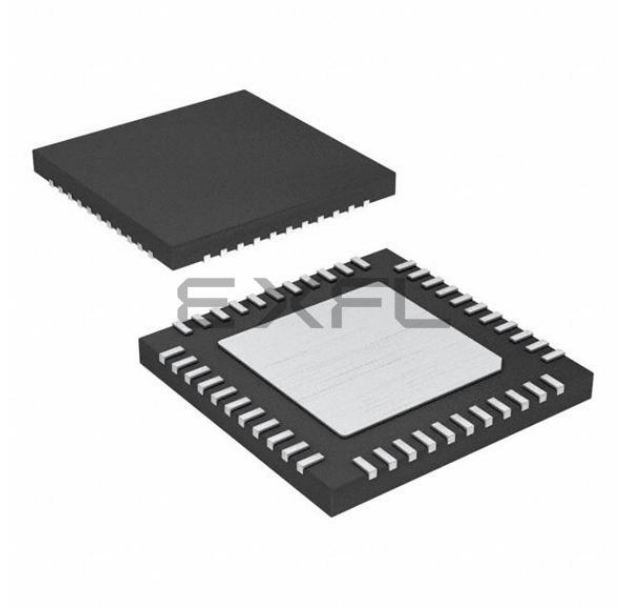


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

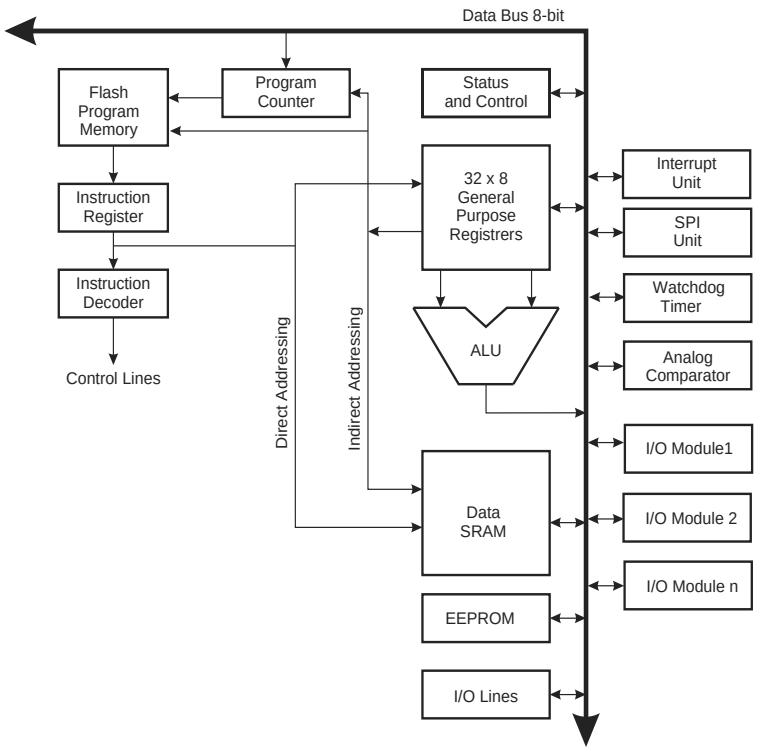
Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	32
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VFQFN Exposed Pad
Supplier Device Package	44-VQFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega16a-mu

6. AVR CPU

6.1 Overview

This section discusses the Atmel AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

Figure 6-1. Block Diagram of the AVR MCU Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash Program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. A Stack PUSH command will decrease the Stack Pointer.

The Stack in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. Initial Stack Pointer value equals the last address of the internal SRAM and the Stack Pointer must be set to point above start of the SRAM, see Figure 7-2 on page 17.

See Table 6-1 for Stack Pointer details.

Table 6-1. Stack Pointer instructions

Instruction	Stack pointer	Description
PUSH	Decremented by 1	Data is pushed onto the stack
CALL ICALL RCALL	Decremented by 2	Return address is pushed onto the stack with a subroutine call or interrupt
POP	Incremented by 1	Data is popped from the stack
RET RETI	Incremented by 2	Return address is popped from the stack with return from subroutine or return from interrupt

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

6.5.1 SPH and SPL – Stack Pointer High and Low Register

Bit	15	14	13	12	11	10	9	8	
	SP15	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
	SP7	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	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	

6.6 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 6-4 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.

When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 8-10.

Table 8-10. Start-up Times for the External Clock Selection

SUT1:0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	–	BOD enabled
01	6 CK	4.1ms	Fast rising power
10	6 CK	65ms	Slowly rising power
11	Reserved		

When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the MCU. A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. It is required to ensure that the MCU is kept in reset during such changes in the clock frequency.

8.9 Timer/Counter Oscillator

For microcontrollers with Timer/Counter Oscillator pins (TOSC1 and TOSC2), the crystal is connected directly between the pins. No external capacitors are needed. The Oscillator is optimized for use with a 32.768kHz watch crystal. Applying an external clock source to TOSC1 is not recommended.

Note: The Timer/Counter Oscillator uses the same type of crystal oscillator as Low-Frequency Oscillator and the internal capacitors have the same nominal value of 36pF.

8.10 Register Description

8.10.1 OSCCAL – Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

- **Bits 7:0 – CAL7:0: Oscillator Calibration Value**

Writing the calibration byte to this address will trim the Internal Oscillator to remove process variations from the Oscillator frequency. This is done automatically during Chip Reset. When OSCCAL is zero, the lowest available frequency is chosen. Writing non-zero values to this register will increase the frequency of the Internal Oscillator. Writing $FF to the register gives the highest available frequency. The calibrated Oscillator is used to time EEPROM and Flash access. If EEPROM or Flash is written, do not calibrate to more than 10% above the nominal frequency. Otherwise, the EEPROM or Flash write may fail. Note that the Oscillator is intended for calibration to 1.0, 2.0, 4.0, or 8.0MHz. Tuning to other values is not guaranteed, as indicated in Table 8-11.

Table 8-11. Internal RC Oscillator Frequency Range.

OSCCAL Value	Min Frequency in Percentage of Nominal Frequency (%)	Max Frequency in Percentage of Nominal Frequency (%)
$00	50	100
$7F	75	150
$FF	100	200

clear	Clear TCNT2 (set all bits to zero).
clk_{T2}	Timer/Counter clock.
top	Signalizes that TCNT2 has reached maximum value.
bottom	Signalizes that TCNT2 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T2}). clk_{T2} can be generated from an external or internal clock source, selected by the Clock Select bits (CS22:0). When no clock source is selected (CS22:0 = 0) the timer is stopped. However, the TCNT2 value can be accessed by the CPU, regardless of whether clk_{T2} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

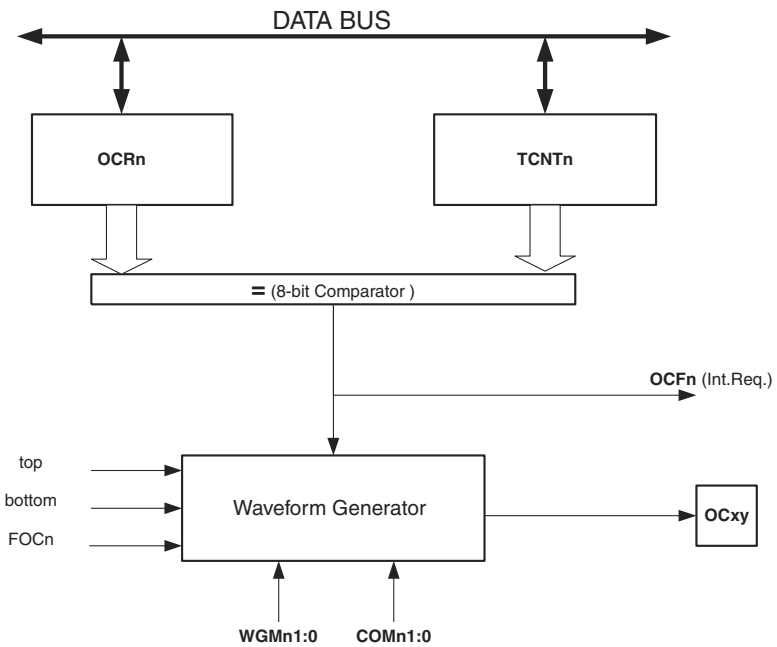
The counting sequence is determined by the setting of the WGM21 and WGM20 bits located in the Timer/Counter Control Register (TCCR2). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare output OC2. For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 116.

The Timer/Counter Overflow (TOV2) Flag is set according to the mode of operation selected by the WGM21:0 bits. TOV2 can be used for generating a CPU interrupt.

17.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT2 with the Output Compare Register (OCR2). Whenever TCNT2 equals OCR2, the comparator signals a match. A match will set the Output Compare Flag (OCF2) at the next timer clock cycle. If enabled (OCIE2 = 1), the Output Compare Flag generates an output compare interrupt. The OCF2 Flag is automatically cleared when the interrupt is executed. Alternatively, the OCF2 Flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM21:0 bits and Compare Output mode (COM21:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (“Modes of Operation” on page 116). Figure 17-3 shows a block diagram of the output compare unit.

Figure 17-3. Output Compare Unit, Block Diagram



the time before re-entering Power-save or Extended Standby mode is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:

1. Write a value to TCCR2, TCNT2, or OCR2.
2. Wait until the corresponding Update Busy Flag in ASSR returns to zero.
3. Enter Power-save or Extended Standby mode.

- When the asynchronous operation is selected, the 32.768 kHz Oscillator for Timer/Counter2 is always running, except in Power-down and Standby modes. After a Power-up Reset or wake-up from Power-down or Standby mode, the user should be aware of the fact that this Oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter2 after power-up or wake-up from Power-down or Standby mode. The contents of all Timer/Counter2 Registers must be considered lost after a wake-up from Power-down or Standby mode due to unstable clock signal upon start-up, no matter whether the Oscillator is in use or a clock signal is applied to the TOSC1 pin.
- Description of wake up from Power-save or Extended Standby mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.
- Reading of the TCNT2 Register shortly after wake-up from Power-save may give an incorrect result. Since TCNT2 is clocked on the asynchronous TOSC clock, reading TCNT2 must be done through a register synchronized to the internal I/O clock domain. Synchronization takes place for every rising TOSC1 edge. When waking up from Power-save mode, and the I/O clock ($clk_{I/O}$) again becomes active, TCNT2 will read as the previous value (before entering sleep) until the next rising TOSC1 edge. The phase of the TOSC clock after waking up from Power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading TCNT2 is thus as follows:

1. Write any value to either of the registers OCR2 or TCCR2.
2. Wait for the corresponding Update Busy Flag to be cleared.
3. Read TCNT2.

- During asynchronous operation, the synchronization of the Interrupt Flags for the asynchronous timer takes three processor cycles plus one timer cycle. The timer is therefore advanced by at least one before the processor can read the timer value causing the setting of the Interrupt Flag. The output compare pin is changed on the timer clock and is not synchronized to the processor clock.

18.3.2 Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin.

If $\overline{SS}$ is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the $\overline{SS}$ pin of the SPI Slave.

If $\overline{SS}$ is configured as an input, it must be held high to ensure Master SPI operation. If the $\overline{SS}$ pin is driven low by peripheral circuitry when the SPI is configured as a Master with the $\overline{SS}$ pin defined as an input, the SPI system interprets this as another Master selecting the SPI as a Slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The SPIF Flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that $\overline{SS}$ is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a Slave Select, it must be set by the user to re-enable SPI Master mode.

18.4 Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 18-2 and Figure 18-3. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing Table 18-3 and Table 18-4, as done below:

Table 18-2. CPOL and CPHA Functionality

	Leading Edge	Trailing Edge	SPI Mode
CPOL = 0, CPHA = 0	Sample (Rising)	Setup (Falling)	0
CPOL = 0, CPHA = 1	Setup (Rising)	Sample (Falling)	1
CPOL = 1, CPHA = 0	Sample (Falling)	Setup (Rising)	2
CPOL = 1, CPHA = 1	Setup (Falling)	Sample (Rising)	3

Figure 18-2. SPI Transfer Format with CPHA = 0

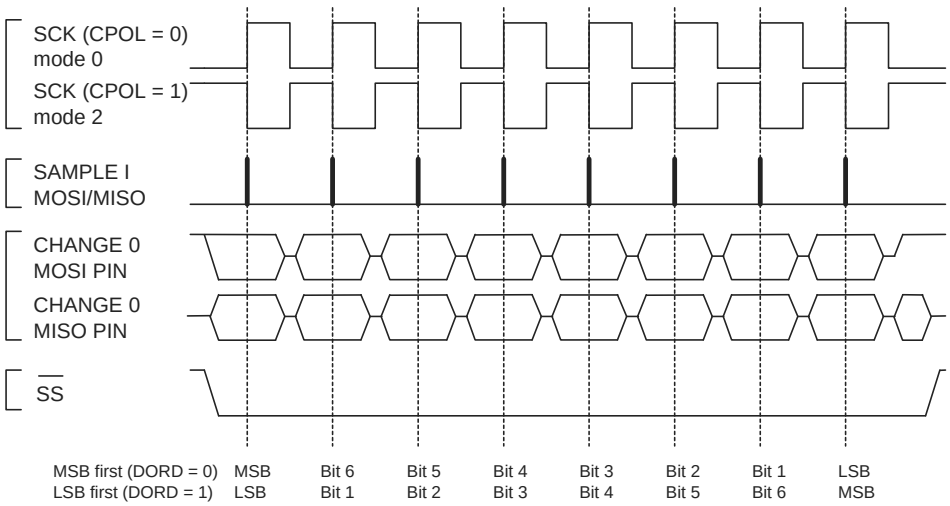


Table 18-5. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

18.5.2 SPSR – SPI Status Register

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag**

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

- **Bit 6 – WCOL: Write COLlision Flag**

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

- **Bit 5:1 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16A and will always read as zero.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 18-5). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{osc}/4$ or lower.

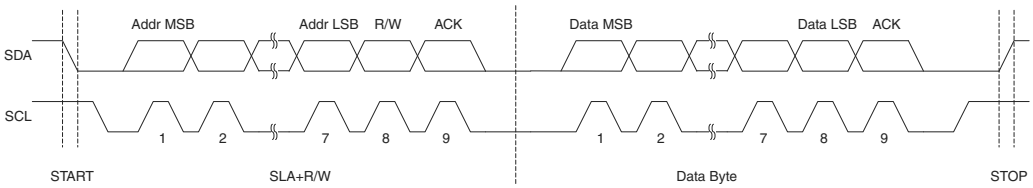
The SPI interface on the ATmega16A is also used for program memory and EEPROM downloading or uploading. See page 262 for SPI Serial Programming and Verification.

20.3.5 Combining Address and Data Packets into a Transmission

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

Figure 20-6 shows a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP condition, depending on the software protocol implemented by the application software.

Figure 20-6. Typical Data Transmission



20.4 Multi-master Bus Systems, Arbitration and Synchronization

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

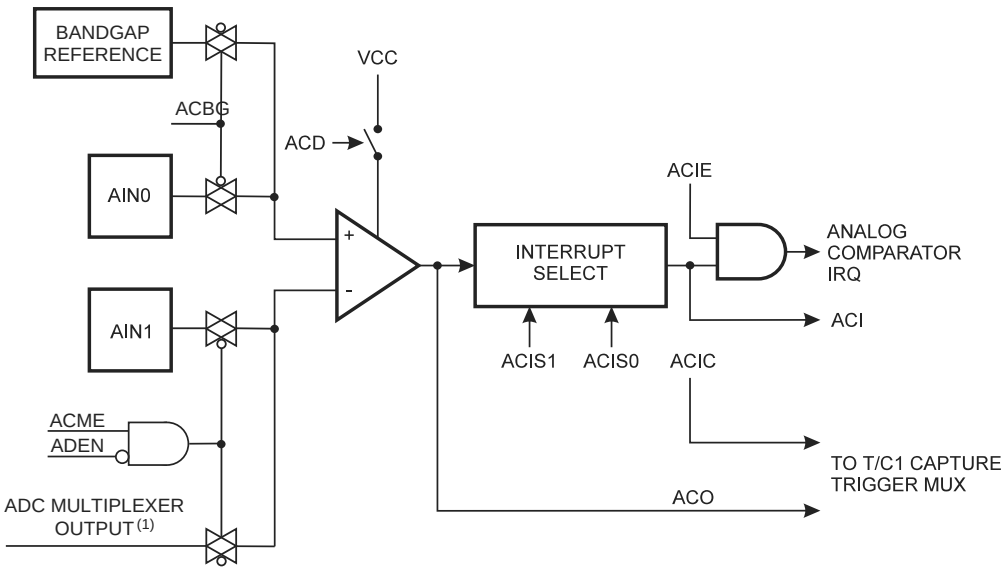
- An algorithm must be implemented allowing only one of the Masters to complete the transmission. All other Masters should cease transmission when they discover that they have lost the selection process. This selection process is called arbitration. When a contending Master discovers that it has lost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

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

21. Analog Comparator

The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator Output, ACO, is set. The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 21-1.

Figure 21-1. Analog Comparator Block Diagram(2)



Notes: 1. See Table 1 on page 193.
2. Refer to Figure 1-1 on page 3 and Table 12-6 on page 57 for Analog Comparator pin placement.

21.1 Analog Comparator Multiplexed Input

It is possible to select any of the 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 and the ADC is switched off (ADEN in ADCSRA is zero), MUX2:0 in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 1. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the Analog Comparator.

Table 1. 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

state 0x01 is shifted out on the TDO pin. The JTAG Instruction selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.

- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the Shift Register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine.
- At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift Data Register – Shift-DR state. While in this state, upload the selected Data Register (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI input at the rising edge of TCK. In order to remain in the Shift-DR state, the TMS input must be held low during input of all bits except the MSB. The MSB of the data is shifted in when this state is left by setting TMS high. While the Data Register is shifted in from the TDI pin, the parallel inputs to the Data Register captured in the Capture-DR state is shifted out on the TDO pin.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine.

As shown in the state diagram, the Run-Test/Idle state need not be entered between selecting JTAG instruction and using Data Registers, and some JTAG instructions may select certain functions to be performed in the Run-Test/Idle, making it unsuitable as an Idle state.

Note: Independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS high for five TCK clock periods.

For detailed information on the JTAG specification, refer to the literature listed in “Bibliography” on page 218.

23.5 Using the Boundary-scan Chain

A complete description of the Boundary-scan capabilities are given in the section “IEEE 1149.1 (JTAG) Boundary-scan” on page 219.

23.6 Using the On-chip Debug System

As shown in Figure 23-1, the hardware support for On-chip Debugging consists mainly of:

- A scan chain on the interface between the internal AVR CPU and the internal peripheral units
- Break Point unit
- Communication interface between the CPU and JTAG system

All read or modify/write operations needed for implementing the Debugger are done by applying AVR instructions via the internal AVR CPU Scan Chain. The CPU sends the result to an I/O memory mapped location which is part of the communication interface between the CPU and the JTAG system.

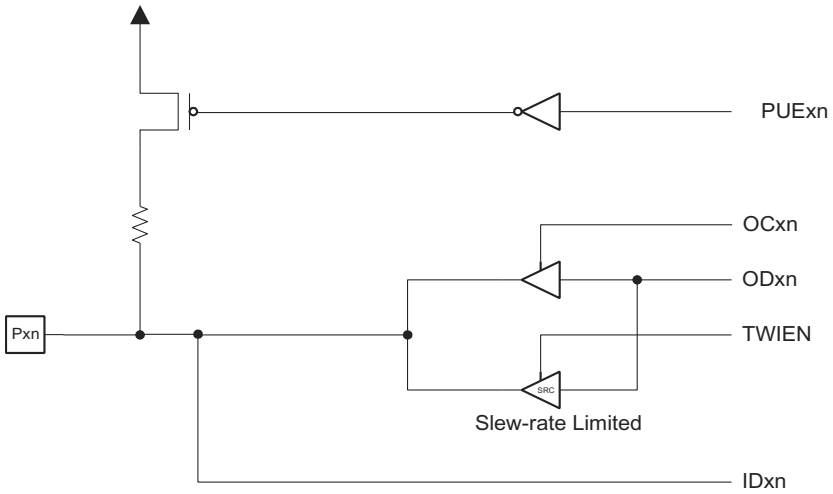
The Break Point Unit implements Break on Change of Program Flow, Single Step Break, 2 Program Memory Break Points, and 2 combined Break Points. Together, the 4 Break Points can be configured as either:

- 4 single Program Memory Break Points
- 3 Single Program Memory Break Point + 1 single Data Memory Break Point
- 2 single Program Memory Break Points + 2 single Data Memory Break Points
- 2 single Program Memory Break Points + 1 Program Memory Break Point with mask (“range Break Point”)
- 2 single Program Memory Break Points + 1 Data Memory Break Point with mask (“range Break Point”)

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 217.

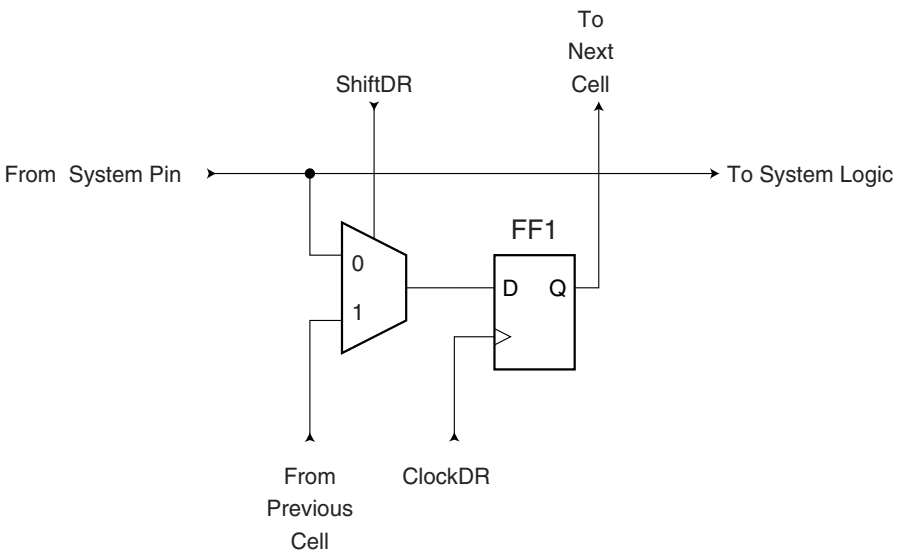
Figure 24-5. Additional Scan Signal for the Two-wire Interface



24.5.3 Scanning the RESET Pin

The RESET pin accepts 5V active low logic for standard reset operation, and 12V active high logic for High Voltage Parallel Programming. An observe-only cell as shown in Figure 24-6 is inserted both for the 5V reset signal; RSTT, and the 12V reset signal; RSTHV.

Figure 24-6. Observe-only Cell



24.5.4 Scanning the Clock Pins

The AVR devices have many clock options selectable by fuses. These are: Internal RC Oscillator, External RC, External Clock, (High Frequency) Crystal Oscillator, Low Frequency Crystal Oscillator, and Ceramic Resonator.

Figure 24-7 shows how each Oscillator with external connection is supported in the scan chain. The Enable signal is supported with a general boundary-scan cell, while the Oscillator/Clock output is attached to an observe-only cell. In addition to the main clock, the Timer Oscillator is scanned in the same way. The output from the internal RC Oscillator is not scanned, as this Oscillator does not have external connections.

As an example, consider the task of verifying a 1.5V ± 5% input signal at ADC channel 3 when the power supply is 5.0V and AREF is externally connected to V_{CC}.

The lower limit is: $\lceil 1024 \cdot 1{,}5V \cdot 0{,}95/5V \rceil = 291 = 0x123$
The upper limit is: $\lceil 1024 \cdot 1{,}5V \cdot 1{,}05/5V \rceil = 323 = 0x143$

The recommended values from Table 24-5 are used unless other values are given in the algorithm in Table 24-6. Only the DAC and Port Pin values of the Scan-chain are shown. The column “Actions” describes what JTAG instruction to be used before filling the Boundary-scan Register with the succeeding columns. The verification should be done on the data scanned out when scanning in the data on the same row in the table.

Table 24-6. Algorithm for Using the ADC

Step	Actions	ADCEN	DAC	MUXEN	HOLD	PRECH	PA3. Data	PA3. Control	PA3. Pullup_ Enable
1	SAMPLE_ PRELOAD	1	0x200	0x08	1	1	0	0	0
2	EXTEST	1	0x200	0x08	0	1	0	0	0
3		1	0x200	0x08	1	1	0	0	0
4		1	0x123	0x08	1	1	0	0	0
5		1	0x123	0x08	1	0	0	0	0
6	Verify the COMP bit scanned out to be 0	1	0x200	0x08	1	1	0	0	0
7		1	0x200	0x08	0	1	0	0	0
8		1	0x200	0x08	1	1	0	0	0
9		1	0x143	0x08	1	1	0	0	0
10		1	0x143	0x08	1	0	0	0	0
11	Verify the COMP bit scanned out to be 1	1	0x200	0x08	1	1	0	0	0

Using this algorithm, the timing constraint on the HOLD signal constrains the TCK clock frequency. As the algorithm keeps HOLD high for five steps, the TCK clock frequency has to be at least five times the number of scan bits divided by the maximum hold time, $t_{hold,max}$.

24.6 Boundary-scan Order

Table 24-7 shows the scan order between TDI and TDO when the Boundary-scan chain is selected as data path. Bit 0 is the LSB; the first bit scanned in, and the first bit scanned out. The scan order follows the pin-out order as far as possible. Therefore, the bits of Port A is scanned in the opposite bit order of the other ports. Exceptions from the rules are the Scan chains for the analog circuits, which constitute the most significant bits of the scan chain regardless of which physical pin they are connected to. In Figure 24-3, PXn. Data corresponds to

25.8.10 Preventing Flash Corruption

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

A Flash program corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the Flash 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.

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

1. If there is no need for a Boot Loader update in the system, program the Boot Loader Lock bits to prevent any Boot Loader software updates.
2. 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 voltage is sufficient.
3. 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 SPMCR Register and thus the Flash from unintentional writes.

25.8.11 Programming Time for Flash when using SPM

The Calibrated RC Oscillator is used to time Flash accesses. Table 25-5 shows the typical programming time for Flash accesses from the CPU.

Table 25-5. SPM Programming Time.

Symbol	Min Programming Time	Max Programming Time
Flash write (Page Erase, Page Write, and write Lock bits by SPM)	3.7 ms	4.5 ms

25.8.12 Simple Assembly Code Example for a Boot Loader

```
            ;-the routine writes one page of data from RAM to Flash
            ; the first data location in RAM is pointed to by the Y pointer
            ; the first data location in Flash is pointed to by the Z pointer
            ;-error handling is not included
            ;-the routine must be placed inside the boot space
            ; (at least the Do_spm sub routine). Only code inside NRWW section can
            ; be read during self-programming (page erase and page write).
            ;-registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
            ; loophi (r25), spmcrval (r20)
            ; storing and restoring of registers is not included in the routine
            ; register usage can be optimized at the expense of code size
            ;-It is assumed that either the interrupt table is moved to the Boot
            ; loader section or that the interrupts are disabled.
.equ                  PAGESIZEB = PAGESIZE*2    ; PAGESIZEB is page size in
                                                BYTES, not
                                                ; words
.org SMALLBOOTSTART
Write_page:
            ; page erase
            ldi   spmcrval, (1<<PGERS) | (1<<SPMEN)
            call  Do_spm

            ; re-enable the RWW section
```

Table 26-2. Lock Bit Protection Modes (Continued)

Memory Lock Bits[2]			Protection Type
BLB1 Mode	**BLB12**	**BLB11**	
1	1	1	No restrictions for SPM or LPM accessing the Boot Loader section.
2	1	0	SPM is not allowed to write to the Boot Loader section.
3	0	0	SPM is not allowed to write to the Boot Loader section, and LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.
4	0	1	LPM executing from the Application section is not allowed to read from the Boot Loader section. If interrupt vectors are placed in the Application section, interrupts are disabled while executing from the Boot Loader section.

Notes: 1. Program the Fuse bits before programming the Lock bits.
2. "1" means unprogrammed, "0" means programmed

26.2 Fuse Bits

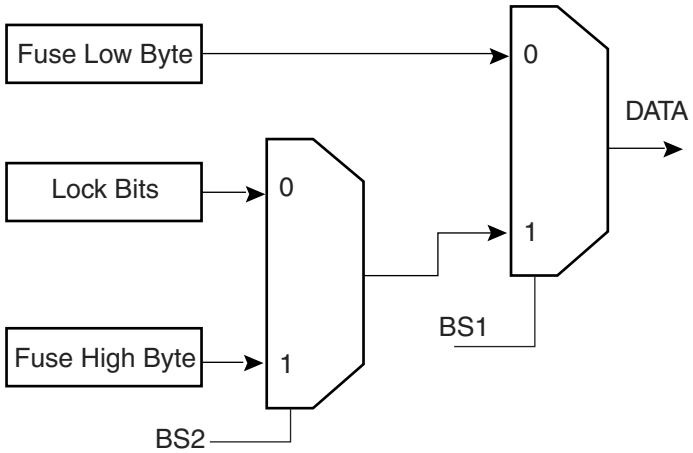
The ATmega16A has two fuse bytes. Table 26-3 and Table 26-4 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the fuses are read as logical zero, "0", if they are programmed.

Table 26-3. Fuse High Byte

Fuse High Byte	Bit No.	Description	Default Value
OCDEN[4]	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN[5]	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN[1]	5	Enable SPI Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT[2]	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 25-6 for details)	0 (programmed)[3]
BOOTSZ0	1	Select Boot Size (see Table 25-6 for details)	0 (programmed)[3]
BOOTRST	0	Select reset vector	1 (unprogrammed)

Notes: 1. The SPIEN Fuse is not accessible in SPI Serial Programming mode.
2. The CKOPT Fuse functionality depends on the setting of the CKSEL bits. See See "Clock Sources" on page 25. for details.
3. The default value of BOOTSZ1:0 results in maximum Boot Size. See Table 25-6 on page 249.
4. Never ship a product with the OCDEN Fuse programmed regardless of the setting of Lock bits and the JTAGEN Fuse. A programmed OCDEN Fuse enables some parts of the clock system to be running in all sleep modes. This may increase the power consumption.
5. If the JTAG interface is left unconnected, the JTAGEN fuse should if possible be disabled. This to avoid static current at the TDO pin in the JTAG interface.

Figure 26-6. Mapping between BS1, BS2 and the Fuse- and Lock Bits during Read



26.7.12 Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to “Programming the Flash” on page 257 for details on Command and Address loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte ($00 - $02).
3. Set $\overline{OE}$ to “0”, and BS1 to “0”. The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

26.7.13 Reading the Calibration Byte

The algorithm for reading the Calibration byte is as follows (refer to “Programming the Flash” on page 257 for details on Command and Address loading):

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

26.8 Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{RESET}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input), and MISO (output). After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed. NOTE, in Table 26-11 on page 262, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

26.8.1 SPI Serial Programming Pin Mapping

Table 26-11. Pin Mapping SPI Serial Programming

Symbol	Pins	I/O	Description
MOSI	PB5	I	Serial Data in
MISO	PB6	O	Serial Data out
SCK	PB7	I	Serial Clock

After data is loaded to the page buffer, program the EEPROM page, see Figure 26-8 on page 266.

Figure 26-8. Serial Programming Instruction example

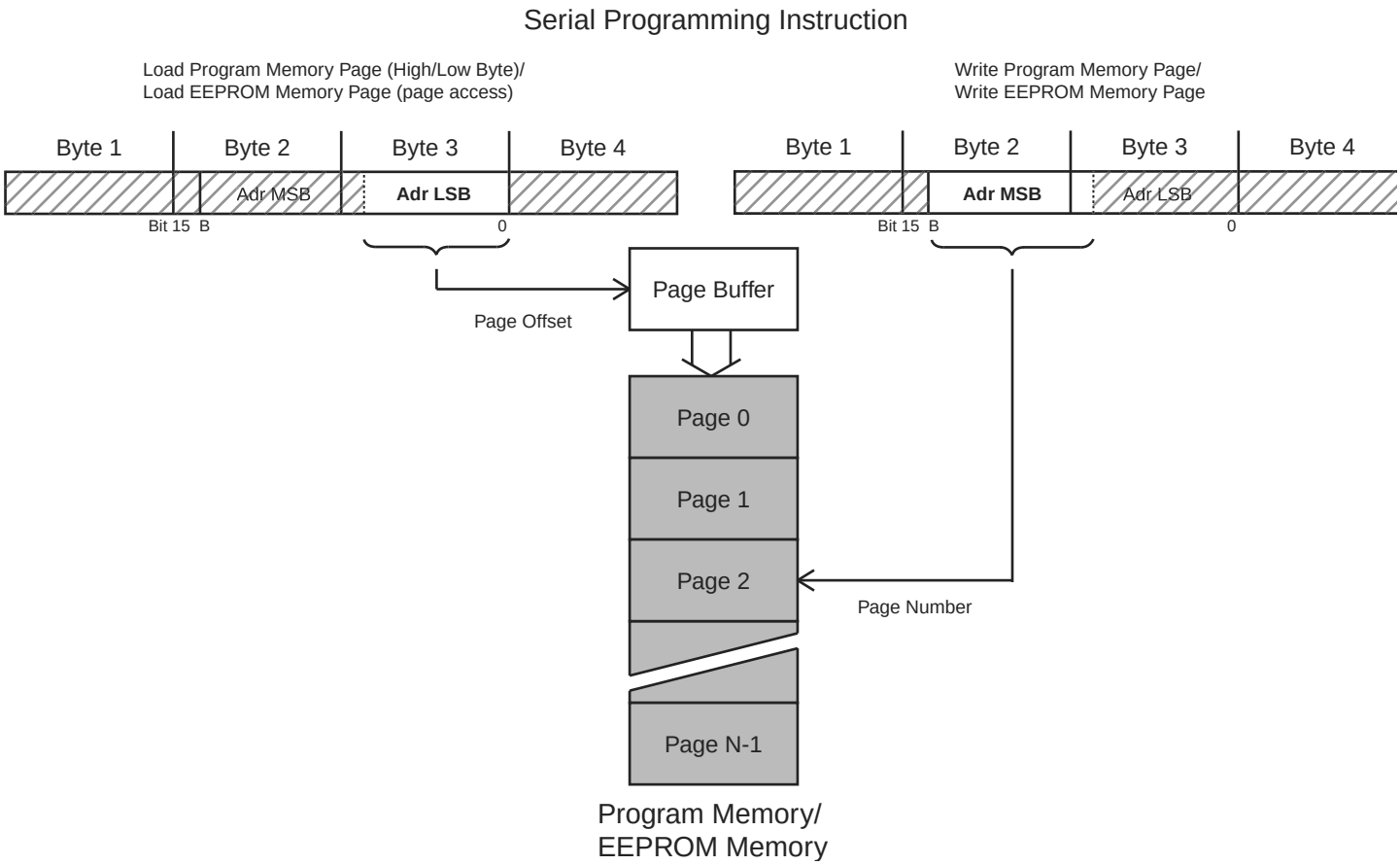
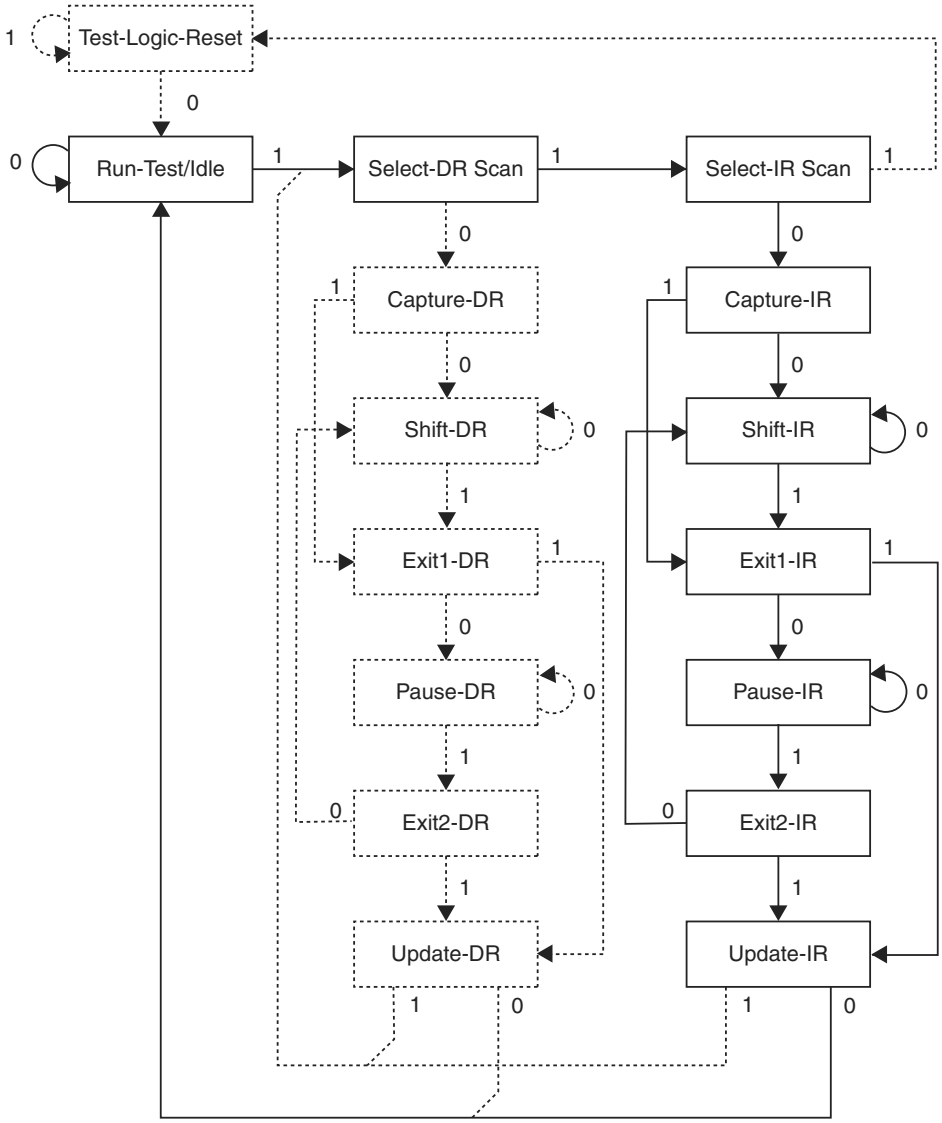


Figure 26-9. State Machine Sequence for Changing the Instruction Word



26.10.2 AVR_RESET ($C)

The AVR specific public JTAG instruction for setting the AVR device in the Reset mode or taking the device out from the Reset Mode. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the Reset will be active as long as there is a logic “one” in the Reset Chain. The output from this chain is not latched.

The active states are:

- Shift-DR: The Reset Register is shifted by the TCK input.

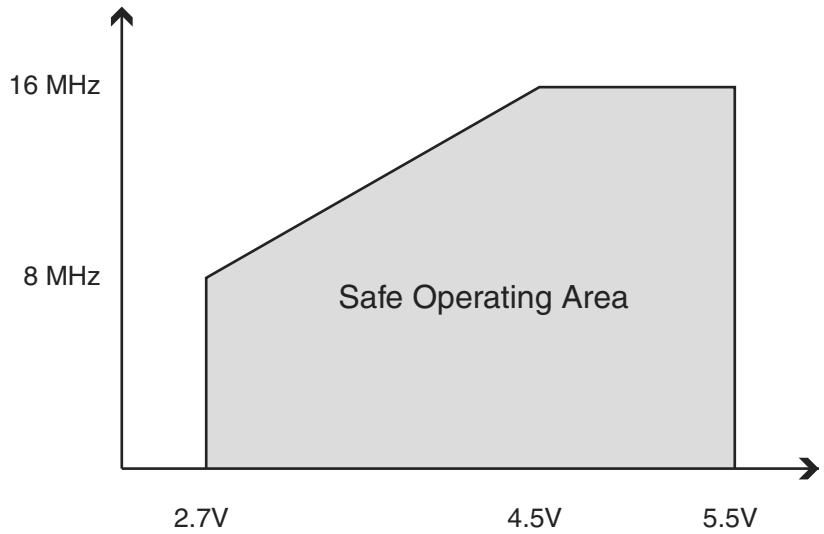
26.10.3 PROG_ENABLE ($4)

The AVR specific public JTAG instruction for enabling programming via the JTAG port. The 16-bit Programming Enable Register is selected as Data Register. The active states are the following:

- Shift-DR: The programming enable signature is shifted into the Data Register.
- Update-DR: The programming enable signature is compared to the correct value, and Programming mode is entered if the signature is valid.

27.3 Speed Grades

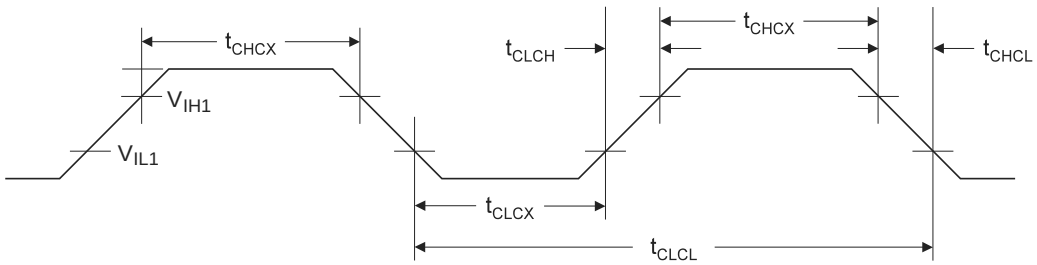
Figure 27-1. Maximum Frequency vs V_{CC}.



27.4 Clock Characteristics

27.4.1 External Clock Drive Waveforms

Figure 27-2. External Clock Drive Waveforms



27.4.2 External Clock Drive

Figure 27-3. External Clock Drive(1)

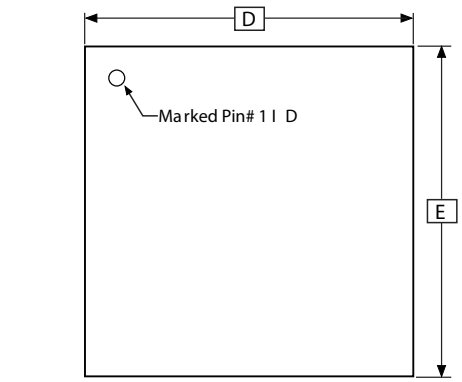
Symbol	Parameter	V_{CC} = 2.7V to 5.5V		V_{CC} = 4.5V to 5.5V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	8	0	16	MHz
t_{CLCL}	Clock Period	125		62.5		ns
t_{CHCX}	High Time	50		25		ns
t_{CLCX}	Low Time	50		25		ns

27.9 ADC Characteristics

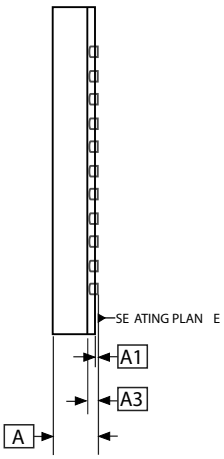
Table 27-6. ADC Characteristics

Symbol	Parameter	Condition	Min[1]	Typ[1]	Max[1]	Units
	Resolution	Single Ended Conversion		10		Bits
		Differential Conversion Gain = 1x or 10x		8		Bits
		Differential Conversion Gain = 200x		7		Bits
	Absolute Accuracy (Including INL, DNL, Quantization Error, Gain, and Offset Error).	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		1.5	2.5	LSB
		Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 1MHz		3	4	LSB
		Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz Noise Reduction mode		1.5		LSB
		Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 1MHz Noise Reduction mode		3		LSB
	Integral Non-linearity (INL)	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		1		LSB
	Differential Non-linearity (DNL)	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		0.5		LSB
	Gain Error	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		1		LSB
	Offset Error	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz				LSB
	Conversion Time	Free Running Conversion	13		260	µs
	Clock Frequency		50		1000	kHz
AVCC	Analog Supply Voltage		V_{CC} - 0.3[2]		V_{CC} + 0.3[3]	V
V_{REF}	Reference Voltage	Single Ended Conversion	2.0		AVCC	V
		Differential Conversion	2.0		AVCC - 0.2	V
V_{IN}	Input voltage	Single ended channels	GND		V_{REF}	V
		Differential channels	0		V_{REF}	V
	Input bandwidth	Single ended channels		38.5		kHz
		Differential channels		4		kHz
V_{INT}	Internal Voltage Reference		2.3	2.6	2.9	V
R_{REF}	Reference Input Resistance			32		kΩ
R_{AIN}	Analog Input Resistance			100		MΩ

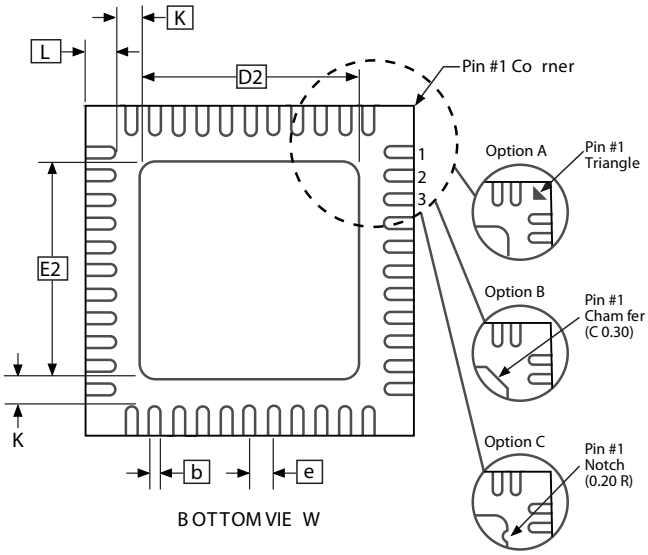
32.3 44M1



TOP VIE W



SIDE VIEW



B OTTOM VIE W

Note: JEDEC Standard MO-220, Fig . 1 (S AW Singulation) VKKD-3 .

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A3		0.20 REF		
b	0.18	0.23	0.30	
D	6.90	7.00	7.10	
D2	5.00	5.20	5.40	
E	6.90	7.00	7.10	
E2	5.00	5.20	5.40	
e		0.50 BSC		
L	0.59	0.64	0.69	
K	0.20	0.26	0.41	

02/13/2014

Atmel Package Drawing Contact: packagedrawings@atmel.com	TITLE 44M1, 44-pad, 7 x 7 x 1.0mm body, lead pitch 0.50mm, 5.20mm exposed pad, thermally enhanced plastic very thin quad flat no lead package (VQFN)	GPC ZWS	DRAWING NO. 44M1	REV. H