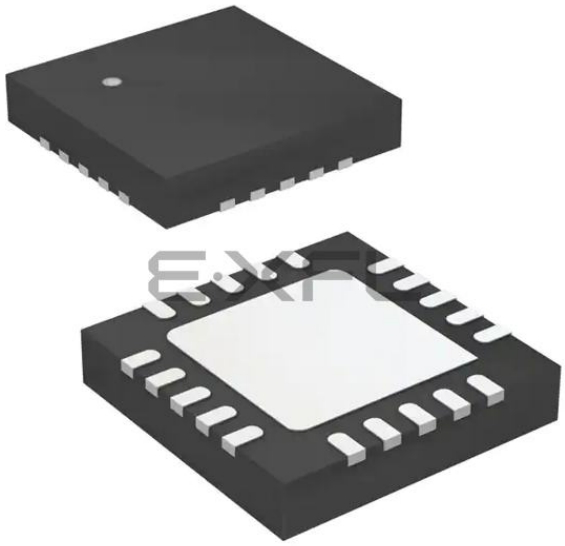


**Welcome to E-XFL.COM**

### What is "Embedded - Microcontrollers"?

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

### Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | AVR |
| Core Size | 8-Bit |
| Speed | 16MHz |
| Connectivity | USI |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 6 |
| Program Memory Size | 8KB (4K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 512 x 8 |
| RAM Size | 512 x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.5V |
| Data Converters | A/D 4x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 20-WFQFN Exposed Pad |
| Supplier Device Package | 20-QFN-EP (4x4) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/attiny85-15mz |

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the arithmetic logic unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The Atmel® ATtiny25/45/85 provides the following features: 2/4/8K byte of in-system programmable flash, 128/256/512 bytes EEPROM, 128/256/256 bytes SRAM, 6 general purpose I/O lines, 32 general purpose working registers, one 8-bit Timer/Counter with compare modes, one 8-bit high speed Timer/Counter, universal serial interface, internal and external interrupts, a 4-channel, 10-bit ADC, a programmable watchdog timer with internal oscillator, and three software selectable power saving modes. The idle mode stops the CPU while allowing the SRAM, Timer/Counter, ADC, analog comparator, and interrupt system to continue functioning. The power-down mode saves the register contents, disabling all chip functions until the next interrupt or hardware reset. The ADC noise reduction mode stops the CPU and all I/O modules except ADC, to minimize switching noise during ADC conversions.

The device is manufactured using Atmel's high density non-volatile memory technology. The on-chip ISP flash allows the program memory to be re-programmed in-system through an SPI serial interface, by a conventional non-volatile memory programmer or by an on-chip boot code running on the AVR core.

The ATtiny25/45/85 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

## 1.2 Automotive Quality Grade

The ATtiny25/45/85 have been developed and manufactured according to the most stringent requirements of the international standard ISO-TS-16949. This data sheet contains limit values extracted from the results of extensive characterization (temperature and voltage). The quality and reliability of the ATtiny25/45/85 have been verified during regular product qualification as per AEC-Q100 grade 1.

As indicated in the ordering information paragraph, the products are available in three different temperature grades, but with equivalent quality and reliability objectives. Different temperature identifiers have been defined as listed in Table 1-1.

**Table 1-1. Temperature Grade Identification for Automotive Products**

| Temperature | Temperature Identifier | Comments |
|---|---|---|
| –40; +85 | T | Similar to industrial temperature grade but with automotive quality |
| –40; +105 | T1 | Reduced automotive temperature range |
| –40; +125 | Z | Full automotive temperature range |

## 1.3 Pin Descriptions

### 1.3.1 VCC

Supply voltage.

### 1.3.2 GND

Ground.

### 1.3.3 Port B (PB5..PB0)

Port B is a 6-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the Atmel ATtiny25/45/85 as listed on Section 9.3.2 "Alternate Functions of Port B" on page 50.

### 1.3.4 $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table  on page 34. Shorter pulses are not guaranteed to generate a reset.

Assembly Code Example

```
in     r16, SREG     ; store SREG value
cli    ; disable interrupts during timed sequence
sbi    EECR, EEMWE   ; start EEPROM write
sbi    EECR, EEWE
out    SREG, r16     ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG;       /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example

```
sei    ; set Global Interrupt Enable
sleep  ; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

```
_SEI(); /* set Global Interrupt Enable */
_SLEEP(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

### 3.8.1 Interrupt Response Time

The interrupt execution response for all the enabled AVR® interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the program counter is pushed onto the stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the program counter (two bytes) is popped back from the stack, the stack pointer is incremented by two, and the I-bit in SREG is set.

# 6. Power Management and Sleep Modes

The high performance and industry leading code efficiency makes the AVR® microcontrollers an ideal choice for low power applications.

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

To enter any of the three sleep modes, the SE bit in MCUCR must be written to logic one and a SLEEP instruction must be executed. The SM1..0 bits in the MCUCR register select which sleep mode (idle, ADC noise reduction, or power-down) will be activated by the SLEEP instruction. See Table 6-1 for a summary. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the register file and SRAM are unaltered when the device wakes up from sleep. If a reset occurs during sleep mode, the MCU wakes up and executes from the reset vector.

Figure 5-1 on page 19 presents the different clock systems in the Atmel ATtiny25/45/85, and their distribution. The figure is helpful in selecting an appropriate sleep mode.

## 6.1 MCU Control Register – MCUCR

The MCU control register contains control bits for power management.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | BODS | PUD | SE | SM1 | SM0 | BODSE | ISC01 | ISC00 | MCUCR |
| 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 – BODS: BOD Sleep**

BOD disable functionality is available in some devices, only. See Section 6.5 "Limitations" on page 29.

In order to disable BOD during sleep (see Table 6-2 on page 29) the BODS bit must be written to logic one. This is controlled by a timed sequence and the enable bit, BODSE in MCUCR. First both BODS and BODSE must be set to one. Second, within four clock cycles, BODS must be set to one and BODSE must be set to zero. The BODS bit is active three clock cycles after it is set. A sleep instruction must be executed while BODS is active in order to turn off the BOD for the actual sleep mode. The BODS bit is automatically cleared after three clock cycles.

In devices where sleeping BOD has not been implemented this bit is unused and will always read zero.

- **Bit 5 – SE: Sleep Enable**

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer's purpose, it is recommended to write the sleep enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

- **Bits 4, 3 – SM1..0: Sleep Mode Select Bits 2..0**

These bits select between the three available sleep modes as shown in Table 6-1.

**Table 6-1. Sleep Mode Select**

| SM1 | SM0 | Sleep Mode |
|---|---|---|
| 0 | 0 | Idle |
| 0 | 1 | ADC noise reduction |
| 1 | 0 | Power-down |
| 1 | 1 | Stand-by mode |

- **Bit 2 – BODSE: BOD Sleep Enable**

BOD disable functionality is available in some devices, only. See Section 6.5 "Limitations" on page 29.

The BODSE bit enables setting of BODS control bit, as explained on BODS bit description. BOD disable is controlled by a timed sequence.

This bit is unused in devices where software BOD disable has not been implemented and will read as zero in those devices.

### 6.7.3 Brown-out Detector

If the brown-out detector is not needed in the application, this module should be turned off. If the brown-out detector is enabled by the BODLEVEL fuses, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to Section 7.5 “Brown-out Detection” on page 35 for details on how to configure the brown-out detector.

### 6.7.4 Internal Voltage Reference

The internal voltage reference will be enabled when needed by the brown-out detection, the analog comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to Section 7.8 “Internal Voltage Reference” on page 37 for details on the start-up time.

### 6.7.5 Watchdog Timer

If the watchdog timer is not needed in the application, this module should be turned off. If the watchdog timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to Section 7.9 “Watchdog Timer” on page 38 for details on how to configure the watchdog timer.

### 6.7.6 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ($clk_{I/O}$) and the ADC clock ($clk_{ADC}$) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the Section 9.2.5 “Digital Input Enable and Sleep Modes” on page 47 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or has an analog signal level close to $V_{CC}/2$, the input buffer will use excessive power.

For analog input pins, the digital input buffer should be disabled at all times. An analog signal level close to $V_{CC}/2$ on an input pin can cause significant current even in active mode. Digital input buffers can be disabled by writing to the digital input disable register (DIDR0). Refer to Section 16.3.1 “Digital Input Disable Register 0 – DIDR0” on page 100 for details.

# 7. System Control and Reset

## 7.1 Resetting the AVR

During reset, all I/O registers are set to their initial values, and the program starts execution from the reset vector. The instruction placed at the reset vector must be a RJMP – relative jump – instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 7-1 on page 33 shows the reset logic. Table  on page 34 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR® are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL fuses. The different selections for the delay period are presented in Section 5.2 “Clock Sources” on page 21.

## 7.2 Reset Sources

The Atmel® ATtiny25/45/85 has four sources of reset:

- Power-on reset. The MCU is reset when the supply voltage is below the power-on reset threshold ($V_{POT}$).
- External reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for longer than the minimum pulse length.
- Watchdog reset. The MCU is reset when the watchdog timer period expires and the watchdog is enabled.
- Brown-out reset. The MCU is reset when the supply voltage $V_{CC}$ is below the brown-out reset threshold ($V_{BOT}$) and the brown-out detector is enabled.

## 7.9 Watchdog Timer

The watchdog timer is clocked from an on-chip oscillator which runs at 128kHz. By controlling the watchdog timer prescaler, the watchdog reset interval can be adjusted as shown in Table 7-7 on page 40. The WDR – watchdog reset – instruction resets the watchdog timer. The watchdog timer is also reset when it is disabled and when a chip reset occurs. Ten different clock cycle periods can be selected to determine the reset period. If the reset period expires without another watchdog reset, the Atmel® ATtiny25/45/85 resets and executes from the reset vector. For timing details on the watchdog reset, refer to Table 7-7 on page 40.

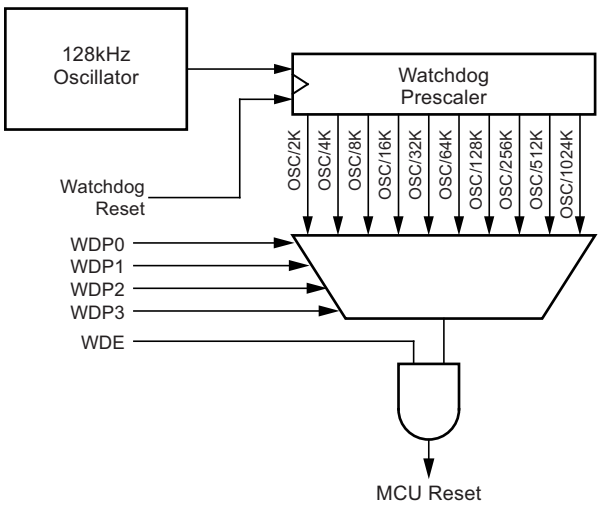
The watchdog timer can also be configured to generate an interrupt instead of a reset. This can be very helpful when using the watchdog to wake-up from power-down.

To prevent unintentional disabling of the watchdog or unintentional change of time-out period, two different safety levels are selected by the fuse WDTON as shown in Table 7-5 Refer to Section 7.10 “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 41 for details.

**Table 7-5. WDT Configuration as a Function of the Fuse Settings of WDTON**

| WDTON | Safety Level | WDT Initial State | How to Disable the WDT | How to Change Time-out |
|---|---|---|---|---|
| Unprogrammed | 1 | Disabled | Timed sequence | No limitations |
| Programmed | 2 | Enabled | Always enabled | Timed sequence |

**Figure 7-7. Watchdog Timer**




### 7.9.1 Watchdog Timer Control Register – WDTCR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | WDIF | WDIE | WDP3 | WDCE | WDE | WDP2 | WDP1 | WDP0 | WDTCR |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | |

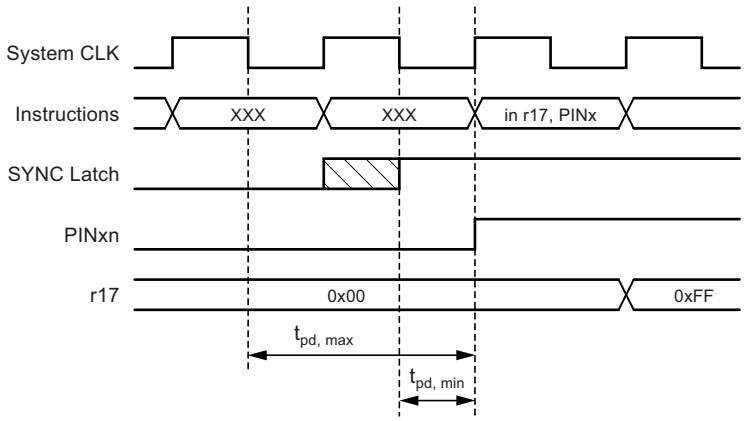
- **Bit 7 – WDIF: Watchdog Timeout Interrupt Flag**

This bit is set when a time-out occurs in the watchdog timer and the watchdog timer is configured for interrupt. WDIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, WDIF is cleared by writing a logic one to the flag. When the I-bit in SREG and WDIE are set, the watchdog time-out interrupt is executed.

### 9.2.4 Reading the Pin Value

Independent of the setting of data direction bit DDxn, the port pin can be read through the PINxn register bit. As shown in Figure 9-2 on page 44, the PINxn register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 9-3 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

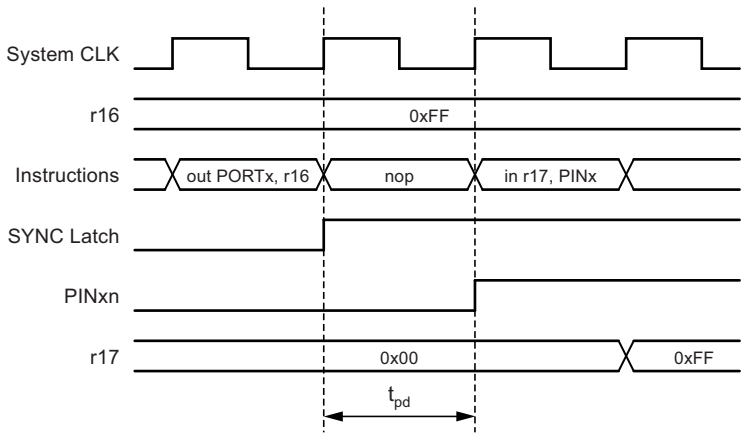
**Figure 9-3. Synchronization when Reading an Externally Applied Pin Value**




Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn register at the succeeding positive clock edge. As indicated by the two arrows tpd,max and tpd,min, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 9-4. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is one system clock period.

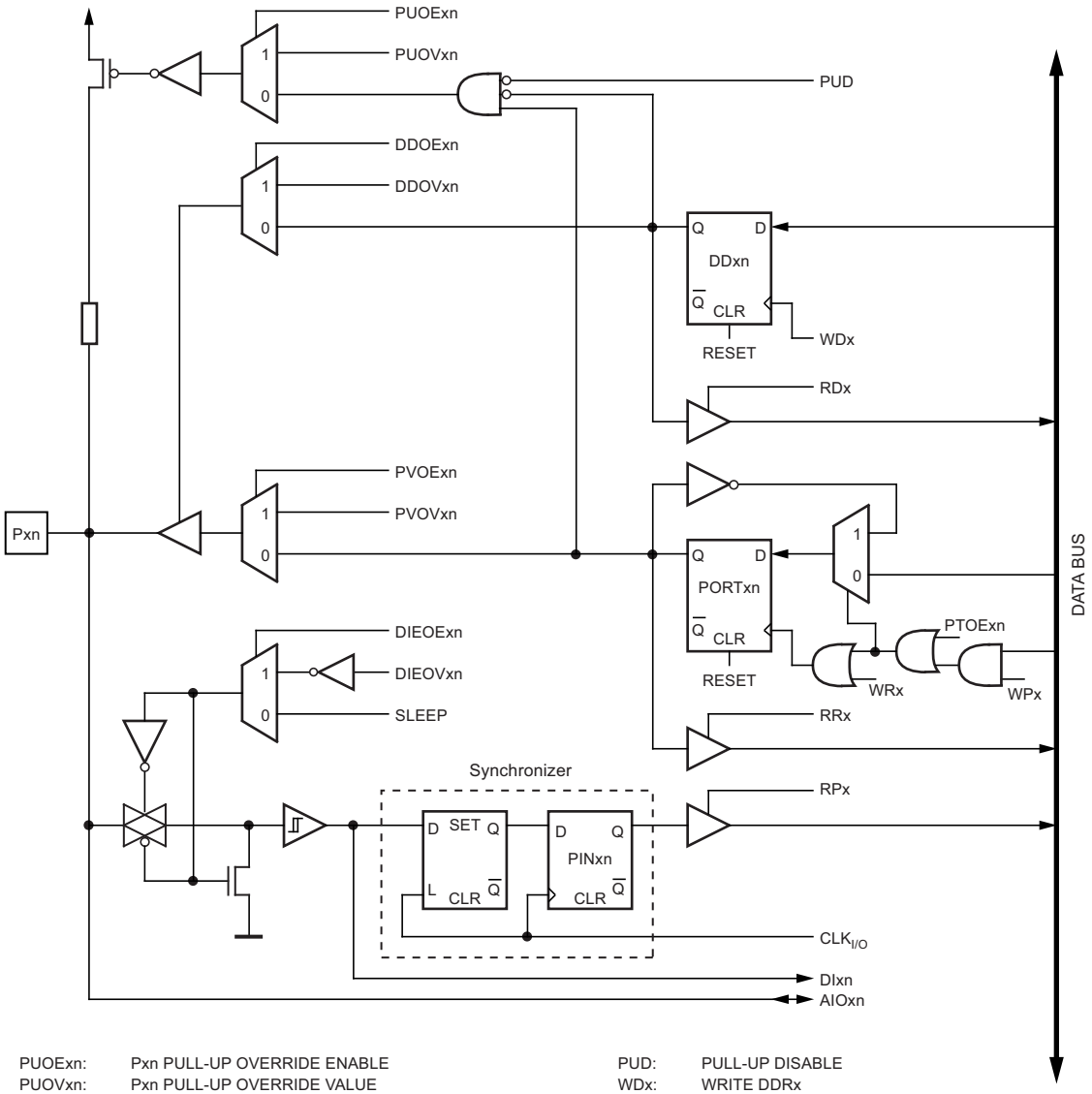
**Figure 9-4. Synchronization when Reading a Software Assigned Pin Value**



## 9.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 9-5 shows how the port pin control signals from the simplified Figure 9-2 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR® microcontroller family.

**Figure 9-5. Alternate Port Functions(1)**




| | |
|---|---|
| PUOExn: | Pxn PULL-UP OVERRIDE ENABLE |
| PUOVxn: | Pxn PULL-UP OVERRIDE VALUE |
| DDOExn: | Pxn DATA DIRECTION OVERRIDE ENABLE |
| DDOVxn: | Pxn DATA DIRECTION OVERRIDE VALUE |
| PVOExn: | Pxn PORT VALUE OVERRIDE ENABLE |
| PVOVxn: | Pxn PORT VALUE OVERRIDE VALUE |
| DIEOExn: | Pxn DIGITAL INPUT ENABLE OVERRIDE ENABLE |
| DIEOVxn: | Pxn DIGITAL INPUT ENABLE OVERRIDE VALUE |
| SLEEP: | SLEEP CONTROL |
| PTOExn: | Pxn, PORT TOGGLE OVERRIDE ENABLE |

| | |
|---|---|
| PUD: | PULL-UP DISABLE |
| WDx: | WRITE DDRx |
| RDx: | READ DDRx |
| RRx: | READ PORTx REGISTER |
| WRx: | WRITE PORTx |
| RPx: | READ PORTx PIN |
| WPx: | WRITE PINx |
| CLK:I/O | I/O CLOCK |
| DIxn: | DIGITAL INPUT PIN n ON PORTx |
| AIOxn: | ANALOG INPUT/OUTPUT PIN n ON PORTx |

Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

AREF: External analog reference for ADC. Pull up and output driver are disabled on PB0 when the pin is used as an external reference or internal voltage reference with external capacitor at the AREF pin.

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.

PCINT0: Pin change interrupt source 0.

Table 9-4 and Table 9-5 relate the alternate functions of port B to the overriding signals shown in Figure 9-5 on page 48.

**Table 9-4. Overriding Signals for Alternate Functions in PB5..PB3**

| Signal Name | PB5/RESET/ADC0/PCINT5 | PB4/ADC2/XTAL2/OC1B/PCINT4 | PB3/ADC3/XTAL1/_OC1B/PCINT3 |
|---|---|---|---|
| PUOE | $\overline{\text{RSTDISBL}}^{(1)} \times \text{DWEN}^{(1)}$ | 0 | 0 |
| PUOV | 1 | 0 | 0 |
| DDOE | $\text{RSTDISBL}^{(1)} \times \text{DWEN}^{(1)}$ | 0 | 0 |
| DDOV | debugWire transmit | 0 | 0 |
| PVOE | 0 | OC1B enable | _OC1B enable |
| PVOV | 0 | OC1B | _OC1B |
| PTOE | 0 | 0 | 0 |
| DIEOE | $\overline{\text{RSTDISBL}}^{(1)}$ + (PCINT5 × PCIE + ADC0D) | PCINT4 × PCIE + ADC2D | PCINT3 × PCIE + ADC3D |
| DIEOV | ADC0D | ADC2D | ADC3D |
| DI | PCINT5 input | PCINT4 input | PCINT3 input |
| AIO | RESET input, ADC0 input | ADC2 input | ADC3 input |

Note: 1. 1 when the fuse is “0” (programmed).

**Table 9-5. Overriding Signals for Alternate Functions in PB3..PB0**

| Signal Name | PB2/SCK/ADC1/T0/ USCK/SCL/INT0/PCINT2 | PB1/MISO/DO/AIN1/ OC1A/OC0B/PCINT1 | PB0/MOSI/DI/SDA/AIN0/AREF/_OC1A/OC0A/PCINT0 |
|---|---|---|---|
| PUOE | 0 | 0 | 0 |
| PUOV | 0 | 0 | 0 |
| DDOE | USI_TWO_WIRE | 0 | USI_TWO_WIRE |
| DDOV | (USI_SCL_HOLD + $\overline{\text{PORTB2}}$) × DDB2 | 0 | ($\overline{\text{SDA}}$ + $\overline{\text{PORTBO}}$) × DDB0 |
| PVOE | USI_TWO_WIRE × DDB2 | OC0B enable + OC1A enable + USI_THREE_WIRE | OC0A enable + _OC1A enable + (USI_TWO_WIRE × DDB0) |
| PVOV | 0 | OC0B + OC1A + DO | OC0A + _OC1A |
| PTOE | USITC | 0 | 0 |
| DIEOE | PCINT2 × PCIE + ADC1D + USISIE | PCINT1 × PCIE + AIN1D | PCINT0 × PCIE + AIN0D + USISIE |
| DIEOV | ADC1D | AIN1D | AIN0D |
| DI | T0/USCK/SCL/INT0/ PCINT2 input | PCINT1 input | DI/SDA/PCINT0 input |
| AIO | ADC1 input | Analog comparator negative input | Analog comparator positive input |

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM0A1:0 = 1). The OC0A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCnx} = \frac{f_{clk\ I/O}}{2 \times N \times (1 + OCRnx)}$$

The *N* variable represents the prescale factor (1, 8, 64, 256, or 1024).

As for the normal mode of operation, the TOV0 flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

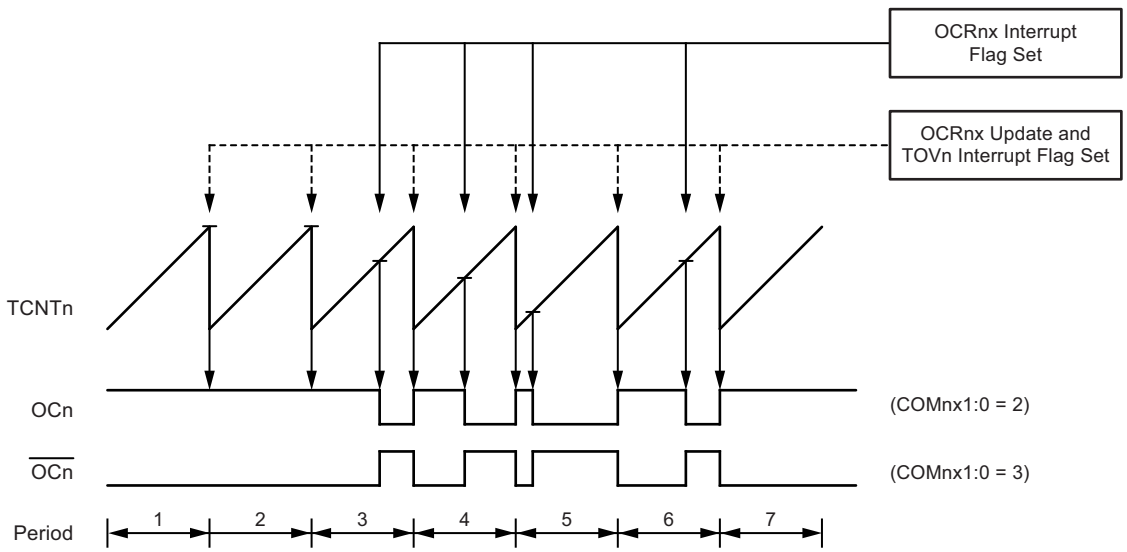
### 11.6.3 Fast PWM Mode

The fast pulse width modulation or fast PWM mode (WGM02:0 = 3 or 7) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. TOP is defined as 0xFF when WGM2:0 = 3, and OCR0A when WGM2:0 = 7. In non-inverting compare output mode, the output compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at BOTTOM. In inverting compare output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that use dual-slope operation.

This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

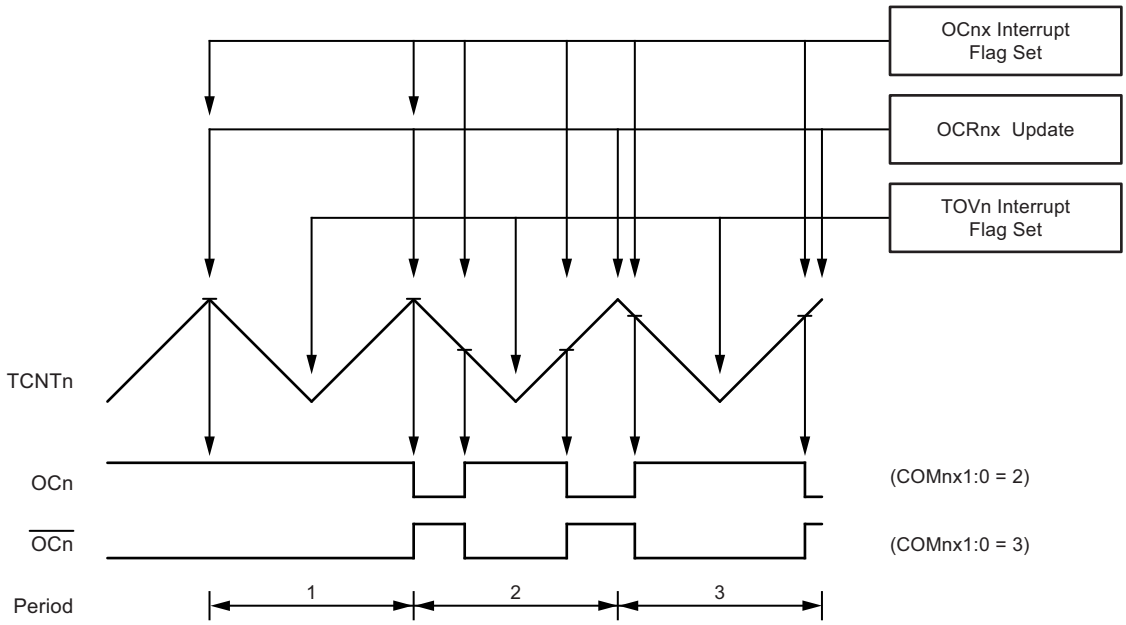
In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 11-6 on page 62. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

**Figure 11-6. Fast PWM Mode, Timing Diagram**




The Timer/Counter overflow flag (TOV0) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

**Figure 11-7. 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 OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0x1:0 to three: Setting the COM0A0 bits to one allows the OC0A pin to toggle on compare matches if the WGM02 bit is set. This option is not available for the OC0B pin (See Table 11-4 on page 67). The actual OC0x 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 OC0x register at the compare match between OCR0x and TCNT0 when the counter increments, and setting (or clearing) the OC0x register at compare match between OCR0x and TCNT0 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \times 510}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in Figure 11-7 on page 64 OCn has a transition from high to low even though there is no compare match. The point of this transition is to guaratee symmetry around BOTTOM. There are two cases that give a transition without compare match.

- OCR0A changes its value from MAX, like in Figure 11-7 on page 64. When the OCR0A value is MAX the OCn pin value is the same as the result of a down-counting compare match. To ensure symmetry around BOTTOM the OCn value at MAX must correspond to the result of an up-counting compare match.
- The timer starts counting from a value higher than the one in OCR0A, and for that reason misses the compare match and hence the OCn change that would have happened on the way up.

### 13.1.8 Timer/Counter Interrupt Flag Register - TIFR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $38 ($58) | - | OCF1A | OCF1B | OCF0A | OCF0B | TOV1 | TOV0 | - | TIFR |
| Read/Write | R | R/W | R/W | R | R | R/W | R/W | R | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7 - Res: Reserved Bit**

This bit is a reserved bit in the Atmel® ATtiny25/45/85 and always reads as zero.

- **Bit 6 - OCF1A: Output Compare Flag 1A**

The OCF1A bit is set (one) when compare match occurs between Timer/Counter1 and the data value in OCR1A - output compare register 1A. OCF1A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1A is cleared, after synchronization clock cycle, by writing a logic one to the flag. When the I-bit in SREG, OCIE1A, and OCF1A are set (one), the Timer/Counter1 A compare match interrupt is executed.

- **Bit 5 - OCF1B: Output Compare Flag 1B**

The OCF1B bit is set (one) when compare match occurs between Timer/Counter1 and the data value in OCR1B - output compare register 1A. OCF1B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1B is cleared, after synchronization clock cycle, by writing a logic one to the flag. When the I-bit in SREG, OCIE1B, and OCF1B are set (one), the Timer/Counter1 B compare match interrupt is executed.

- **Bit 2 - TOV1: Timer/Counter1 Overflow Flag**

In normal mode (PWM1A=0 and PWM1B=0) the bit TOV1 is set (one) when an overflow occurs in Timer/Counter1. The bit TOV1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV1 is cleared, after synchronization clock cycle, by writing a logical one to the flag.

In PWM mode (either PWM1A=1 or PWM1B=1) the bit TOV1 is set (one) when compare match occurs between Timer/Counter1 and data value in OCR1C - output compare register 1C. Clearing the Timer/Counter1 with the bit CTC1 does not generate an overflow.

When the SREG I-bit, and TOIE1 (Timer/Counter1 overflow interrupt enable), and TOV1 are set (one), the Timer/Counter1 overflow interrupt is executed.

- **Bit 0 - Res: Reserved Bit**

This bit is a reserved bit in the Atmel ATtiny25/45/85 and always reads as zero.

### 13.1.9 PLL Control and Status Register - PLLCSR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $27 ($27) | LSM | - | - | - | - | PCKE | PLLE | PLOCK | PLLCSR |
| Read/Write | R/W | R | R | R | R | R/W | R/W | R | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0/1 | 0 | |

- **Bit 7- LSM: Low Speed Mode**

The high speed mode is enabled as default and the fast peripheral clock is 64MHz, but the low speed mode can be set by writing the LSM bit to one. Then the fast peripheral clock is scaled down to 32MHz. The low speed mode must be set, if the supply voltage is below 2.7 volts, because the Timer/Counter1 is not running fast enough on low voltage levels. It is highly recommended that Timer/Counter1 is stopped whenever the LSM bit is changed.

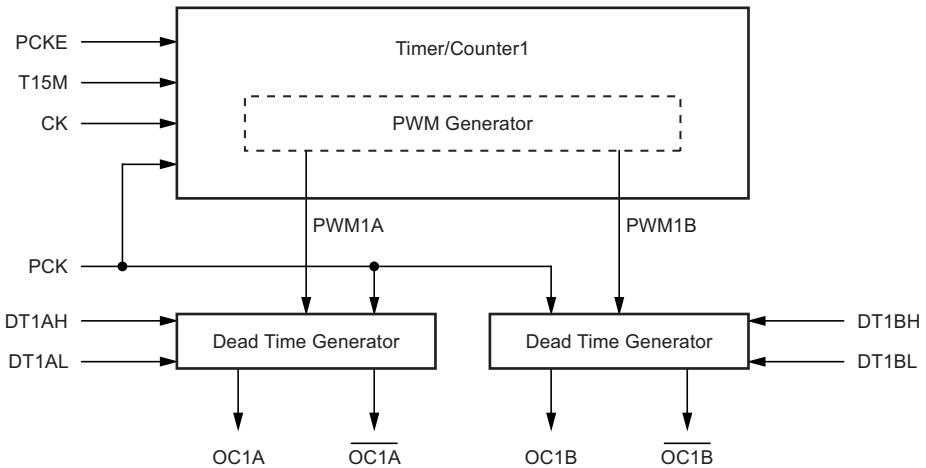
- **Bit 6.. 3- Res: Reserved Bits**

These bits are reserved bits in the Atmel ATtiny25/45/85 and always read as zero.

# 14. Dead Time Generator

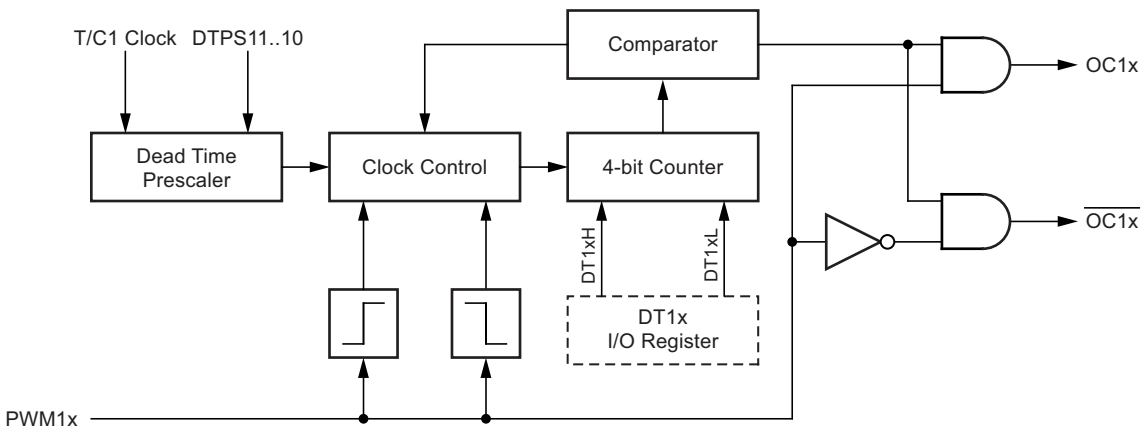
The dead time generator is provided for the Timer/Counter1 PWM output pairs to allow driving external power control switches safely. The dead time generator is a separate block that can be connected to Timer/Counter1 and it is used to insert dead times (non-overlapping times) for the Timer/Counter1 complementary output pairs (OC1A-$\overline{\text{OC1A}}$ and OC1B-$\overline{\text{OC1B}}$). The sharing of tasks is as follows: the Timer/Counter generates the PWM output and the dead time generator generates the non-overlapping PWM output pair from the Timer/Counter PWM signal. Two dead time generators are provided, one for each PWM output. The non-overlap time is adjustable and the PWM output and it's complementary output are adjusted separately, and independently for both PWM outputs.

**Figure 14-1. Timer/Counter1 and Dead Time Generators**




The dead time generation is based on the 4-bit down counters that count the dead time, as shown in Figure 14-1 There is a dedicated prescaler in front of the dead time generator that can divide the Timer/Counter1 clock (PCK or CK) by 1, 2, 4 or 8. This provides for large range of dead times that can be generated. The prescaler is controlled by two control bits DTPS11..10 from the I/O register at address 0x23. The block has also a rising and falling edge detector that is used to start the dead time counting period. Depending on the edge, one of the transitions on the rising edges, OC1x or $\overline{\text{OC1x}}$ is delayed until the counter has counted to zero. The comparator is used to compare the counter with zero and stop the dead time insertion when zero has been reached. The counter is loaded with a 4-bit DT1xH or DT1xL value from DT1x I/O register, depending on the edge of the PWM generator output when the dead time insertion is started.

**Figure 14-2. Dead Time Generator**



## 16.3 Analog Comparator Multiplexed Input

It is possible to select any of the ADC3..0 pins to replace the negative input to the analog comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the analog comparator multiplexer enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX1..0 in ADMUX select the input pin to replace the negative input to the analog comparator, as shown in Table 16-2. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the analog comparator.

**Table 16-2. Analog Comparator Multiplexed Input**

| ACME | ADEN | MUX1..0 | Analog Comparator Negative Input |
|---|---|---|---|
| 0 | x | xx | AIN1 |
| 1 | 1 | xx | AIN1 |
| 1 | 0 | 00 | ADC0 |
| 1 | 0 | 01 | ADC1 |
| 1 | 0 | 10 | ADC2 |
| 1 | 0 | 11 | ADC3 |

### 16.3.1 Digital Input Disable Register 0 – DIDR0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | – | ADC0D | ADC2D | ADC3D | ADC1D | AIN1D | AIN0D | DIDR0 |
| Read/Write | R | R | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 1, 0 – AIN1D, AIN0D: AIN1, AIN0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN1/0 pin is disabled. The corresponding PIN register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN1/0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the chip erase instruction. The chip erase operation turns the content of every memory location in both the program and EEPROM arrays into 0xFF.

Depending on CKSEL fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2CPU clock cycles for $f_{ck}$ < 12MHz, 3CPU clock cycles for $f_{ck}$ ≥ 12MHz

High: > 2CPU clock cycles for $f_{ck}$ < 12MHz, 3CPU clock cycles for $f_{ck}$ ≥ 12MHz

### 20.6.1 Serial Programming Algorithm

When writing serial data to the Atmel® ATtiny25/45/85, data is clocked on the rising edge of SCK.

When reading data from the Atmel ATtiny25/45/85, data is clocked on the falling edge of SCK. See Figure 20-2 on page 128 and Figure 20-3 on page 129 for timing details.

To program and verify the ATtiny25/45/85 in the serial programming mode, the following sequence is recommended (see four byte instruction formats in Table 20-10 on page 128):

1. Power-up sequence:
   Apply power between $V_{CC}$ and GND while $\overline{\text{RESET}}$ and SCK are set to “0”. In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to “0”.
2. Wait for at least 20ms and enable serial programming by sending the programming enable serial instruction to pin MOSI.
3. The serial programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (0x53), will echo back when issuing the third byte of the programming enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new programming enable command.
4. The flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 5 LSB of the address and data together with the load program memory page instruction. To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address. The program memory page is stored by loading the write program memory page instruction with the 6 MSB of the address. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least $t_{WD\_FLASH}$ before issuing the next page. (See Table 20-9 on page 128.) Accessing the serial programming interface before the flash write operation completes can result in incorrect programming.
5. **A:** The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least $t_{WD\_EEPROM}$ before issuing the next byte. (See Table 20-9 on page 128.) In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
   **B:** The EEPROM array is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 2 LSB of the address and data together with the Load EEPROM memory page instruction. The EEPROM memory page is stored by loading the write EEPROM memory page Instruction with the 6 MSB of the address. When using EEPROM page access only byte locations loaded with the Load EEPROM memory page instruction is altered. The remaining locations remain unchanged. If polling (RDY/$\overline{\text{BSY}}$) is not used, the used must wait at least $t_{WD\_EEPROM}$ before issuing the next page (See Table 20-7 on page 126). In a chip erased device, no 0xFF in the data file(s) need to be programmed.
6. Any memory location can be verified by using the read instruction which returns the content at the selected address at serial output MISO.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to “1”.
   Turn $V_{CC}$ power off.

**Table 20-11. Serial Programming Characteristics, $T_A$ = –40°C to 125°C, $V_{CC}$ = 2.7 to 5.5V (Unless Otherwise Noted)**

| Parameter | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Oscillator frequency (ATtiny25/45/85V) | $1/t_{CLCL}$ | 0 | | 4 | MHz |
| Oscillator period (ATtiny25/45/85V) | $t_{CLCL}$ | 250 | | | ns |
| Oscillator frequency (ATtiny25/45/85L, VCC = 2.7 to 5.5V) | $1/t_{CLCL}$ | 0 | | 10 | MHz |
| Oscillator period (ATtiny25/45/85L, VCC = 2.7 to 5.5V) | $t_{CLCL}$ | 100 | | | ns |
| Oscillator frequency (ATtiny25/45/85, $V_{CC}$ = 4.5V to 5.5V) | $1/t_{CLCL}$ | 0 | | 20 | MHz |
| Oscillator period (ATtiny25/45/85, $V_{CC}$ = 4.5V to 5.5V) | $t_{CLCL}$ | 50 | | | ns |
| SCK pulse width high | $t_{SHSL}$ | 2 $t_{CLCL}$* | | | ns |
| SCK pulse width low | $t_{SLSH}$ | 2 $t_{CLCL}$* | | | ns |
| MOSI setup to SCK high | $t_{OVSH}$ | $t_{CLCL}$ | | | ns |
| MOSI hold after SCK high | $t_{SHOX}$ | 2 $t_{CLCL}$ | | | ns |

Note: 2 $t_{CLCL}$ for $f_{ck}$ < 12MHz, 3 $t_{CLCL}$ for $f_{ck}$ ≥12MHz

## 20.7 High-voltage Serial Programming

This section describes how to program and verify flash program memory, EEPROM data memory, lock bits and fuse bits in the Atmel® ATtiny25/45/85.

**Figure 20-4. High-voltage Serial Programming**

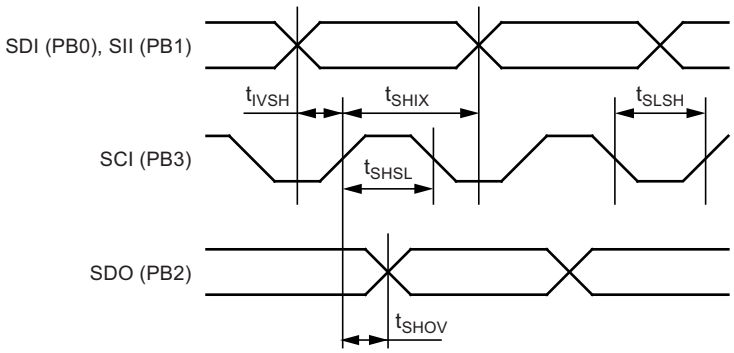



**Table 20-12. Pin Name Mapping**

| Signal Name in High-voltage Serial Programming Mode | Pin Name | I/O | Function |
|---|---|---|---|
| SDI | PB0 | I | Serial data input |
| SII | PB1 | I | Serial instruction input |
| SDO | PB2 | O | Serial data output |
| SCI | PB3 | I | Serial clock input (min. 220ns period) |

### 20.8.3 Chip Erase

The chip erase will erase the flash and EEPROM[1] memories plus lock bits. The lock bits are not reset until the program memory has been completely erased. The fuse bits are not changed. A chip erase must be performed before the flash and/or EEPROM are re-programmed.

Note: 1. The EEPROM memory is preserved during chip erase if the EESAVE fuse is programmed.

1. Load command “chip erase” (see Table 20-16 on page 134).
2. Wait after instr. 3 until SDO goes high for the “chip erase” cycle to finish.
3. Load command “no operation”.

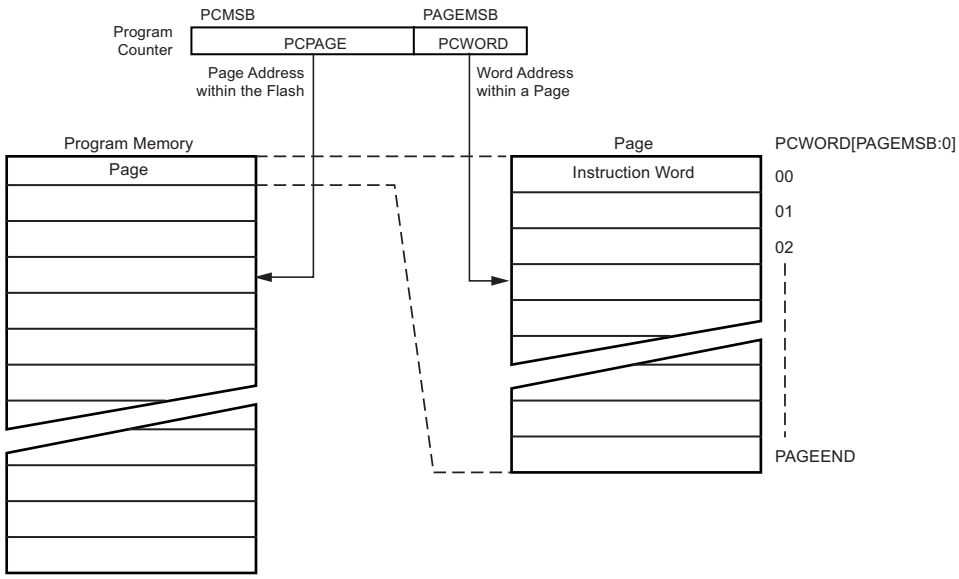
### 20.8.4 Programming the Flash

The flash is organized in pages, see Table 20-10 on page 128. When programming the flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire flash memory:

1. Load command “write flash” (see Table 20-16 on page 134).
2. Load flash page buffer.
3. Load flash high address and program page. Wait after instr. 3 until SDO goes high for the “page programming” cycle to finish.
4. Repeat 2 through 3 until the entire flash is programmed or until all data has been programmed.
5. End page programming by loading command “no operation”.

When writing or reading serial data to the Atmel® ATtiny25/45/85, data is clocked on the rising edge of the serial clock, see Figure 20-6 on page 133, Figure 20-7 on page 136 and Figure 20-17 on page 136 for details.

**Figure 20-5. Addressing the Flash which is Organized in Pages**



## 22.10 Current Consumption in Reset and Reset Pulse width

**Figure 22-41. Reset Supply Current versus $V_{CC}$ (0.1 to 1.0MHz, Excluding Current through the Reset Pull-up)**

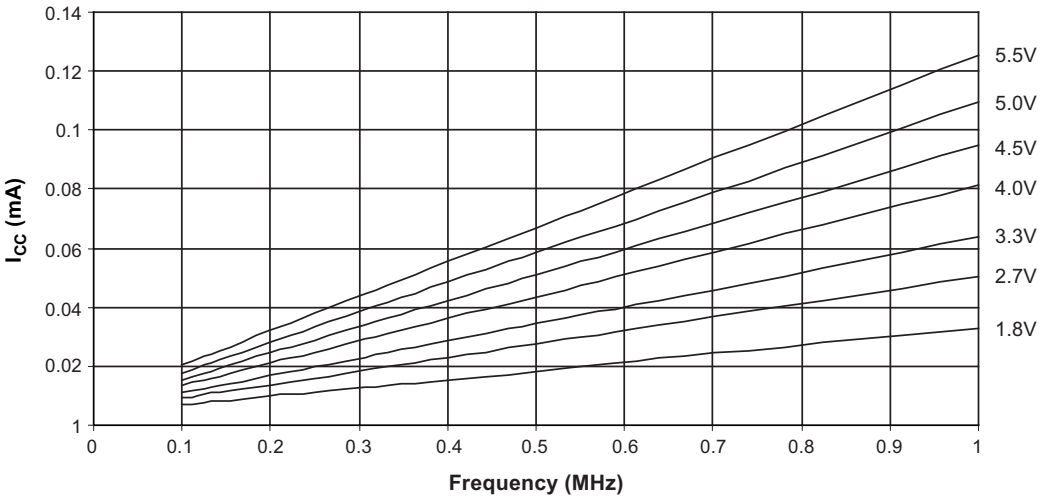
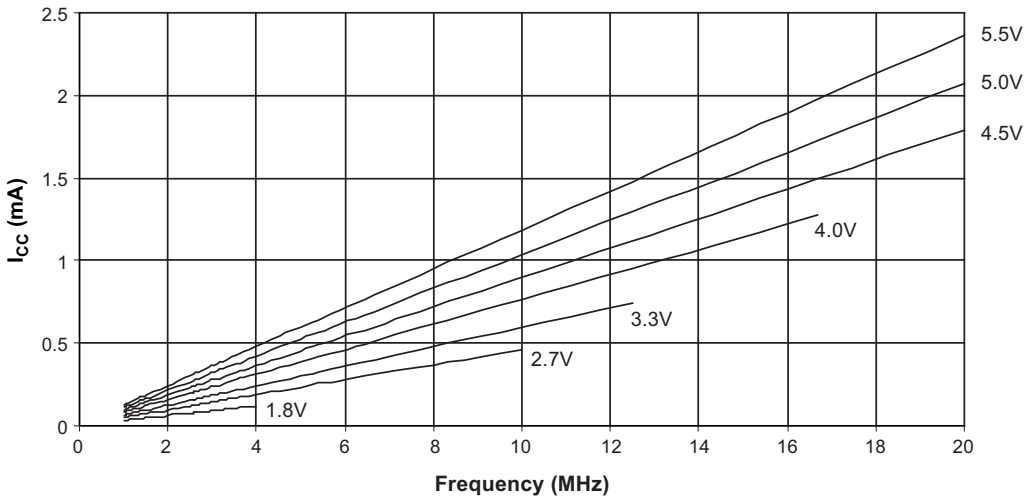



**Figure 22-42. Reset Supply Current versus $V_{CC}$ (1 to 24MHz, Excluding Current through the Reset Pull-up)**



## 24. Instruction Set Summary

| Mnemonics | Operands | Description | Operation | Flags | #Clocks |
|---|---|---|---|---|---|
| Arithmetic and Logic Instructions | | | | | |
| ADD | Rd, Rr | Add two registers | Rd ← Rd + Rr | Z,C,N,V,H | 1 |
| ADC | Rd, Rr | Add with carry two registers | Rd ← Rd + Rr + C | Z,C,N,V,H | 1 |
| ADIW | Rdl, K | Add immediate to word | Rdh: Rdl ← Rdh: Rdl + K | Z,C,N,V,S | 2 |
| SUB | Rd, Rr | Subtract two registers | Rd ← Rd – Rr | Z,C,N,V,H | 1 |
| SUBI | Rd, K | Subtract constant from register | Rd ← Rd – K | Z,C,N,V,H | 1 |
| SBC | Rd, Rr | Subtract with carry two registers | Rd ← Rd – Rr – C | Z,C,N,V,H | 1 |
| SBCI | Rd, K | Subtract with carry constant from Reg. | Rd ← Rd – K – C | Z,C,N,V,H | 1 |
| SBIW | Rdl, K | Subtract immediate from word | Rdh:Rdl ← Rdh:Rdl – K | Z,C,N,V,S | 2 |
| AND | Rd, Rr | Logical AND registers | Rd ← Rd × Rr | Z,N,V | 1 |
| ANDI | Rd, K | Logical AND register and constant | Rd ← Rd × K | Z,N,V | 1 |
| OR | Rd, Rr | Logical OR registers | Rd ← Rd v Rr | Z,N,V | 1 |
| ORI | Rd, K | Logical OR register and constant | Rd ← Rd v K | Z,N,V | 1 |
| EOR | Rd, Rr | Exclusive OR registers | Rd ← Rd ⊕ Rr | Z,N,V | 1 |
| COM | Rd | One’s complement | Rd ← 0xFF – Rd | Z,C,N,V | 1 |
| NEG | Rd | Two’s complement | Rd ← 0x00 – Rd | Z,C,N,V,H | 1 |
| SBR | Rd, K | Set Bit(s) in register | Rd ← Rd v K | Z,N,V | 1 |
| CBR | Rd, K | Clear Bit(s) in register | Rd ← Rd × (0xFF – K) | Z,N,V | 1 |
| INC | Rd | Increment | Rd ← Rd + 1 | Z,N,V | 1 |
| DEC | Rd | Decrement | Rd ← Rd – 1 | Z,N,V | 1 |
| TST | Rd | Test for zero or minus | Rd ← Rd × Rd | Z,N,V | 1 |
| CLR | Rd | Clear register | Rd ← Rd ⊕ Rd | Z,N,V | 1 |
| SER | Rd | Set register | Rd ← 0xFF | None | 1 |
| Branch Instructions | | | | | |
| RJMP | k | Relative jump | PC ← PC + k + 1 | None | 2 |
| IJMP | | Indirect jump to (Z) | PC ← Z | None | 2 |
| RCALL | k | Relative subroutine call | PC ← PC + k + 1 | None | 3 |
| ICALL | | Indirect call to (Z) | PC ← Z | None | 3 |
| RET | | Subroutine return | PC ← STACK | None | 4 |
| RETI | | Interrupt return | PC ← STACK | I | 4 |
| CPSE | Rd, Rr | Compare, skip if equal | if (Rd = Rr) PC ← PC + 2 or 3 | None | 1/2/3 |
| CP | Rd, Rr | Compare | Rd – Rr | Z, N,V,C,H | 1 |
| CPC | Rd, Rr | Compare with carry | Rd – Rr – C | Z, N,V,C,H | 1 |
| CPI | Rd, K | Compare register with immediate | Rd – K | Z, N,V,C,H | 1 |
| SBRC | Rr, b | Skip if bit in register cleared | if (Rr(b)=0) PC ← PC + 2 or 3 | None | 1/2/3 |
| SBRS | Rr, b | Skip if bit in register is set | if (Rr(b)=1) PC ← PC + 2 or 3 | None | 1/2/3 |
| SBIC | P, b | Skip if bit in I/O register cleared | if (P(b)=0) PC ← PC + 2 or 3 | None | 1/2/3 |
| SBIS | P, b | Skip if bit in I/O register is set | if (P(b)=1) PC ← PC + 2 or 3 | None | 1/2/3 |
| BRBS | s, k | Branch if status flag set | if (SREG (s) = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRBC | s, k | Branch if status flag cleared | if (SREG (s) = 0) then PC ← PC + k + 1 | None | 1/2 |
| BREQ | k | Branch if equal | if (Z = 1) then PC ← PC + k + 1 | None | 1/2 |
| BRNE | k | Branch if not equal | if (Z = 0) then PC ← PC + k + 1 | None | 1/2 |
| BRCS | k | Branch if carry set | if (C = 1) then PC ← PC + k + 1 | None | 1/2 |

# 29. Table of Contents


Features . . . 1

Pin Configurations . . . 2

1. Overview . . . 3
   - 1.1 Block Diagram . . . 3
   - 1.2 Automotive Quality Grade . . . 4
   - 1.3 Pin Descriptions . . . 4

2. About Code Examples . . . 5

3. AVR CPU Core . . . 5
   - 3.1 Introduction . . . 5
   - 3.2 Architectural Overview . . . 5
   - 3.3 ALU – Arithmetic Logic Unit . . . 6
   - 3.4 Status Register . . . 6
   - 3.5 General Purpose Register File . . . 7
   - 3.6 Stack Pointer . . . 9
   - 3.7 Instruction Execution Timing . . . 9
   - 3.8 Reset and Interrupt Handling . . . 10

4. AVR ATtiny25/45/85 Memories . . . 12
   - 4.1 In-System Re-programmable Flash Program Memory . . . 12
   - 4.2 SRAM Data Memory . . . 12
   - 4.3 EEPROM Data Memory . . . 13
   - 4.4 I/O Memory . . . 18

5. System Clock and Clock Options . . . 19
   - 5.1 Clock Systems and their Distribution . . . 19
   - 5.2 Clock Sources . . . 21
   - 5.3 Default Clock Source . . . 21
   - 5.4 Crystal Oscillator . . . 21
   - 5.5 Low-frequency Crystal Oscillator . . . 23
   - 5.6 Calibrated Internal RC Oscillator . . . 23
   - 5.7 External Clock . . . 24
   - 5.8 128 kHz Internal Oscillator . . . 25
   - 5.9 Clock Output Buffer . . . 26
   - 5.10 System Clock Prescaler . . . 26

6. Power Management and Sleep Modes . . . 28
   - 6.1 MCU Control Register – MCUCR . . . 28
   - 6.2 Idle Mode . . . 29
   - 6.3 ADC Noise Reduction Mode . . . 29
   - 6.4 Power-down Mode . . . 29
   - 6.5 Limitations . . . 29
   - 6.6 Power Reduction Register . . . 30
   - 6.7 Minimizing Power Consumption . . . 30

7. System Control and Reset . . . 32
   - 7.1 Resetting the AVR . . . 32
   - 7.2 Reset Sources . . . 32
   - 7.3 Power-on Reset . . . 34
   - 7.4 External Reset . . . 35
   - 7.5 Brown-out Detection . . . 35
   - 7.6 Watchdog Reset . . . 36
   - 7.7 MCU Status Register – MCUSR . . . 37