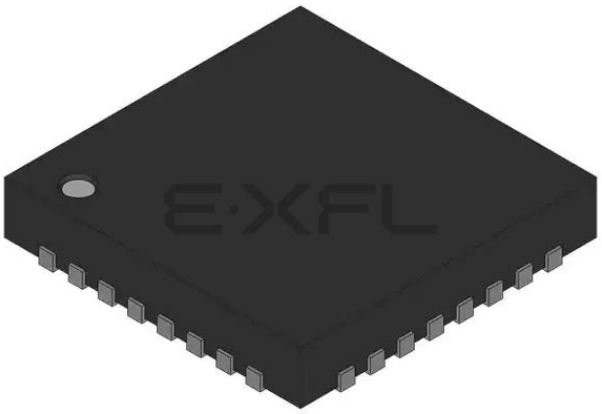


Welcome to E-XFL.COM

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.

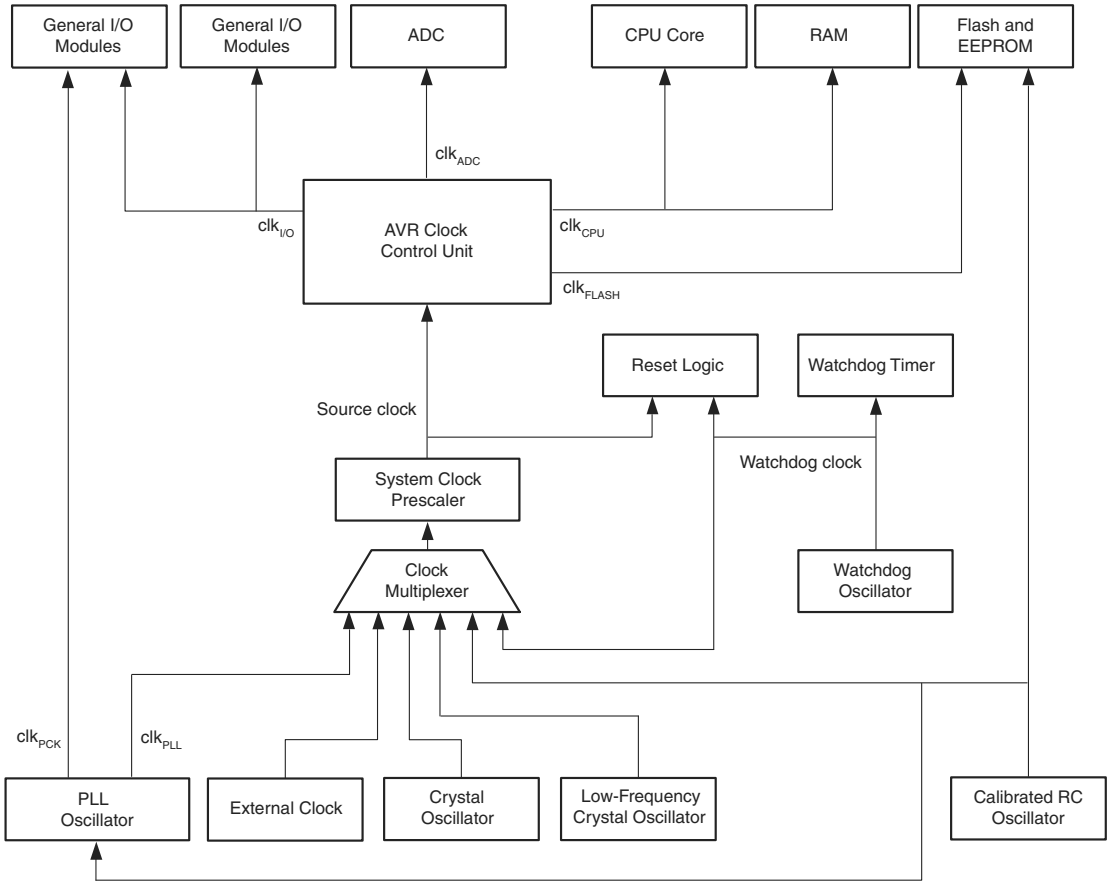
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	10MHz
Connectivity	USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	4KB (2K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-VQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/atmel/attiny461v-10mu

6. Clock System

Figure 6-1 presents the principal clock systems and their distribution in ATtiny261/461/861. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in “Power Management and Sleep Modes” on page 36.

Figure 6-1. Clock Distribution



6.1 Clock Subsystems

The clock subsystems are detailed in the sections below.

6.1.1 CPU Clock – clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the Data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

6.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counter. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted.

6.1.3 Flash Clock – clk_{FLASH}

The Flash clock controls operation of the Flash interface. The Flash clock is usually active simultaneously with the CPU clock.

6.1.4 ADC Clock – clk_{ADC}

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

6.1.5 Fast Peripheral Clock – clk_{PCK}

Selected peripherals can be clocked at a frequency higher than the CPU core. The fast peripheral clock is generated by an on-chip PLL circuit.

6.1.6 PLL System Clock – clk_{ADC}

The PLL can also be used to generate a system clock. The clock signal can be prescaled to avoid overclocking the CPU.

6.2 Clock Sources

The device has the following clock source options, selectable by Flash Fuse bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 6-1. Device Clocking Options Select(1) vs. PB4 and PB5 Functionality

Device Clocking Option	CKSEL3:0	PB4	PB5
External Clock (see page 26)	0000	XTAL1	I/O
High-Frequency PLL Clock (see page 26)	0001	I/O	I/O
Calibrated Internal 8 MHz Oscillator (see page 28)	0010	I/O	I/O
Internal 128 kHz Oscillator (see page 29)	0011	I/O	I/O
Low-Frequency Crystal Oscillator (see page 29)	01xx	XTAL1	XTAL2
Crystal Oscillator / Ceramic Resonator 0.4...0.9 MHz (see page 30)	1000 1001	XTAL1	XTAL2
Crystal Oscillator / Ceramic Resonator 0.9...3.0 MHz (see page 30)	1010 1011	XTAL1	XTAL2
Crystal Oscillator / Ceramic Resonator 3...8 MHz (see page 30)	1100 1101	XTAL1	XTAL2
Crystal Oscillator / Ceramic Resonator 8...20 MHz (see page 30)	1110 1111	XTAL1	XTAL2

Note: 1. For all fuses “1” means unprogrammed and “0” means programmed.

The various choices for each clocking option is given in the following sections. When the CPU wakes up from Power-down or Power-save, the selected clock source is used to time the start-up, ensuring stable oscillator operation before instruction execution starts. When the CPU starts from reset, there is an additional delay allowing the power to reach a stable level before com-

- **Port A, Bit 5 – ADC4/AIN2/PCINT5**
  - ADC4: Analog to Digital Converter, Channel 4.
  - AIN2: Analog Comparator Input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - PCINT5: Pin Change Interrupt source 5.
- **Port A, Bit 4 – ADC3/ICP0/PCINT4**
  - ADC3: Analog to Digital Converter, Channel 3.
  - ICP0: Timer/Counter0 Input Capture Pin.
  - PCINT4: Pin Change Interrupt source 4.
- **Port A, Bit 3 – AREF/PCINT3**
  - AREF: External analog reference for ADC. Pullup and output driver are disabled on PA3 when the pin is used as an external reference or internal voltage reference with external capacitor at the AREF pin.
  - PCINT3: Pin Change Interrupt source 3.
- **Port A, Bit 2 – ADC2/INT1/USCK/SCL/PCINT2**
  - ADC2: Analog to Digital Converter, Channel 2.
  - INT1: The PA2 pin can serve as an External Interrupt source 1.
  - USCK: Three-wire mode Universal Serial Interface Clock.
  - SCL: Two-wire mode Serial Clock for USI Two-wire mode.
  - PCINT2: Pin Change Interrupt source 2.
- **Port A, Bit 1 – ADC1/DO/PCINT1**
  - ADC1: Analog to Digital Converter, Channel 1.
  - DO: Three-wire mode Universal Serial Interface Data output. Three-wire mode Data output overrides PORTA1 value and it is driven to the port when data direction bit DDA1 is set. PORTA1 still enables the pull-up, if the direction is input and PORTA1 is set.
  - PCINT1: Pin Change Interrupt source 1.
- **Port A, Bit 0 – ADC0/DI/SDA/PCINT0**
  - ADC0: Analog to Digital Converter, Channel 0.
  - DI: Data Input in USI Three-wire mode. USI Three-wire mode does not override normal port functions, so pin must be configure as an input for DI function.
  - SDA: Two-wire mode Serial Interface Data.
  - PCINT0: Pin Change Interrupt source 0.

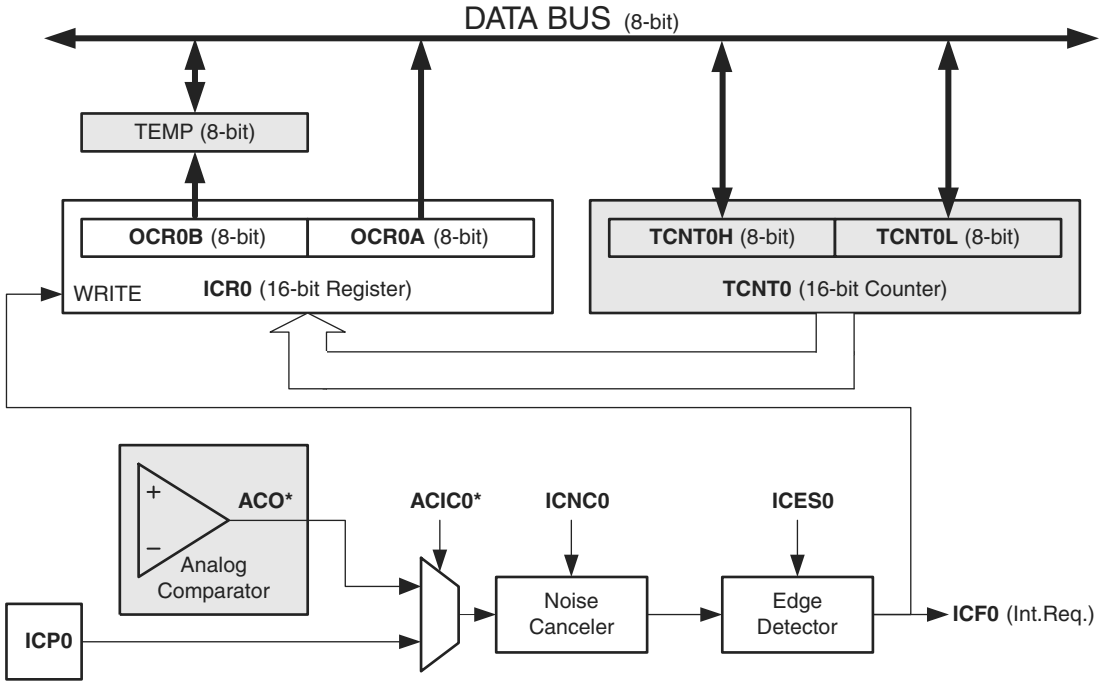
internal clock source, selected by the Clock Select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations. The Timer/Counter Overflow Flag (TOV0) is set when the counter reaches the maximum value and it can be used for generating a CPU interrupt.

11.5 Input Capture Unit

The Timer/Counter incorporates an Input Capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP0 pin or alternatively, via the analog-comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

The Input Capture unit is illustrated by the block diagram shown in Figure 11-4. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded.

Figure 11-4. Input Capture Unit Block Diagram



The Output Compare Register OCR0A is a dual-purpose register that is also used as an 8-bit Input Capture Register ICR0. In 16-bit Input Capture mode the Output Compare Register OCR0B serves as the high byte of the Input Capture Register ICR0. In 8-bit Input Capture mode the Output Compare Register OCR0B is free to be used as a normal Output Compare Register, but in 16-bit Input Capture mode the Output Compare Unit cannot be used as there are no free Output Compare Register(s). Even though the Input Capture register is called ICR0 in this section, it is refering to the Output Compare Register(s).

When a change of the logic level (an event) occurs on the *Input Capture pin* (ICP0), alternatively on the *Analog Comparator output* (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the value of the counter (TCNT0) is written to the *Input Capture Register* (ICR0). The *Input Capture Flag* (ICF0) is set at

if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

- **Bit 3 – OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable**

When the OCIE0B bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter Compare Match B interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter occurs, i.e., when the OCF0B bit is set in the Timer/Counter Interrupt Flag Register – TIFR0.

- **Bit 1 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

- **Bit 0 – TICIE0: Timer/Counter0, Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Input Capture interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 50.) is executed when the ICF0 flag, located in TIFR, is set.

11.10.8 TIFR – Timer/Counter0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x38 (0x58)	OCF1D	OCF1A	OCF1B	OCF0A	OCF0B	TOV1	TOV0	ICF0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 4 – OCF0A: Output Compare Flag 0 A**

The OCF0A bit is set when a Compare Match occurs between the Timer/Counter0 and the data in OCR0A – Output Compare Register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 Compare Match Interrupt Enable), and OCF0A are set, the Timer/Counter0 Compare Match Interrupt is executed.

The OCF0A is also set in 16-bit mode when a Compare Match occurs between the Timer/Counter and 16-bit data in OCR0B/A. The OCF0A is not set in Input Capture mode when the Output Compare Register OCR0A is used as an Input Capture Register.

- **Bit 3 – OCF0B: Output Compare Flag 0 B**

The OCF0B bit is set when a Compare Match occurs between the Timer/Counter and the data in OCR0B – Output Compare Register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter Compare B Match Interrupt Enable), and OCF0B are set, the Timer/Counter Compare Match Interrupt is executed.

The OCF0B is not set in 16-bit Output Compare mode when the Output Compare Register OCR0B is used as the high byte of the 16-bit Output Compare Register or in 16-bit Input Capture mode when the Output Compare Register OCR0B is used as the high byte of the Input Capture Register.

For actual placement of the I/O pins, refer to “Pinout ATtiny261/461/861 and ATtiny261V/461V/861V” on page 2. The device-specific I/O register and bit locations are listed in the “Register Description” on page 112.

12.2.1 Speed

The maximum speed of the Timer/Counter1 is 64 MHz. However, if a supply voltage below 2.7 volts is used, it is recommended to use the Low Speed Mode (LSM), because the Timer/Counter1 is not running fast enough on low voltage levels. In the Low Speed Mode the fast peripheral clock is scaled down to 32 MHz. For more details about the Low Speed Mode, see “PLLCSR – PLL Control and Status Register” on page 120.

12.2.2 Accuracy

The Timer/Counter1 is a 10-bit Timer/Counter module that can alternatively be used as an 8-bit Timer/Counter. The Timer/Counter1 registers are basically 8-bit registers, but on top of that there is a 2-bit High Byte Register (TC1H) that can be used as a common temporary buffer to access the two MSBs of the 10-bit Timer/Counter1 registers by the AVR CPU via the 8-bit data bus, if the 10-bit accuracy is used. Whereas, if the two MSBs of the 10-bit registers are written to zero the Timer/Counter1 is working as an 8-bit Timer/Counter. When reading the low byte of any 8-bit register the two MSBs are written to the TC1H register, and when writing the low byte of any 8-bit register the two MSBs are written from the TC1H register. Special procedures must be followed when accessing the 10-bit Timer/Counter1 values via the 8-bit data bus. These procedures are described in the section “Accessing 10-Bit Registers” on page 108.

12.2.3 Registers

The Timer/Counter (TCNT1) and Output Compare Registers (OCR1A, OCR1B, OCR1C and OCR1D) are 8-bit registers that are used as a data source to be compared with the TCNT1 contents. The OCR1A, OCR1B and OCR1D registers determine the action on the OC1A, OC1B and OC1D pins and they can also generate the compare match interrupts. The OCR1C holds the Timer/Counter TOP value, i.e. the clear on compare match value. The Timer/Counter1 High Byte Register (TC1H) is a 2-bit register that is used as a common temporary buffer to access the MSB bits of the Timer/Counter1 registers, if the 10-bit accuracy is used.

Interrupt request (overflow TOV1, and compare matches OCF1A, OCF1B, OCF1D and fault protection FPF1) signals are visible in the Timer Interrupt Flag Register (TIFR) and Timer/Counter1 Control Register D (TCCR1D). The interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK) and the FPIE1 bit in the Timer/Counter1 Control Register D (TCCR1D).

Control signals are found in the Timer/Counter Control Registers TCCR1A, TCCR1B, TCCR1C, TCCR1D and TCCR1E.

12.2.4 Synchronization

In asynchronous clocking mode the Timer/Counter1 and the prescaler allow running the CPU from any clock source while the prescaler is operating on the fast peripheral clock (PCK) having frequency of 64 MHz (or 32 MHz in Low Speed Mode). This is possible because there is a synchronization boundary between the CPU clock domain and the fast peripheral clock domain. Figure 12-2 shows Timer/Counter 1 synchronization register block diagram and describes synchronization delays in between registers. Note that all clock gating details are not shown in the figure.

The Timer/Counter1 register values go through the internal synchronization registers, which cause the input synchronization delay, before affecting the counter operation. The registers

counter value and so on. The definitions in Table 12-1 are used extensively throughout the document.

Table 12-1. Definitions

Constant	Description
BOTTOM	The counter reaches BOTTOM when it becomes 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 OCR0A Register. The assignment depends on the mode of operation

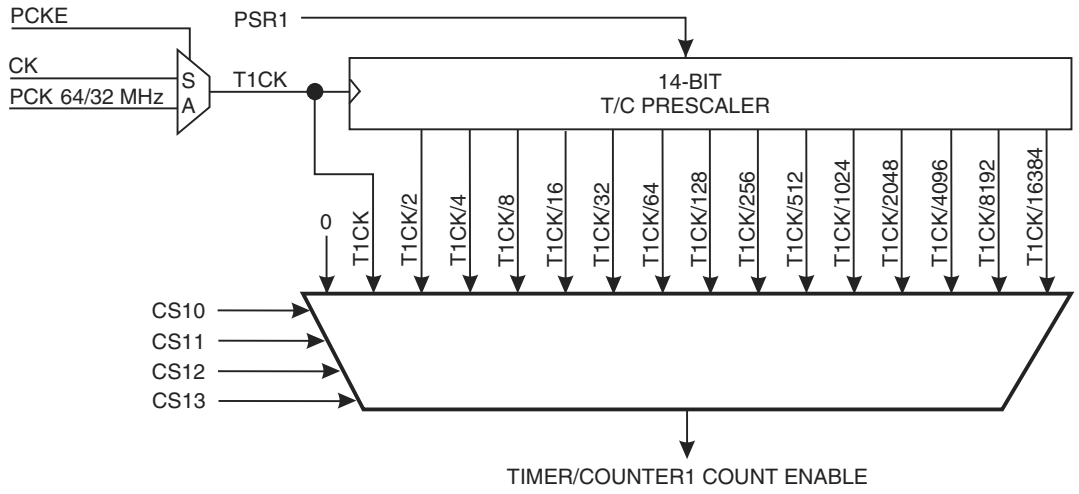
12.3 Clock Sources

The Timer/Counter is clocked internally, either from CK or PCK. See bits CSxx in Table 12-17 on page 116 and bit PCKE in “PLLCSR – PLL Control and Status Register” on page 120.

12.3.1 Prescaler

Figure 12-3 shows the Timer/Counter1 prescaler that supports two clocking modes, a synchronous clocking mode and an asynchronous clocking mode. The synchronous clocking mode uses the system clock (CK) as a clock timebase and asynchronous mode uses the fast peripheral clock (PCK) as a clock time base. The PCKE bit from the PLLCSR register enables the asynchronous mode when it is set (‘1’).

Figure 12-3. Timer/Counter1 Prescaler



In the asynchronous clocking mode the clock selections are from PCK to PCK/16384 and stop, and in the synchronous clocking mode the clock selections are from CK to CK/16384 and stop. The clock options are illustrated in Figure 12-3 and desribed in “TCCR1B – Timer/Counter1 Control Register B” on page 115.

The frequency of the fast peripheral clock is 64 MHz or 32 MHz in Low Speed mode (the LSM bit in PLLCSR register is set to one). The Low Speed Mode is recommended to use when the supply voltage below 2.7 volts are used.

The dedicated Dead Time prescaler in front of the Dead Time Generator can divide the Timer/Counter1 clock (PCK or CK) by 1, 2, 4 or 8 providing a large range of dead times that can be generated. The Dead Time prescaler is controlled by two bits DTPS11 and DTPS10 from the Dead Time Prescaler register. These bits define the division factor of the Dead Time prescaler. The division factors are given in Table 12-16.

Table 12-16. Division factors of the Dead Time prescaler

DTPS11	DTPS10	Prescaler divides the T/C1 clock by
0	0	1x (no division)
0	1	2x
1	0	4x
1	1	8x

- **Bits 3:0 – CS13, CS12, CS11, CS10: Clock Select Bits 3, 2, 1, and 0**

The Clock Select bits 3, 2, 1, and 0 define the prescaling source of Timer/Counter1.

Table 12-17. Timer/Counter1 Prescaler Select

CS13	CS12	CS11	CS10	Asynchronous Clocking Mode	Synchronous Clocking Mode
0	0	0	0	T/C1 stopped	T/C1 stopped
0	0	0	1	PCK	CK
0	0	1	0	PCK/2	CK/2
0	0	1	1	PCK/4	CK/4
0	1	0	0	PCK/8	CK/8
0	1	0	1	PCK/16	CK/16
0	1	1	0	PCK/32	CK/32
0	1	1	1	PCK/64	CK/64
1	0	0	0	PCK/128	CK/128
1	0	0	1	PCK/256	CK/256
1	0	1	0	PCK/512	CK/512
1	0	1	1	PCK/1024	CK/1024
1	1	0	0	PCK/2048	CK/2048
1	1	0	1	PCK/4096	CK/4096
1	1	1	0	PCK/8192	CK/8192
1	1	1	1	PCK/16384	CK/16384

The Stop condition provides a Timer Enable/Disable function.

12.12.7 TCNT1 – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	MSB							LSB	TCNT1
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	

This 8-bit register contains the value of Timer/Counter1.

The Timer/Counter1 is realized as a 10-bit up/down counter with read and write access. Due to synchronization of the CPU, Timer/Counter1 data written into Timer/Counter1 is delayed by one and half CPU clock cycles in synchronous mode and at most one CPU clock cycles for asynchronous mode. When a 10-bit accuracy is preferred, special procedures must be followed for accessing the 10-bit TCNT1 register via the 8-bit AVR data bus. These procedures are described in section “Accessing 10-Bit Registers” on page 108. Alternatively the Timer/Counter1 can be used as an 8-bit Timer/Counter. Note that the Timer/Counter1 always starts counting up after writing the TCNT1 register.

12.12.8 TC1H – Timer/Counter1 High Byte

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	-	-	-	-	-	-	TC19	TC18	TC1H
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The temporary Timer/Counter1 register is an 2-bit read/write register.

- **Bits 7:2 – Res: Reserved Bits**

These bits are reserved and always reads zero.

- **Bits 1:0 – TC19, TC18: Two MSB bits of the 10-bit accesses**

If 10-bit accuracy is used, the Timer/Counter1 High Byte Register (TC1H) is used for temporary storing the MSB bits (TC19, TC18) of the 10-bit acceses. The same TC1H register is shared between all 10-bit registers within the Timer/Counter1. Note that special procedures must be followed when accessing the 10-bit TCNT1 register via the 8-bit AVR data bus. These procedures are described in section “Accessing 10-Bit Registers” on page 108.

12.12.9 OCR1A – Timer/Counter1 Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	MSB							LSB	OCR1A
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	

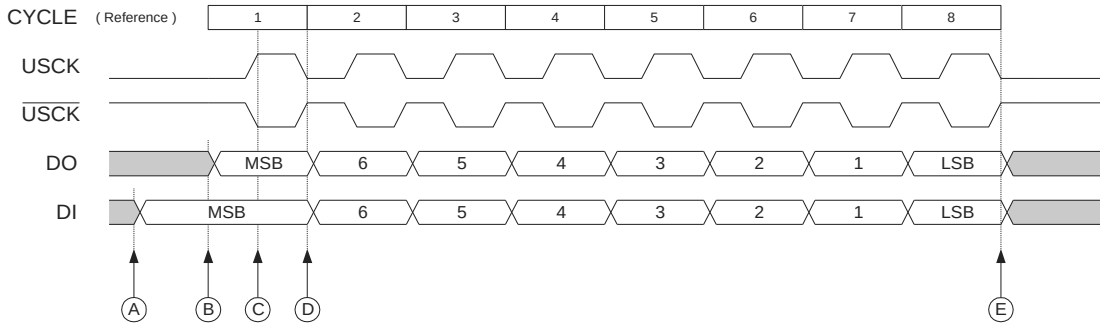
The output compare register A is an 8-bit read/write register.

The Timer/Counter Output Compare Register A contains data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in TCCR1A. A compare match does only occur if Timer/Counter1 counts to the OCR1A value. A software write that sets TCNT1 and OCR1A to the same value does not generate a compare match.

A compare match will set the compare interrupt flag OCF1A after a synchronization delay following the compare event.

Note that, if 10-bit accuracy is used special procedures must be followed when accessing the internal 10-bit Ouput Compare Registers via the 8-bit AVR data bus. These procedures are described in section “Accessing 10-Bit Registers” on page 108.

Figure 13-3. Three-wire Mode, Timing Diagram



The Three-wire mode timing is shown in Figure 13-3. At the top of the figure is a USCK cycle reference. One bit is shifted into the USI Data Register (USIDR) for each of these cycles. The USCK timing is shown for both external clock modes. In External Clock mode 0 (USICS0 = 0), DI is sampled at positive edges, and DO is changed (Data Register is shifted by one) at negative edges. In external clock mode 1 (USICS0 = 1) the opposite edges with respect to mode 0 are used. In other words, data is sampled at negative and output is changed at positive edges. The USI clock modes corresponds to the SPI data mode 0 and 1.

Referring to the timing diagram (Figure 13-3), a bus transfer involves the following steps:

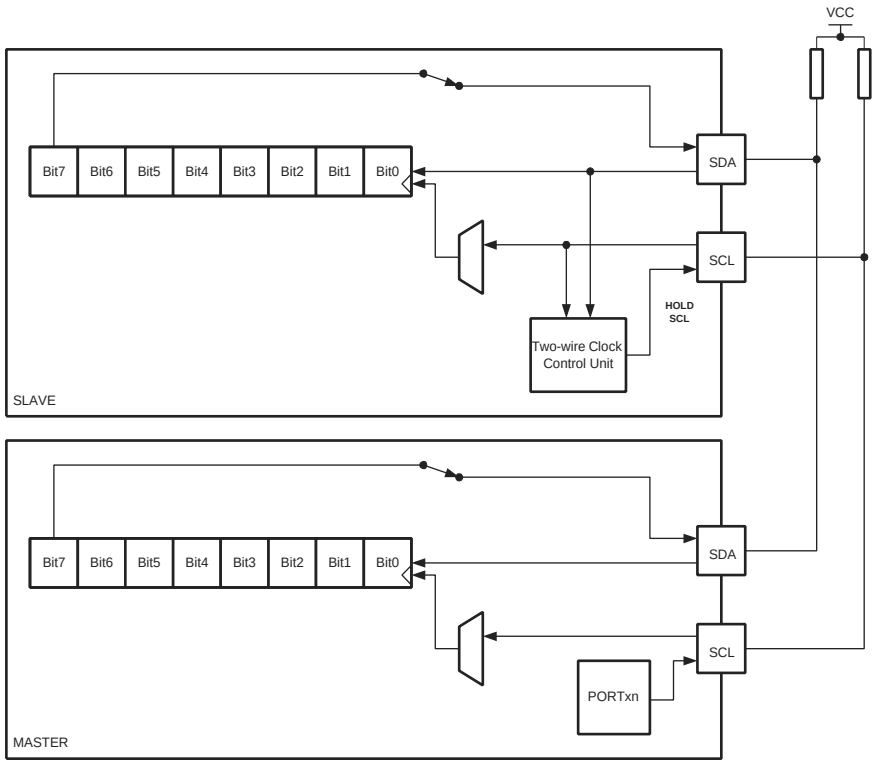
1. The slave and master devices set up their data outputs and, depending on the protocol used, enable their output drivers (mark A and B). The output is set up by writing the data to be transmitted to the USI Data Register. The output is enabled by setting the corresponding bit in the Data Direction Register of Port A. Note that there is not a preferred order of points A and B in the figure, but both must be at least one half USCK cycle before point C, where the data is sampled. This is in order to ensure that the data setup requirement is satisfied. The 4-bit counter is reset to zero.
2. The master software generates a clock pulse by toggling the USCK line twice (C and D). The bit values on the data input (DI) pins are sampled by the USI on the first edge (C), and the data output is changed on the opposite edge (D). The 4-bit counter will count both edges.
3. Step 2. is repeated eight times for a complete register (byte) transfer.
4. After eight clock pulses (i.e., 16 clock edges) the counter will overflow and indicate that the transfer has been completed. The data bytes transferred must now be processed before a new transfer can be initiated. The overflow interrupt will wake up the processor if it is set to Idle mode. Depending on the protocol used the slave device can now set its output to high impedance.

13.3.2 SPI Master Operation Example

The following code demonstrates how to use the USI module as a SPI Master:

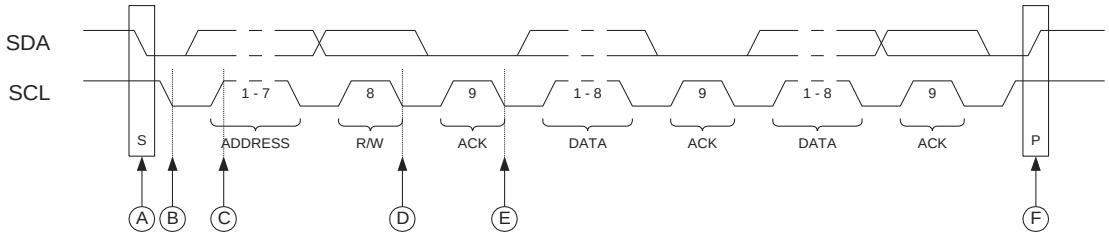
```
SPITransfer:
   sts    USIDR,r16
   ldi    r16,(1<<USIOIF)
   sts    USISR,r16
   ldi    r16,(1<<USIWM0)|(1<<USICS1)|(1<<USICLK)|(1<<USITC)
```

Figure 13-4. Two-wire Mode Operation, Simplified Diagram



The data direction is not given by the physical layer. A protocol, like the one used by the TWI-bus, must be implemented to control the data flow.

Figure 13-5. Two-wire Mode, Typical Timing Diagram



Referring to the timing diagram (Figure 13-5), a bus transfer involves the following steps:

1. The start condition is generated by the master by forcing the SDA low line while keeping the SCL line high (A). SDA can be forced low either by writing a zero to bit 7 of the USI Data Register, or by setting the corresponding bit in the PORTA register to zero. Note that the Data Direction Register bit must be set to one for the output to be enabled. The start detector logic of the slave device (see Figure 13-6 on page 131) detects the start condition and sets the USISIF Flag. The flag can generate an interrupt if necessary.
2. In addition, the start detector will hold the SCL line low after the master has forced a negative edge on this line (B). This allows the slave to wake up from sleep or complete other tasks before setting up the USI Data Register to receive the address. This is done by clearing the start condition flag and resetting the counter.

If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded will be lost.

17.3 Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write “00000101” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

Note: The CPU is halted during the Page Write operation.

17.4 Addressing the Flash During Self-Programming

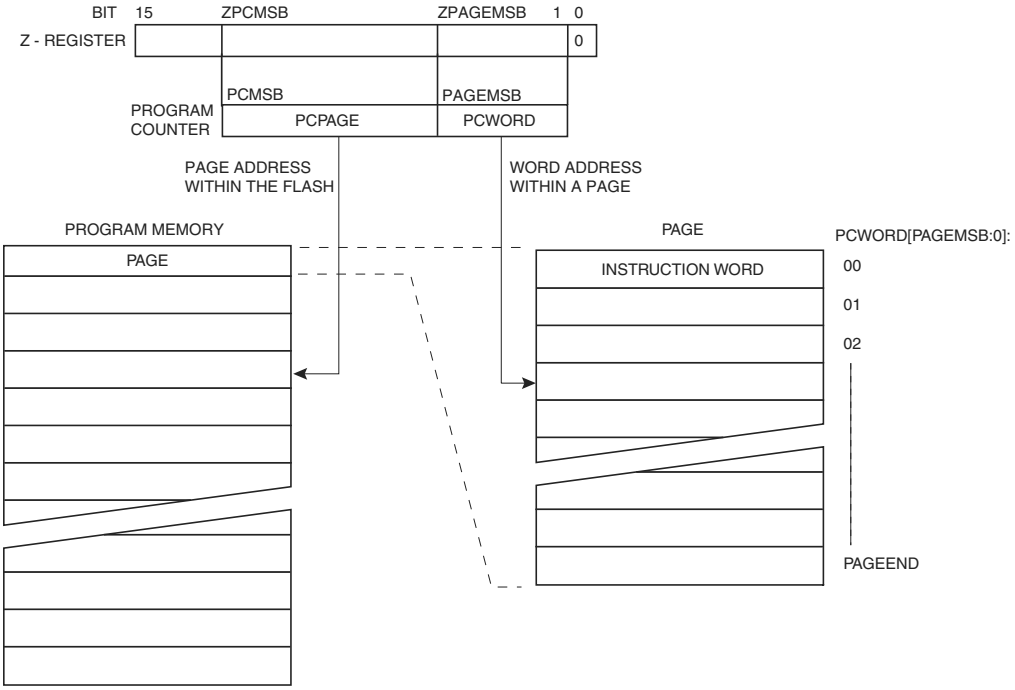
The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
ZL (R30)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
	7	6	5	4	3	2	1	0

Since the Flash is organized in pages (see Table 18-7 on page 173), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 17-1. Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the software addresses the same page in both the Page Erase and Page Write operation.

The LPM instruction uses the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

Figure 17-1. Addressing the Flash During SPM



Note: The different variables used in Figure 17-1 are listed in Table 18-7 on page 173.

17.5 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to Flash. Reading the Fuses and Lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEPE) in the EECR Register and verifies that the bit is cleared before writing to the SPMCSR Register.

17.6 Reading Fuse and Lock Bits from Software

It is possible for firmware to read device fuse and lock bits.

Note: Fuse and Lock bits that are programmed, will be read as zero. Fuse and Lock bits that are unprogrammed, will be read as one.

17.6.1 Reading Lock Bits from Firmware

Lock bit values are returned in the destination register after an LPM instruction has been issued within three CPU cycles after RFLB and SELFPRGEN bits have been set in SPMCSR. The RFLB and SELFPRGEN bits automatically clear upon completion of reading the lock bits, or if no LPM instruction is executed within three CPU cycles, or if no SPM instruction is executed within four CPU cycles. When RFLB and SELFPRGEN are cleared LPM functions normally.

To read the lock bits, follow the below procedure:

1. Load the Z-pointer with 0x0001.
2. Set RFLB and SELFPRGEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.
4. Read the lock bits from the LPM destination register.

If successful, the contents of the destination register are as follows.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	–	–	LB2	LB1

See section “Program And Data Memory Lock Bits” on page 170 for more information.

17.6.2 Reading Fuse Bits from Firmware

The algorithm for reading fuse bytes is similar to the one described above for reading lock bits, only the addresses are different. To read the Fuse Low Byte (FLB), follow the below procedure:

1. Load the Z-pointer with 0x0000.
2. Set RFLB and SELFPRGEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.
4. Read the FLB from the LPM destination register.

If successful, the contents of the destination register are as follows.

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

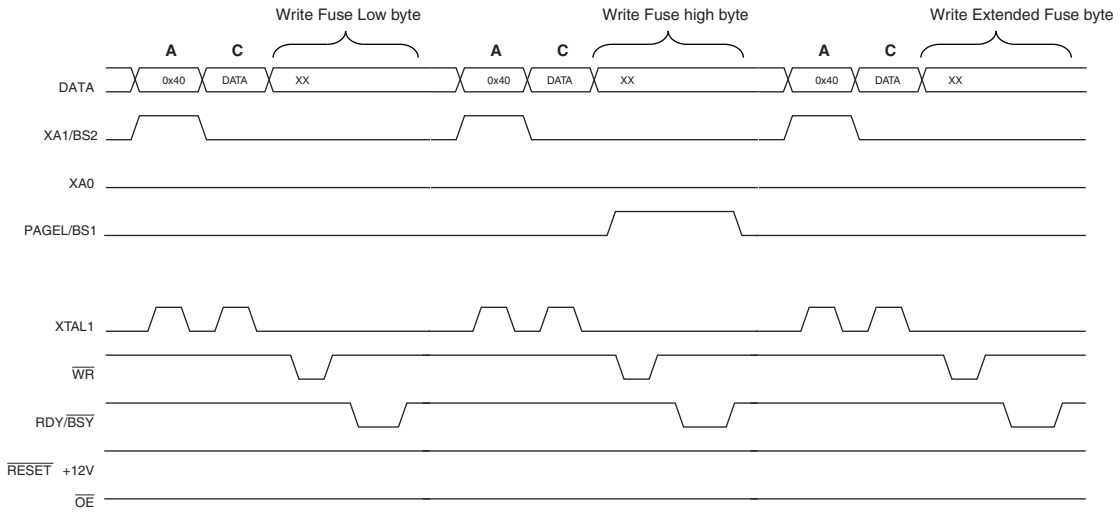
Refer to Table 18-5 on page 172 for a detailed description and mapping of the Fuse Low Byte.

Table 18-11. Serial Programming Instruction Set

Instruction/Operation	Instruction Format			
	Byte 1	**Byte 2**	**Byte 3**	**Byte4**
Programming Enable	$AC	$53	$00	$00
Chip Erase (Program Memory/EEPROM)	$AC	$80	$00	$00
Poll RDY/$\overline{\text{BSY}}$	$F0	$00	$00	data byte out
Load Instructions				
Load Extended Address byte(1)	$4D	$00	Extended adr	$00
Load Program Memory Page, High byte	$48	adr MSB	adr LSB	high data byte in
Load Program Memory Page, Low byte	$40	adr MSB	adr LSB	low data byte in
Load EEPROM Memory Page (page access)	$C1	$00	0000 000aa	data byte in
Read Instructions				
Read Program Memory, High byte	$28	adr MSB	adr LSB	high data byte out
Read Program Memory, Low byte	$20	adr MSB	adr LSB	low data byte out
Read EEPROM Memory	$A0	$00	00aa aaaa	data byte out
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(6)				
Write Program Memory Page	$4C	adr MSB	adr LSB	$00
Write EEPROM Memory	$C0	$00	00aa aaaa	data byte in
Write EEPROM Memory Page (page access)	$C2	$00	00aa aa00	$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. Instructions accessing program memory use a word address. This address may be random within the page range.
7. See http://www.atmel.com/avr for Application Notes regarding programming and programmers.

Figure 18-7. Programming the FUSES Waveforms



18.7.12 Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to “Programming the Flash” on page 180 for details on Command and Data loading):

1. A: Load Command “0010 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs the Lock bit. If LB mode 3 is programmed (LB1 and LB2 is programmed), it is not possible to program the Boot Lock bits by any External Programming mode.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

The Lock bits can only be cleared by executing Chip Erase.

18.7.13 Reading the Fuse and Lock Bits

The algorithm for reading the Fuse and Lock bits is as follows (refer to “Programming the Flash” on page 180 for details on Command loading):

1. A: Load Command “0000 0100”.
2. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “0”. The status of the Fuse Low bits can now be read at DATA (“0” means programmed).
3. Set $\overline{OE}$ to “0”, BS2 to “1” and BS1 to “1”. The status of the Fuse High bits can now be read at DATA (“0” means programmed).
4. Set OE to “0”, BS2 to “1”, and BS1 to “0”. The status of the Extended Fuse bits can now be read at DATA (“0” means programmed).
5. Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “1”. The status of the Lock bits can now be read at DATA (“0” means programmed).
6. Set $\overline{OE}$ to “1”.

19.6 ADC Characteristics

Table 19-7. ADC Characteristics, Single Ended Channels. T = -40°C to +85°C

Symbol	Parameter	Condition	Min(1)	Typ(1)	Max(1)	Units
	Resolution				10	Bits
	Absolute accuracy (Including INL, DNL, and Quantization, Gain and Offset Errors)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		2		LSB
		V_{REF} = 4V, V_{CC} = 4V, ADC clock = 1 MHz		3		LSB
		V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz Noise Reduction Mode		1.5		LSB
		V_{REF} = 4V, V_{CC} = 4V, ADC clock = 1 MHz Noise Reduction Mode		2.5		LSB
	Integral Non-Linearity (INL) (Accuracy after Offset and Gain Calibration)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		1		LSB
	Differential Non-linearity (DNL)	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		0.5		LSB
	Gain Error	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		2.5		LSB
	Offset Error	V_{REF} = 4V, V_{CC} = 4V, ADC clock = 200 kHz		1.5		LSB
	Conversion Time	Free Running Conversion	13		260	µs
	Clock Frequency		50		1000	kHz
AV_{CC}	Analog Supply Voltage		V_{CC} - 0.3		V_{CC} + 0.3	V
A_{REF}	External Voltage Reference	Single Ended Conversions	2.0		AVCC	V
		Differential Conversions	2.0		AVCC - 1.0	V
V_{IN}	Input Voltage	Single Ended Conversions	GND		V_{REF}	
		Differential Conversions	0		AV_{CC} (2)	V
	Input Bandwidth	Single Ended Conversions		38.5		kHz
		Differential Conversions		4		
V_{INT}	Internal 1.1V Reference		1.0	1.1	1.2	V
	Internal 2.56V Reference (2)		2.3	2.56	2.8	V
R_{REF}	Reference Input Resistance			35		kΩ
R_{AIN}	Analog Input Resistance			100		MΩ
	ADC Conversion Output		0		1023	LSB

Note: 1. Values are guidelines, only.

2. V_{DIFF} must be below V_{REF}.

19.7 Serial Programming Characteristics

Figure 19-4. Serial Programming Waveforms

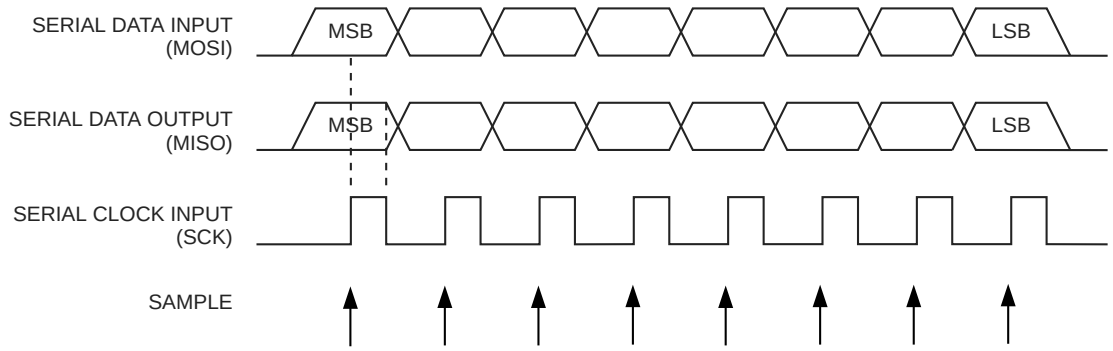


Figure 19-5. Serial Programming Timing

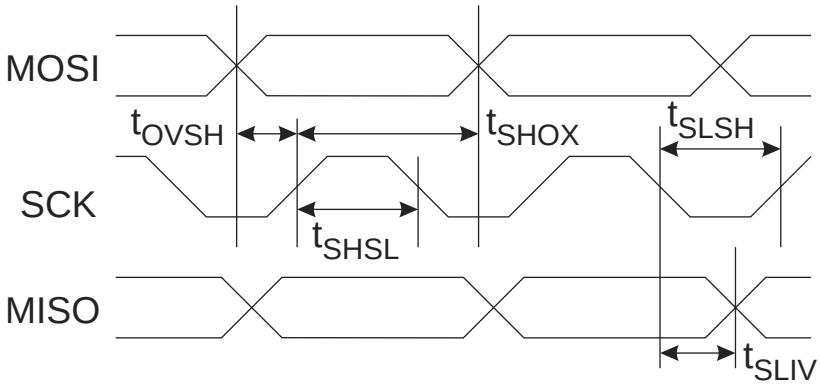


Table 19-8. Serial Programming Characteristics, T_A = -40°C to +85°C, V_{CC} = 1.8 - 5.5V (Unless Otherwise Noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Oscillator Frequency (ATtiny261V/461V/861V)	0		4	MHz
t_{CLCL}	Oscillator Period (ATtiny261V/461V/861V)	250			ns
$1/t_{CLCL}$	Oscillator Frequency (ATtiny261/461/861, V_{CC} = 4.5 - 5.5V)	0		20	MHz
t_{CLCL}	Oscillator Period (ATtiny261/461/861, V_{CC} = 4.5 - 5.5V)	50			ns
t_{SHSL}	SCK Pulse Width High	$2\ t_{CLCL}$ (1)			ns
t_{SLSH}	SCK Pulse Width Low	$2\ t_{CLCL}$ (1)			ns
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			ns
t_{SHOX}	MOSI Hold after SCK High	$2\ t_{CLCL}$			ns
t_{SLIV}	SCK Low to MISO Valid			100	ns

Note: 1. $2\ t_{CLCL}$ for f_{ck} < 12 MHz, $3\ t_{CLCL}$ for f_{ck} >= 12 MHz

Figure 20-8. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

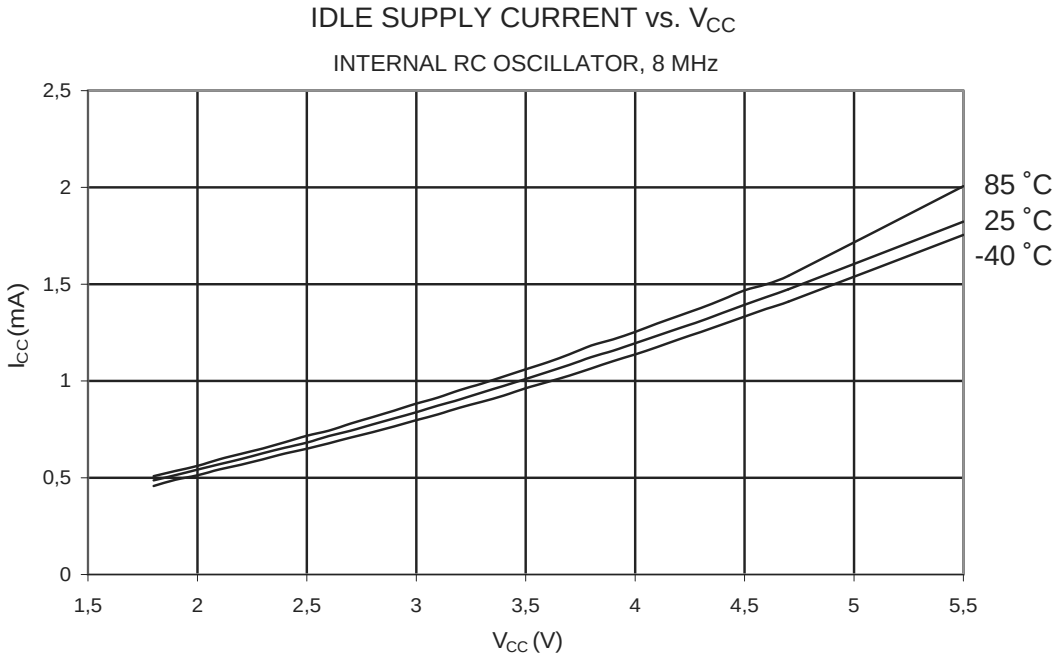


Figure 20-9. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

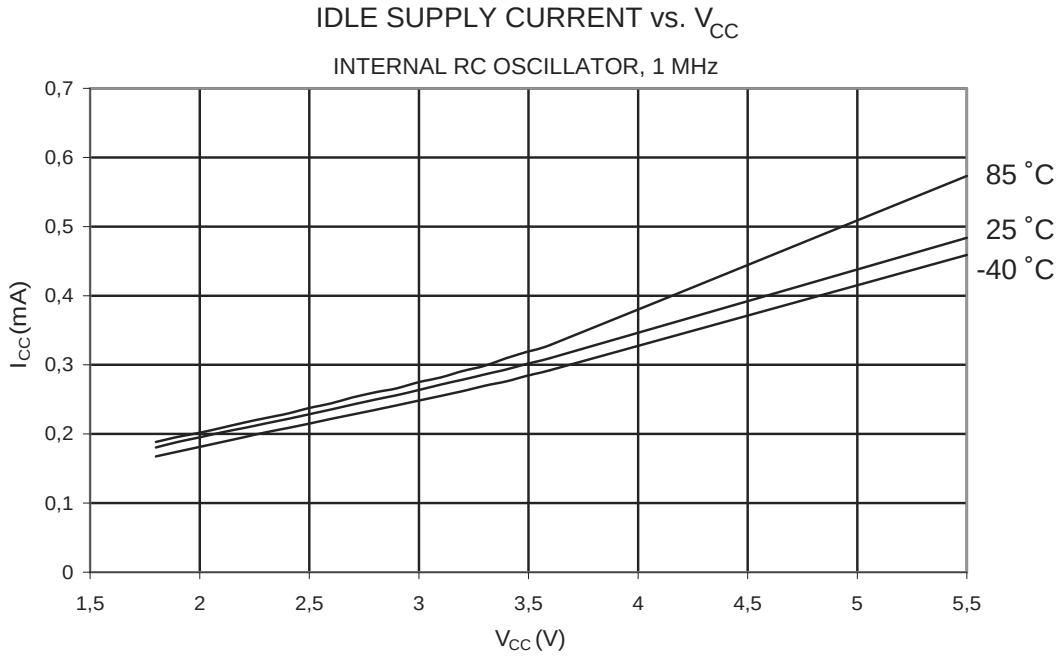


Figure 20-42. AREF External Reference Current vs. V_{CC}

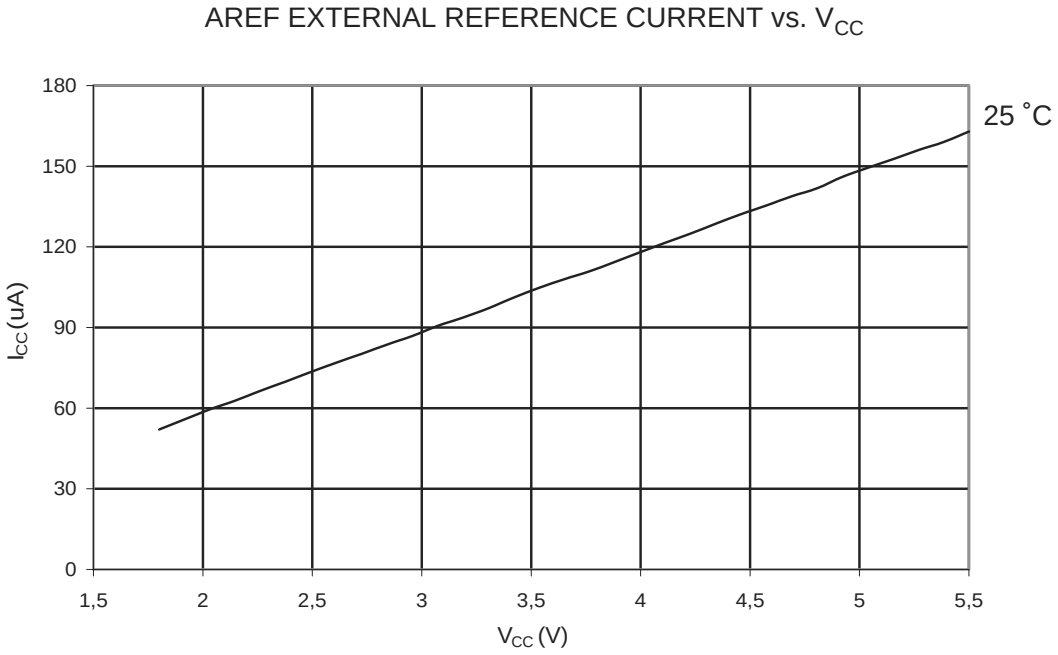


Figure 20-43. Analog Comparator vs. V_{CC}