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	SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	53
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	4K 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-TQFP
Supplier Device Package	64-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega645-16au

3. Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

4. Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

5. About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

For I/O Registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

6. Capacitive touch sensing

The Atmel®QTouch® Library provides a simple to use solution to realize touch sensitive interfaces on most Atmel AVR® microcontrollers. The QTouch Library includes support for the QTouch and QMatrix® acquisition methods.

Touch sensing can be added to any application by linking the appropriate Atmel QTouch Library for the AVR Microcontroller. This is done by using a simple set of APIs to define the touch channels and sensors, and then calling the touch sensing API’s to retrieve the channel information and determine the touch sensor states.

The QTouch Library is FREE and downloadable from the Atmel website at the following location: www.atmel.com/qtouchlibrary. For implementation details and other information, refer to the Atmel QTouch Library User Guide - also available for download from the Atmel website.

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.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F. In addition, the Atmel ATmega325/3250/645/6450 has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

7.3 ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

7.4 Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

7.4.1 SREG – AVR Status Register

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
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 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry Is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive or between the Negative Flag N and the Two’s Complement Overflow Flag V. See the “Instruction Set Description” for detailed information.

- **Bit 3 – V: Two’s Complement Overflow Flag**

The Two’s Complement Overflow Flag V supports two’s complement arithmetics. See the “Instruction Set Description” for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

Applying an external clock source to TOSC1 can be done if EXTCLK in the ASSR Register is written to logic one. See “Asynchronous Operation of Timer/Counter2” on page 141 for further description on selecting external clock as input instead of a 32kHz crystal.

9.9 System Clock Prescaler

The Atmel ATmega325/3250/645/6450 system clock can be divided by setting the “CLKPR – Clock Prescale Register” on page 33. This feature can be used to decrease power consumption when the requirement for processing power is low. This can be used with all clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, clk_{ADC}, clk_{CPU}, and clk_{FLASH} are divided by a factor as shown in Table 9-11 on page 33.

9.9.1 Switching Time

When switching between prescaler settings, the System Clock Prescaler ensures that no glitches occur in the clock system and that no intermediate frequency is higher than neither the clock frequency corresponding to the previous setting, nor the clock frequency corresponding to the new setting.

The ripple counter that implements the prescaler runs at the frequency of the undivided clock, which may be faster than the CPU’s clock frequency. Hence, it is not possible to determine the state of the prescaler – even if it were readable, and the exact time it takes to switch from one clock division to another cannot be exactly predicted.

From the time the CLKPS values are written, it takes between T1 + T2 and T1 + 2*T2 before the new clock frequency is active. In this interval, 2 active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

9.10 Register Description

9.10.1 OSCCAL – Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
(0x66)	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**

The Oscillator Calibration Register is used to trim the Calibrated Internal RC Oscillator to remove process variations from the oscillator frequency. A pre-programmed calibration value is automatically written to this register during chip reset, giving the Factory calibrated frequency as specified in Table 28-2 on page 300. The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to frequencies as specified in Table 28-2 on page 300. Calibration outside that range is not guaranteed.

Note that this oscillator is used to time EEPROM and Flash write accesses, and these write times will be affected accordingly. If the EEPROM or Flash are written, do not calibrate to more than 8.8MHz. Otherwise, the EEPROM or Flash write may fail.

The CAL7 bit determines the range of operation for the oscillator. Setting this bit to 0 gives the lowest frequency range, setting this bit to 1 gives the highest frequency range. The two frequency ranges are overlapping, in other words a setting of OSCCAL = 0x7F gives a higher frequency than OSCCAL = 0x80.

When the BOOTRST Fuse is programmed, the Boot section size set to 4K bytes and the IVSEL bit in the MCUCR Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address  Labels Code                  Comments
;
.org 0x3800/0x7800
0x3800/0x7800   jmp    RESET          ; Reset handler
0x3802/0x7802   jmp    EXT_INT0       ; IRQ0 Handler
0x3804/0x7804   jmp    PCINT0         ; PCINT0 Handler
...             ...    ...            ;
0x382C/0x782C   jmp    SPM_RDY    ; Store Program Memory Ready Handler
;
0x382E/0x782ERESET:ldir16,high(RAMEND); Main program start
0x382F/0x782F   out    SPH,r16      ; Set Stack Pointer to top of RAM
0x3830/0x7830   ldi    r16,low(RAMEND)
0x3831/0x7831   out    SPL,r16
0x3832/0x7832   sei                   ; Enable interrupts
0x3833/0x7833   <instr>  xxx
```

12.2 Moving Interrupts Between Application and Boot Space

The MCU Control Register controls the placement of the Interrupt Vector table.

12.3 Register Description

12.3.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	JTD	–	–	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R/W	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1 – IVSEL: Interrupt Vector Select**

When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. Refer to the section “Boot Loader Support – Read-While-Write Self-Programming” on page 251 for details. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

1. Write the Interrupt Vector Change Enable (IVCE) bit to one.
2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the Status Register is unaffected by the automatic disabling.

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed

corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

This bit is reserved bit in ATmega325/645 and will always be read as zero.

- **Bit 6– PCIF2: Pin Change Interrupt Flag 2**

When a logic change on any PCINT24..16 pin triggers an interrupt request, PCIF2 becomes set (one). If the I-bit in SREG and the PCIE2 bit in EIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

This bit is reserved bit in ATmega325/645 and will always be read as zero.

- **Bit 5– PCIF1: Pin Change Interrupt Flag 1**

When a logic change on any PCINT15..8 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in EIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 4– PCIF0: Pin Change Interrupt Flag 0**

When a logic change on any PCINT7..0 pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in EIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 0 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in EIMSK are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

13.2.4 PCMSK3 – Pin Change Mask Register 3[(1)]

Bit	7	6	5	4	3	2	1	0	
(0x73)	–	PCINT30	PCINT29	PCINT28	PCINT27	PCINT26	PCINT25	PCINT24	PCMSK3
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

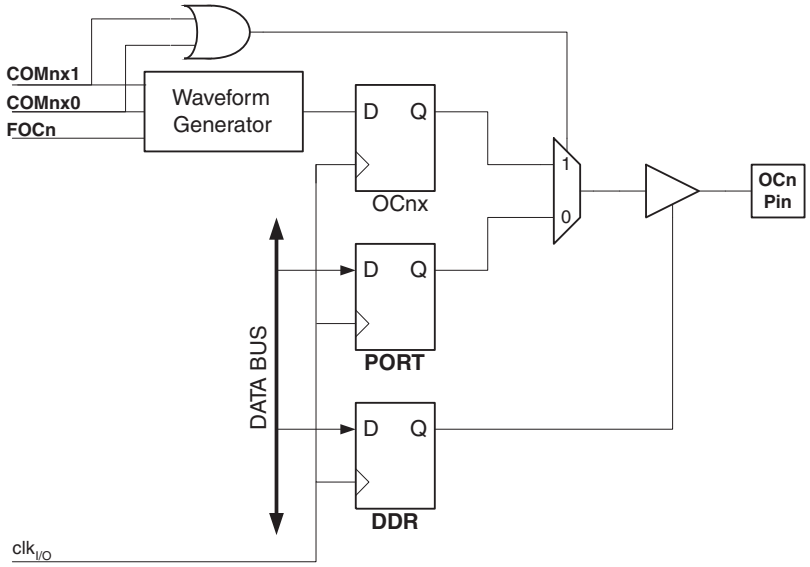
- **Bit 6:0 – PCINT30:24: Pin Change Enable Mask 30:24**

Each PCINT30:24-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT30:24 is set and the PCIE3 bit in EIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT30:24 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

13.2.5 PCMSK2 – Pin Change Mask Register 2[(1)]

Bit	7	6	5	4	3	2	1	0	
(0x6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
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	

Figure 15-4. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC0A) from the Waveform Generator if either of the COM0A1:0 bits are set. However, the OC0A pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC0A pin (DDR_OC0A) must be set as output before the OC0A value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the Output Compare pin logic allows initialization of the OC0A state before the output is enabled. Note that some COM0A1:0 bit settings are reserved for certain modes of operation. See “Register Description” on page 96.

15.6.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM0A1:0 bits differently in Normal, CTC, and PWM modes. For all modes, setting the COM0A1:0 = 0 tells the Waveform Generator that no action on the OC0A Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 15-3 on page 97. For fast PWM mode, refer to Table 15-4 on page 97, and for phase correct PWM refer to Table 15-5 on page 98.

A change of the COM0A1:0 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 FOC0A strobe bits.

15.7 Modes of Operation

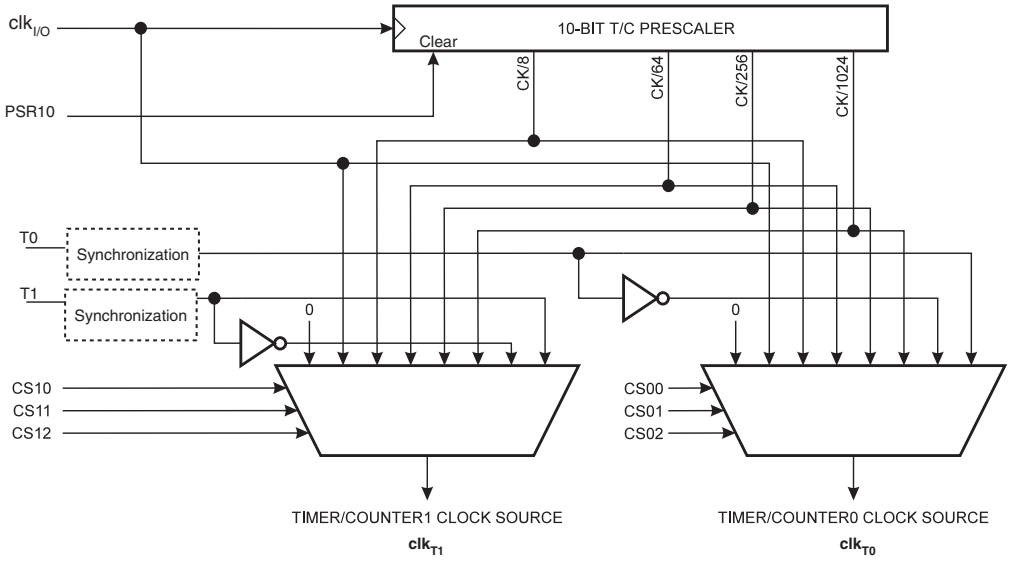
The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM01:0) and Compare Output mode (COM0A1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM0A1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0A1:0 bits control whether the output should be set, cleared, or toggled at a compare match (See “Compare Match Output Unit” on page 89.).

For detailed timing information refer to Figure 15-8, Figure 15-9, Figure 15-10 and Figure 15-11 in “Timer/Counter Timing Diagrams” on page 95.

and duty cycle caused by Oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{clk\_I/O}/2.5$.

An external clock source can not be prescaled.

Figure 16-1. Prescaler for Timer/Counter0 and Timer/Counter1 (1)



Note: 1. The synchronization logic on the input pins (T1/T0) is shown in Figure 1.

16.1 Register Description

16.1.1 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.

The following code example shows a simple USART receive function based on polling of the Receive Complete (RXCn) Flag. When using frames with less than eight bits the most significant bits of the data read from the UDRn will be masked to zero. The USART has to be initialized before the function can be used.

Assembly Code Example[1]

```
USART_Receive:
  ; Wait for data to be received
  sbis UCSR0A, RXC0
  rjmp USART_Receive
  ; Get and return received data from buffer
  in   r16, UDR0
  ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSR0A & (1<<RXC0)) )
       ;
  /* Get and return received data from buffer */
  return UDR0;
}
```

Note: 1. See “About Code Examples” on page 9.

The function simply waits for data to be present in the receive buffer by checking the RXCn Flag, before reading the buffer and returning the value.

20.7.2 Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZ=7) the ninth bit must be read from the RXB8n bit in UCSRnB **before** reading the low bits from the UDRn. This rule applies to the FEn, DORn and UPEn Status Flags as well. Read status from UCSRnA, then data from UDRn. Reading the UDRn I/O location will change the state of the receive buffer FIFO and consequently the TXB8n, FEn, DORn and UPEn bits, which all are stored in the FIFO, will change.

with Auto triggering from a source other than the ADC Conversion Complete, each conversion will require 25 ADC clocks. This is because the ADC must be disabled and re-enabled after every conversion.

In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see Table 23-1.

Figure 23-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)

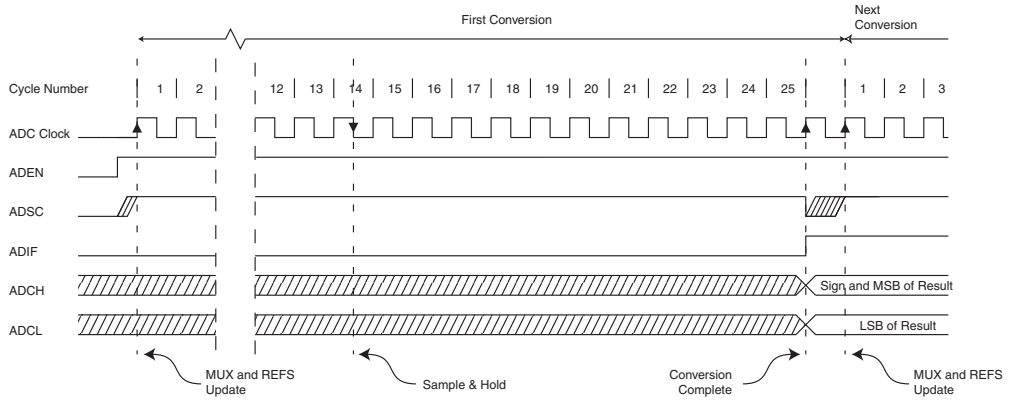


Figure 23-5. ADC Timing Diagram, Single Conversion

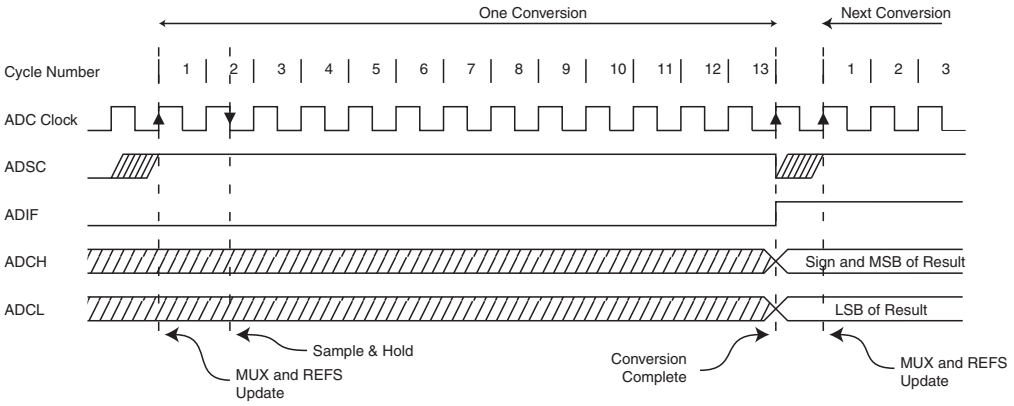


Figure 23-6. ADC Timing Diagram, Auto Triggered Conversion

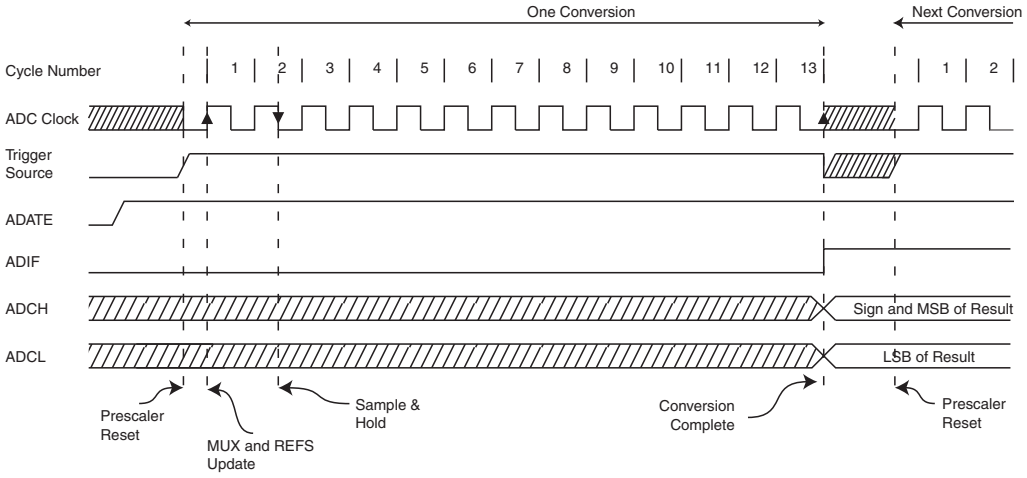
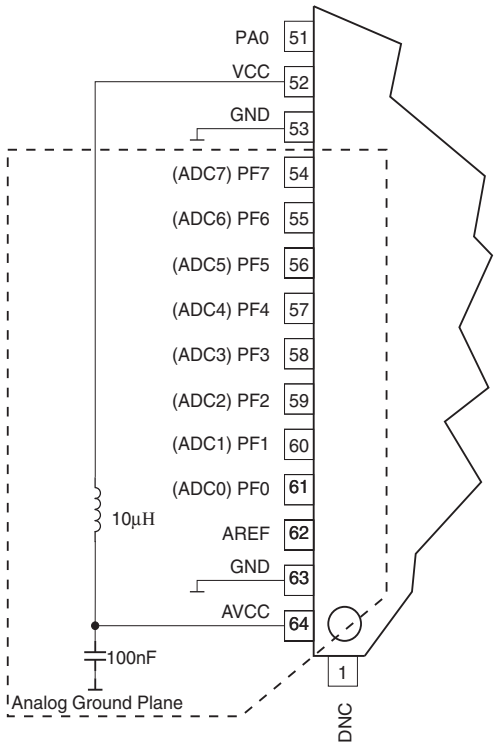


Figure 23-9. ADC Power Connections



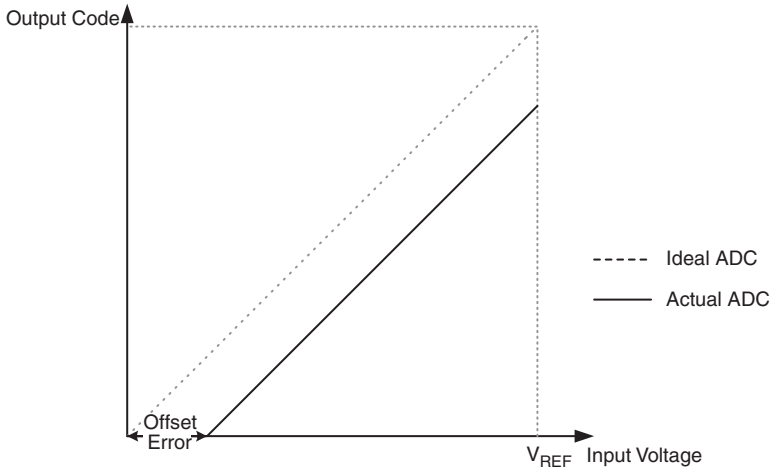
23.6.3 ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between GND and V_{REF} in 2^n steps (LSBs). The lowest code is read as 0, and the highest code is read as 2^n-1.

Several parameters describe the deviation from the ideal behavior:

- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 23-10. Offset Error



- 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 223.

24.5 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 224.

24.6 Using the On-chip Debug System

As shown in Figure 24-1, 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, two Program Memory Break Points, and two combined Break Points. Together, the four 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”).

Table 25-7. ATmega325/645 Boundary-scan Order, 64-pin (Continued)

Bit Number	Signal Name	Module
64	PG1.Control	
63	PG1.Pull-up_Enable	
62	PC0.Data	Port C
61	PC0.Control	
60	PC0.Pull-up_Enable	
59	PC1.Data	
58	PC1.Control	
57	PC1.Pull-up_Enable	
56	PC2.Data	
55	PC2.Control	
54	PC2.Pull-up_Enable	
53	PC3.Data	
52	PC3.Control	
51	PC3.Pull-up_Enable	
50	PC4.Data	
49	PC4.Control	
48	PC4.Pull-up_Enable	
47	PC5.Data	
46	PC5.Control	
45	PC5.Pull-up_Enable	
44	PC6.Data	
43	PC6.Control	
42	PC6.Pull-up_Enable	
41	PC7.Data	
40	PC7.Control	
39	PC7.Pull-up_Enable	
38	PG2.Data	Port G
37	PG2.Control	
36	PG2.Pull-up_Enable	
35	PA7.Data	Port A
34	PA7.Control	
33	PA7.Pull-up_Enable	
32	PA6.Data	
31	PA6.Control	
30	PA6.Pull-up_Enable	
29	PA5.Data	

27. Memory Programming

27.1 Program And Data Memory Lock Bits

The Atmel ATmega325/3250/645/6450 provides six Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 27-2. The Lock bits can only be erased to “1” with the Chip Erase command.

Table 27-1. Lock Bit Byte[1]

Lock Bit Byte	Bit No	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
BLB12	5	Boot Lock bit	1 (unprogrammed)
BLB11	4	Boot Lock bit	1 (unprogrammed)
BLB02	3	Boot Lock bit	1 (unprogrammed)
BLB01	2	Boot Lock bit	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note: 1. “1” means unprogrammed, “0” means programmed

Table 27-2. Lock Bit Protection Modes[1][2]

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

Table 27-15. Serial Programming Instruction Set

Instruction/Operation	Instruction Format			
	Byte 1	**Byte 2**	**Byte 3**	**Byte4**
Read Lock bits	$58	$00	$00	data byte out
Read Signature Byte	$30	$00	0000 000aa	data byte out
Read Fuse bits	$50	$00	$00	data byte out
Read Fuse High bits	$58	$08	$00	data byte out
Read Extended Fuse Bits	$50	$08	$00	data byte out
Read Calibration Byte	$38	$00	$00	data byte out
Write Instructions				
Write Program Memory Page	$4C	adr MSB	adr LSB	$00
Write EEPROM Memory	$C0	0000 00aa / 0000 0aaa	aaaa aaaa	data byte in
Write EEPROM Memory Page (page access)	$C2	0000 00aa / 0000 0aaa	aaaa aa00 / aaaa a000	$00
Write Lock bits	$AC	$E0	$00	data byte in
Write Fuse bits	$AC	$A0	$00	data byte in
Write Fuse High bits	$AC	$A8	$00	data byte in
Write Extended Fuse Bits	$AC	$A4	$00	data byte in

Notes:
1. Not all instructions are applicable for all parts
2. a = address
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused Fuses and Lock bits should be unprogrammed (‘1’) .
5. Refer to the correspondig section for Fuse and Lock bits, Calibration and Signature bytes and Page size.
6. See htt://www.atmel.com/avr for Application Notes regarding programming and programmers.

If the LSB in RDY/BSY data byte out is ‘1’, a programming operation is still pending. Wait until this bit returns ‘0’ before the next instruction is carried out.

Within the same page, the low data byte must be loaded prior to the high data byte.

After data is loaded to the page buffer, program the EEPROM page, see Figure 27-12.

8. Poll for Fuse write complete using programming instruction 6g, or wait for t_{WLRH} (refer to Table 27-12 on page 279).

27.8.21 Programming the Lock Bits

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Lock bit write using programming instruction 7a.
3. Load data using programming instructions 7b. A bit value of “0” will program the corresponding lock bit, a “1” will leave the lock bit unchanged.
4. Write Lock bits using programming instruction 7c.
5. Poll for Lock bit write complete using programming instruction 7d, or wait for t_{WLRH} (refer to Table 27-12 on page 279).

27.8.22 Reading the Fuses and Lock Bits

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Fuse/Lock bit read using programming instruction 8a.
3. To read all Fuses and Lock bits, use programming instruction 8e.
   To only read Fuse High byte, use programming instruction 8b.
   To only read Fuse Low byte, use programming instruction 8c.
   To only read Lock bits, use programming instruction 8d.

27.8.23 Reading the Signature Bytes

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Signature byte read using programming instruction 9a.
3. Load address 0x00 using programming instruction 9b.
4. Read first signature byte using programming instruction 9c.
5. Repeat steps 3 and 4 with address 0x01 and address 0x02 to read the second and third signature bytes, respectively.

27.8.24 Reading the Calibration Byte

1. Enter JTAG instruction PROG_COMMANDS.
2. Enable Calibration byte read using programming instruction 10a.
3. Load address 0x00 using programming instruction 10b.

Read the calibration byte using programming instruction 10c.

28.5 System and Reset Characteristics

Table 28-4. Reset, Brown-out and Internal Voltage Reference Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}[1]	Power-on Reset Threshold Voltage (rising)	T_A = -40°C to 85°C	0.7	1.0	1.4	V
	Power-on Reset Threshold Voltage (falling)[1]	T_A = -40°C to 85°C	0.05	0.9	1.3	V
V_{PSR}	Power-on Slope Rate		0.01		4.5	V/ms
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage	V_{CC} = 3V	$0.2V_{CC}$		$0.85V_{CC}$	V
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin	V_{CC} = 3V		800		ns
V_{HYST}	Brown-out Detector Hysteresis			50		mV
t_{BOD}	Min Pulse Width on Brown-out Reset			2		µs
V_{BG}	Bandgap reference voltage	V_{CC}= 2.7V, T_A = 25°C	1.0	1.1	1.2	V
t_{BG}	Bandgap reference start-up time	V_{CC}= 2.7V, T_A = 25°C		40	70	µs
I_{BG}	Bandgap reference current consumption	V_{CC}= 2.7V, T_A = 25°C		15		µA

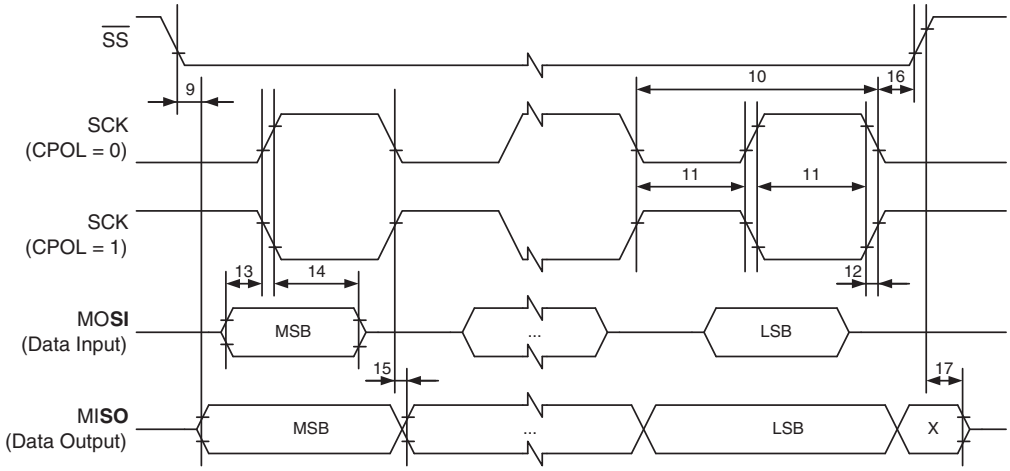
Notes: 1. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling)

Table 28-5. BODLEVEL Fuse Coding[1]

BODLEVEL 2:0 Fuses	Min V_{BOT}	Typ V_{BOT}	Max V_{BOT}	Units
11	BOD Disabled			
10	1.7	1.8	2.0	V
01	2.5	2.7	2.9	
00	4.1	4.3	4.5	

Note: 1. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to V_{CC} = V_{BOT} during the production test. This guarantees that a Brown-Out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 10 for Atmel ATmega325/3250/645/6450 and BODLEVEL = 01 for Atmel ATmega325/3250/645/6450V.

Figure 28-5. SPI Interface Timing Requirements (Slave Mode)



29.7 Pin Pull-up

Figure 29-15. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 5V)

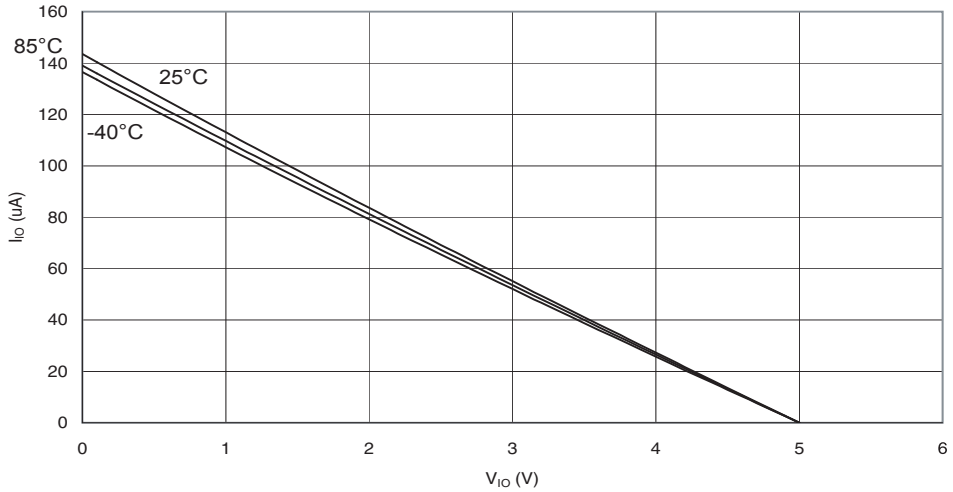


Figure 29-16. I/O Pin Pull-up Resistor Current vs. Input Voltage (V_{CC} = 2.7V)

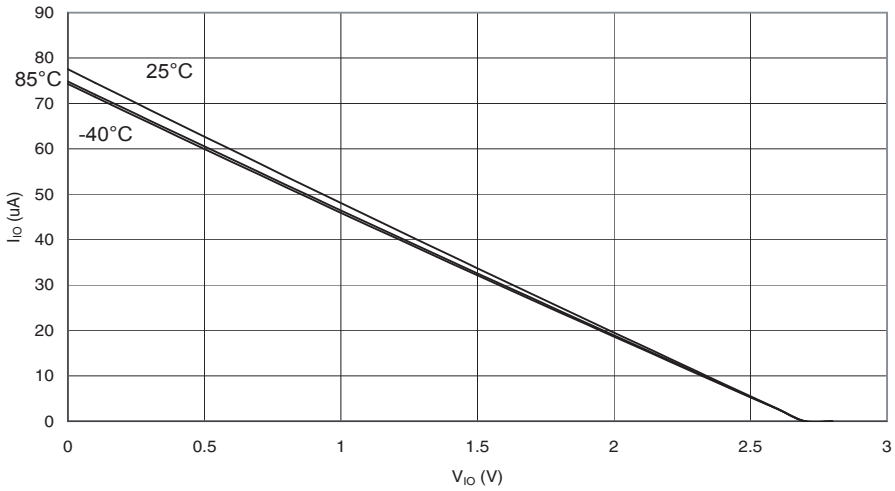


Figure 29-35. I/O Pin Input Hysteresis vs. V_{CC}

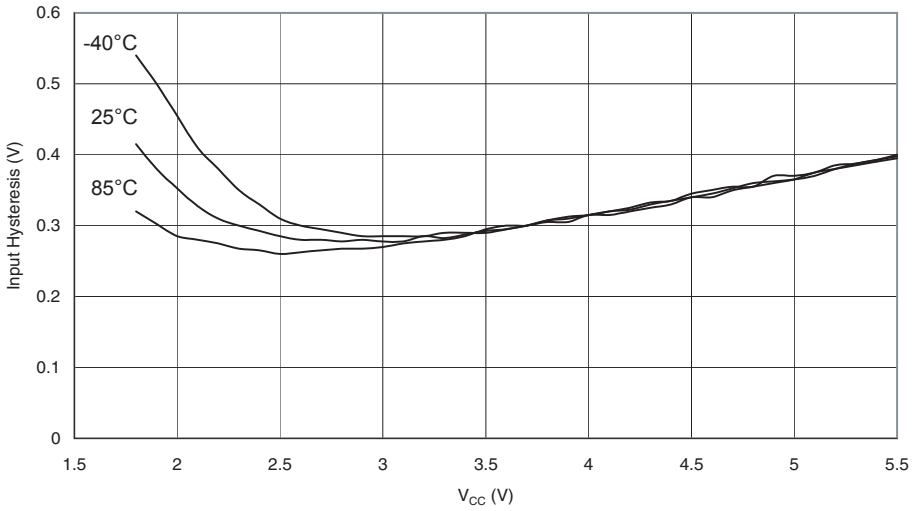


Figure 29-36. Reset Input Threshold Voltage vs. V_{CC} (V_{IH},Reset Pin Read as “1”)