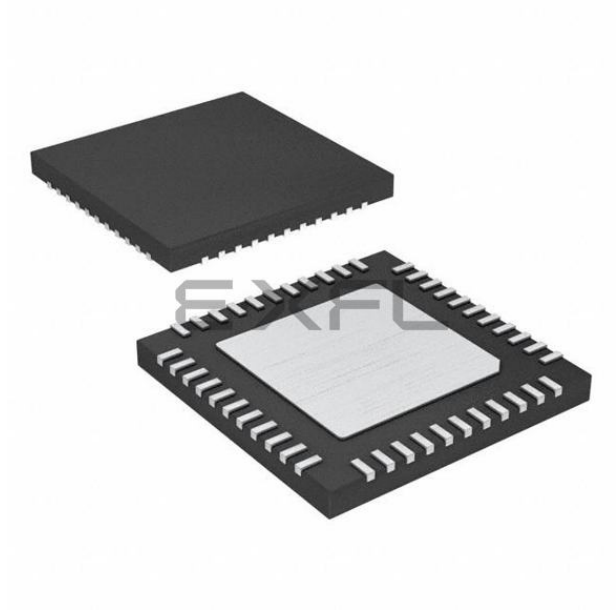


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	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)	4.5V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°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/atmega16-16mc

AVR ATmega16 Memories

This section describes the different memories in the ATmega16. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega16 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

In-System Reprogrammable Flash Program Memory

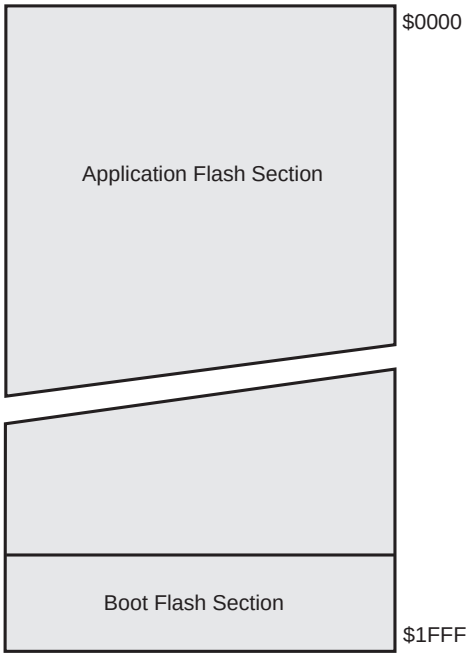
The ATmega16 contains 16 Kbytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 8K × 16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega16 Program Counter (PC) is 13 bits wide, thus addressing the 8K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in “Boot Loader Support – Read-While-Write Self-Programming” on page 246. “Memory Programming” on page 259 contains a detailed description on Flash data serial downloading using the SPI pins or the JTAG interface.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory Instruction Description).

Timing diagrams for instruction fetch and execution are presented in “Instruction Execution Timing” on page 13.

Figure 8. Program Memory Map



The following code example shows one assembly and one C function for turning off the WDT. The example assumes that interrupts are controlled (for example by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
WDT_off:
  ; Reset WDT
  WDR
  ; Write logical one to WDTOE and WDE
  in   r16, WDTCR
  ori r16, (1<<WDTOE)|(1<<WDE)
  out  WDTCR, r16
  ; Turn off WDT
  ldi  r16, (0<<WDE)
  out  WDTCR, r16
  ret
```

C Code Example

```
void WDT_off(void)
{
  /* Reset WDT*/
  _WDR();
  /* Write logical one to WDTOE and WDE */
  WDTCR |= (1<<WDTOE) | (1<<WDE);
  /* Turn off WDT */
  WDTCR = 0x00;
}
```

Register Description for I/O Ports

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B Input Pins Address – PINB

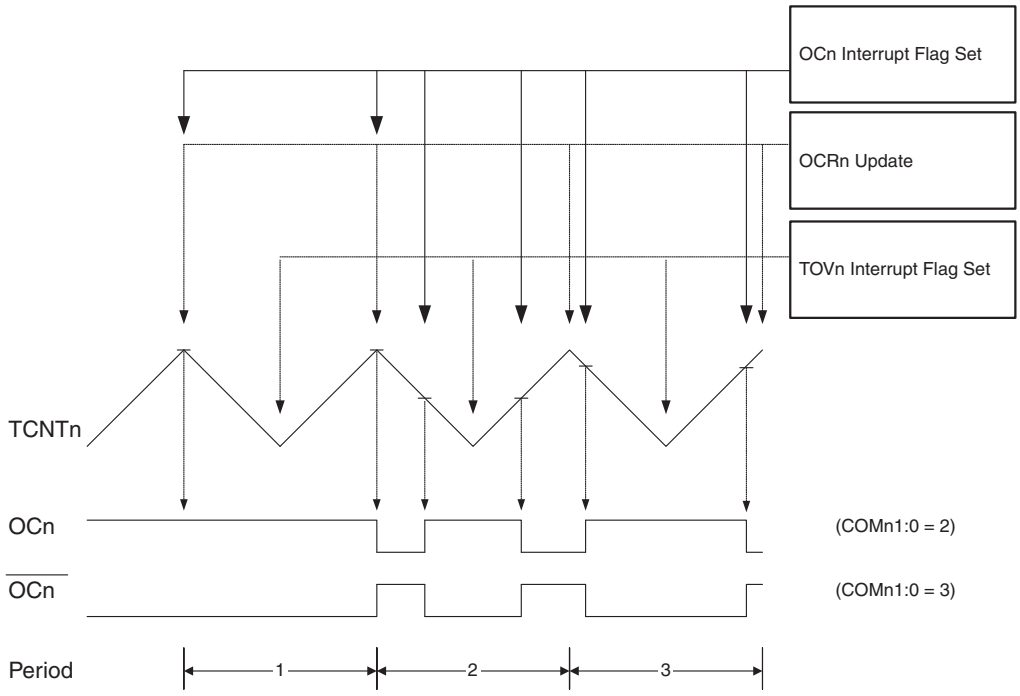
Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Phase Correct PWM Mode

The phase correct PWM mode (WGM01:0 = 1) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to MAX and then from MAX to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0) is cleared on the compare match between TCNT0 and OCR0 while upcounting, and set on the compare match while downcounting. 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 is fixed to eight bits. In phase correct PWM mode the counter is incremented until the counter value matches MAX. When the counter reaches MAX, it changes the count direction. The TCNT0 value will be equal to MAX for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 33. The TCNT0 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 TCNT0 slopes represent compare matches between OCR0 and TCNT0.

Figure 33. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0 pin. Setting the COM01:0 bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM01:0 to 3 (see Table 41 on page 84). The actual OC0 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0 Register at the compare match between OCR0 and TCNT0 when the counter increments, and setting (or clearing) the OC0 Register at compare match between OCR0 and TCNT0 when the counter decrements. The

Table 45. Compare Output Mode, Fast PWM[1]

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OCnA/OCnB disconnected.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at BOTTOM, (non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at BOTTOM, (inverting mode)

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. See “Fast PWM Mode” on page 102. for more details.

Table 46 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 46. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM [1]

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 14: Toggle OCnA on Compare Match, OCnB disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match when up-counting. Set OC1A/OC1B on compare match when downcounting.
1	1	Set OC1A/OC1B on compare match when up-counting. Clear OC1A/OC1B on compare match when downcounting.

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. See “Phase Correct PWM Mode” on page 104. for more details.

- **Bit 3 – FOC1A: Force Output Compare for Channel A**
- **Bit 2 – FOC1B: Force Output Compare for Channel B**

The FOC1A/FOC1B bits are only active when the WGM13:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written when operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the Waveform Generation unit. The OC1A/OC1B output is changed according to its COM1x1:0 bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x1:0 bits that determine the effect of the forced compare.

The double buffered Output Compare Register (OCR2) is compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the Output Compare Pin (OC2). See “Output Compare Unit” on page 119. for details. The compare match event will also set the Compare Flag (OCF2) which can be used to generate an output compare interrupt request.

Definitions

Many register and bit references in this document are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 2. However, when using the register or bit defines in a program, the precise form must be used (that is, TCNT2 for accessing Timer/Counter2 counter value and so on). The definitions in Table 49 are also used extensively throughout the document.

Table 49. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2 Register. The assignment is dependent on the mode of operation.

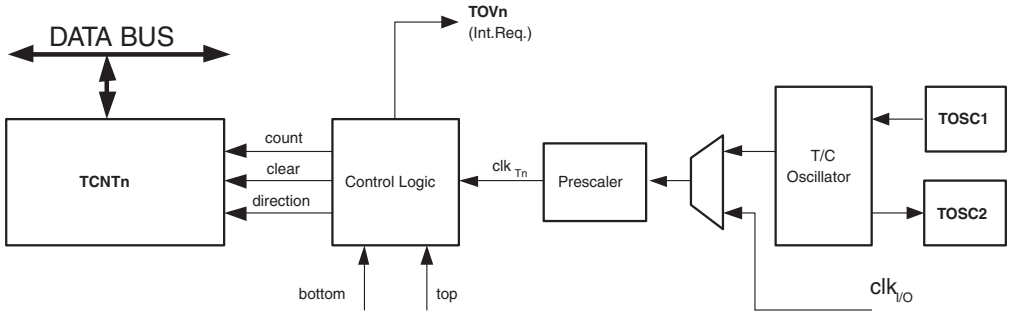
Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source clk_{T2} is by default equal to the MCU clock, $clk_{I/O}$. When the AS2 bit in the ASSR Register is written to logic one, the clock source is taken from the Timer/Counter Oscillator connected to TOSC1 and TOSC2. For details on asynchronous operation, see “Asynchronous Status Register – ASSR” on page 131. For details on clock sources and prescaler, see “Timer/Counter Prescaler” on page 134.

Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 54 shows a block diagram of the counter and its surrounding environment.

Figure 54. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT2 by 1.
direction	Selects between increment and decrement.
clear	Clear TCNT2 (set all bits to zero).
clk_{T2}	Timer/Counter clock.
top	Signalizes that TCNT2 has reached maximum value.

- The Oscillator is optimized for use with a 32.768 kHz watch crystal. Applying an external clock to the TOSC1 pin may result in incorrect Timer/Counter2 operation. The CPU main clock frequency must be more than four times the Oscillator frequency.
- When writing to one of the registers TCNT2, OCR2, or TCCR2, the value is transferred to a temporary register, and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the three mentioned registers have their individual temporary register, which means for example that writing to TCNT2 does not disturb an OCR2 write in progress. To detect that a transfer to the destination register has taken place, the Asynchronous Status Register – ASSR has been implemented.
- When entering Power-save or Extended Standby mode after having written to TCNT2, OCR2, or TCCR2, the user must wait until the written register has been updated if Timer/Counter2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if the Output Compare2 interrupt is used to wake up the device, since the output compare function is disabled during writing to OCR2 or TCNT2. If the write cycle is not finished, and the MCU enters sleep mode before the OCR2UB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.
- If Timer/Counter2 is used to wake the device up from Power-save or Extended Standby mode, precautions must be taken if the user wants to re-enter one of these modes: The interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and re-entering sleep mode is less than one TOSC1 cycle, the interrupt will not occur, and the device will fail to wake up. If the user is in doubt whether 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:

Two-wire Serial Interface

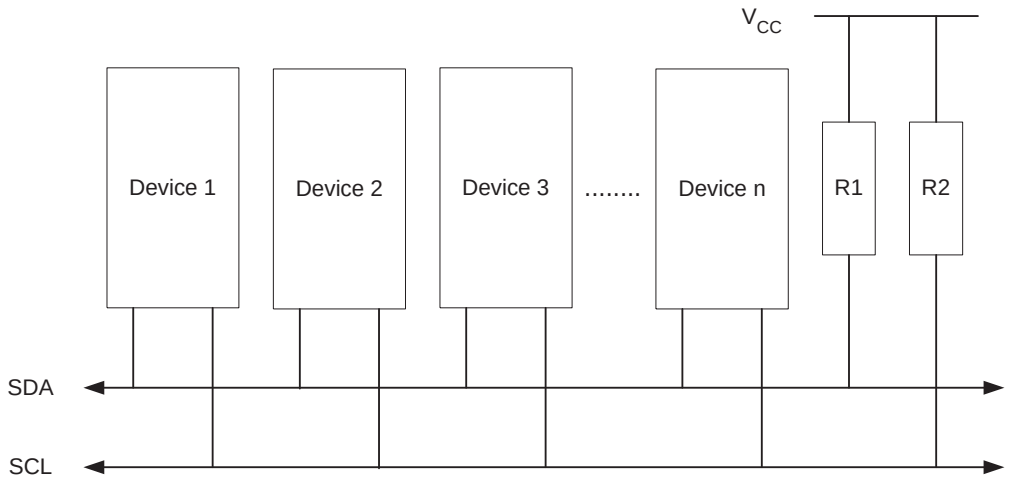
Features

- **Simple Yet Powerful and Flexible Communication Interface, Only Two Bus Lines Needed**
- **Both Master and Slave Operation Supported**
- **Device Can Operate as Transmitter or Receiver**
- **7-bit Address Space allows up to 128 Different Slave Addresses**
- **Multi-master Arbitration Support**
- **Up to 400 kHz Data Transfer Speed**
- **Slew-rate Limited Output Drivers**
- **Noise Suppression Circuitry Rejects Spikes on Bus Lines**
- **Fully Programmable Slave Address with General Call Support**
- **Address Recognition causes Wake-up when AVR is in Sleep Mode**

Two-wire Serial Interface Bus Definition

The Two-wire Serial Interface (TWI) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.

Figure 76. TWI Bus Interconnection



TWI Terminology

The following definitions are frequently encountered in this section.

Table 72. TWI Terminology

Term	Description
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock.
Slave	The device addressed by a Master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

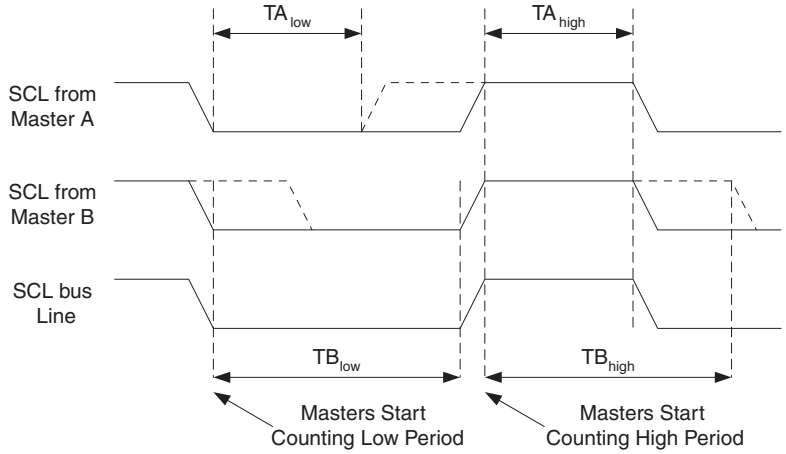
Multi-master Bus Systems, Arbitration and Synchronization

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, that is, 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.

Figure 82. SCL Synchronization between Multiple Masters



Arbitration is carried out by all Masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value the Master had output, it has lost the arbitration. Note that a Master can only lose arbitration when it outputs a high SDA value while another Master outputs a low value. The losing Master should immediately go to Slave mode, checking if it is being addressed by the winning Master. The SDA line should be left high, but losing Masters are allowed to generate a clock signal until the end of the current data or address packet. Arbitration will continue until only one Master remains, and this may take many bits. If several Masters are trying to address the same Slave, arbitration will continue into the data packet.

- **Bits 1..0 – TWPS: TWI Prescaler Bits**

These bits can be read and written, and control the bit rate prescaler.

Table 73. TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

To calculate bit rates, see “Bit Rate Generator Unit” on page 178. The value of TWPS1..0 is used in the equation.

TWI Data Register – TWDR

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

In Transmit mode, TWDR contains the next byte to be transmitted. In Receive mode, the TWDR contains the last byte received. It is writable while the TWI is not in the process of shifting a byte. This occurs when the TWI Interrupt Flag (TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remains stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI interrupt. In this case, the contents of TWDR is undefined. In the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK bit is controlled automatically by the TWI logic, the CPU cannot access the ACK bit directly.

- **Bits 7..0 – TWD: TWI Data Register**

These eight bits contain the next data byte to be transmitted, or the latest data byte received on the Two-wire Serial Bus.

TWI (Slave) Address Register – TWAR

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

The TWAR should be loaded with the 7-bit Slave address (in the seven most significant bits of TWAR) to which the TWI will respond when programmed as a Slave Transmitter or receiver. In multi-master systems, TWAR must be set in Masters which can be addressed as Slaves by other Masters.

The LSB of TWAR is used to enable recognition of the general call address ($00). There is an associated address comparator that looks for the Slave address (or general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

- **Bits 7..1 – TWA: TWI (Slave) Address Register**

These seven bits constitute the Slave address of the TWI unit.

Transmission Modes

The TWI can operate in one of four major modes. These are named Master Transmitter (MT), Master Receiver (MR), Slave Transmitter (ST) and Slave Receiver (SR). Several of these modes can be used in the same application. As an example, the TWI can use MT mode to write data into a TWI EEPROM, MR mode to read the data back from the EEPROM. If other Masters are present in the system, some of these might transmit data to the TWI, and then SR mode would be used. It is the application software that decides which modes are legal.

The following sections describe each of these modes. Possible status codes are described along with figures detailing data transmission in each of the modes. These figures contain the following abbreviations:

S: START condition

Rs: REPEATED START condition

R: Read bit (high level at SDA)

W: Write bit (low level at SDA)

A: Acknowledge bit (low level at SDA)

$\overline{A}$: Not acknowledge bit (high level at SDA)

Data: 8-bit data byte

P: STOP condition

SLA: Slave Address

In Figure 87 to Figure 93, circles are used to indicate that the TWINT Flag is set. The numbers in the circles show the status code held in TWSR, with the prescaler bits masked to zero. At these points, actions must be taken by the application to continue or complete the TWI transfer. The TWI transfer is suspended until the TWINT Flag is cleared by software.

When the TWINT Flag is set, the status code in TWSR is used to determine the appropriate software action. For each status code, the required software action and details of the following serial transfer are given in Table 74 to Table 77. Note that the prescaler bits are masked to zero in these tables.

Master Transmitter Mode

In the Master Transmitter mode, a number of data bytes are transmitted to a Slave Receiver (see Figure 86). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 86. Data Transfer in Master Transmitter Mode

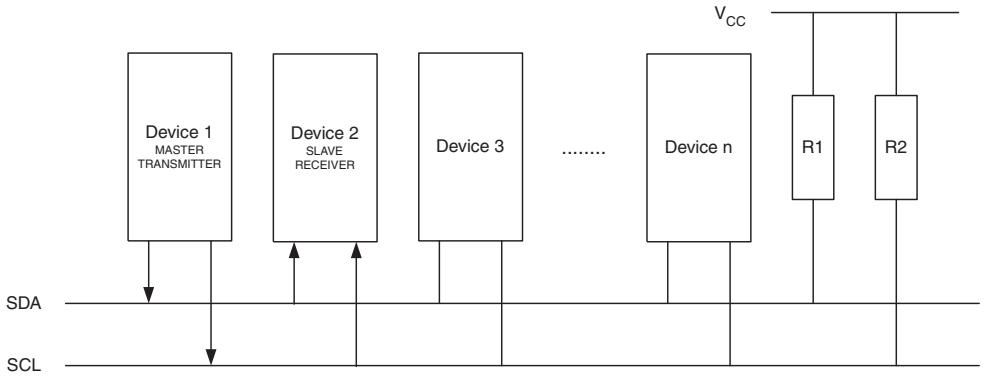


Table 76. Status Codes for Slave Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$70	General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$78	Arbitration lost in SLA+R/W as Master; General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$80	Previously addressed with own SLA+W; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
$88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
$90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
$98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
$A0	A STOP condition or repeated START condition has been received while still addressed as Slave	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
			1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
			1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

Table 82. Correlation between Input Voltage and Output Codes

V_{ADCn}	Read code	Corresponding Decimal Value
V_{ADCm} + V_{REF}/GAIN	0x1FF	511
V_{ADCm} + 511/512 V_{REF}/GAIN	0x1FF	511
V_{ADCm} + 510/512 V_{REF}/GAIN	0x1FE	510
...	...	...
V_{ADCm} + 1/512 V_{REF}/GAIN	0x001	1
V_{ADCm}	0x000	0
V_{ADCm} - 1/512 V_{REF}/GAIN	0x3FF	-1
...	...	...
V_{ADCm} - 511/512 V_{REF}/GAIN	0x201	-511
V_{ADCm} - V_{REF}/GAIN	0x200	-512

Example:

ADMUX = 0xED (ADC3 - ADC2, 10x gain, 2.56V reference, left adjusted result)

Voltage on ADC3 is 300 mV, voltage on ADC2 is 500 mV.

ADCR = 512 × 10 × (300 - 500) / 2560 = -400 = 0x270

ADCL will thus read 0x00, and ADCH will read 0x9C. Writing zero to ADLAR right adjusts the result: ADCL = 0x70, ADCH = 0x02.

ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
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 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 83. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 83. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conver-

Table 85. ADC Prescaler Selections

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

The ADC Data Register – ADCL and ADCH

ADLAR = 0

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

ADLAR = 1

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

When an ADC conversion is complete, the result is found in these two registers. If differential channels are used, the result is presented in two's complement form.

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

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

- **ADC9:0: ADC Conversion Result**

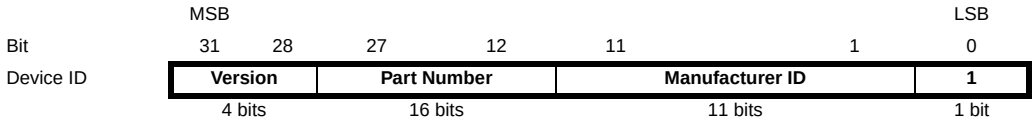
These bits represent the result from the conversion, as detailed in "ADC Conversion Result" on page 216.

controller state. The Bypass Register can be used to shorten the scan chain on a system when the other devices are to be tested.

Device Identification Register

Figure 114 shows the structure of the Device Identification Register.

Figure 114. The Format of the Device Identification Register



Version

Version is a 4-bit number identifying the revision of the component. The JTAG version number follows the revision of the device. Revision A is 0x0, revision B is 0x1 and so on. However, some revisions deviate from this rule, and the relevant version number is shown in Table 87.

Table 87. JTAG Version Numbers

Version	JTAG Version Number (Hex)
ATmega16 revision G	0x6
ATmega16 revision H	0xE
ATmega16 revision I	0x8
ATmega16 revision J	0x9
ATmega16 revision K	0xA
ATmega16 revision L	0xB

Part Number

The part number is a 16-bit code identifying the component. The JTAG Part Number for ATmega16 is listed in Table 88.

Table 88. AVR JTAG Part Number

Part Number	JTAG Part Number (Hex)
ATmega16	0x9403

Manufacturer ID

The Manufacturer ID is a 11 bit code identifying the manufacturer. The JTAG manufacturer ID for ATMEL is listed in Table 89.

Table 89. Manufacturer ID

Manufacturer	JTAG Manufacturer ID (Hex)
ATMEL	0x01F

Reset Register

The Reset Register is a Test Data Register used to reset the part. Since the AVR tri-states Port Pins when reset, the Reset Register can also replace the function of the unimplemented optional JTAG instruction HIGHZ.

A high value in the Reset Register corresponds to pulling the External Reset low. The part is reset as long as there is a high value present in the Reset Register. Depending on the Fuse settings for the clock options, the part will remain reset for a Reset Time-Out Period (refer to “Clock Sources” on page 25) after releasing the Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, as shown in Figure 115.

Table 94. ATmega16 Boundary-scan Order (Continued)

Bit Number	Signal Name	Module
39	PD7.Control	
38	PD7.Pullup_Enable	
37	PC0.Data	Port C
36	PC0.Control	
35	PC0.Pullup_Enable	
34	PC1.Data	
33	PC1.Control	
32	PC1.Pullup_Enable	
31	PC6.Data	
30	PC6.Control	
29	PC6.Pullup_Enable	
28	PC7.Data	
27	PC7.Control	
26	PC7.Pullup_Enable	
25	TOSC	32 kHz Timer Oscillator
24	TOSCON	
23	PA7.Data	Port A
22	PA7.Control	
21	PA7.Pullup_Enable	
20	PA6.Data	
19	PA6.Control	
18	PA6.Pullup_Enable	
17	PA5.Data	
16	PA5.Control	
15	PA5.Pullup_Enable	
14	PA4.Data	
13	PA4.Control	
12	PA4.Pullup_Enable	
11	PA3.Data	
10	PA3.Control	
9	PA3.Pullup_Enable	
8	PA2.Data	
7	PA2.Control	
6	PA2.Pullup_Enable	
5	PA1.Data	

```
  sbrs  temp1, RWWSB              ; If RWWSB is set, the RWW section is not
                                  ; ready yet
  ret
  ; re-enable the RWW section
  ldi   spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  call  Do_spm
  rjmp  Return

Do_spm:
  ; check for previous SPM complete
Wait_spm:
  in    temp1, SPMCR
  sbrc  temp1, SPMEN
  rjmp  Wait_spm
  ; input: spmcrval determines SPM action
  ; disable interrupts if enabled, store status
  in    temp2, SREG
  cli
  ; check that no EEPROM write access is present
Wait_ee:
  sbic  EECR, EEWE
  rjmp  Wait_ee
  ; SPM timed sequence
  out   SPMCR, spmcrval
  spm
  ; restore SREG (to enable interrupts if originally enabled)
  out   SREG, temp2
  ret
```

ATmega16 Boot Loader Parameters

In Table 100 through Table 102, the parameters used in the description of the self programming are given.

Table 100. Boot Size Configuration[1]

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application section	Boot Reset Address (start Boot Loader Section)
1	1	128 words	2	$0000 - $1F7F	$1F80 - $1FFF	$1F7F	$1F80
1	0	256 words	4	$0000 - $1EFF	$1F00 - $1FFF	$1EFF	$1F00
0	1	512 words	8	$0000 - $1DFF	$1E00 - $1FFF	$1DFF	$1E00
0	0	1024 words	16	$0000 - $1BFF	$1C00 - $1FFF	$1BFF	$1C00

Note: 1. The different BOOTSZ Fuse configurations are shown in Figure 125

Table 101. Read-While-Write Limit[1]

Section	Pages	Address
Read-While-Write section (RWW)	112	$0000 - $1BFF
No Read-While-Write section (NRWW)	16	$1C00 - $1FFF

Note: 1. For details about these two section, see “NRWW – No Read-While-Write Section” on page 247 and “RWW – Read-While-Write Section” on page 247

Figure 150. Active Supply Current vs. Frequency (1 MHz - 20 MHz)

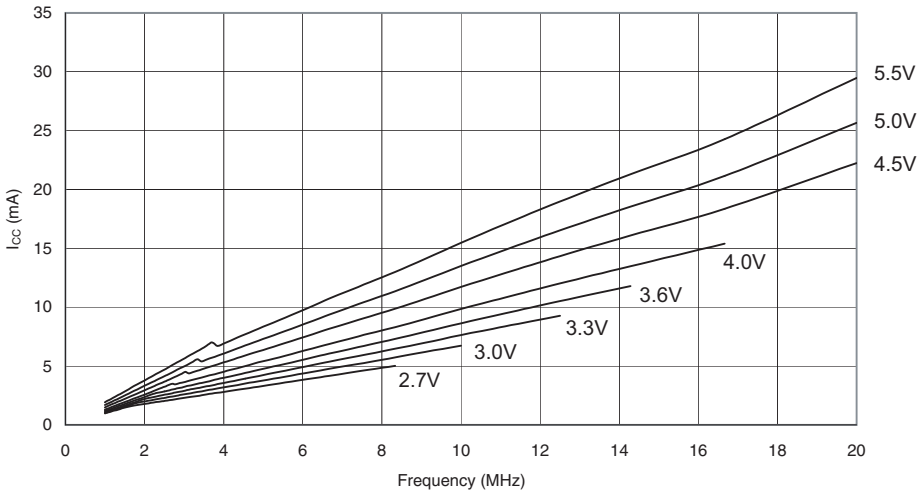


Figure 151. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

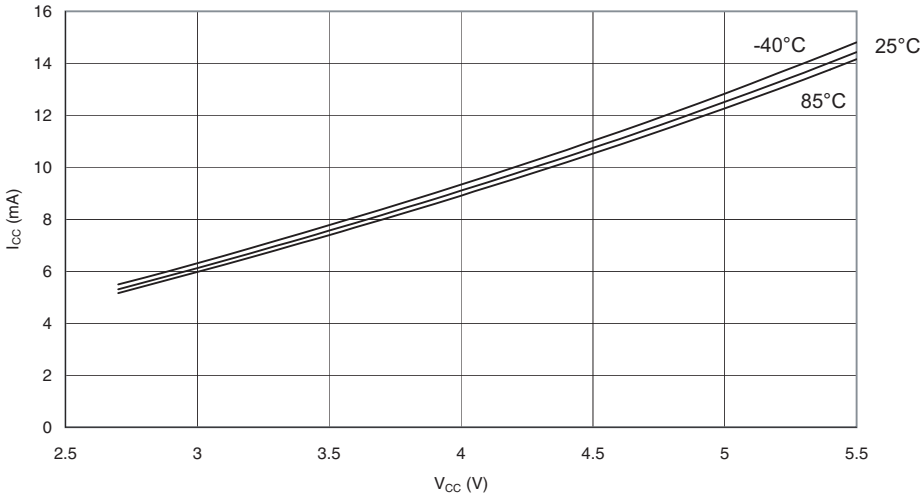


Figure 180. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 5V)

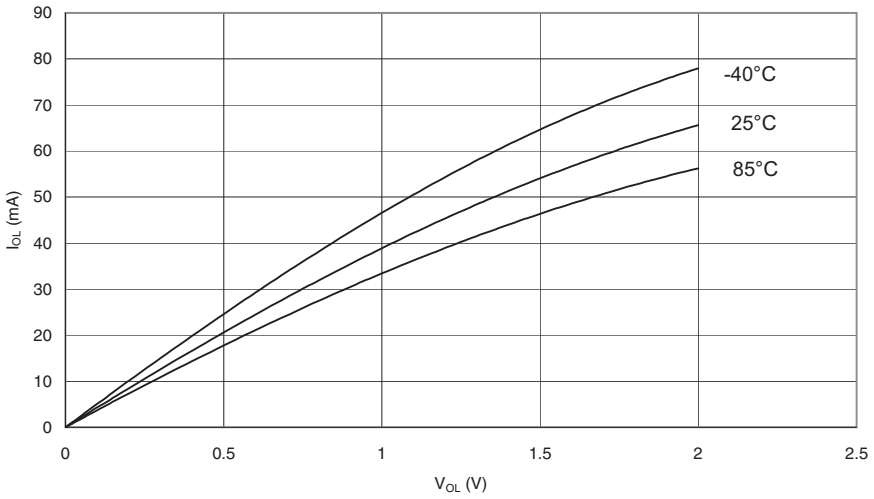


Figure 181. I/O Pin Sink Current vs. Output Voltage (V_{CC} = 2.7V)

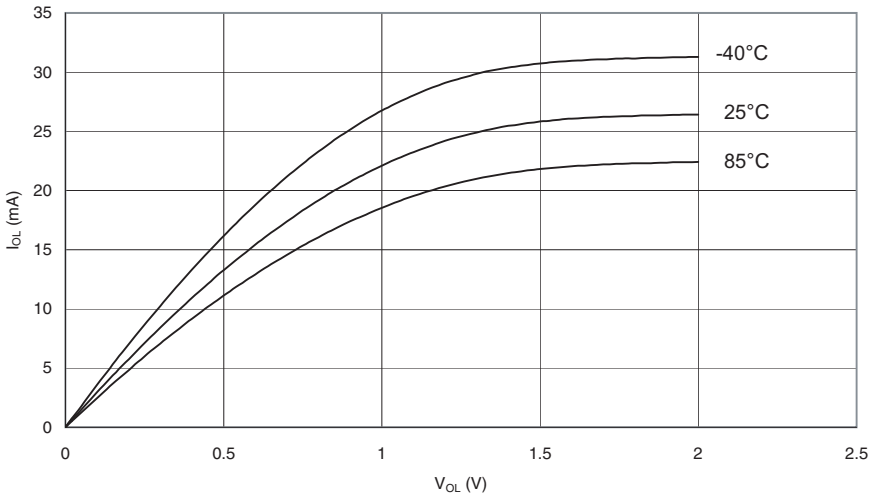


Figure 186. Reset Input Threshold Voltage vs. V_{CC} (V_{IL}, Reset Pin Read As '0')

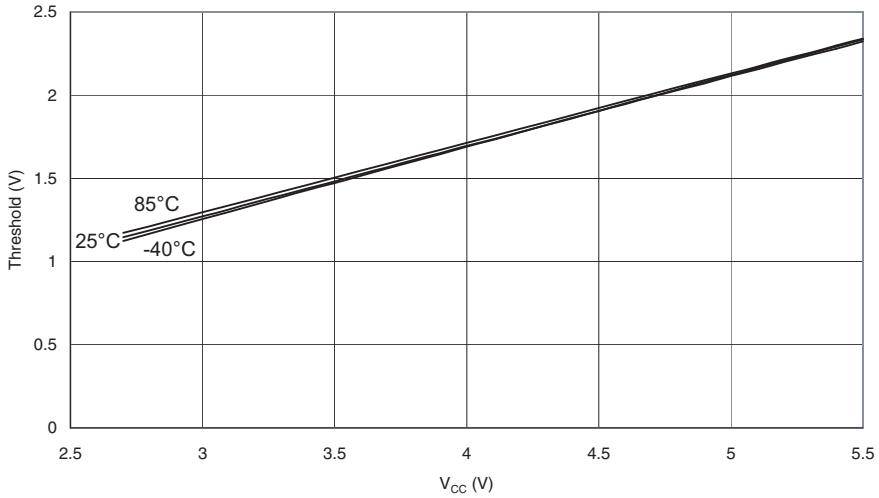


Figure 187. Reset Input Pin Hysteresis vs. V_{CC}