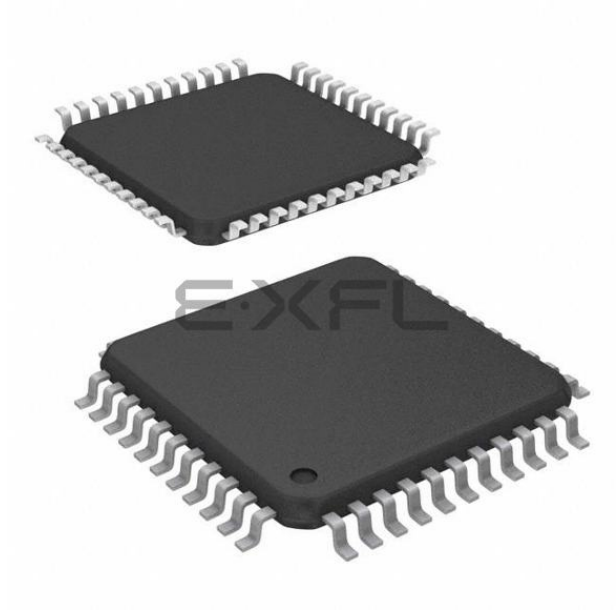


Welcome to E-XFL.COM

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.

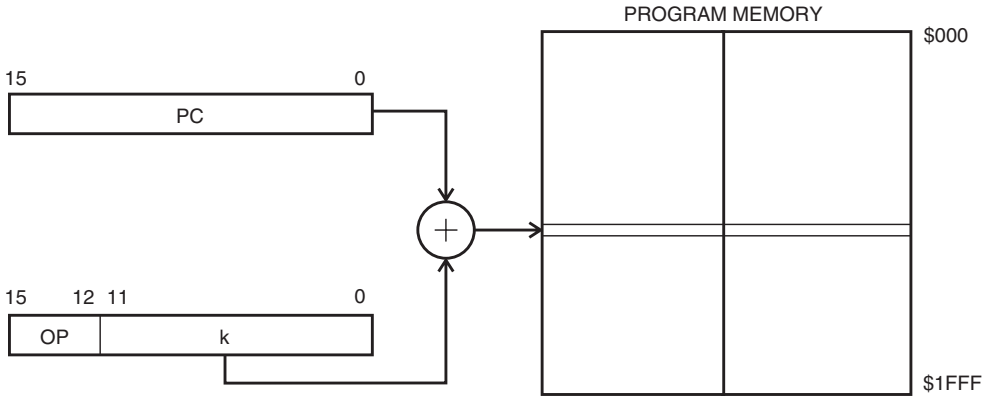
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	8MHz
Connectivity	SPI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	35
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	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
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/atmega161-8ai

Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the contents of the Z-register).

Relative Program Addressing, RJMP and RCALL

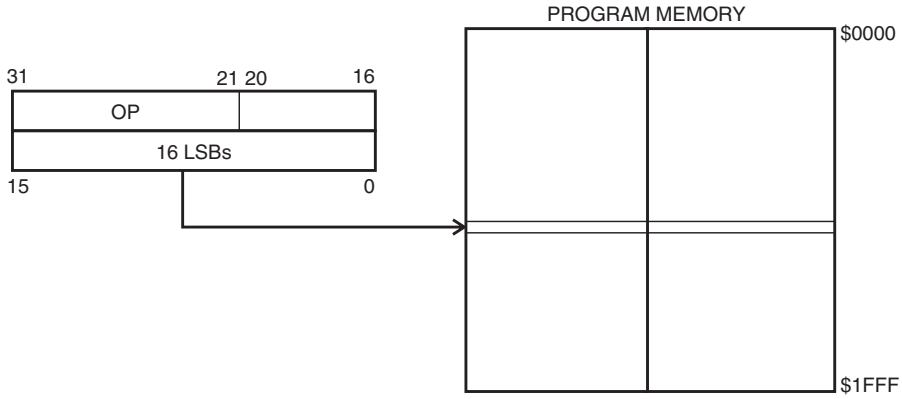
Figure 19. Relative Program Memory Addressing



Program execution continues at address PC + k + 1. The relative address k is -2048 to 2047.

Direct Program Addressing, JMP and CALL

Figure 20. Direct Program Addressing



Program execution continues at the address immediate in the instruction words.

Memory Access Times and Instruction Execution Timing

This section describes the general access timing concepts for instruction execution and internal memory access.

The AVR CPU is driven by the System Clock Ø, directly generated from the external clock crystal for the chip. No internal clock division is used.

Figure 21 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks and functions per power unit.

- **Bits 4..0 – Res: Reserved Bits**

These bits are reserved bits in the ATmega161 and always read as zero.

General Interrupt Flag Register – GIFR

Bit	7	6	5	4	3	2	1	0	
$3A ($5A)	INTF1	INTF0	INTF2	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – INTF1: External Interrupt Flag1**

When an event on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GIMSK are set (one), the MCU will jump to the Interrupt Vector at address $004. The Flag is cleared when the interrupt routine is executed. Alternatively, the Flag can be cleared by writing a logical “1” to it.

- **Bit 6 – INTF0: External Interrupt Flag0**

When an event on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in GIMSK are set (one), the MCU will jump to the Interrupt Vector at address $002. The Flag is cleared when the interrupt routine is executed. Alternatively, the Flag can be cleared by writing a logical “1” to it.

- **Bit 5 – INTF2: External Interrupt Flag2**

When an event on the INT2 pin triggers an interrupt request, INTF2 becomes set (one). If the I-bit in SREG and the INT2 bit in GIMSK are set (one), the MCU will jump to the Interrupt Vector at address $006. The Flag is cleared when the interrupt routine is executed. Alternatively, the Flag can be cleared by writing a logical “1” to it.

- **Bits 4..0 – Res: Reserved Bits**

These bits are reserved bits in the ATmega161 and always read as zero.

Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
$39 ($59)	TOIE1	OCIE1A	OCIE1B	TOIE2	TICIE1	OCIE2	TOIE0	OCIE0	TIMSK
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	

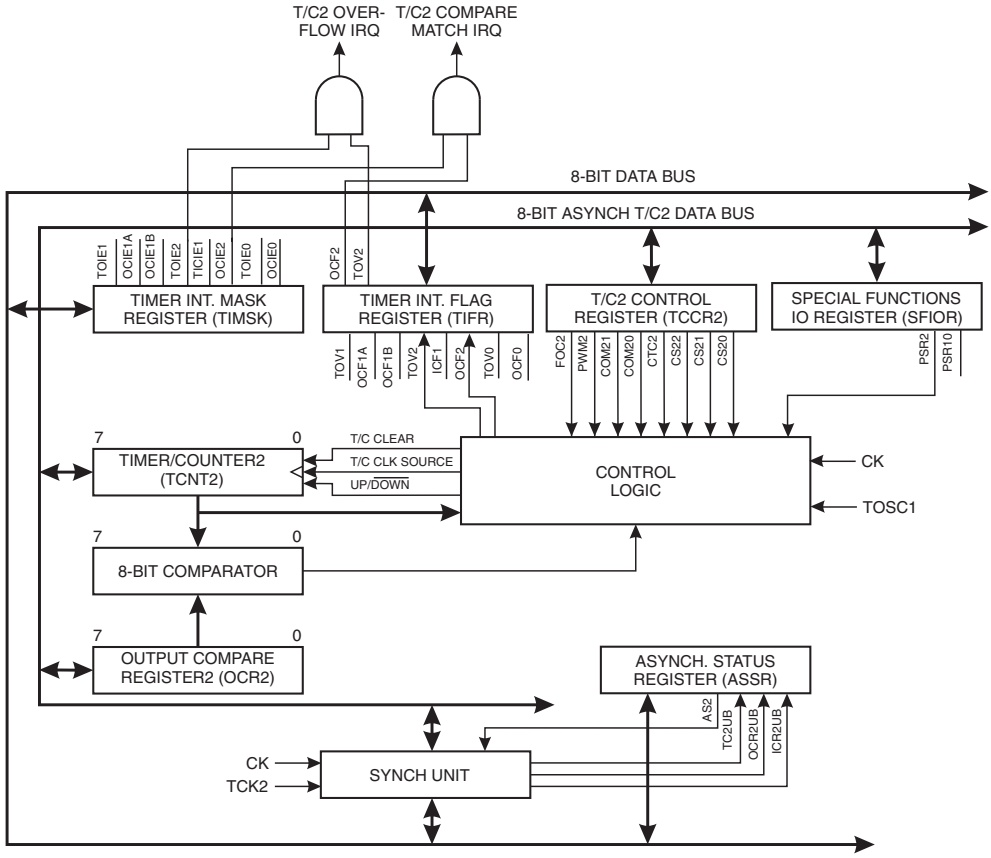
- **Bit 7 – TOIE1: Timer/Counter1 Overflow Interrupt Enable**

When the TOIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Overflow interrupt is enabled. The corresponding interrupt (at Vector $012) is executed if an overflow in Timer/Counter1 occurs, i.e., when the TOV1 bit is set in the Timer/Counter Interrupt Flag Register (TIFR).

- **Bit 6 – OCE1A: Timer/Counter1 Output CompareA Match Interrupt Enable**

When the OCIE1A bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 CompareA Match interrupt is enabled. The corresponding interrupt (at Vector $00e) is executed if a Compare A Match in Timer/Counter1 occurs, i.e., when the OCF1A bit is set in the Timer/Counter Interrupt Flag Register (TIFR).

Figure 32. Timer/Counter2 Block Diagram



The 8-bit Timer/Counter0 can select clock source from CK, prescaled CK or an external pin.The 8-bit Timer/Counter2 can select clock source from CK, prescaled CK or external TOSC1.

Both Timer/Counters can be stopped as described in sections “Timer/Counter0 Control Register – TCCR0” and “Timer/Counter2 Control Register – TCCR2”.

The various Status Flags (Overflow and Compare Match) are found in the Timer/Counter Interrupt Flag Register (TIFR). Control signals are found in the Timer/Counter Control Register (TCCR0 and TCCR2). The interrupt enable/disable settings are found in the Timer/Counter Interrupt Mask Register (TIMSK).

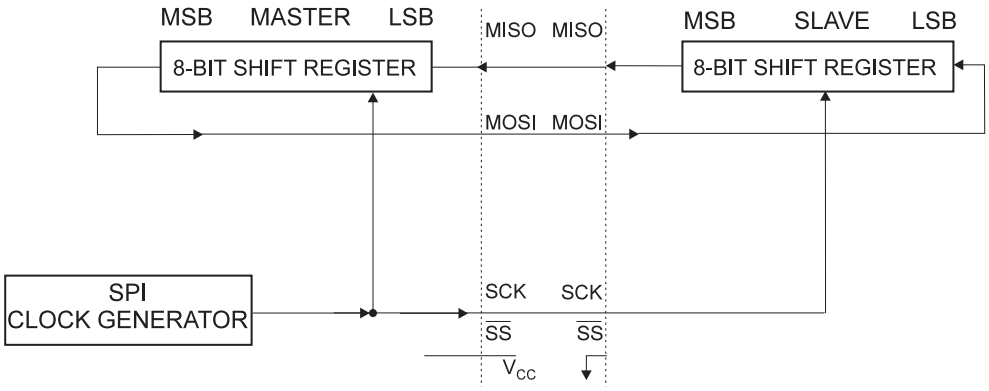
When Timer/Counter0 is externally clocked, the external signal is synchronized with the Oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 8-bit Timer/Counters feature both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities make the Timer/Counter0 useful for lower speed functions or exact timing functions with infrequent actions.

Timer/Counters 0 and 2 can also be used as 8-bit Pulse Width Modulators. In this mode, the Timer/Counter and the Output Compare Register serve as a glitch-free, stand-alone PWM with centered pulses. Refer to page 44 for a detailed description of this function.

The interconnection between Master and Slave CPUs with SPI is shown in Figure 41. The PB7(SCK) pin is the Clock Output in the Master mode and is the clock input in the Slave mode. Writing to the SPI Data Register of the Master CPU starts the SPI clock generator, and the data written shifts out of the PB5(MOSI) pin and into the PB5(MOSI) pin of the Slave CPU. After shifting one byte, the SPI clock generator stops, setting the End-of-Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Slave Select input, PB4($\overline{SS}$), is set low to select an individual Slave SPI device. The two Shift Registers in the Master and the Slave can be considered as one distributed 16-bit circular Shift Register. This is shown in Figure 41. When data is shifted from the Master to the Slave, data is also shifted in the opposite direction, simultaneously. This means that during one shift cycle, data in the Master and the Slave are interchanged.

Figure 41. SPI Master-Slave Interconnection



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

When the SPI is enabled, the data direction of the MOSI, MISO, SCK and $\overline{SS}$ pins is overridden according to Table 22.

Table 22. SPI Pin Overrides[1]

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

Note: 1. See “Alternate Functions of Port B” on page 92 for a detailed description of how to define the direction of the user defined SPI pins.

UARTs

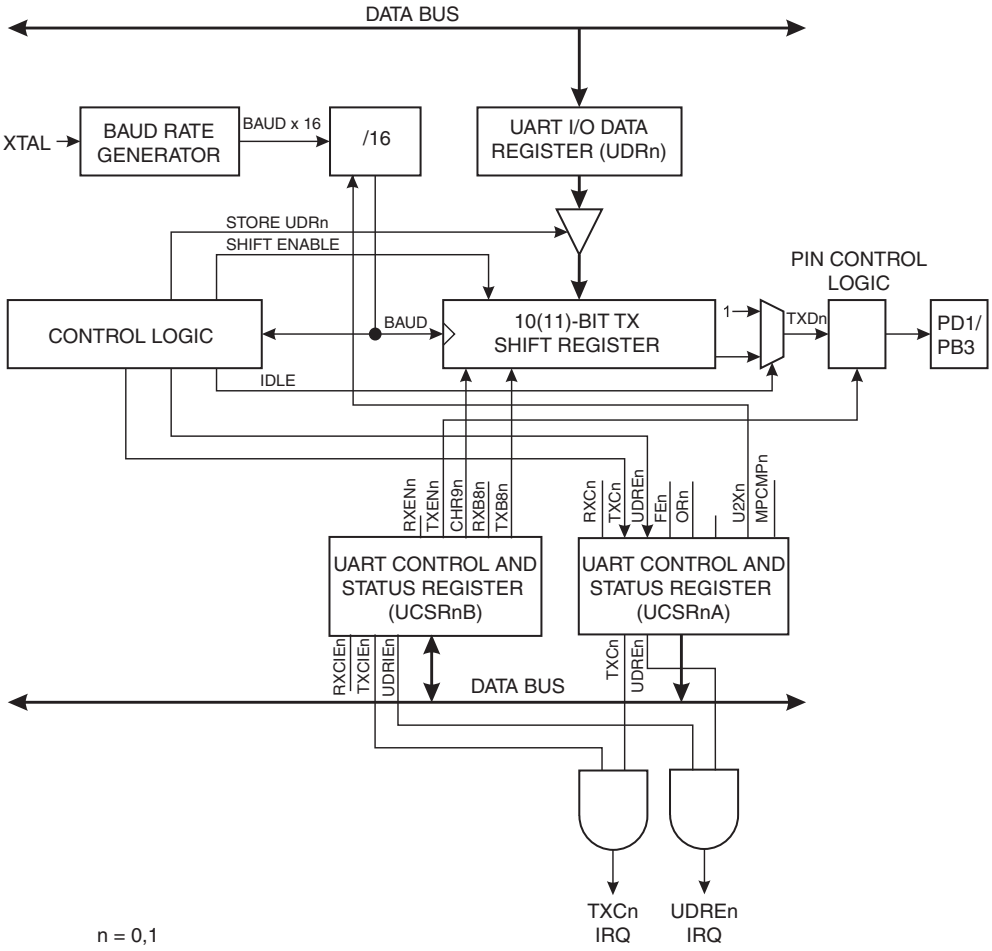
The ATmega161 features two full-duplex (separate Receive and Transmit Registers) Universal Asynchronous Receiver and Transmitters (UARTs). The main features are:

- **Baud Rate Generator Generates any Baud Rate**
- **High Baud Rates at low XTAL Frequencies**
- **8 or 9 Bits Data**
- **Noise Filtering**
- **Overrun Detection**
- **Framing Error Detection**
- **False Start Bit Detection**
- **Three Separate Interrupts on TX Complete, TX Data Register Empty, and RX Complete**
- **Multi-processor Communication Mode**
- **Double-speed UART Mode**

Data Transmission

A block schematic of the UART Transmitter is shown in Figure 44. The two UARTs are identical and the functionality is described in general for the two UARTs.

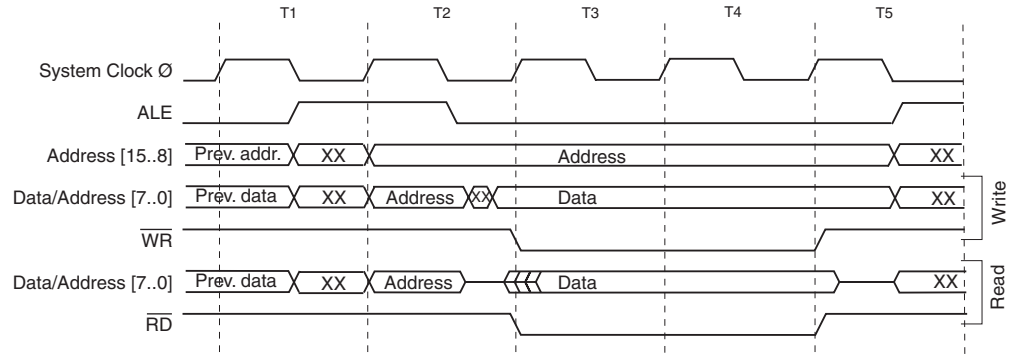
Figure 44. UART Transmitter



Data transmission is initiated by writing the data to be transmitted to the UART I/O Data Register, UDRn. Data is transferred from UDRn to the Transmit Shift Register when:

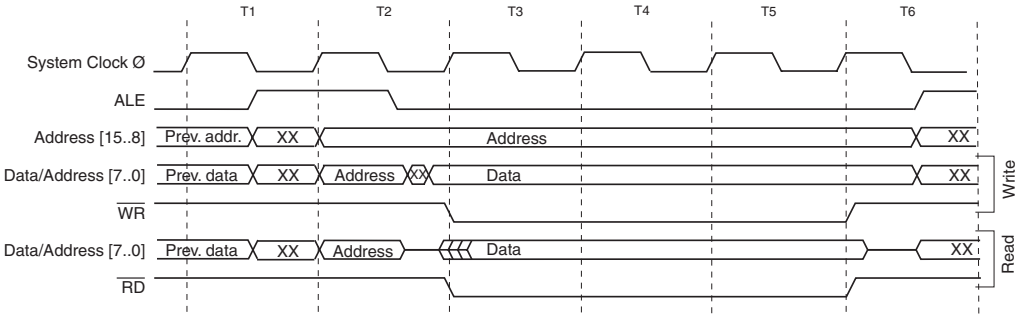
- A new character has been written to UDRn after the stop bit from the previous character has been shifted out. The Shift Register is loaded immediately.

Figure 51. External Data Memory Cycles with SRWn1 = 0 and SRWn0 = 1[1]



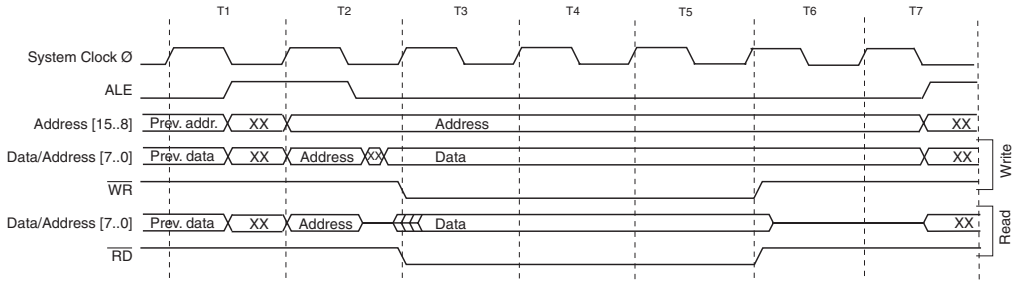
Note: 1. SRWn1 = SRW11 (upper page) or SRW01 (lower page), SRWn0 = SRW10 (upper page) or SRW00 (lower page). The ALE pulse in period T5 is only present if the next instruction accesses the RAM (internal or external). The Data and Address will only change in T5 if ALE is present (the next instruction accesses the RAM).

Figure 52. External Data Memory Cycles with SRWn1 = 1 and SRWn0 = 0[1]



Note: 1. SRWn1 = SRW11 (upper page) or SRW01 (lower page), SRWn0 = SRW10 (upper page) or SRW00 (lower page). The ALE pulse in period T6 is only present if the next instruction accesses the RAM (internal or external). The Data and Address will only change in T6 if ALE is present (the next instruction accesses the RAM).

Figure 53. External Data Memory Cycles with SRWn1 = 1 and SRWn0 = 1[1]



Note: 1. SRWn1 = SRW11 (upper page) or SRW01 (lower page), SRWn0 = SRW10 (upper page) or SRW00 (lower page). The ALE pulse in period T7 is only present if the next instruction accesses the RAM (internal or external). The Data and Address will only change in T7 if ALE is present (the next instruction accesses the RAM).

I/O Ports

All AVR ports have true read-modify-write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies for changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input).

Port A

Port A is an 8-bit bi-directional I/O port.

Three I/O memory address locations are allocated for the Port A, one each for the Data Register – PORTA, $1B($3B), Data Direction Register – DDRA, $1A($3A) and the Port A Input Pins – PINA, $19($39). The Port A Input Pins address is read-only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pull-up resistors. The Port A output buffers can sink 20 mA and thus 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 have alternate functions related to the optional external memory interface. Port A can be configured to be the multiplexed low-order address/data bus during accesses to the external Data memory. In this mode, Port A has internal pull-up resistors.

When Port A is set to the alternate function by the SRE (External SRAM Enable) bit in the MCUCR (MCU Control Register), the alternate settings override the Data Direction Register.

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
$1B ($3B)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
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	

Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
$1A ($3A)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
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	

Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
$19 ($39)	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
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 A Input Pins address (PINA) is not a register; this address enables access to the physical value on each Port A pin. When reading PORTA, the Port A Data Latch is read and when reading PINA, the logical values present on the pins are read.

Port A as General Digital I/O

All eight pins in Port A have equal functionality when used as digital I/O pins.

PAn, general I/O pin: The DDAn bit in the DDRA Register selects the direction of this pin. If DDAn is set (one), PAn is configured as an output pin. If DDAn is cleared (zero), PAn is configured as an input pin. If PORTAn is set (one) when the pin is configured as an input pin, the MOS pull-up resistor is activated. To switch the pull-up resistor off, the PORTAn has to be cleared (zero) or the pin has to be configured as an output pin. The

Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Table 29. DDAn Effects on Port A Pins[1]

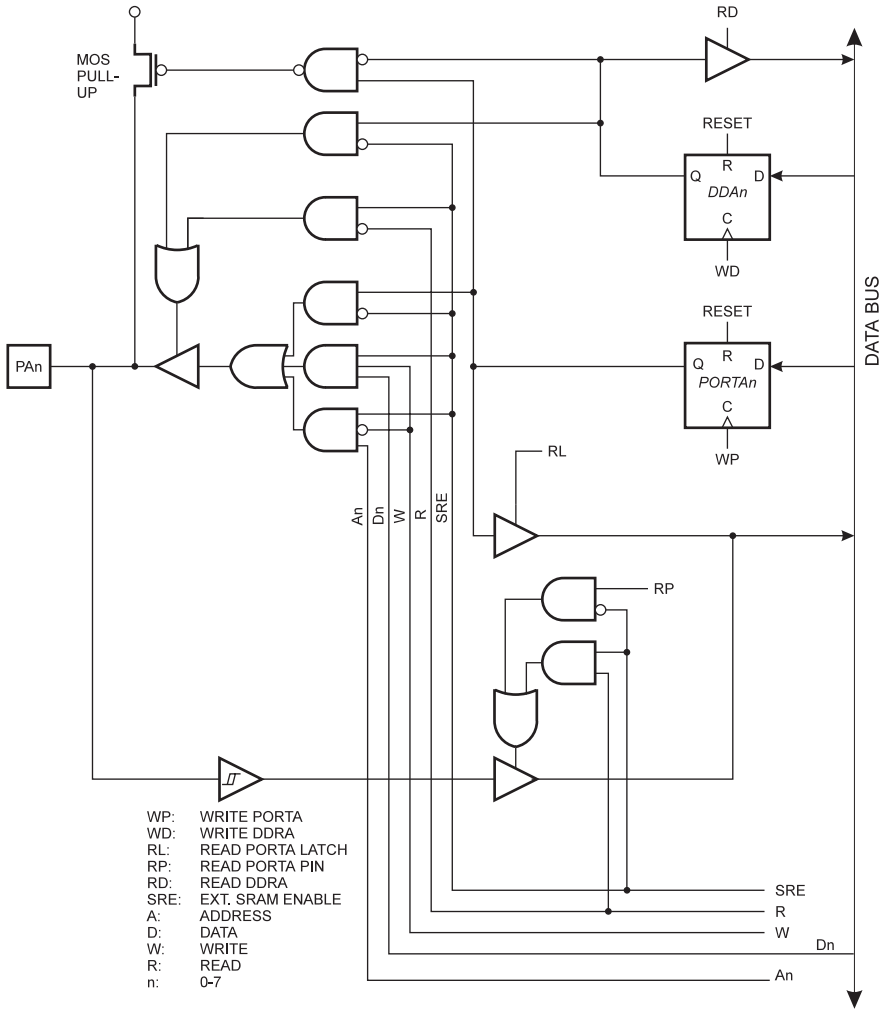
DDAn	PORTAn	I/O	Pull-up	Comment
0	0	Input	No	Tri-state (high-Z)
0	1	Input	Yes	PAn will source current if ext. pulled low.
1	0	Output	No	Push-pull Zero Output
1	1	Output	No	Push-pull One Output

Note: 1. n: 7,6…0, pin number

Port A Schematics

Note that all port pins are synchronized. The synchronization latch is, however, not shown in the figure.

Figure 55. Port A Schematic Diagrams (Pins PA0 - PA7)



Port B Schematics

Note that all port pins are synchronized. The synchronization latches are, however, not shown in the figures.

Figure 56. Port B Schematic Diagram (Pins PB0 and PB1)

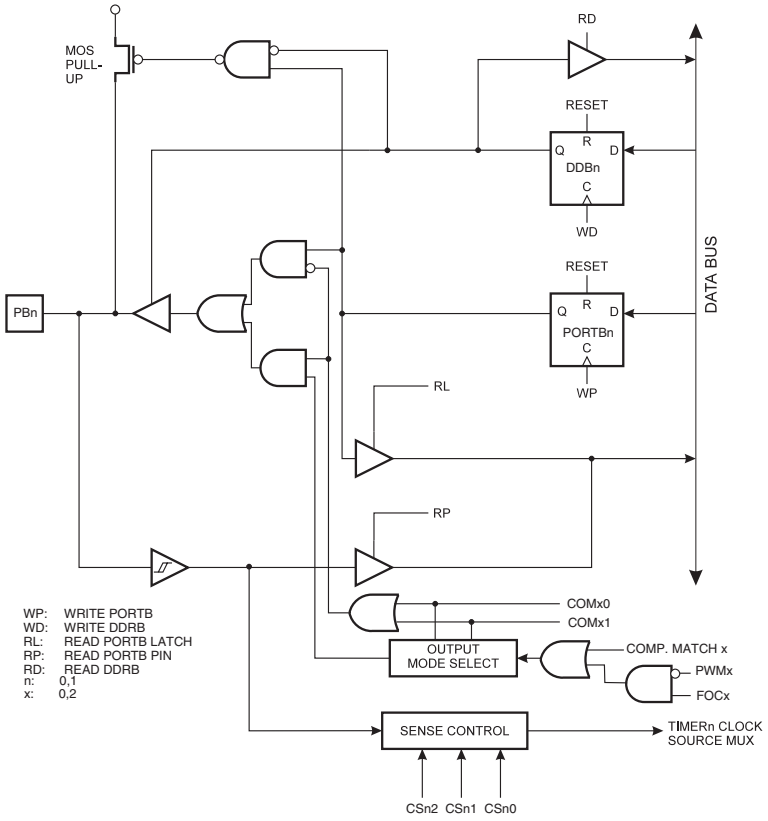


Table 32. DDCn Effects on Port C Pins[1]

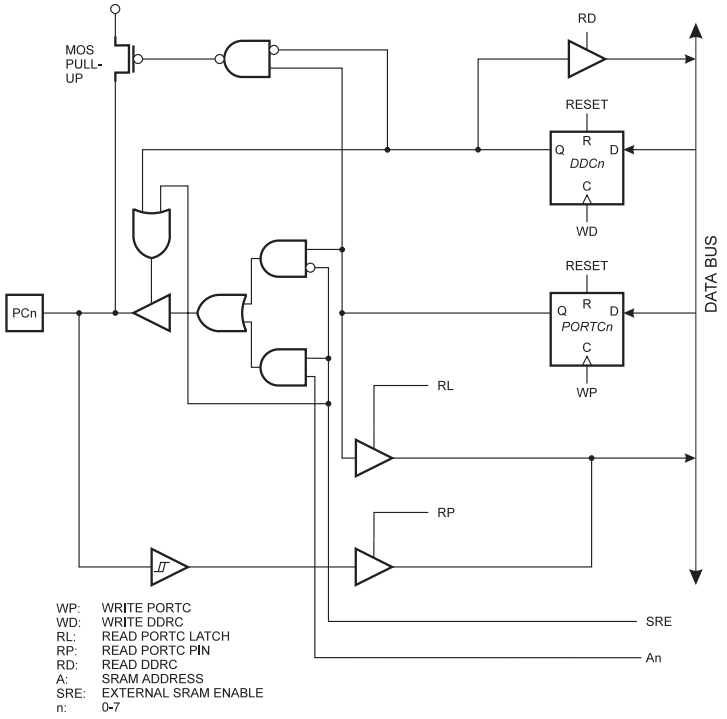
DDCn	PORTCn	I/O	Pull-up	Comment
0	0	Input	No	Tri-state (high-Z)
0	1	Input	Yes	PCn will source current if ext. pulled low
1	0	Output	No	Push-pull Zero Output
1	1	Output	No	Push-pull One Output

Note: 1. n: 7, 6,…0, pin number

Port C Schematics

Note that all port pins are synchronized. The synchronization latch is, however, not shown in the figure.

Figure 63. Port C Schematic Diagram (Pins PC0 - PC7)



Port E Schematics

Figure 71. Port E Schematic Diagram (Pin PE0)

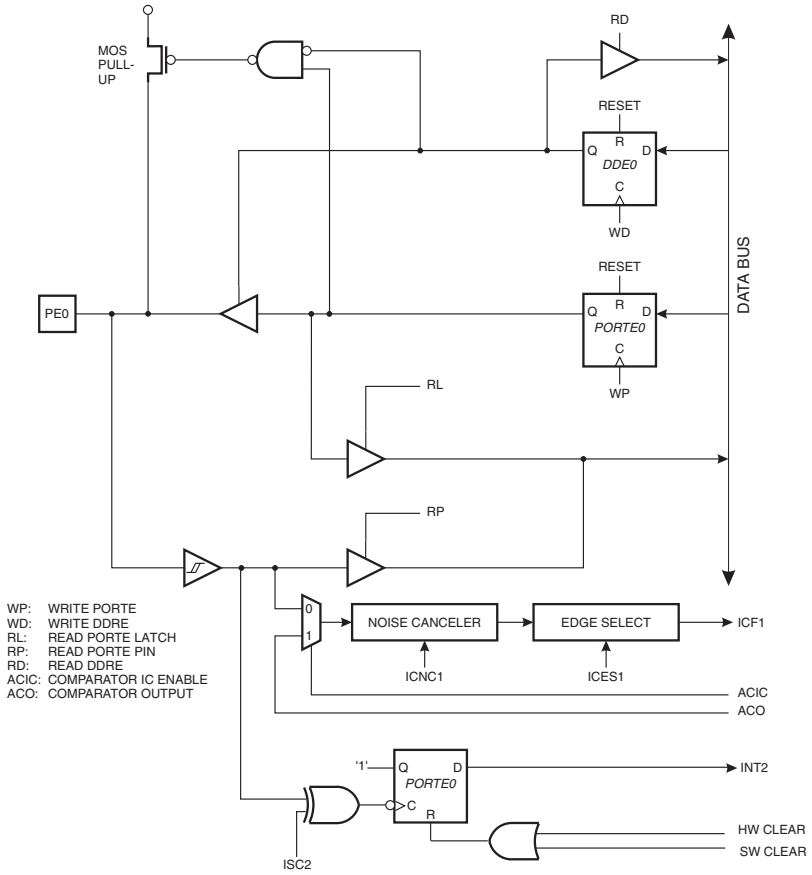
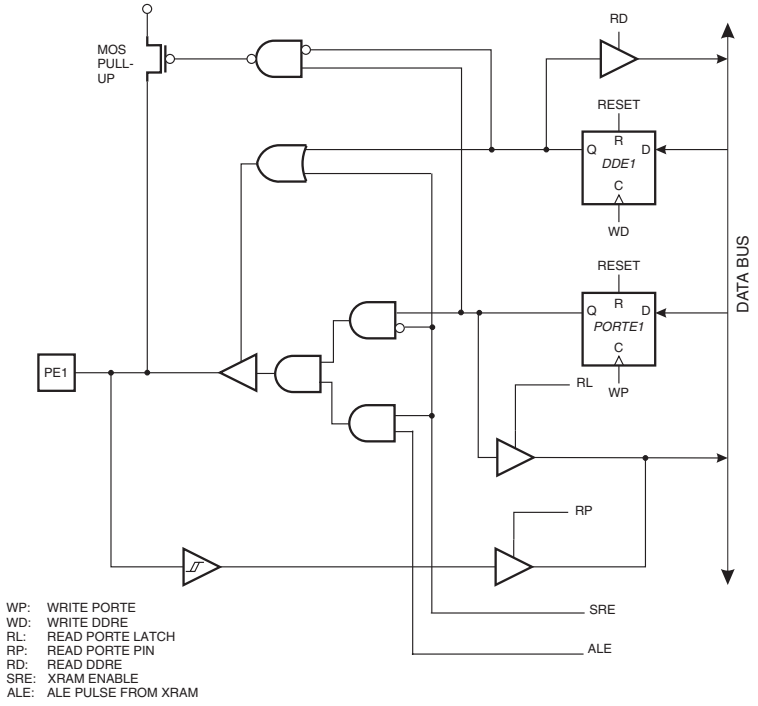


Figure 72. Port E Schematic Diagram (Pin PE1)



Program Memory Lock bits

The ATmega161 MCU provides six Lock bits that can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 40. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 40. Lock Bit Protection Modes [1]

Memory Lock bits			
LB Mode	**LB1**	**LB2**	**Protection Type**
1	1	1	No memory lock features enabled
2	0	1	Further programming of the Flash and EEPROM is disabled in parallel and Serial Programming modes. The Fuse bits are locked in both Serial and Parallel Programming modes.[1]
3	0	0	Further programming and verification of the Flash and EEPROM is disabled in parallel and Serial Programming modes. The Fuse bits are locked in both Serial and Parallel Programming modes.[1]
BLB0 Mode	**BLB02**	**BLB01**	
1	1	1	No restrictions for SPM, LPM accessing the Application Code section
2	1	0	SPM is not allowed to write to the Application Code section.
3	0	0	SPM is not allowed to write to the Application Code section and LPM executing from Boot Loader section is not allowed to read from the Application Code section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application Code section.
BLB1 Mode	**BLB12**	**BLB11**	
1	1	1	No restrictions for SPM, LPM accessing the Boot Loader section
2	1	0	SPM is not allowed to write the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section and LPM executing from the Application Code section is not allowed to read from the Boot Loader section.
4	0	1	LPM executing from the Application Code section is not allowed to read from the Boot Loader section.

Note: 1. Program the Fuse bits before programming the Lock bits.

Fuse bits

The ATmega161 has six Fuse bits: BOOTRST, SPIEN, SUT, and CKSEL [2:0].

- When BOOTRST is programmed (“0”), the Reset Vector is set to address $1E00, which is the first address location in the Boot Loader section of the Flash. If the BOOTRST is unprogrammed (“1”), the Reset Vector is set to address $0000. Default value is unprogrammed (“1”).
- When the SPIEN Fuse is programmed (“0”), Serial Program and Data Downloading is enabled. Default value is programmed (“0”). The SPIEN Fuse is not accessible in Serial Programming mode.
- The SUT Fuse changes the start-up times. Default value is unprogrammed (“1”).

Table 42. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
PAGEL	PD7	I	Program Memory Page Load
BS2	PA0	I	Byte Select 2 (Always low)
DATA	PB7 - 0	I/O	Bi-directional Data Bus (Output when $\overline{OE}$ is low)

Table 43. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or low address byte determined by BS1)
0	1	Load Data (High or low data byte for Flash determined by BS1)
1	0	Load Command
1	1	No Action, Idle

Table 44. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse bits
0010 0000	Write Lock bits
0001 0000	Write Flash
0001 0001	Write EEPROM
0000 1000	Read Signature Bytes
0000 0100	Read Fuse and Lock bits
0000 0010	Read Flash
0000 0011	Read EEPROM

Enter Programming Mode

The following algorithm puts the device in Parallel Programming mode:

1. Apply 4.5 - 5.5V between V_{CC} and GND.
2. Set $\overline{RESET}$ and BS pins to “0” and wait at least 500 ns.
3. Apply 11.5 - 12.5V to $\overline{RESET}$, and wait for at least 500 ns.

Chip Erase

The Chip Erase will erase the Flash and EEPROM memories plus Lock bits. The Lock bits are not reset until the Program memory has been completely erased. The Fuse bits are not changed. A Chip Erase must be performed before the Flash is reprogrammed.

Load Command “Chip Erase”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the command for Chip Erase.
4. Give $\overline{WR}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{BSY}$ goes low.
5. Wait until RDY/$\overline{BSY}$ goes high before loading a new command.

Programming the Flash

The Flash is organized as 128 pages of 128 bytes each. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

A. Load Command “Write Flash”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “0001 0000”. This is the command for Write Flash.
4. Give XTAL1 a positive pulse. This loads the command.

B. Load Address Low Byte

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

C. Load Data Low Byte

1. Set BS1 to “0”. This selects low data byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data Low byte ($00 - $FF).
4. Give XTAL1 a positive pulse. This loads the data byte.

D. Latch Data Low Byte

Give PAGEL a positive pulse. This latches the data Low byte. (See Figure 76 for signal waveforms.)

E. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data High byte ($00 - $FF).
4. Give XTAL1 a positive pulse. This loads the data byte.

F. Latch Data High Byte

Give PAGEL a positive pulse. This latches the data High byte.

G. Repeat “B” through “F” 64 times to fill the page buffer.

To address a page in the Flash, seven bits are needed (128 pages). The five most significant bits are read from address High byte as described in section “H” below. The two least significant page address bits, however, are the two most significant bits (bit7 and bit6) of the latest loaded address Low byte as described in section “B”.

H. 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 ($00 - $1F).
4. Give XTAL1 a positive pulse. This loads the address High byte.

I. Program Page

If I_{OH} exceeds the test condition, V_{OH} may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.

5. Minimum V_{CC} for power-down is 2V.

External Clock Drive Waveforms

Figure 82. External Clock

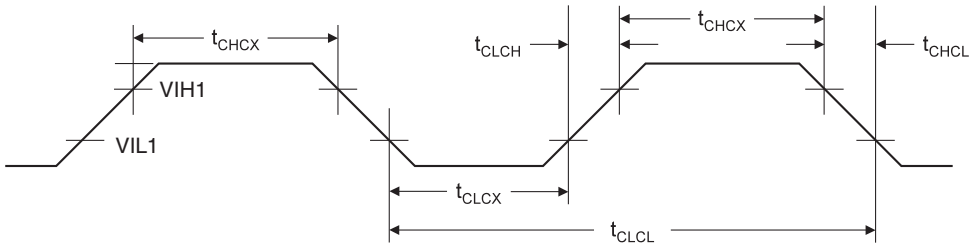


Table 50. External Clock Drive[1]

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

Notes: 1. See “External Data Memory Timing” for a description of how the duty cycle influences the timing for the external Data memory.

Figure 93. I/O Pin Sink Current vs. Output Voltage

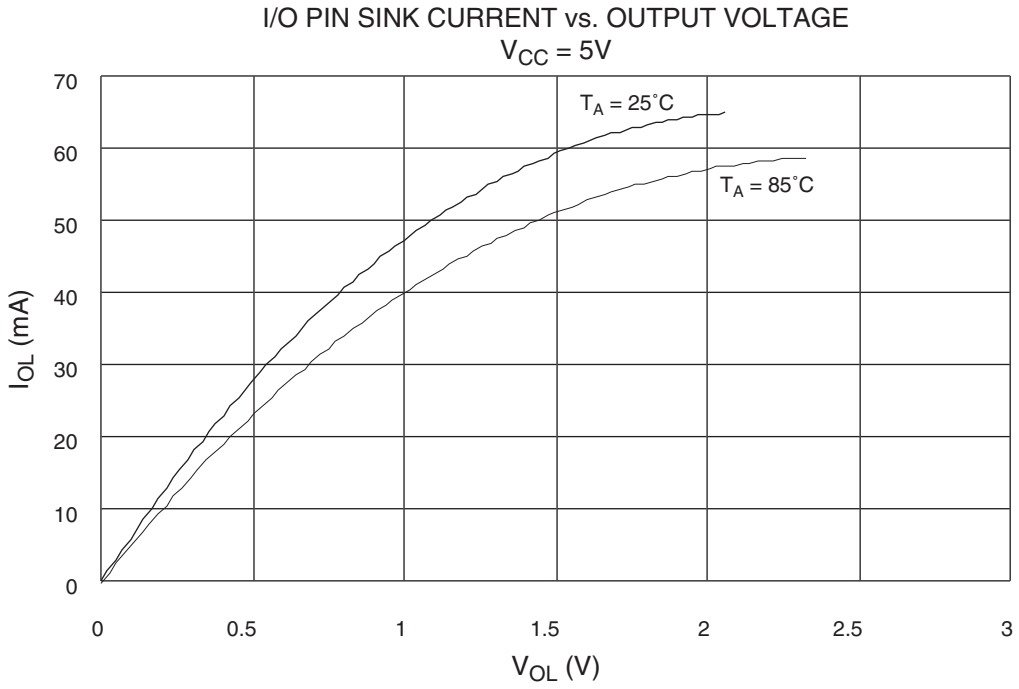
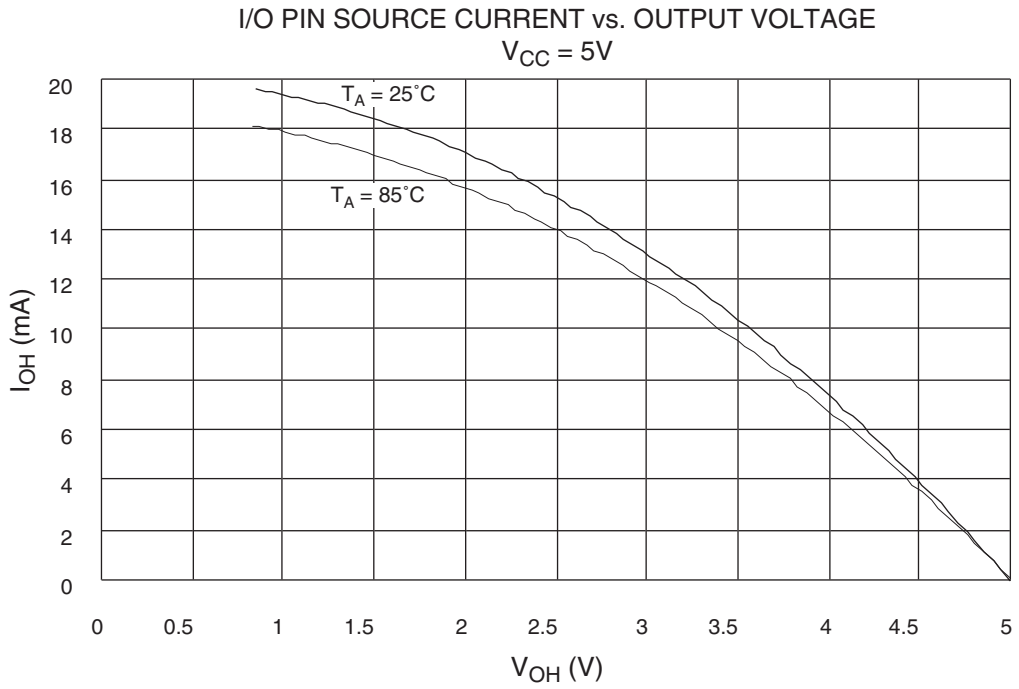


Figure 94. I/O Pin Source Current vs. Output Voltage



Errata

ATmega161 Rev. E

- **PWM not Phase Correct**
- **Increased Interrupt Latency**
- **Interrupt Return Fails when Stack Pointer Addresses the External Memory**
- **Writing UBBRH Affects both UART0 and UART1**
- **Store Program Memory Instruction May Fail**

5. 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.

4. 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.

3. Interrupt Return Fails when Stack Pointer Addresses the External Memory

When Stack Pointer addresses external memory (SPH:SPL > $45F), returning from interrupt will fail. The program counter will be updated with a wrong value and thus the program flow will be corrupted.

Problem Fix/Workaround

Address the stack pointer to internal SRAM or disable interrupts while Stack Pointer addresses external memory.

2. Writing UBBRH Affects Both UART0 and UART1

Writing UBRRHI updates baud rate generator for both UART0 and UART1. The baud rate generator's counter is updated each time either UBRR or UBRRHI are written. Since the UBRRHI regiSter is shared by UART0 and UART1, changing the baud rate for one UART will affect the operation of the other UART.

Problem Fix/Workaround

Do not update UBRRHI for one UART when transmitting/receiving data on the other.

1. Store Program Memory Instruction May Fail

At certain frequencies and voltages, the store program memory (SPM) instruction may fail.

Problem Fix/Workaround

Avoid using the SPM instruction.

Table of Contents