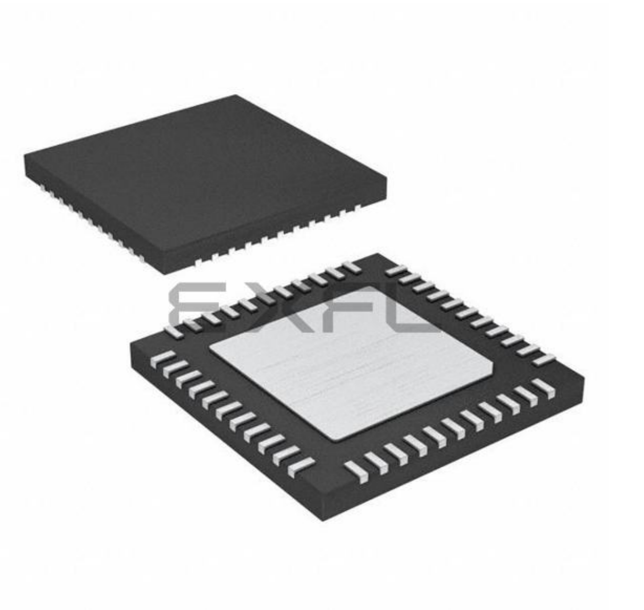


Welcome to E-XFL.COM

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	20MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	32
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°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/atmega324p-20mq

- Up to 64 Sense Channels
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC
    - Differential Mode with Selectable Gain at 1×, 10× or 200×
  - One Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Two Programmable Serial USART
  - One Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP
  - 44-lead TQFP
  - 44-pad VQFN/QFN
- Operating Voltage:
  - 1.8 - 5.5V for ATmega324PV
  - 2.7 - 5.5V for ATmega324P
- Speed Grades
  - ATmega324PV:
    - 0 - 4MHz @ 1.8V - 5.5V
    - 0 - 10MHz @ 2.7V - 5.5V
  - ATmega324P:
    - 0 - 10MHz @ 2.7V - 5.5V
    - 0 - 20MHz @ 4.5 - 5.5V
- Power Consumption at 1MHz, 1.8V, 25°C
  - Active Mode: 0.4mA
  - Power-down Mode: 0.1µA
  - Power-save Mode: 0.6µA (Including 32kHz RTC)

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, this device has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

8.2. 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 *Instruction Set Summary* section for a detailed description.

Related Links

Instruction Set Summary on page 435

8.3. 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. 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.

16. TC0 - 8-bit Timer/Counter0 with PWM

Related Links

Timer/Counter0 and Timer/Counter1 Prescalers on page 185

16.1. Features

- Two independent Output Compare Units
- Double Buffered Output Compare Registers
- Clear Timer on Compare Match (Auto Reload)
- Glitch free, phase correct Pulse Width Modulator (PWM)
- Variable PWM period
- Frequency generator
- Three independent interrupt sources (TOV0, OCF0A, and OCF0B)

16.2. Overview

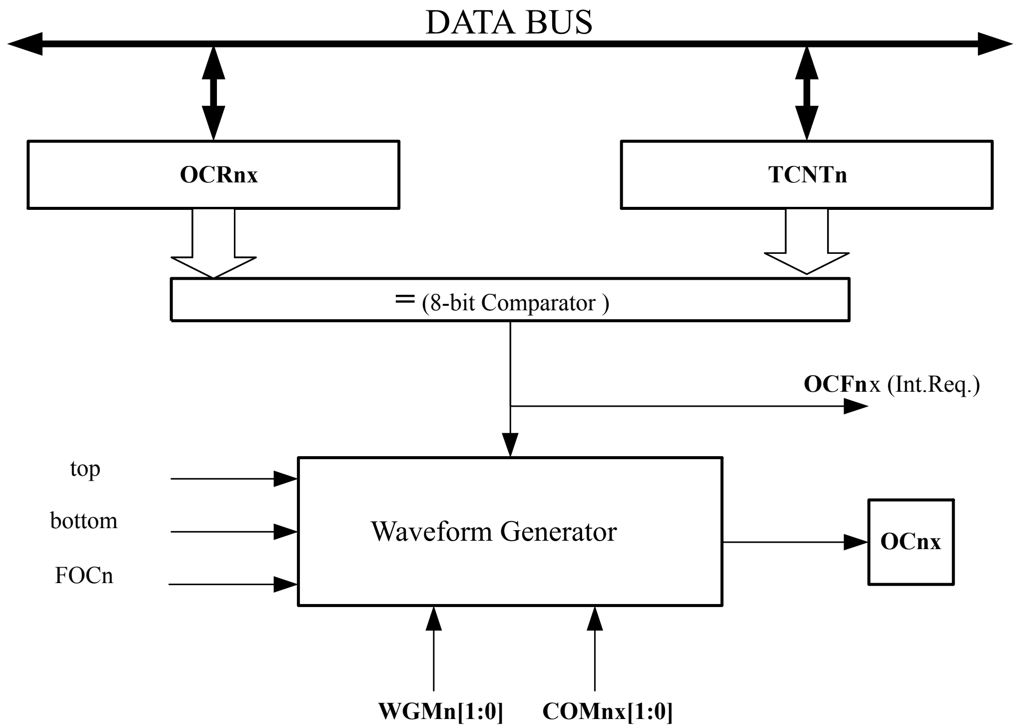
Timer/Counter0 (TC0) is a general purpose 8-bit Timer/Counter module, with two independent Output Compare Units, and PWM support. It allows accurate program execution timing (event management) and wave generation.

A simplified block diagram of the 8-bit Timer/Counter is shown below. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the Register Description. For the actual placement of I/O pins, refer to the pinout diagram.

The TC0 is enabled by writing the PRTIM0 bit in ”Minimizing Power Consumption” to '0'.

The TC0 is enabled when the PRTIM0 bit in the Power Reduction Register (0.PRTIM0) is written to '1'.

Figure 16-3. Output Compare Unit, Block Diagram



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The OCR0x Registers are double buffered when using any of the Pulse Width Modulation (PWM) modes. When double buffering is enabled, the CPU has access to the OCR0x Buffer Register. The double buffering synchronizes the update of the OCR0x Compare Registers to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The double buffering is disabled for the normal and Clear Timer on Compare (CTC) modes of operation, and the CPU will access the OCR0x directly.

16.5.1. Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a '1' to the Force Output Compare (TCCR0C.FOC0x) bit. Forcing compare match will not set the OCF0x Flag or reload/clear the timer, but the OC0x pin will be updated as if a real compare match had occurred (the TCCR0A.COM0x[1:0] bits define whether the OC0x pin is set, cleared or toggled).

16.5.2. Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 Register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0x to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

16.5.3. Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is down counting.

Accessing the low byte triggers the 16-bit read or write operation: When the low byte of a 16-bit register is written by the CPU, the high byte that is currently stored in TEMP and the low byte being written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the TEMP register in the same clock cycle as the low byte is read, and must be read subsequently.

Note: To perform a 16-bit write operation, the high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

Not all 16-bit accesses uses the temporary register for the high byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

16-bit Access

The following code examples show how to access the 16-bit Timer Registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR1A/B and ICR1 Registers. Note that when using C, the compiler handles the 16-bit access.

Assembly Code Example(1)

```
   ...
; Set TCNT1 to 0x01FF
ldi    r17,0x01
ldi    r16,0xFF
out    TCNT1H,r17
out    TCNT1L,r16
; Read TCNT1 into r17:r16
in    r16,TCNT1L
in    r17,TCNT1H
   ...
```

The assembly code example returns the TCNT**1** value in the r17:r16 register pair.

C Code Example(1)

```
unsigned int i;
   ...
/* Set TCNT1 to 0x01FF */
TCNT1 = 0x1FF;
/* Read TCNT1 into i */
i = TCNT1;
   ...
```

Note:

1. The example code assumes that the part specific header file is included. 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”.

Atomic Read

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit Timer Registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to perform an atomic read of the TCNT1 Register contents. The OCR1A/B or ICR1 Registers can be ready by using the same principle.

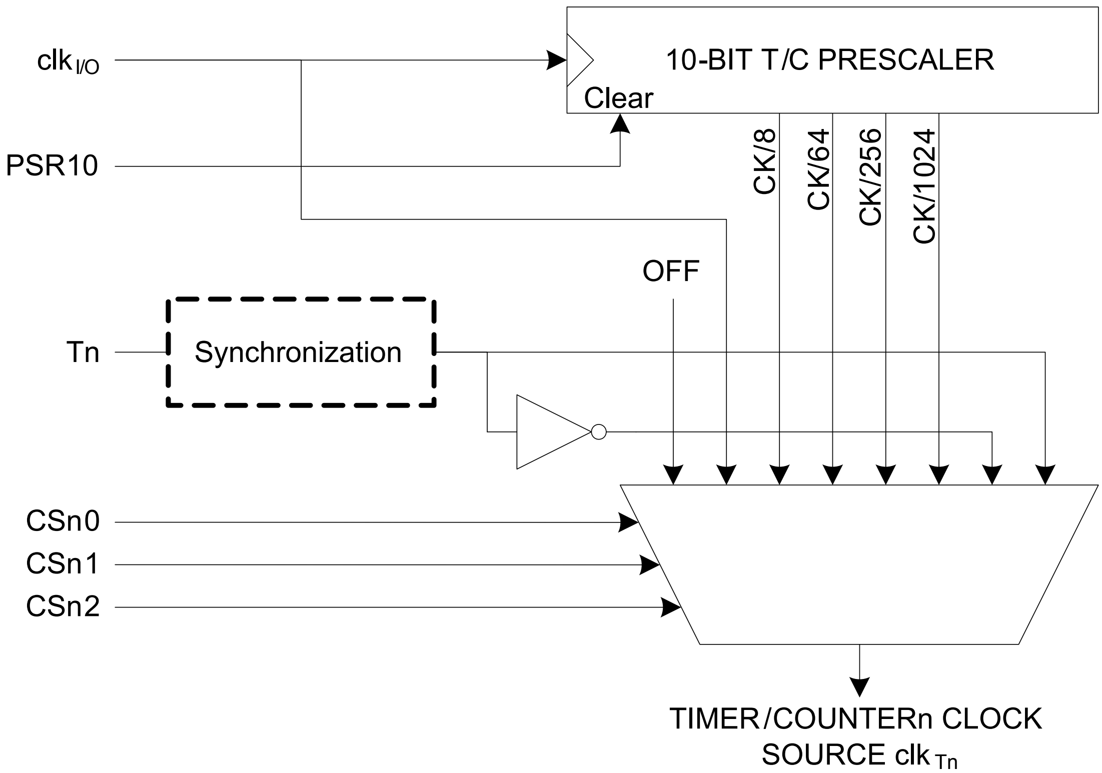
The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1/T0 pin to the counter is updated.

Enabling and disabling of the clock input must be done when T1/T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{Tn} < f_{clk\_I/O}/2$) given a 50% duty cycle. Since the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by the tolerances of the oscillator source (crystal, resonator, and capacitors), 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 18-2. Prescaler for Timer/Counter0 and Timer/Counter1(1)



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

18.4. Register Description

20. SPI – Serial Peripheral Interface

20.1. Features

- Full-duplex, Three-wire Synchronous Data Transfer
- Master or Slave Operation
- LSB First or MSB First Data Transfer
- Seven Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wake-up from Idle Mode
- Double Speed (CK/2) Master SPI Mode

20.2. Overview

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the device and peripheral units, or between several AVR devices.

The USART can also be used in Master SPI mode, please refer to *USART in SPI Mode* chapter.

To enable the SPI module, Power Reduction Serial Peripheral Interface bit in the Power Reduction Register (0.PRSPI0) must be written to '0'.

20.5.1. SPI Control Register 0

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: SPCR0
Offset: 0x4C
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2C

Bit	7	6	5	4	3	2	1	0
	SPIE0	SPE0	DORD0	MSTR0	CPOL0	CPHA0	SPR01	SPR00
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – SPIE0: SPI0 Interrupt Enable
This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and if the Global Interrupt Enable bit in SREG is set.

Bit 6 – SPE0: SPI0 Enable
When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

Bit 5 – DORD0: Data0 Order
When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

Bit 4 – MSTR0: Master/Slave0 Select
This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If SS is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

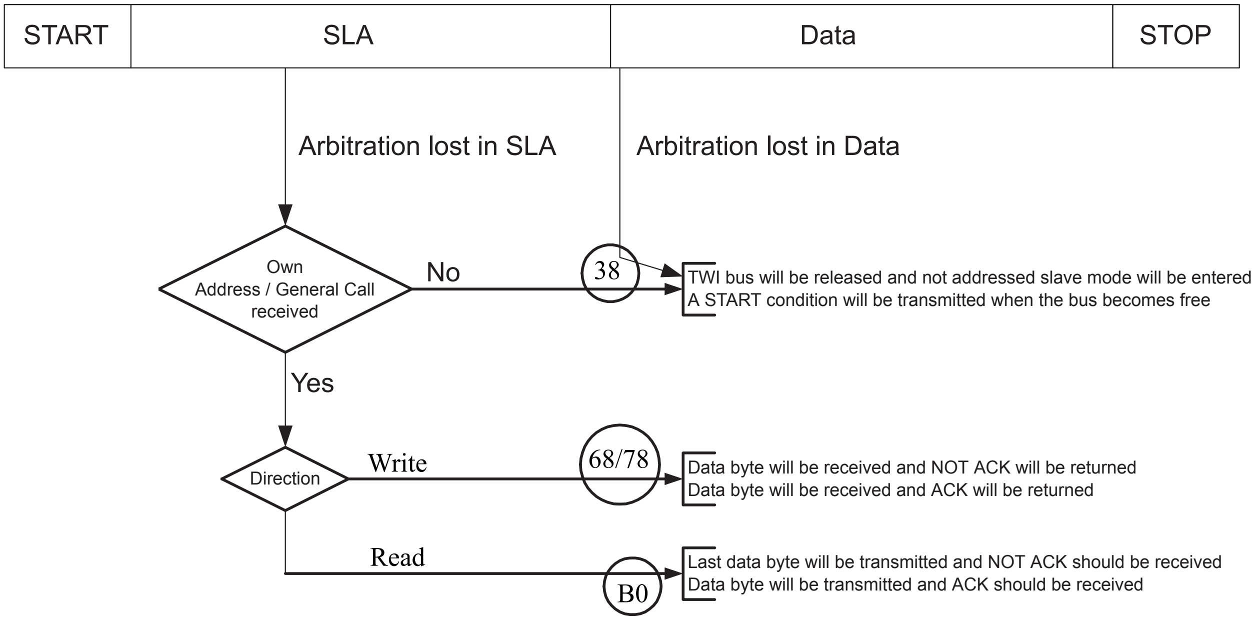
Bit 3 – CPOL0: Clock0 Polarity
When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 20-3 and Figure 20-4 for an example. The CPOL functionality is summarized below:

Table 20-3. CPOL0 Functionality

CPOL0	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

Bit 2 – CPHA0: Clock0 Phase
The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 20-3 and Figure 20-4 for an example. The CPHA functionality is summarized below:

Figure 23-21. Possible Status Codes Caused by Arbitration



23.9. Register Description

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADC Left Adjust Result bit ADMUX.ADLAR.

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, to ensure that the content of the Data Registers belongs to the same conversion: Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and a second conversion completes before ADCH is read, neither register is updated and the result from the second conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

Related Links

Power Management and Sleep Modes on page 56

Power Reduction Register on page 59

25.3. Starting a Conversion

A single conversion is started by writing a '0' to the Power Reduction ADC bit in the Power Reduction Register (PRR.PRADC), and writing a '1' to the ADC Start Conversion bit in the ADC Control and Status Register A (ADCSRA.ADSC). ADCS will stay high as long as the conversion is in progress, and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the ADC Auto Trigger Enable bit (ADCSRA.ADATE). The trigger source is selected by setting the ADC Trigger Select bits in the ADC Control and Status Register B (ADCSRB.ADTS). See the description of the ADCSRB.ADTS for a list of available trigger sources.

When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an interrupt flag will be set even if the specific interrupt is disabled or the Global Interrupt Enable bit in the AVR Status REgister (SREG.I) is cleared. A conversion can thus be triggered without causing an interrupt. However, the Interrupt Flag must be cleared in order to trigger a new conversion at the next interrupt event.

is switched on by writing the ADC Enable bit ADCSRA.ADEN to '1'. The prescaler keeps running for as long as ADEN=1, and is continuously reset when ADEN=0.

When initiating a single ended conversion by writing a '1' to the ADC Start Conversion bit (ADCSRA.ADSC), the conversion starts at the following rising edge of the ADC clock cycle.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (i.e., ADCSRA.ADEN is written to '1') takes 25 ADC clock cycles in order to initialize the analog circuitry.

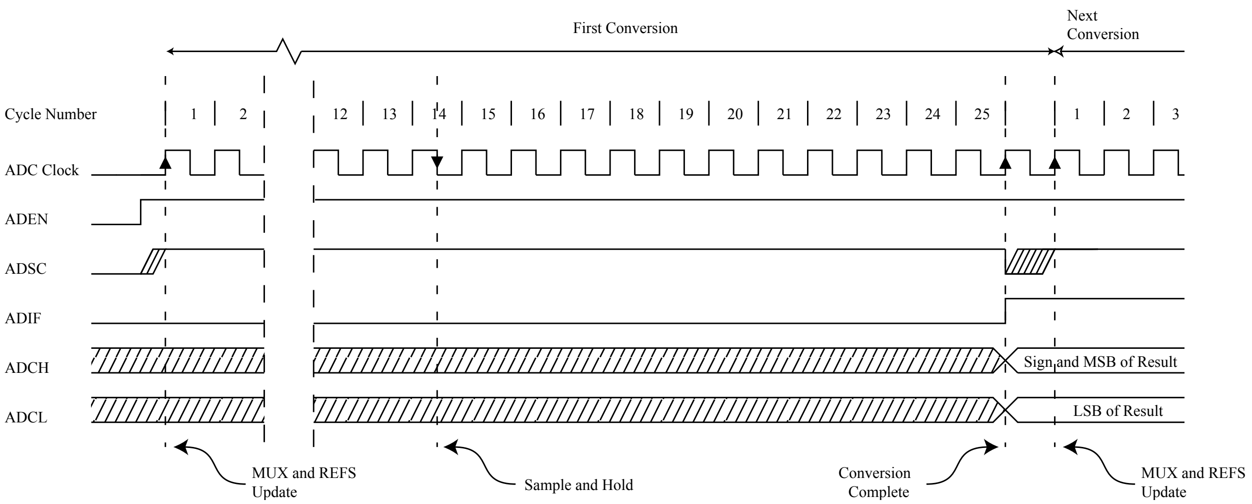
When the bandgap reference voltage is used as input to the ADC, it will take a certain time for the voltage to stabilize. If not stabilized, the first value read after the first conversion may be wrong.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of an first conversion. When a conversion is complete, the result is written to the ADC Data Registers (ADCL and ADCH), and the ADC Interrupt Flag (ADCSRA.ADIF) is set. In Single Conversion mode, ADCSRA.ADSC is cleared simultaneously. The software may then set ADCSRA.ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

When Auto Triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADCRSA.ADSC remains high. See also the ADC Conversion Time table below.

Figure 25-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)



If Auto Triggering is used, the exact time of the triggering event can be indeterministic. Special care must be taken when updating the ADMUX Register, in order to control which conversion will be affected by the new settings.

If both the ADC Auto Trigger Enable and ADC Enable bits (ADCRSA.ADATE, ADCRSA.ADEN) are written to '1', an interrupt event can occur at any time. If the ADMUX Register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

1. When ADATE or ADEN is cleared.
   1.1. During conversion, minimum one ADC clock cycle after the trigger event.
   1.2. After a conversion, before the Interrupt Flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next ADC conversion.

Special care should be taken when changing differential channels. Once a differential channel has been selected, the gain stage may take as much as 125 μs to stabilize to the new value. Thus conversions should not be started within the first 125 μs after selecting a new differential channel. Alternatively, conversion results obtained within this period should be discarded. The same settling time should be observed for the first differential conversion after changing ADC reference (by changing the REFS[1:0] bits in ADMUX).

25.5.1. ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

- In Single Conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.
- In Free Running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

The user is advised not to write new channel or reference selection values during Free Running mode.

When switching to a differential gain channel, the first conversion result may have a poor accuracy due to the required settling time for the automatic offset cancellation circuitry. The user should preferably disregard the first conversion result.

25.5.2. ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result in codes close to 0x3FF. V_{REF} can be selected as either AV_{CC}, internal 2.56V reference, or external AREF pin.

AV_{CC} is connected to the ADC through a passive switch. The internal 2.56V reference is generated from the internal bandgap reference (V_{BG}) through an internal amplifier. In either case, the external AREF pin is directly connected to the ADC, and the reference voltage can be made more immune to noise by connecting a capacitor between the AREF pin and ground. V_{REF} can also be measured at the AREF pin with a high impedance voltmeter. Note that V_{REF} is a high impedance source, and only a capacitive load should be connected in a system.

If the user has a fixed voltage source connected to the AREF pin, the user may not use the other reference voltage options in the application, as they will be shorted to the external voltage. If no external voltage is applied to the AREF pin, the user may switch between AV_{CC} and 2.56V as reference selection.

25.8.1. ADC Multiplexer Selection Register

Name: ADMUX
Offset: 0x7C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:6 – REFSn: Reference Selection [n = 1:0]
These bits select the voltage reference for the ADC. 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 25-3. ADC Voltage Reference Selection

REFS[1:0]	Voltage Reference Selection
00	AREF, Internal V_{ref} turned off
01	AV_{CC} with external capacitor at AREF pin
10	Internal 1.1V Voltage Reference with external capacitor at AREF pin
11	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Note: If differential channels are selected, only 2.56V should be used as Internal Voltage Reference.

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 conversions. For a complete description of this bit, see the ADCL and ADCH.

Bits 4:0 – MUXn: Analog Channel and Gain Selection Bits [n = 4:0]
The value of these bits selects which combination of analog inputs are connected to the ADC. These bits also select the gain for the differential channels. 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).

26.16.3. MCU Status Register

To make use of the Reset Flags to identify a reset condition, the user should read and then Reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.
When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: MCUSR
Offset: 0x54
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x34

Bit	7	6	5	4	3	2	1	0
				JTRF	WDRF	BORF	EXTRF	PORF
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit 4 – JTRF: JTAG Reset Flag
This bit is set if a reset is being caused by a logic one in the JTAG Reset Register selected by the JTAG instruction AVR_RESET. This bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

Bit 3 – WDRF: Watchdog System Reset Flag
This bit is set if a Watchdog System Reset occurs. The bit is reset by a Power-on Reset, or by writing a '0' to it.

Bit 2 – BORF: Brown-out Reset Flag
This bit is set if a Brown-out Reset occurs. The bit is reset by a Power-on Reset, or by writing a '0' to it.

Bit 1 – EXTRF: External Reset Flag
This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a '0' to it.

Bit 0 – PORF: Power-on Reset Flag
This bit is set if a Power-on Reset occurs. The bit is reset only by writing a '0' to it.

Table 27-2. Boot Reset Fuse

BOOTRST	Reset Address
1	Reset Vector = Application Reset (address 0x0000)
0	Reset Vector = Boot Loader Reset, as described by the Boot Loader Parameters

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

27.6. Boot Loader Lock Bits

If no Boot Loader capability is needed, the entire Flash is available for application code. The Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

The user can select:

- To protect the entire Flash from a software update by the MCU
- To protect only the Boot Loader Flash section from a software update by the MCU
- To protect only the Application Flash section from a software update by the MCU
- Allow software update in the entire Flash

The Boot Lock bits can be set in software and in Serial or Parallel Programming mode, but they can be cleared by a Chip Erase command only. The general Write Lock (Lock Bit mode 2) does not control the programming of the Flash memory by SPM instruction. Similarly, the general Read/Write Lock (Lock Bit mode 1) does not control reading nor writing by LPM/SPM, if it is attempted.

Table 27-3. Boot Lock Bit0 Protection Modes (Application Section)

BLB0 Mode	BLB02	BLB01	Protection
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.

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

Table 27-4. Boot Lock Bit1 Protection Modes (Boot Loader Section)

BLB1 Mode	BLB12	BLB11	Protection
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.

```
	lpm r0, Z+
	ld r1, Y+
	cpse r0, r1
	jmp Error
	sbiw loophi:looplo, 1 ;use subi for PAGESIZEB<=256
	brne Rdloop

	; return to RWW section
	; verify that RWW section is safe to read
Return:
	in temp1, SPMCSR
	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, SPMCSR
	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, EEPE
	rjmp Wait_ee
	; SPM timed sequence
	out SPMCSR, spmcrval
	spm
	; restore SREG (to enable interrupts if originally enabled)
	out SREG, temp2
```

Table 28-11. No. of Words in a Page and No. of Pages in the EEPROM

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATmega324PA	1Kbytes	4bytes	EEA[1:0]	256	EEA[9:2]	9

28.7. Parallel Programming Parameters, Pin Mapping, and Commands

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the device. Pulses are assumed to be at least 250ns unless otherwise noted.

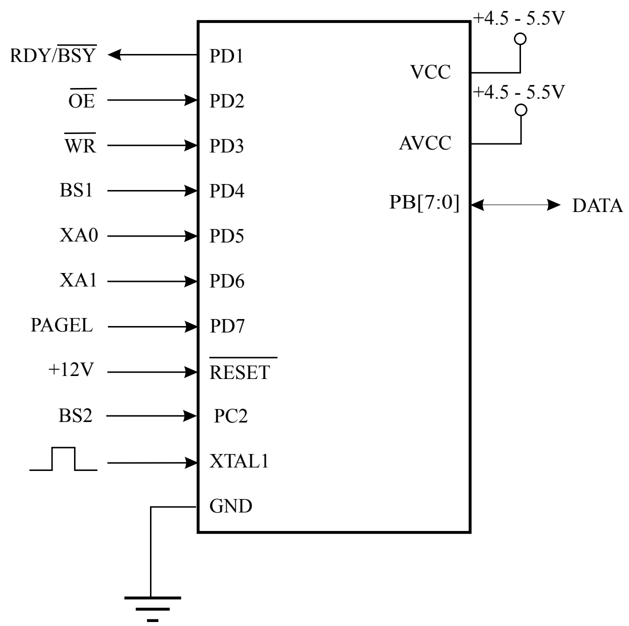
28.7.1. Signal Names

In this section, some pins of this device are referenced by signal names describing their functionality during parallel programming, please refer to Figure. Parallel Programming and Table. Pin Name Mapping in this section. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in the table, XA1 and XA0 Coding.

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different Commands are shown in the table, Command Byte Bit Coding Command Byte Command Executed.

Figure 28-1. Parallel Programming



Note: V_{CC} - 0.3V < AV_{CC} < V_{CC} + 0.3V, however, AV_{CC} should always be within 4.5 - 5.5V

Table 28-12. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/$\overline{BSY}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command
$\overline{OE}$	PD2	I	Output Enable (Active low)

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

28.8.8. Programming the Fuse Low Bits

The algorithm for programming the Fuse Low bits is as follows (Please refer to Programming the Flash for details on Command and Data loading):

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

28.8.9. Programming the Fuse High Bits

The algorithm for programming the Fuse High bits is as follows (Please refer to Programming the Flash for details on Command and Data loading):

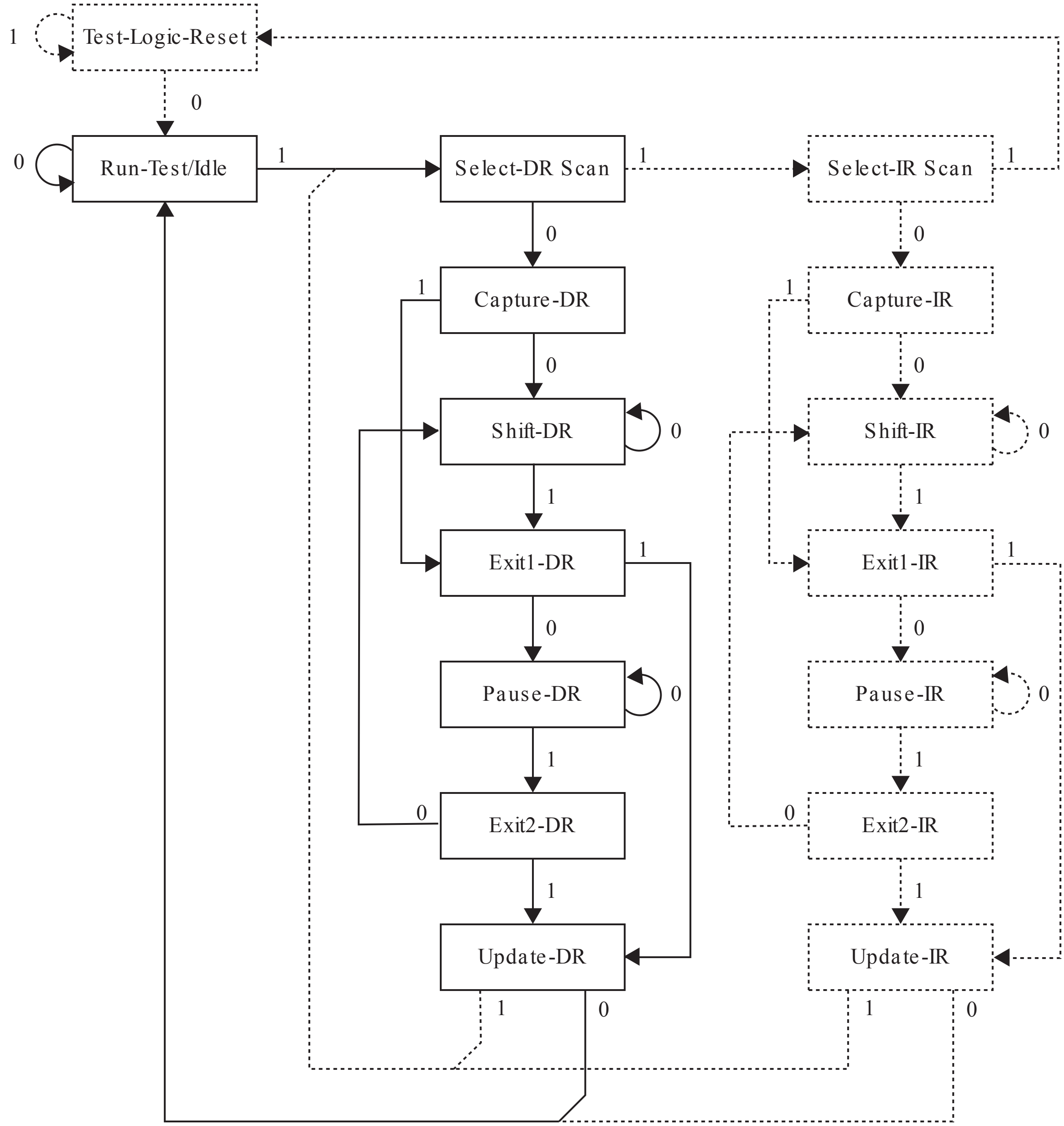
1. Step A: Load Command “0100 0000”.
2. Step C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
3. Set BS1 to “1” and BS2 to “0”. This selects high data byte.
4. Give WR a negative pulse and wait for RDY/BSY to go high.
5. Set BS1 to “0”. This selects low data byte.

28.8.10. Programming the Extended Fuse Bits

The algorithm for programming the Extended Fuse bits is as follows (Please refer to Programming the Flash for details on Command and Data loading):

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

Figure 28-12. State Machine Sequence for Changing/Reading the Data Word



28.10.11. Virtual Flash Page Load Register

The Virtual Flash Page Load Register is a virtual scan chain with length equal to the number of bits in one Flash page. Internally the Shift Register is 8-bit, and the data are automatically transferred to the Flash page buffer byte by byte. Shift in all instruction words in the page, starting with the LSB of the first instruction in the page and ending with the MSB of the last instruction in the page. This provides an efficient way to load the entire Flash page buffer before executing Page Write.

30.9. Pin Threshold and Hysteresis

Figure 30-25. I/O Pin Input Threshold vs. V_{CC} (V_{IH} , I/O Pin Read as ‘1’)

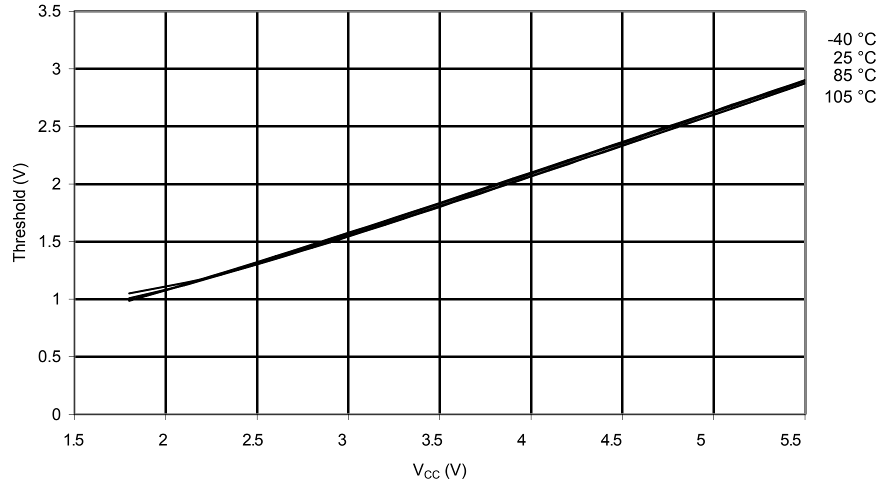


Figure 30-26. I/O Pin Input Threshold vs. V_{CC} (V_{IL}, I/O Pin Read as ‘0’)