the operation. Operand address is the decremented contents of the X-, Y-, or the Z-register.

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
...     ...       ...
```

- **Bit 2 – BORF: Brown-out Reset Flag**

This bit is set if a Brown-out Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the Flag.

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the Reset Flags to identify a reset condition, the user should read and then reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.

Internal Voltage Reference

ATmega163 features an internal bandgap reference with a nominal voltage of 1.22V. This reference is used for Brown-out Detection, and it can be used as an input to the Analog Comparator and ADC. The 2.56V reference to the ADC is also generated from the internal bandgap reference.

Voltage Reference Enable Signals and Start-up Time

To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (by programming the BODEN Fuse)
2. When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).
3. When the ADC is enabled.

Thus, when the BOD is not enabled, after setting the ACBG bit, the user must always allow the reference to start up before the output from the Analog Comparator is used. The bandgap reference uses typically 10 µA, and to reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

Interrupt Handling

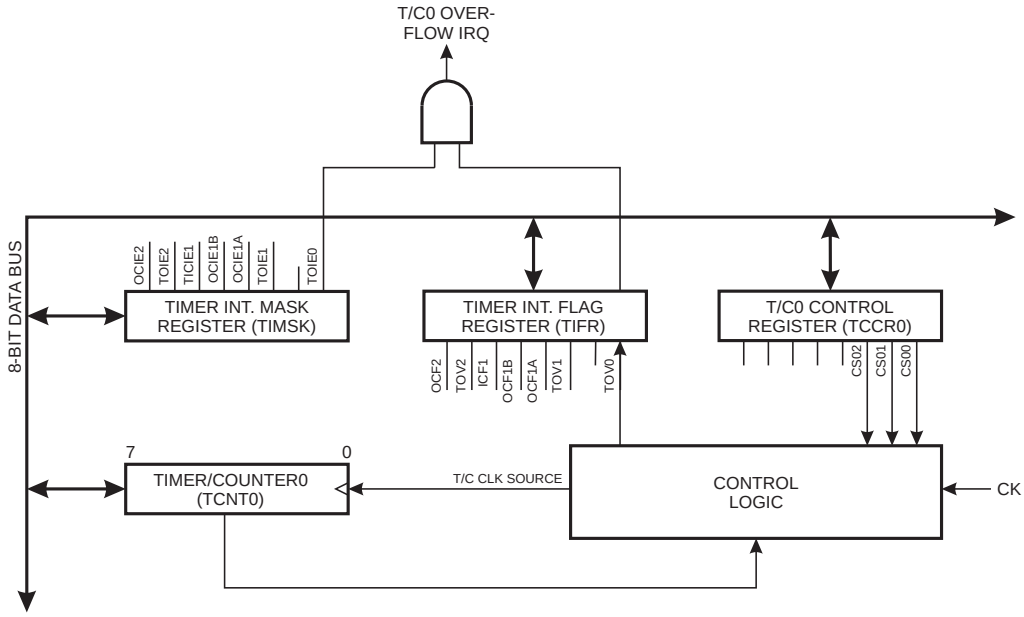
The ATmega163 has two 8-bit Interrupt Mask Control Registers: GIMSK – General Interrupt Mask Register and TIMSK – Timer/Counter Interrupt Mask Register.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared (zero) and all interrupts are disabled. The user software must set (one) the I-bit to enable nested interrupts. The I-bit is set (one) when a Return from Interrupt instruction – RETI – is executed.

When the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, hardware clears the corresponding flag that generated the interrupt. Some of the interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared.

If an interrupt condition occurs when the corresponding interrupt enable bit is cleared (zero), the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software.

Figure 32. Timer/Counter0 Block Diagram



Timer/Counter0 Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	–	–	–	–	–	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3 – Res: Reserved Bits**

These bits are reserved bits in the ATmega163 and always read as zero.

- **Bits 2..0 – CS02, CS01, CS00: Clock Select0, Bit 2, 1, and 0**

The Clock Select0 bits 2,1, and 0 define the prescaling source of Timer0.

Table 11. Clock0 Prescale Select

CS02	CS01	CS00	Description
0	0	0	Stop, Timer/Counter0 is stopped.
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	External Pin T0, falling edge
1	1	1	External Pin T0, rising edge

The Stop condition provides a Timer Enable/Disable function. The prescaled CK modes are scaled directly from the CK Oscillator clock. If the external pin modes are used for Timer/Counter0, transitions on PB0/(T0) will clock the counter even if the pin is configured as an output. This feature can give the user SW control of the counting.

- **Bit 3 – CTC2: Clear Timer/Counter on Compare Match**

When the CTC2 control bit is set (one), Timer/Counter2 is Reset to $00 in the CPU clock cycle following a Compare Match. If the control bit is cleared, the Timer/Counter2 continues counting and is unaffected by a Compare Match. When a prescaling of 1 is used, and the Compare Register is set to C, the Timer will count as follows if CTC2 is set:

... | C-1 | C | 0 | 1 |...

When the prescaler is set to divide by eight, the Timer will count like this:

... | C-1, C-1, C-1, C-1, C-1, C-1, C-1, C-1 | C, C, C, C, C, C, C, C | 0, 0, 0, 0, 0, 0, 0, 0 | 1, 1, 1, ...

In PWM mode, this bit has a different function. If the CTC2 bit is cleared in PWM mode, the Timer/Counter acts as an up/down counter. If the CTC2 bit is set (one), the Timer/Counter wraps when it reaches $FF. Refer to page 54 for a detailed description.

- **Bits 2, 1, 0 – CS22, CS21, CS20: Clock Select Bits 2, 1, and 0**

The Clock Select bits 2, 1, and 0 define the prescaling source of Timer/Counter2.

Table 20. Timer/Counter2 Prescale Select

CS22	CS21	CS20	Description
0	0	0	Timer/Counter2 is stopped.
0	0	1	PCK2
0	1	0	PCK2/8
0	1	1	PCK2/32
1	0	0	PCK2/64
1	0	1	PCK2/128
1	1	0	PCK2/256
1	1	1	PCK2/1024

The Stop condition provides a Timer Enable/Disable function. The prescaled modes are scaled directly from the PCK2 clock.

Timer/Counter2 – TCNT2

Bit	7	6	5	4	3	2	1	0	
$24 ($44)	MSB							LSB	TCNT2
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	

This 8-bit register contains the value of Timer/Counter2.

Timer/Counters2 is implemented as an up or up/down (in PWM mode) counter with read and write access. If the Timer/Counter2 is written to and a clock source is selected, it continues counting in the timer clock cycle following the write operation.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is not possible to set the EERE bit, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 24 lists the typical programming time for EEPROM access from the CPU

Table 24. EEPROM Programming Time.

Symbol	Number of Calibrated RC Oscillator Cycles	Min Programmingn Time	Max Programming Time
EEPROM write (from CPU)	2048	1.9 ms	3.8 ms

Preventing EEPROM Corruption

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

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

EEPROM data corruption can easily be avoided by following these design recommendations (one is sufficient):

1. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} Reset Protection circuit can be used. If a Reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply is voltage is sufficient.
2. Keep the AVR core in Power-down Sleep Mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the EEPROM Registers from unintentional writes.
3. Store constants in Flash memory if the ability to change memory contents from software is not required. Flash memory can not be updated by the CPU unless the boot loader software supports writing to the Flash and the Boot Lock bits are configured so that writing to the Flash memory from CPU is allowed. See “Boot Loader Support” on page 134 for details.

When data is transferred from UDR to the Shift Register, the UDRE (UART Data Register Empty) bit in the UART Status Register, USR, is set. When this bit is set (one), the UART is ready to receive the next character. At the same time as the data is transferred from UDR to the 10(11)-bit Shift Register, bit 0 of the Shift Register is cleared (start bit) and bit 9 or 10 is set (stop bit). If 9-bit data word is selected (the CHR9 bit in the UART Control Register, UCR is set), the TXB8 bit in UCR is transferred to bit 9 in the Transmit Shift Register.

On the Baud Rate clock following the transfer operation to the Shift Register, the start bit is shifted out on the TXD pin. Then follows the data, LSB first. When the stop bit has been shifted out, the Shift Register is loaded if any new data has been written to the UDR during the transmission. During loading, UDRE is set. If there is no new data in the UDR Register to send when the stop bit is shifted out, the UDRE Flag will remain set until UDR is written again. When no new data has been written, and the stop bit has been present on TXD for one bit length, the Transmit Complete Flag, TXC, in USR is set.

The TXEN bit in UCR enables the UART transmitter when set (one). When this bit is cleared (zero), the PD1 pin can be used for general I/O. When TXEN is set, the UART Transmitter will be connected to PD1, which is forced to be an output pin regardless of the setting of the DDD1 bit in DDRD.

- **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.

Slave Transmitter Mode

In the Slave Transmitter mode, a number of data bytes are transmitted to a Master Receiver (see Figure 55). The transfer is initialized as in the Slave Receiver mode. When TWAR and TWCR have been initialized, the Two-wire Serial Interface waits until it is addressed by its own slave address (or the general call address if enabled) followed by the Data Direction bit which must be “1” (read) for the Two-wire Serial Interface to operate in the Slave Transmitter mode. After its own slave address and the read bit have been received, the Two-wire Serial Interface Interrupt Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 35. The slave transmitter mode may also be entered if arbitration is lost while the Two-wire Serial Interface is in the Master mode (see state $B0).

If the TWEA bit is reset during a transfer, the Two-wire Serial Interface will transmit the last byte of the transfer and enter state $C0 or state $C8. the Two-wire Serial Interface is switched to the not addressed Slave mode, and will ignore the Master if it continues the transfer. Thus the Master Receiver receives all “1” as serial data. While TWEA is reset, the Two-wire Serial Interface does not respond to its own slave address. However, the Two-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the Two-wire Serial Interface from the Two-wire Serial Bus.

Assembly code illustrating operation of the Slave Receiver mode is given at the end of the TWI section.

Miscellaneous States

There are two status codes that do not correspond to a defined Two-wire Serial Interface state, see Table 36.

Status $F8 indicates that no relevant information is available because the Two-wire Serial Interface Interrupt Flag (TWINT) is not set yet. This occurs between other states, and when the Two-wire Serial Interface is not involved in a serial transfer.

Status $00 indicates that a bus error has occured during a Two-wire Serial Bus transfer. A bus error occurs when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte or an acknowledge bit. When a bus error occurs, TWINT is set. To recover from a bus error, the TWSTO Flag must set and TWINT must be cleared by writing a logic one to it. This causes the Two-wire Serial Interface to enter the not addressed Slave mode and to clear the TWSTO Flag (no other bits in TWCR are affected). The SDA and SCL lines are released and no STOP condition is transmitted.

Table 33. Status Codes for Master Receiver 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	
$08	A START condition has been transmitted	Load SLA+R	X	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
$10	A repeated START condition has been transmitted	Load SLA+R or	X	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
		Load SLA+W	X	0	1	X	SLA+W will be transmitted Logic will switch to Master Transmitter mode.
$38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	X	Two-wire Serial Bus will be released and not addressed Slave mode will be entered
		No TWDR actio	1	0	1	X	A START condition will be transmitted when the bus becomes free
$40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
$48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be Reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be Reset
$50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
$58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be Reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be Reset

Figure 53. Formats and States in the Master Receiver Mode

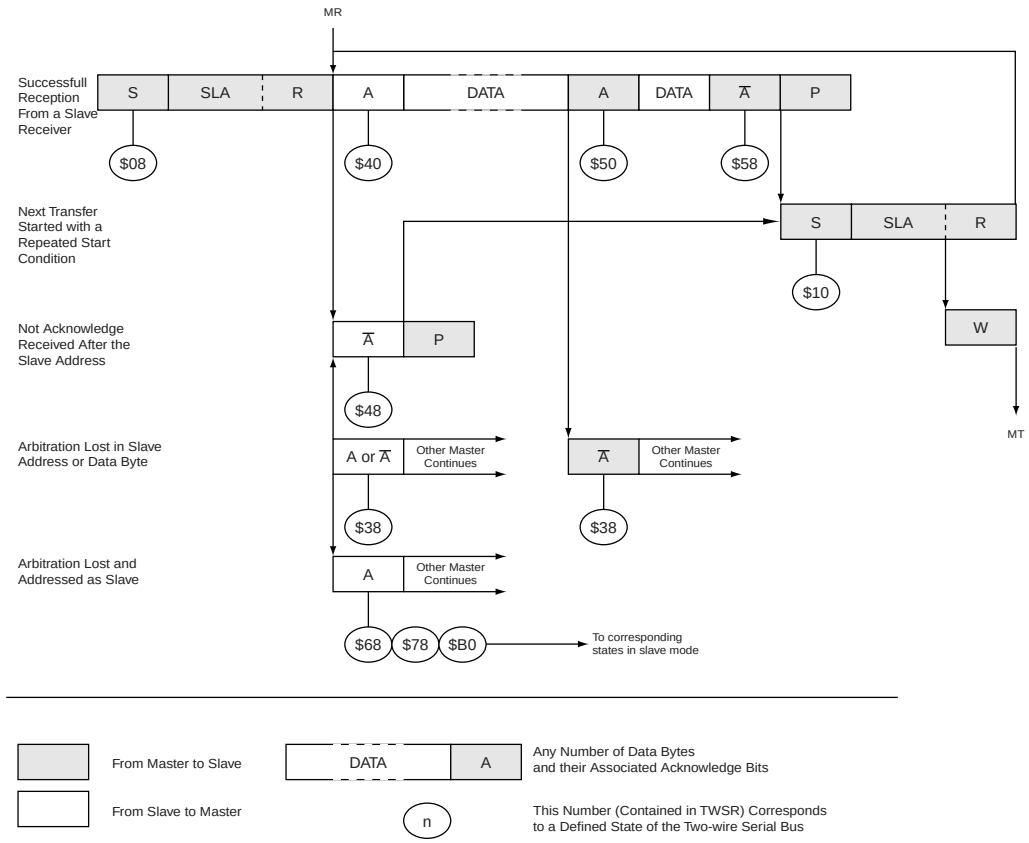
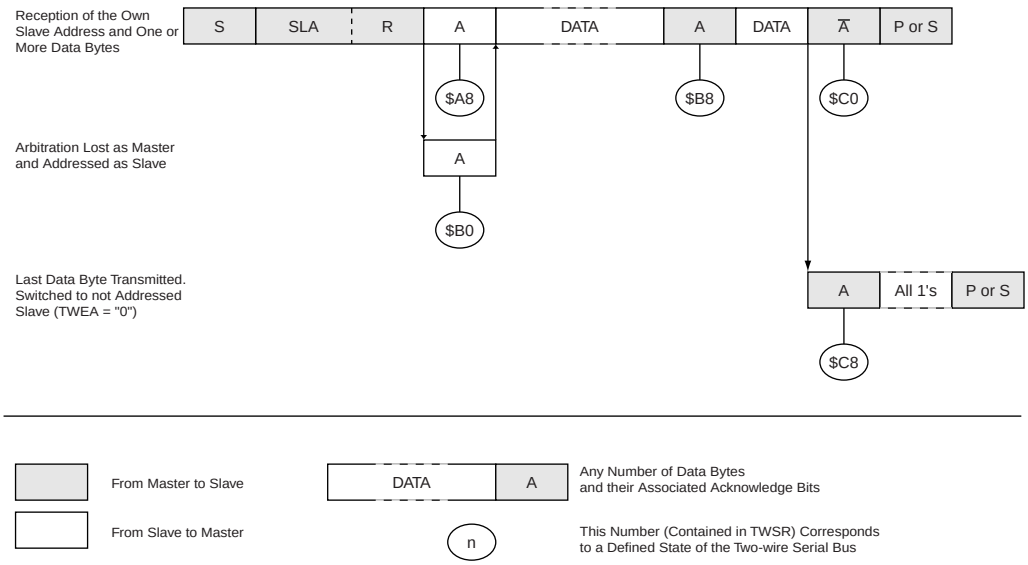


Table 35. Status Codes for Slave Transmitter Mode

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



Analog Comparator Multiplexed Input

It is possible to select any of the PA7..0 (ADC7..0) pins to replace the negative input to the Analog Comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in SFIOR) is set (one) and the ADC is switched off (ADEN in ADCSR is zero), MUX2..0 in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 38. If ACME is cleared (zero) or ADEN is set (one), PB3 (AIN1) is applied to the negative input to the Analog Comparator.

Table 38. Analog Comparator Multiplexed Input

ACME	ADEN	MUX2..0	Analog Comparator Negative Input
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

Analog to Digital Converter

Feature List

- **10-bit Resolution**
- **0.5 LSB Integral Non-linearity**
- **±2 LSB Absolute Accuracy**
- **65 - 260 µs Conversion Time**
- **Up to 15 kSPS at Maximum Resolution**
- **Up to 76 kSPS at 8-bit Resolution**
- **Eight Multiplexed Single Ended Input Channels**
- **Optional Left Adjustment for ADC Result Readout**
- **0 - V_{CC} ADC Input Voltage Range**
- **Selectable 2.56V ADC Reference Voltage**
- **Free Run or Single Conversion Mode**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

The ATmega163 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows each pin of Port A to be used as input for the ADC.

The ADC contains a Sample and Hold Amplifier which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 57.

The ADC has two separate analog supply voltage pins, AVCC and AGND. AGND must be connected to GND, and the voltage on AVCC must not differ more than ±0.3V from V_{CC}. See the paragraph ADC Noise Canceling Techniques on how to connect these pins.

Internal reference voltages of nominally 2.56V or AVCC are provided On-chip. The 2.56V reference may be externally decoupled at the AREF pin by a capacitor for better noise perfomance. See “Internal Voltage Reference” on page 29 for a description of the internal voltage reference.

- **Bits 2..0 – ADPS2..0: ADC Prescaler Select Bits**

These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

Table 42. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

The ADC Data Register – ADCL and ADCH

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
$05 ($25)	SIGN	–	–	–	–	–	ADC9	ADC8	ADCH
$04 ($24)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
$05 ($25)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
$04 ($24)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX affects the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9..0: ADC Conversion result**

These bits represent the result from the conversion. $000 represents analog ground, and $3FF represents the selected reference voltage minus one LSB.

Scanning Multiple Channels

Since change of analog channel always is delayed until a conversion is finished, the Free Running mode can be used to scan multiple channels without interrupting the converter. Typically, the ADC Conversion Complete interrupt will be used to perform the channel shift. However, the user should take the following fact into consideration:

The interrupt triggers once the result is ready to be read. In Free Running mode, the next conversion will start immediately when the interrupt triggers. If ADMUX is changed after the interrupt triggers, the next conversion has already started, and the old setting is used.

ADC Noise Canceling Techniques

Digital circuitry inside and outside the ATmega163 generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

1. The analog part of the ATmega163 and all analog components in the application should have a separate analog ground plane on the PCB. This ground plane is connected to the digital ground plane via a single point on the PCB.
2. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.
3. The AVCC pin on the ATmega163 should be connected to the digital V_{CC} supply voltage via an LC network as shown in Figure 62.
4. Use the ADC noise canceler function to reduce induced noise from the CPU.
5. If some Port A pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

Figure 62. ADC Power Connections

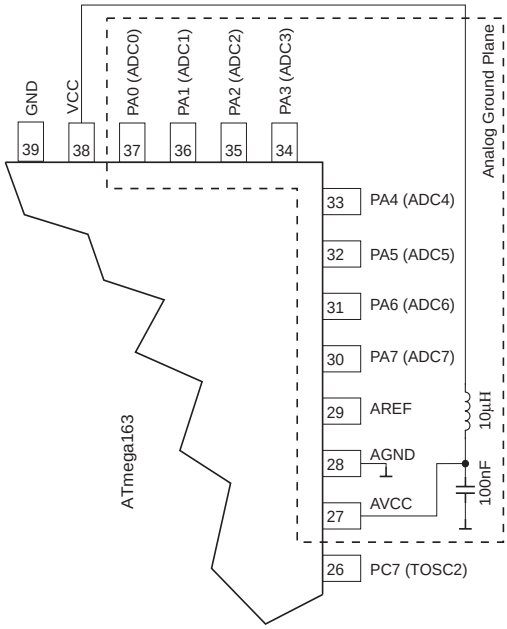
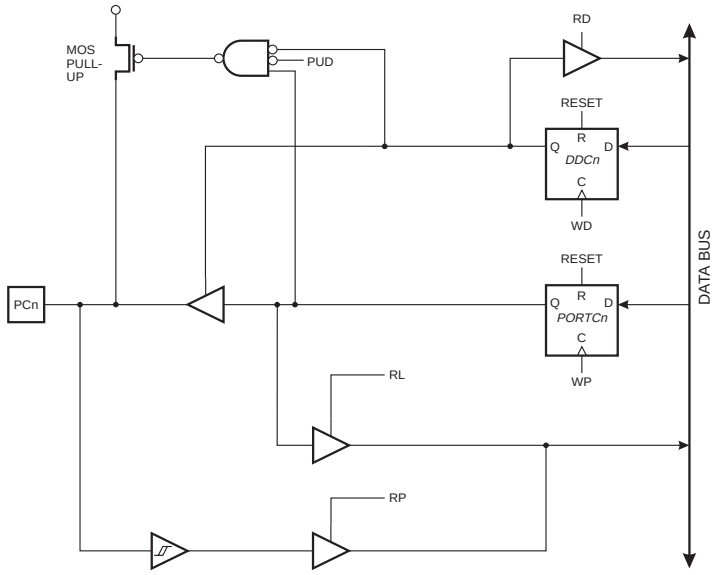
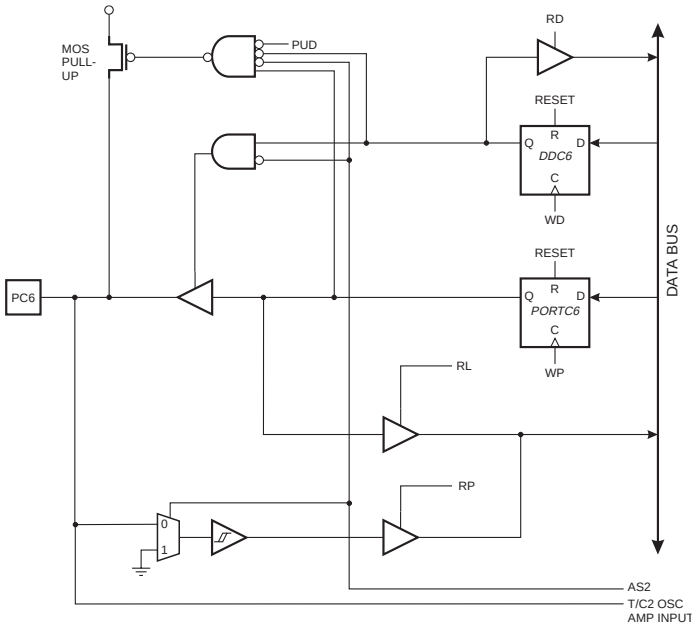


Figure 71. PORTC Schematic Diagram (Pins PC2 - PC5)



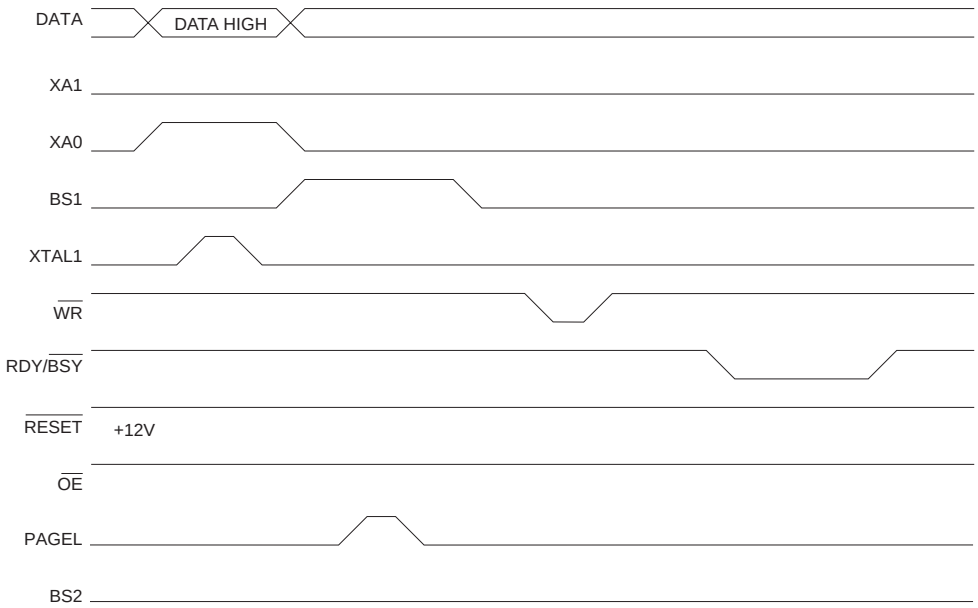
WP: WRITE PORTC
WD: WRITE DDRC
RL: READ PORTC LATCH
RP: READ PORTC PIN
RD: READ DDRC
PUD: PULL-UP DISABLE
n: 2..5

Figure 72. PORTC Schematic Diagram (Pins PC6)



WP: WRITE PORTC
WD: WRITE DDRC
RL: READ PORTC LATCH
RP: READ PORTC PIN
RD: READ DDRC
AS2: ASYNCH SELECT T/C2
PUD: PULL-UP DISABLE

Figure 83. Programming the Flash Waveforms (continued)



Programming the EEPROM

The programming algorithm for the EEPROM Data Memory is as follows (refer to "Programming the Flash" on page 147 for details on Command, Address and Data loading):

1. A: Load Command "0001 0001".
2. H: Load Address High Byte ($00 - $01)
3. B: Load Address Low Byte ($00 - $FF)
4. E: Load Data Low Byte ($00 - $FF)

L: Write Data Low Byte

1. Set BS to "0". This selects low data.
2. Give WR a negative pulse. This starts programming of the data byte. RDY/BSY goes low.
3. Wait until to RDY/BSY goes high before programming the next byte. (See Figure 84 for signal waveforms)

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered.

- The command needs only be loaded once when writing or reading multiple memory locations.
- Address high byte needs only be loaded before programming a new 256 word page in the EEPROM.
- Skip writing the data value $FF, that is the contents of the entire EEPROM after a Chip Erase.

These considerations also applies to Flash, EEPROM and Signature bytes reading.

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”.

External Clock Drive

Table 63. External Clock Drive

Symbol	Parameter	V_{CC} = 2.7V to 5.5V		V_{CC} = 4.0V to 5.5V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	4	0	8	MHz
t_{CLCL}	Clock Period	250		125		ns
t_{CHCX}	High Time	100		50		ns
t_{CLCX}	Low Time	100		50		ns
t_{CLCH}	Rise Time		1.6		0.5	µs
t_{CHCL}	Fall Time		1.6		0.5	µs

Table 64. External RC Oscillator, typical frequencies

R [kΩ]	C [pF]	f
100	70	100 kHz
31.5	20	1.0 MHz
6.5	20	4.0 MHz

Note: R should be in the range 3kΩ - 100kΩ, and C should be at least 20pF. The C values given in the table includes pin capacitance. This will vary with package type.

Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$00 ($20)	TWBR	Two-wire Serial Interface Bit Rate Register								82

Note: 1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

2. Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.