

Welcome to [E-XFL.COM](https://www.e-xfl.com)

What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "[Embedded - Microcontrollers](#)"

Details

Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, LCD, POR, PWM, WDT
Number of I/O	54
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	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-VQFN Dual Rows, Exposed Pad
Supplier Device Package	64-QFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega169p-16mchr

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega169P provides the following features: 16 Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes EEPROM, 1 Kbyte SRAM, 53 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, a complete On-chip LCD controller with internal step-up voltage, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, Universal Serial Interface with Start Condition Detector, an 8-channel, 10-bit ADC, a programmable Watchdog Timer with internal Oscillator, an SPI serial port, and five 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 and the LCD controller continues to run, allowing the user to maintain a timer base and operate the LCD display while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer, LCD controller and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega169P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega169P 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.

2.2.7 Port E (PE7:PE0)

Port E is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port E also serves the functions of various special features of the ATmega169P as listed on ["Alternate Functions of Port E" on page 81](#).

2.2.8 Port F (PF7:PF0)

Port F serves as the analog inputs to the A/D Converter.

Port F 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 F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PF7(TDI), PF5(TMS), and PF4(TCK) will be activated even if a reset occurs.

Port F also serves the functions of the JTAG interface, see ["Alternate Functions of Port F" on page 83](#).

2.2.9 Port G (PG5:PG0)

Port G is a 6-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port G output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port G pins that are externally pulled low will source current if the pull-up resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port G also serves the functions of various special features of the ATmega169P as listed on [page 85](#).

2.2.10 $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in [Table 28-4 on page 333](#). Shorter pulses are not guaranteed to generate a reset.

2.2.11 XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

2.2.12 XTAL2

Output from the inverting Oscillator amplifier.

2.2.13 AVCC

AVCC is the supply voltage pin for Port F and the A/D Converter. It should be externally connected to V_{CC} , even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

7.3 EEPROM Data Memory

The ATmega169P contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. This section describes the access between the EEPROM and the CPU, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For a detailed description of SPI, JTAG and Parallel data downloading to the EEPROM, see ["Serial Downloading" on page 310](#), ["Programming via the JTAG Interface" on page 316](#), and ["Parallel Programming Parameters, Pin Mapping, and Commands" on page 299](#) respectively.

7.3.1 EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access time for the EEPROM is given in [Table 7-1 on page 24](#). A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See ["Preventing EEPROM Corruption" on page 27](#) for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential). See ["EEPROM Register Description" on page 28](#) for supplementary description for each register bit:

1. Wait until EEWB becomes zero.
2. Wait until SPMEN in SPMCSR becomes zero.
3. Write new EEPROM address to EEAR (optional).
4. Write new EEPROM data to EEDR (optional).
5. Write a logical one to the EEMWE bit while writing a zero to EEWB in EECR.
6. Within four clock cycles after setting EEMWE, write a logical one to EEWB.

The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted. See ["Boot Loader Support – Read-While-Write Self-Programming" on page 280](#) for details about Boot programming.

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

13.2.6 Unconnected Pins

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

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

- **OC1B/PCINT14, Bit 6**

OC1B, Output Compare Match B output: The PB6 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDB6 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

PCINT14, Pin Change Interrupt Source 14: The PB6 pin can serve as an external interrupt source.

- **OC1A/PCINT13, Bit 5**

OC1A, Output Compare Match A output: The PB5 pin can serve as an external output for the Timer/Counter1 Output Compare A. The pin has to be configured as an output (DDB5 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.

PCINT13, Pin Change Interrupt Source 13: The PB5 pin can serve as an external interrupt source.

- **OC0A/PCINT12, Bit 4**

OC0A, Output Compare Match A output: The PB4 pin can serve as an external output for the Timer/Counter0 Output Compare A. The pin has to be configured as an output (DDB4 set (one)) to serve this function. The OC0A pin is also the output pin for the PWM mode timer function.

PCINT12, Pin Change Interrupt Source 12: The PB4 pin can serve as an external interrupt source.

- **MISO/PCINT11 – Port B, Bit 3**

MISO: Master Data input, Slave Data output pin for SPI. When the SPI is enabled as a Master, this pin is configured as an input regardless of the setting of DDB3. When the SPI is enabled as a Slave, the data direction of this pin is controlled by DDB3. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB3 bit.

PCINT11, Pin Change Interrupt Source 11: The PB3 pin can serve as an external interrupt source.

- **MOSI/PCINT10 – Port B, Bit 2**

MOSI: SPI Master Data output, Slave Data input for SPI. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB2. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB2. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB2 bit.

PCINT10, Pin Change Interrupt Source 10: The PB2 pin can serve as an external interrupt source.

- **SCK/PCINT9 – Port B, Bit 1**

SCK: Master Clock output, Slave Clock input pin for SPI. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB1. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB1. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB1 bit.

PCINT9, Pin Change Interrupt Source 9: The PB1 pin can serve as an external interrupt source.

- **\overline{SS} /PCINT8 – Port B, Bit 0**

\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 DDB0. As a Slave, the SPI is activated when this pin is driven

15.8.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM1x1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM1x1:0 = 0 tells the Waveform Generator that no action on the OC1x Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to [Table 15-1 on page 128](#). For fast PWM mode refer to [Table 15-2 on page 128](#), and for phase correct and phase and frequency correct PWM refer to [Table 15-3 on page 129](#).

A change of the COM1x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC1x strobe bits.

15.9 Modes of Operation

The mode of operation, that is, the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the *Waveform Generation mode* (WGM13:0) and *Compare Output mode* (COM1x1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM1x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM1x1:0 bits control whether the output should be set, cleared or toggle at a compare match. (See ["Compare Match Output Unit" on page 117](#).)

For detailed timing information refer to ["Timer/Counter Timing Diagrams" on page 126](#).

15.9.1 Normal Mode

The simplest mode of operation is the *Normal mode* (WGM13:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the *Timer/Counter Overflow Flag* (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 Flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

The Input Capture unit is easy to use in Normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

15.9.2 Clear Timer on Compare Match (CTC) Mode

In *Clear Timer on Compare* or CTC mode (WGM13:0 = 4 or 12), the OCR1A or ICR1 Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM13:0 = 4) or the ICR1 (WGM13:0 = 12). The OCR1A or ICR1 define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

15.9.4 Phase Correct PWM Mode

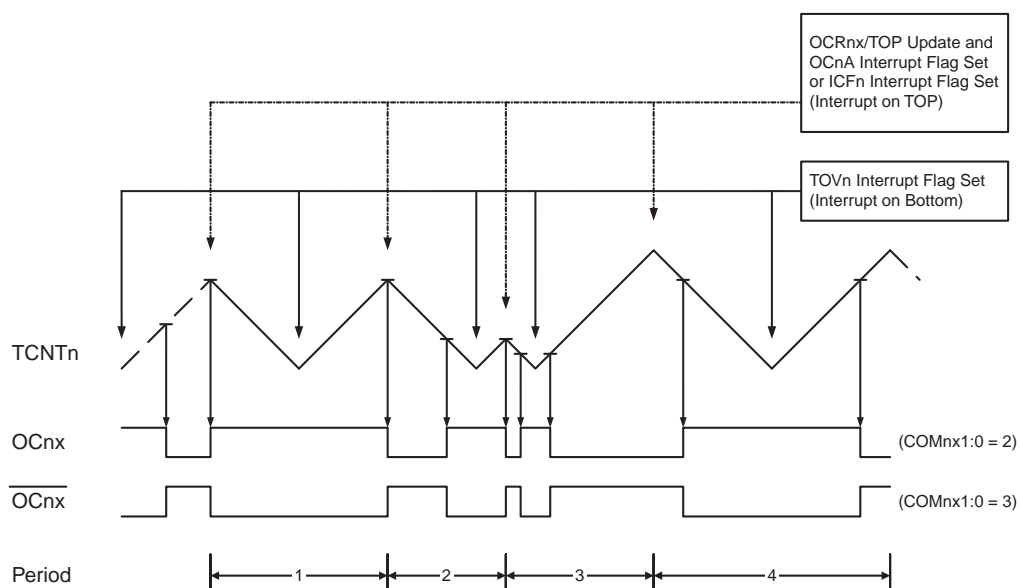
The *phase correct Pulse Width Modulation* or phase correct PWM mode (WGM13:0 = 1, 2, 3, 10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while upcounting, and set on the compare match while down counting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode can be fixed to 8-bit, 9-bit, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

$$R_{PCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase correct PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 1, 2, or 3), the value in ICR1 (WGM13:0 = 10), or the value in OCR1A (WGM13:0 = 11). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on [Figure 15-8](#). The figure shows phase correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a compare match occurs.

Figure 15-8. Phase Correct PWM Mode, Timing Diagram



16.4 Register Description

16.4.1 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	TSM	–	–	–	–	–	PSR2	PSR10	GTCCR
Read/Write	R/W	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TSM: Timer/Counter Synchronization Mode**

Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSR2 and PSR10 bits is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero, the PSR2 and PSR10 bits are cleared by hardware, and the Timer/Counters start counting simultaneously.

- **Bit 0 – PSR10: Prescaler Reset Timer/Counter1 and Timer/Counter0**

When this bit is one, Timer/Counter1 and Timer/Counter0 prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers.

19.11.5 UBRRLn and UBRRHn – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
(0xC5)	–	–	–	–	UBRRn[11:8]				UBRRHn
(0xC4)	UBRRn[7:0]								UBRRLn
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	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	
	0	0	0	0	0	0	0	0	

- **Bit 15:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRHn is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRHn contains the four most significant bits, and the UBRRLn contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRLn will trigger an immediate update of the baud rate prescaler.

Table 20-1. Relations between USIWM1:0 and the USI Operation

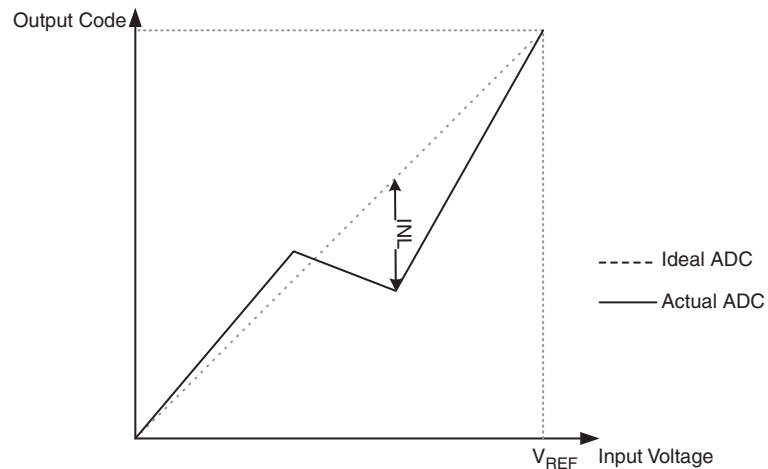
USIWM1	USIWM0	Description
0	0	Outputs, clock hold, and start detector disabled. Port pins operates as normal.
0	1	<p>Three-wire mode. Uses DO, DI, and USCK pins.</p> <p>The <i>Data Output</i> (DO) pin overrides the corresponding bit in the PORT Register in this mode. However, the corresponding DDR bit still controls the data direction. When the port pin is set as input the pins pull-up is controlled by the PORT bit.</p> <p>The <i>Data Input</i> (DI) and <i>Serial Clock</i> (USCK) pins do not affect the normal port operation. When operating as master, clock pulses are software generated by toggling the PORT Register, while the data direction is set to output. The USITC bit in the USICR Register can be used for this purpose.</p>
1	0	<p>Two-wire mode. Uses SDA (DI) and SCL (USCK) pins⁽¹⁾.</p> <p>The <i>Serial Data</i> (SDA) and the <i>Serial Clock</i> (SCL) pins are bi-directional and uses open-collector output drives. The output drivers are enabled by setting the corresponding bit for SDA and SCL in the DDR Register.</p> <p>When the output driver is enabled for the SDA pin, the output driver will force the line SDA low if the output of the Shift Register or the corresponding bit in the PORT Register is zero. Otherwise the SDA line will not be driven (that is, it is released). When the SCL pin output driver is enabled the SCL line will be forced low if the corresponding bit in the PORT Register is zero, or by the start detector. Otherwise the SCL line will not be driven.</p> <p>The SCL line is held low when a start detector detects a start condition and the output is enabled. Clearing the Start Condition Flag (USISIF) releases the line. The SDA and SCL pin inputs is not affected by enabling this mode. Pull-ups on the SDA and SCL port pin are disabled in Two-wire mode.</p>
1	1	<p>Two-wire mode. Uses SDA and SCL pins.</p> <p>Same operation as for the Two-wire mode described above, except that the SCL line is also held low when a counter overflow occurs, and is held low until the Counter Overflow Flag (USIOIF) is cleared.</p>

Note: 1. The DI and USCK pins are renamed to *Serial Data* (SDA) and *Serial Clock* (SCL) respectively to avoid confusion between the modes of operation.

• Bit 3:2 – USICS1:0: Clock Source Select

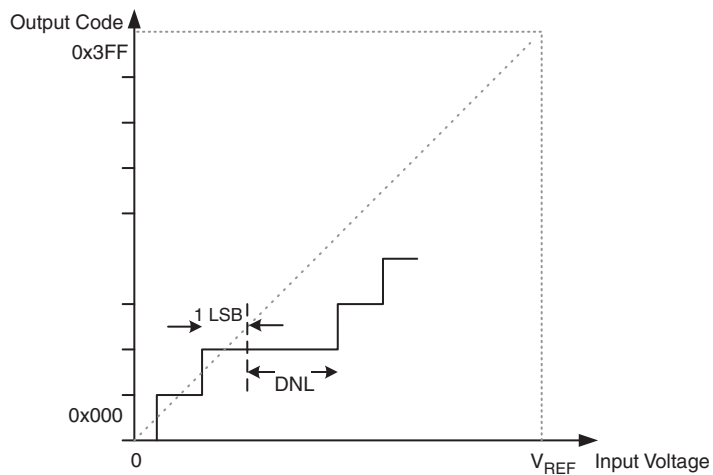
These bits set the clock source for the Shift Register and counter. The data output latch ensures that the output is changed at the opposite edge of the sampling of the data input (DI/SDA) when using external clock source (USCK/SCL). When software strobe or Timer/Counter0 Compare Match clock option is selected, the output latch is transparent and therefore the output is changed immediately. Clearing the USICS1..0 bits enables software strobe option. When using this option, writing a one to the USICLK bit clocks both the Shift Register and the counter. For external clock source (USICS1 = 1), the USICLK bit is no longer used as a strobe, but selects between external clocking and software clocking by the USITC strobe bit.

Figure 22-12. Integral Non-linearity (INL)



- **Differential Non-linearity (DNL):** The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSB). Ideal value: 0 LSB.

Figure 22-13. Differential Non-linearity (DNL)



- **Quantization Error:** Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSB wide) will code to the same value. Always ± 0.5 LSB.
- **Absolute Accuracy:** The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of offset, gain error, differential error, non-linearity, and quantization error. Ideal value: ± 0.5 LSB.

Table 22-4. Input Channel Selections

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input
00000	ADC0	N/A	
00001	ADC1		
00010	ADC2		
00011	ADC3		
00100	ADC4		
00101	ADC5		
00110	ADC6		
00111	ADC7		
01000	N/A		
01001			
01010			
01011			
01100			
01101			
01110			
01111			
10000		ADC0	ADC1
10001		ADC1	ADC1
10010	N/A	ADC2	ADC1
10011		ADC3	ADC1
10100		ADC4	ADC1
10101		ADC5	ADC1
10110		ADC6	ADC1
10111		ADC7	ADC1
11000		ADC0	ADC2
11001		ADC1	ADC2
11010		ADC2	ADC2
11011		ADC3	ADC2
11100		ADC4	ADC2
11101		ADC5	ADC2
11110	1.1V (V _{BG})	N/A	
11111	0V (GND)		

A debugger, like the AVR Studio, may however use one or more of these resources for its internal purpose, leaving less flexibility to the end-user.

A list of the On-chip Debug specific JTAG instructions is given in ["On-chip Debug Specific JTAG Instructions" on page 257](#).

The JTAGEN Fuse must be programmed to enable the JTAG Test Access Port. In addition, the OCDEN Fuse must be programmed and no Lock bits must be set for the On-chip debug system to work. As a security feature, the On-chip debug system is disabled when either of the LB1 or LB2 Lock bits are set. Otherwise, the On-chip debug system would have provided a back-door into a secured device.

The AVR Studio enables the user to fully control execution of programs on an AVR device with On-chip Debug capability, AVR In-Circuit Emulator, or the built-in AVR Instruction Set Simulator. AVR Studio supports source level execution of Assembly programs assembled with Atmel Corporation's AVR Assembler and C programs compiled with third party vendors' compilers.

AVR Studio runs under Microsoft® Windows® 95/98/2000, Windows NT® and Windows XP®.

For a full description of the AVR Studio, please refer to the AVR Studio User Guide. Only highlights are presented in this document.

All necessary execution commands are available in AVR Studio, both on source level and on disassembly level. The user can execute the program, single step through the code either by tracing into or stepping over functions, step out of functions, place the cursor on a statement and execute until the statement is reached, stop the execution, and reset the execution target. In addition, the user can have an unlimited number of code Break Points (using the BREAK instruction) and up to two data memory Break Points, alternatively combined as a mask (range) Break Point.

24.6 On-chip Debug Specific JTAG Instructions

The On-chip debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third party vendors only. Instruction opcodes are listed for reference.

24.6.1 PRIVATE0; 0x8

Private JTAG instruction for accessing On-chip debug system.

24.6.2 PRIVATE1; 0x9

Private JTAG instruction for accessing On-chip debug system.

24.6.3 PRIVATE2; 0xA

Private JTAG instruction for accessing On-chip debug system.

24.6.4 PRIVATE3; 0xB

Private JTAG instruction for accessing On-chip debug system.

Table 25-5. ATmega169P Boundary-scan Order

Bit Number	Signal Name	Module
193	COMP	ADC
192	ACLK	
191	ACTEN	
190	PRIVATE_SIGNAL1 ⁽¹⁾	
189	ADCBGEN	
188	ADCEN	
187	AMPEN	
186	DAC_9	
185	DAC_8	
184	DAC_7	
183	DAC_6	
182	DAC_5	
181	DAC_4	
180	DAC_3	
179	DAC_2	
178	DAC_1	
177	DAC_0	
176	EXTCH	
175	GNDEN	
174	HOLD	
173	IREFEN	
172	MUXEN_7	
171	MUXEN_6	
170	MUXEN_5	
169	MUXEN_4	
168	MUXEN_3	ADC
167	MUXEN_2	
166	MUXEN_1	
165	MUXEN_0	
164	NEGSEL_2	
163	NEGSEL_1	
162	NEGSEL_0	
161	PASSEN	
160	PRECH	
159	ST	
158	VCCREN	

Table 25-5. ATmega169P Boundary-scan Order

Bit Number	Signal Name	Module
132	PB0.Control	Port B
131	PB0.Pull-up_Enable	
130	PB1.Data	
129	PB1.Control	
128	PB1.Pull-up_Enable	
127	PB2.Data	
126	PB2.Control	
125	PB2.Pull-up_Enable	
124	PB3.Data	
123	PB3.Control	
122	PB3.Pull-up_Enable	
121	PB4.Data	
120	PB4.Control	
119	PB4.Pull-up_Enable	
118	PB5.Data	
117	PB5.Control	
116	PB5.Pull-up_Enable	
115	PB6.Data	
114	PB6.Control	
113	PB6.Pull-up_Enable	
112	PB7.Data	
111	PB7.Control	
110	PB7.Pull-up_Enable	
109	PG3.Data	Port G
108	PG3.Control	
107	PG3.Pull-up_Enable	
106	PG4.Data	
105	PG4.Control	
104	PG4.Pull-up_Enable	
103	PG5	(Observe Only)
102	RSTT	Reset Logic (Observe-only)
101	RSTHV	
100	EXTCLKEN	Enable signals for main Clock/Oscillators
99	OSCON	
98	RCOSCEN	
97	OSC32EN	

26.6 Entering the Boot Loader Program

Entering the Boot Loader takes place by a jump or call from the application program. This may be initiated by a trigger such as a command received via USART, or SPI interface. Alternatively, the Boot Reset Fuse can be programmed so that the Reset Vector is pointing to the Boot Flash start address after a reset. In this case, the Boot Loader is started after a reset. After the application code is loaded, the program can start executing the application code. Note that the fuses cannot be changed by the MCU itself. This means that once the Boot Reset Fuse is programmed, the Reset Vector will always point to the Boot Loader Reset and the fuse can only be changed through the serial or parallel programming interface.

Table 26-4. Boot Reset Fuse⁽¹⁾

BOOTRST	Reset Address
1	Reset Vector = Application Reset (address 0x0000)
0	Reset Vector = Boot Loader Reset (see Table 26-6 on page 292)

Note: 1. "1" means unprogrammed, "0" means programmed

27.7.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to ["Programming the Flash" on page 302](#) for details on Command and Address loading):

1. A: Load Command "0000 0011".
2. G: Load Address High Byte (0x00 - 0xFF).
3. B: Load Address Low Byte (0x00 - 0xFF).
4. Set \overline{OE} to "0", and BS1 to "0". The EEPROM Data byte can now be read at DATA.
5. Set \overline{OE} to "1".

27.7.8 Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (refer to ["Programming the Flash" on page 302](#) for details on Command and Data loading):

1. A: Load Command "0100 0000".
2. C: Load Data Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
3. Give \overline{WR} a negative pulse and wait for RDY/ \overline{BSY} to go high.

27.7.9 Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (refer to ["Programming the Flash" on page 302](#) for details on Command and Data loading):

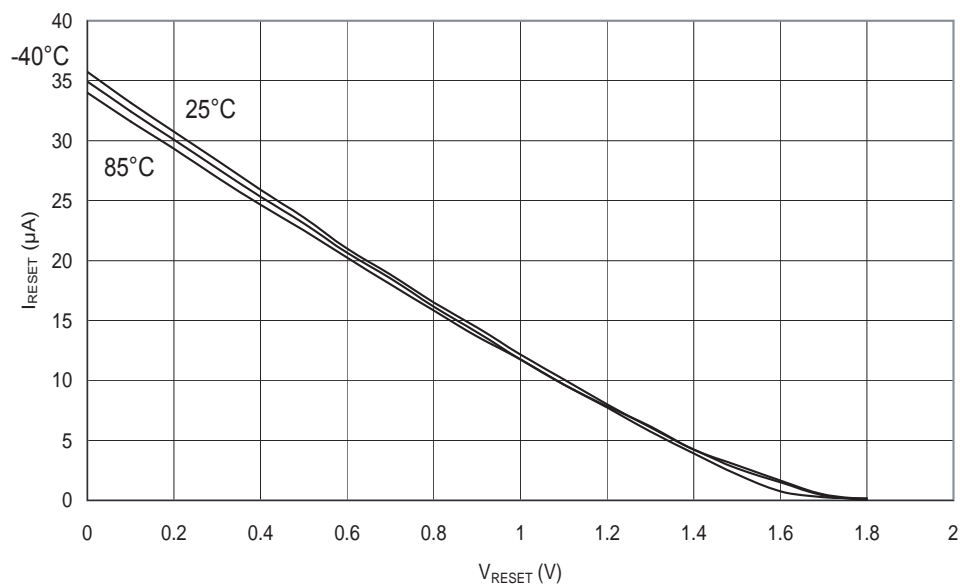
1. A: Load Command "0100 0000".
2. C: Load Data Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
3. Set BS1 to "1" and BS2 to "0". This selects high fuse byte.
4. Give \overline{WR} a negative pulse and wait for RDY/ \overline{BSY} to go high.
5. Set BS1 to "0". This selects low data byte.

27.7.10 Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (refer to ["Programming the Flash" on page 302](#) for details on Command and Data loading):

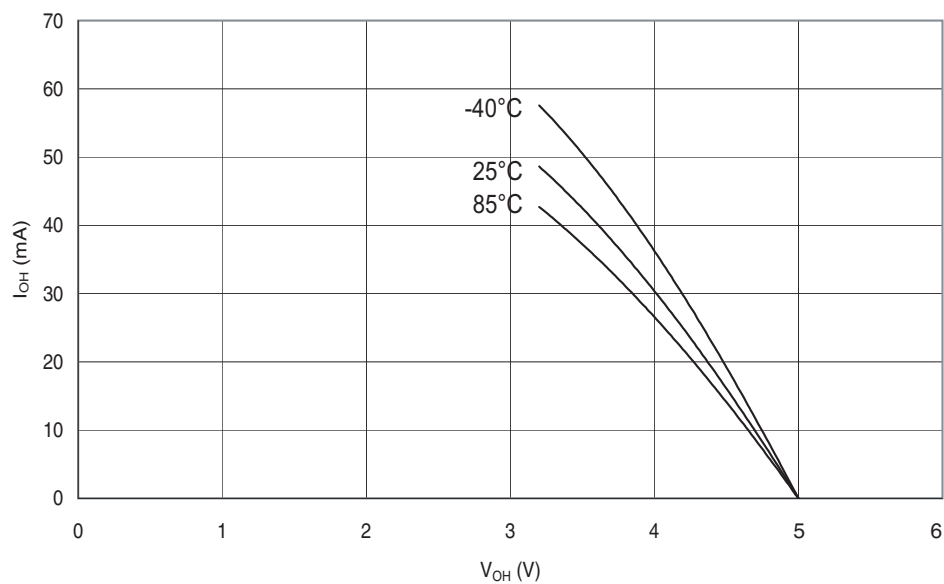
1. 1. A: Load Command "0100 0000".
2. 2. C: Load Data Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
3. 3. Set BS1 to "0" and BS2 to "1". This selects extended fuse byte.
4. 4. Give \overline{WR} a negative pulse and wait for RDY/ \overline{BSY} to go high.
5. 5. Set BS2 to "0". This selects low data byte.

Figure 29-28. Reset Pull-up Resistor Current vs. Reset Pin Voltage ($V_{CC} = 1.8V$)



29.8 Pin Driver Strength

Figure 29-29. I/O Pin Source Current vs. Output Voltage, Ports A, C, D, E, F, G ($V_{CC} = 5V$)



35. Datasheet Revision History

Please note that the referring page numbers in this section are referring to this document. The referring revision in this section are referring to the document revision.

35.1 Rev. 8018P 08/10

1. Status changed to active
2. EEPROM minimum wait delay, [Table 27-15 on page 312](#), has been changed from 9.0 ms to 3.6 ms
3. Datasheet layout and technical terminology updated

35.2 Rev. 8018O 10/09

1. Changed datasheet status to "Mature"
2. Added Capacitance for Low-frequency Crystal Oscillator, [Table 8-5 on page 33](#).

35.3 Rev. 8018N 08/09

1. Updated ["Ordering Information" on page 380](#), MCU replaced by MCH.

35.4 Rev. 8018M 07/09

1. Updated the last page with new Atmel's addresses.

35.5 Rev. L 08/08

1. Updated package information in ["Features" on page 1](#).
2. Added ["Pinout - DRQFN" on page 3](#):
 - The Staggered QFN is named Dual Row QFN (DRQFN).

35.6 Rev. K 06/08

1. Updated package information in ["Features" on page 1](#).
2. Removed "Disclaimer" from section ["Pin Configurations" on page 2](#)
3. Added ["64MC \(DRQFN\) Pinout ATmega169P" on page 3](#)
4. Added ["Data Retention" on page 9](#).
5. Updated ["Stack Pointer" on page 13](#).
6. Updated ["Low-frequency Crystal Oscillator" on page 34](#).
7. Updated ["USART Register Description" on page 194](#), register descriptions and tables.
8. Updated ["UCSRnB – USART Control and Status Register n B" on page 195](#).
9. Updated V_{IL2} in ["DC Characteristics" on page 329](#), by removing $0.2V_{CC}$ from the table.

17.9	Timer/Counter Prescaler	152
17.10	8-bit Timer/Counter Register Description	153
18	<i>SPI – Serial Peripheral Interface</i>	158
18.1	Features	158
18.2	Overview	158
18.3	\overline{SS} Pin Functionality	163
18.4	Data Modes	164
18.5	Register Description	165
19	<i>USART</i>	168
19.1	Features	168
19.2	Overview	168
19.3	Clock Generation	170
19.4	Frame Formats	173
19.5	USART Initialization	175
19.6	Data Transmission – The USART Transmitter	177
19.7	Data Reception – The USART Receiver	180
19.8	Asynchronous Data Reception	185
19.9	Multi-processor Communication Mode	188
19.10	Examples of Baud Rate Setting	190
19.11	USART Register Description	194
20	<i>USI – Universal Serial Interface</i>	199
20.1	Overview	199
20.2	Functional Descriptions	200
20.3	Alternative USI Usage	206
20.4	USI Register Descriptions	207
21	<i>AC - Analog Comparator</i>	212
21.1	Analog Comparator Multiplexed Input	213
21.2	Analog Comparator Register Description	214
22	<i>ADC - Analog to Digital Converter</i>	216
22.1	Features	216
22.2	Overview	216
22.3	Operation	217
22.4	Starting a Conversion	218
22.5	Prescaling and Conversion Timing	219