**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"

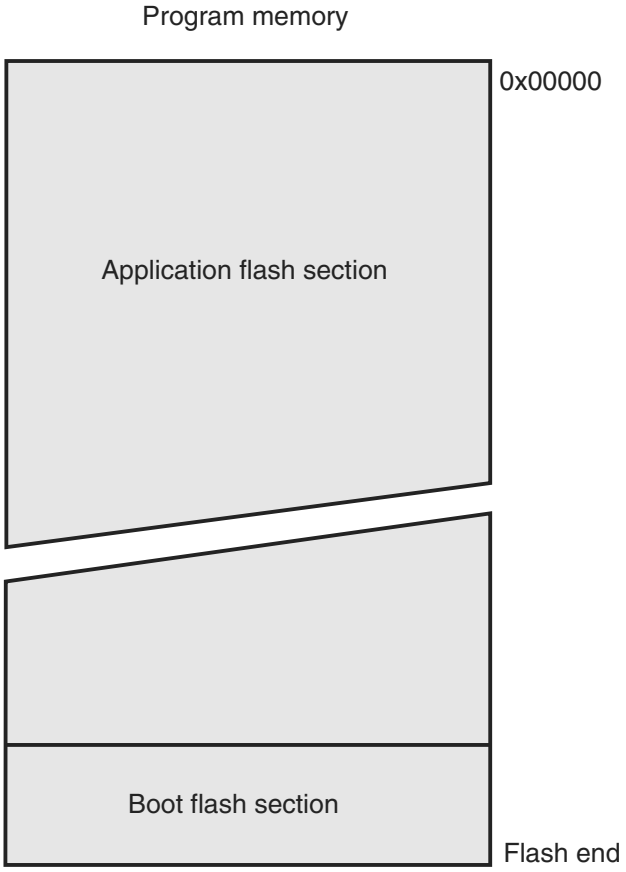
| Details | |
|---|---|
| Product Status | Active |
| Core Processor | AVR |
| Core Size | 8-Bit |
| Speed | 16MHz |
| Connectivity | EBI/EMI, I²C, SPI, UART/USART, USB OTG |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 48 |
| Program Memory Size | 128KB (128K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 4K x 8 |
| RAM Size | 8K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.5V |
| Data Converters | A/D 8x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 64-TQFP |
| Supplier Device Package | 64-TQFP (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/at90usb1287-au |

software protection are described in detail in “Memory programming” on page 359. “Memory programming” on page 359 contains a detailed description on Flash data serial downloading using the SPI pins or the JTAG interface.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory instruction description and ELPM - Extended Load Program Memory instruction description).

Timing diagrams for instruction fetch and execution are presented in “Instruction execution timing” on page 16.

**Figure 6-1.** Program memory map.




## 6.2 SRAM data memory

Figure 6-2 shows how the Atmel AT90USB64/128 SRAM memory is organized.

The AT90USB64/128 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from $060 - $0FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The first 4,352/8,448 Data Memory locations address both the Register File, the I/O Memory, Extended I/O Memory, and the internal data SRAM. The first 32 locations address the Register file, the next 64 location the standard I/O Memory, then 160 locations of Extended I/O memory and the next 4,096/8,192 locations address the internal data SRAM.

## 6.4 I/O memory

The I/O space definition of the Atmel AT90USB64/128 is shown in “Register summary” on page 419.

All AT90USB64/128 I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The AT90USB64/128 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

The I/O and peripherals control registers are explained in later sections.

### 6.4.1 General purpose I/O registers

The AT90USB64/128 contains three General Purpose I/O Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags. General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

### 6.4.2 GPIOR2 – General purpose I/O Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | MSB | | | | | | | LSB | GPIOR2 |
| 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 | |

### 6.4.3 GPIOR1 – General purpose I/O Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | MSB | | | | | | | LSB | GPIOR1 |
| 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 | |

### 6.4.4 GPIOR0 – General purpose I/O Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | MSB | | | | | | | LSB | GPIOR0 |
| 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 | |

## 8.7 Power Reduction Register

The Power Reduction Register, PRR, provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the bit in PRR, puts the module in the same state as before shutdown.

Module shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. In all other sleep modes, the clock is already stopped.

### 8.7.1 PRR0 – Power Reduction Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | **PRTWI** | **PRTIM2** | **PRTIM0** | **–** | **PRTIM1** | **PRSPI** | **-** | **PRADC** | **PRR0** |
| Read/write | R/W | R/W | R/W | R | R/W | R/W | R | R/W | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7 - PRTWI: Power Reduction TWI**

Writing a logic one to this bit shuts down the TWI by stopping the clock to the module. When waking up the TWI again, the TWI should be re initialized to ensure proper operation.

- **Bit 6 - PRTIM2: Power Reduction Timer/Counter2**

Writing a logic one to this bit shuts down the Timer/Counter2 module in synchronous mode (AS2 is 0). When the Timer/Counter2 is enabled, operation will continue like before the shutdown.

- **Bit 5 - PRTIM0: Power Reduction Timer/Counter0**

Writing a logic one to this bit shuts down the Timer/Counter0 module. When the Timer/Counter0 is enabled, operation will continue like before the shutdown.

- **Bit 4 - Res: Reserved bit**

This bit is reserved and will always read as zero.

- **Bit 3 - PRTIM1: Power Reduction Timer/Counter1**

Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

- **Bit 2 - PRSPI: Power Reduction Serial Peripheral Interface**

Writing a logic one to this bit shuts down the Serial Peripheral Interface by stopping the clock to the module. When waking up the SPI again, the SPI should be re initialized to ensure proper operation.

- **Bit 1 - Res: Reserved bit**

These bits are reserved and will always read as zero.

- **Bit 0 - PRADC: Power Reduction ADC**

Writing a logic one to this bit shuts down the ADC. The ADC must be disabled before shut down. The analog comparator cannot use the ADC input MUX when the ADC is shut down.

voltage invokes the delay counter, which determines how long the device is kept in RESET after $V_{CC}$ rise. The RESET signal is activated again, without any delay, when $V_{CC}$ decreases below the detection level.

**Figure 9-2.** MCU start-up, $\overline{\text{RESET}}$ tied to $V_{CC}$.

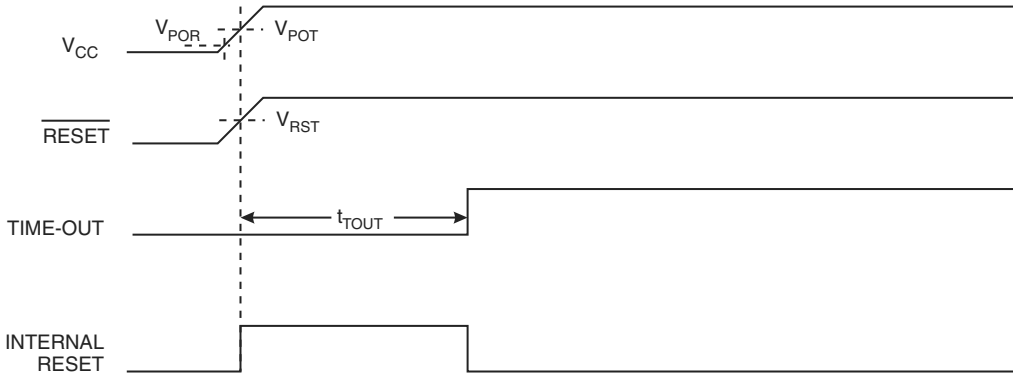
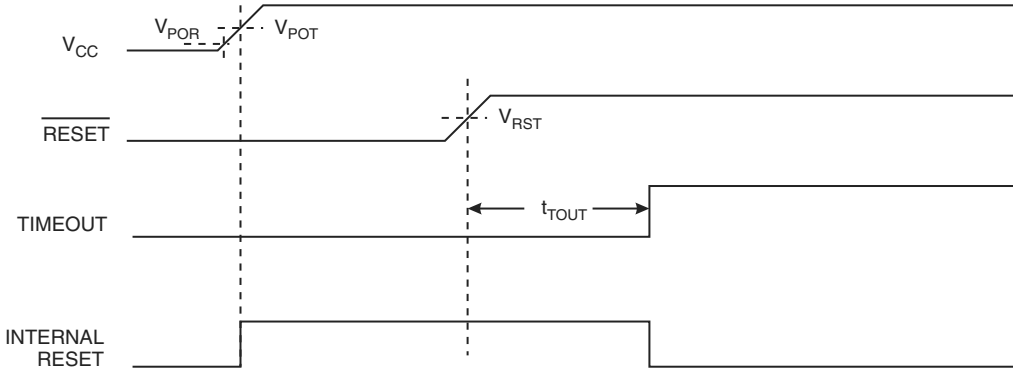



**Figure 9-3.** MCU start-up, $\overline{\text{RESET}}$ extended externally.




Note: If $V_{POR}$ or $V_{CCRR}$ parameter range can not be followed, an external reset is required.

## 9.4 External reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than the minimum pulse width (see Table 9-1 on page 58) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage – $V_{RST}$ – on its positive edge, the delay counter starts the MCU after the Time-out period – $t_{TOUT}$ – has expired.

### 11.4.11 DDRD – Port D Data Direction Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | DDD7 | DDD6 | DDD5 | DDD4 | DDD3 | DDD2 | DDD1 | DDD0 | DDRD |
| 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 | |

### 11.4.12 PIND – Port D Input Pins Address

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | PIND7 | PIND6 | PIND5 | PIND4 | PIND3 | PIND2 | PIND1 | PIND0 | PIND |
| Read/write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial value | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | |

### 11.4.13 PORTE – Port E Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | PORTE7 | PORTE6 | PORTE5 | PORTE4 | PORTE3 | PORTE2 | PORTE1 | PORTE0 | PORTE |
| 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 | |

### 11.4.14 DDRE – Port E Data Direction Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | DDE7 | DDE6 | DDE5 | DDE4 | DDE3 | DDE2 | DDE1 | DDE0 | DDRE |
| 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 | |

### 11.4.15 PINE – Port E Input Pins Address

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | PINE7 | PINE6 | PINE5 | PINE4 | PINE3 | PINE2 | PINE1 | PINE0 | PINE |
| Read/write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial value | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | |

### 11.4.16 PORTF – Port F Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | PORTF7 | PORTF6 | PORTF5 | PORTF4 | PORTF3 | PORTF2 | PORTF1 | PORTF0 | PORTF |
| 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 | |

### 11.4.17 DDRF – Port F Data Direction Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | DDF7 | DDF6 | DDF5 | DDF4 | DDF3 | DDF2 | DDF1 | DDF0 | DDRF |
| 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 | |

### 11.4.18 PINF – Port F Input Pins Address

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | PINF7 | PINF6 | PINF5 | PINF4 | PINF3 | PINF2 | PINF1 | PINF0 | PINF |
| Read/write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial value | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | |

The OCRnx Register is double buffered when using any of the twelve *Pulse Width Modulation* (PWM) modes. For the Normal and *Clear Timer on Compare* (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCRnx Compare Register 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 OCRnx Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCRnx Buffer Register, and if double buffering is disabled the CPU will access the OCRnx directly. The content of the OCR1x (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1 Register). Therefore OCR1x is not read via the high byte temporary register (TEMP). However, it is a good practice to read the low byte first as when accessing other 16-bit registers. Writing the OCRnx Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCRnxH) has to be written first. When the high byte I/O location is written by the CPU, the TEMP Register will be updated by the value written. Then when the low byte (OCRnxL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCRnx buffer or OCRnx Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to Section “Accessing 16-bit registers” on page 117.

### 15.6.1 Force Output Compare

In non-PWM Waveform Generation modes, the match output of the comparator can be forced by writing a one to the *Force Output Compare* (FOCnx) bit. Forcing compare match will not set the OCFnx Flag or reload/clear the timer, but the OCnx pin will be updated as if a real compare match had occurred (the COMn1:0 bits settings define whether the OCnx pin is set, cleared or toggled).

### 15.6.2 Compare Match Blocking by TCNTn write

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

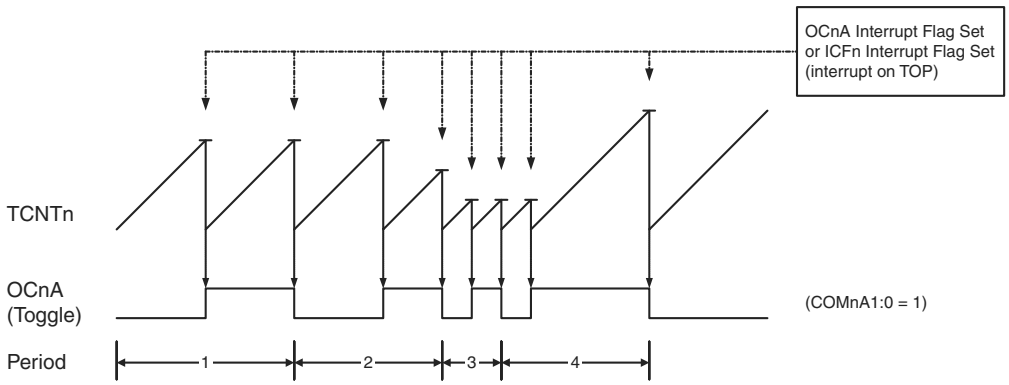
### 15.6.3 Using the Output Compare unit

Since writing TCNTn in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNTn when using any of the Output Compare channels, independent of whether the Timer/Counter is running or not. If the value written to TCNTn equals the OCRnx value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNTn equal to TOP in PWM modes with variable TOP values. The compare match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNTn value equal to BOTTOM when the counter is counting down.

The setup of the OCnx should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OCnx value is to use the Force Output Compare (FOCnx) strobe bits in Normal mode. The OCnx Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COMnx1:0 bits are not double buffered together with the compare value. Changing the COMnx1:0 bits will take effect immediately.

**Figure 15-6.** CTC mode, timing diagram.




An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCFnA or ICFn Flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCRnA or ICRn is lower than the current value of TCNTn, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode using OCRnA for defining TOP (WGMn3:0 = 15) since the OCRnA then will be double buffered.

For generating a waveform output in CTC mode, the OCnA output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COMnA1:0 = 1). The OCnA value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OCnA = 1). The waveform generated will have a maximum frequency of $f_{OCnA} = f_{clk\_I/O}/2$ when OCRnA is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

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

As for the Normal mode of operation, the TOVn Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

### 15.8.3 Fast PWM mode

The *fast Pulse Width Modulation* or fast PWM mode (WGMn3:0 = 5, 6, 7, 14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OCnx) is set on the compare match between TCNTn and OCRnx, and cleared at TOP. In inverting Compare Output mode output is cleared on compare match and set at TOP. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes 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), hence reduces total system cost.

### 15.10.7 TCNT1H and TCNT1L – Timer/Counter1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | TCNT1[15:8] | | | | | | | | TCNT1H |
| | TCNT1[7:0] | | | | | | | | TCNT1L |
| 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 | |

### 15.10.8 TCNT3H and TCNT3L – Timer/Counter3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | TCNT3[15:8] | | | | | | | | TCNT3H |
| | TCNT3[7:0] | | | | | | | | TCNT3L |
| 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 two *Timer/Counter* I/O locations (TCNTnH and TCNTnL, combined TCNTn) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit registers” on page 117.

Modifying the counter (TCNTn) while the counter is running introduces a risk of missing a compare match between TCNTn and one of the OCRnx Registers.

Writing to the TCNTn Register blocks (removes) the compare match on the following timer clock for all compare units.

### 15.10.9 OCR1AH and OCR1AL – Output Compare Register 1 A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | OCR1A[15:8] | | | | | | | | OCR1AH |
| | OCR1A[7:0] | | | | | | | | OCR1AL |
| 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 | |

### 15.10.10 OCR1BH and OCR1BL – Output Compare Register 1 B

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | OCR1B[15:8] | | | | | | | | OCR1BH |
| | OCR1B[7:0] | | | | | | | | OCR1BL |
| 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 | |

### 15.10.11 OCR1CH and OCR1CL – Output Compare Register 1 C

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | OCR1C[15:8] | | | | | | | | OCR1CH |
| | OCR1C[7:0] | | | | | | | | OCR1CL |
| 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 | |

### 16.1.1 Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2A and OCR2B) are 8-bit registers. Interrupt request (abbreviated to Int.Req.) signals are all visible in the Timer Interrupt Flag Register (TIFR2). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK2). TIFR2 and TIMSK2 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or asynchronously clocked from the TOSC1/2 pins, as detailed later in this section. The asynchronous operation is controlled by the Asynchronous Status Register (ASSR). The Clock Select logic block controls which clock source the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock ($clk_{T2}$).

The double buffered Output Compare Register (OCR2A and OCR2B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC2A and OC2B). See “Output Compare unit” on page 147. for details. The compare match event will also set the Compare Flag (OCF2A or OCF2B) which can be used to generate an Output Compare interrupt request.

### 16.1.2 Definitions

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

The definitions in the table below are also used extensively throughout the section.

| | |
|---|---|
| BOTTOM | The counter reaches the BOTTOM when it becomes zero (0x00). |
| MAX | The counter reaches its MAXimum when it becomes 0xFF (decimal 255). |
| TOP | The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2A Register. The assignment is dependent on the mode of operation. |

## 16.2 Timer/Counter clock sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source $clk_{T2}$ is by default equal to the MCU clock, $clk_{I/O}$. When the AS2 bit in the ASSR Register is written to logic one, the clock source is taken from the Timer/Counter Oscillator connected to TOSC1 and TOSC2. For details on asynchronous operation, see “ASSR – Asynchronous Status Register” on page 161. For details on clock sources and prescaler, see “Timer/Counter prescaler” on page 164.

## 16.3 Counter unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 16-2 on page 147 shows a block diagram of the counter and its surrounding environment.

Assembly code example [1]

```
SPI_MasterInit:
  ; Set MOSI and SCK output, all others input
  ldi  r17,(1<<DD_MOSI)|(1<<DD_SCK)
  out  DDR_SPI,r17
  ; Enable SPI, Master, set clock rate fck/16
  ldi  r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
  out  SPCR,r17
  ret

SPI_MasterTransmit:
  ; Start transmission of data (r16)
  out  SPDR,r16
Wait_Transmit:
  ; Wait for transmission complete
  sbis SPSR,SPIF
  rjmp Wait_Transmit
  ret
```

C code example [1]

```
void SPI_MasterInit(void)
{
  /* Set MOSI and SCK output, all others input */
  DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
  /* Enable SPI, Master, set clock rate fck/16 */
  SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
  /* Start transmission */
  SPDR = cData;
  /* Wait for transmission complete */
  while(!(SPSR & (1<<SPIF)))
    ;
}
```

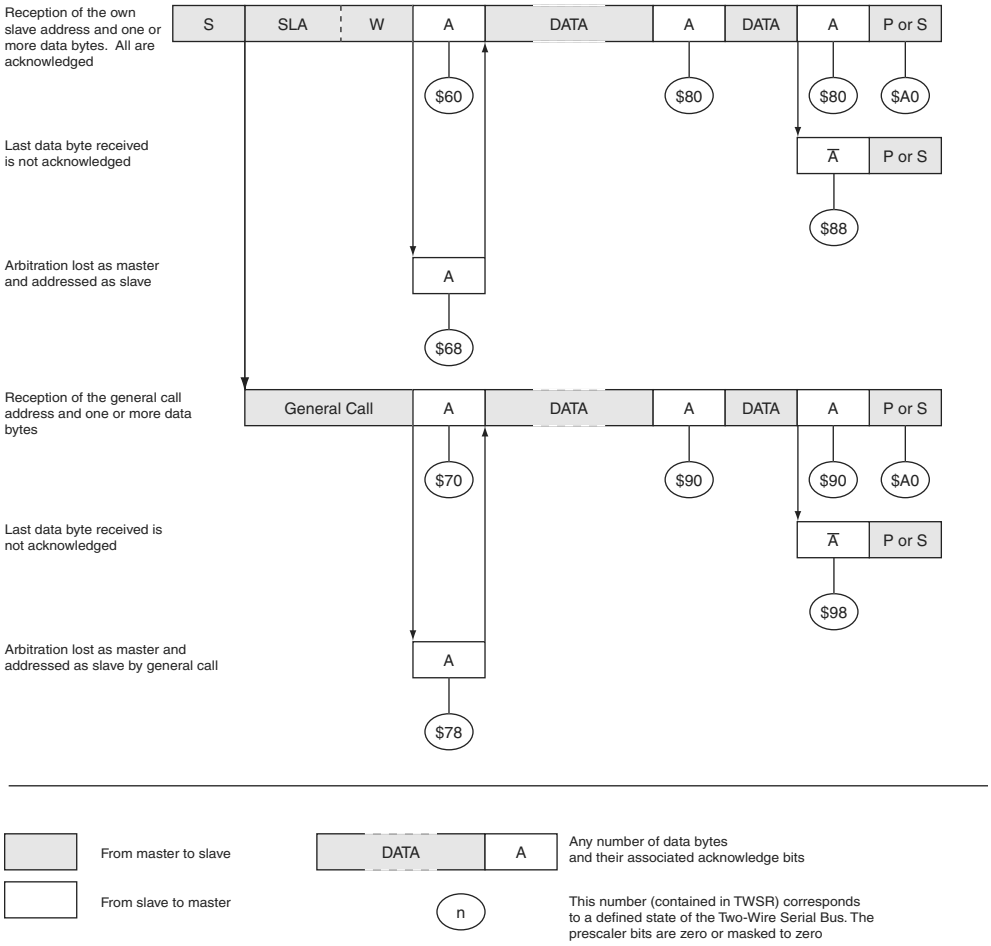
Note: 1. See “About code examples” on page 10.

**Table 19-11.** Examples of UBRRn settings for commonly used oscillator frequencies.

| Baud rate [bps] | $f_{osc}$ = 8.0000MHz | | | | $f_{osc}$ = 11.0592MHz | | | | $f_{osc}$ = 14.7456MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | U2Xn = 0 | | U2Xn = 1 | | U2Xn = 0 | | U2Xn = 1 | | U2Xn = 0 | | U2Xn = 1 | |
| | UBRR | Error | UBRR | Error | UBRR | Error | UBRR | Error | UBRR | Error | UBRR | Error |
| 2400 | 207 | 0.2% | 416 | -0.1% | 287 | 0.0% | 575 | 0.0% | 383 | 0.0% | 767 | 0.0% |
| 4800 | 103 | 0.2% | 207 | 0.2% | 143 | 0.0% | 287 | 0.0% | 191 | 0.0% | 383 | 0.0% |
| 9600 | 51 | 0.2% | 103 | 0.2% | 71 | 0.0% | 143 | 0.0% | 95 | 0.0% | 191 | 0.0% |
| 14.4k | 34 | -0.8% | 68 | 0.6% | 47 | 0.0% | 95 | 0.0% | 63 | 0.0% | 127 | 0.0% |
| 19.2k | 25 | 0.2% | 51 | 0.2% | 35 | 0.0% | 71 | 0.0% | 47 | 0.0% | 95 | 0.0% |
| 28.8k | 16 | 2.1% | 34 | -0.8% | 23 | 0.0% | 47 | 0.0% | 31 | 0.0% | 63 | 0.0% |
| 38.4k | 12 | 0.2% | 25 | 0.2% | 17 | 0.0% | 35 | 0.0% | 23 | 0.0% | 47 | 0.0% |
| 57.6k | 8 | -3.5% | 16 | 2.1% | 11 | 0.0% | 23 | 0.0% | 15 | 0.0% | 31 | 0.0% |
| 76.8k | 6 | -7.0% | 12 | 0.2% | 8 | 0.0% | 17 | 0.0% | 11 | 0.0% | 23 | 0.0% |
| 115.2k | 3 | 8.5% | 8 | -3.5% | 5 | 0.0% | 11 | 0.0% | 7 | 0.0% | 15 | 0.0% |
| 230.4k | 1 | 8.5% | 3 | 8.5% | 2 | 0.0% | 5 | 0.0% | 3 | 0.0% | 7 | 0.0% |
| 250k | 1 | 0.0% | 3 | 0.0% | 2 | -7.8% | 5 | -7.8% | 3 | -7.8% | 6 | 5.3% |
| 0.5M | 0 | 0.0% | 1 | 0.0% | – | – | 2 | -7.8% | 1 | -7.8% | 3 | -7.8% |
| 1M | – | – | 0 | 0.0% | – | – | – | – | 0 | -7.8% | 1 | -7.8% |
| Max. (1) | 0.5Mbps | | 1Mbps | | 691.2kbps | | 1.3824Mbps | | 921.6kbps | | 1.8432Mbps | |

1. UBRR = 0, Error = 0.0%.

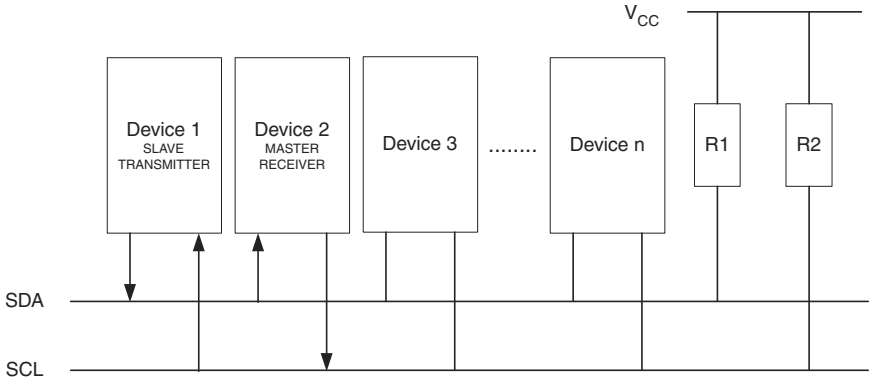
**Figure 21-17.** Formats and states in the Slave Receiver mode.




### 21.8.4 Slave Transmitter mode

In the Slave Transmitter mode, a number of data bytes are transmitted to a Master Receiver (see Figure 21-18). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

**Figure 21-18.** Data transfer in Slave Transmitter mode.



- **3 - EORSTI - End Of Reset Interrupt flag**

Set by hardware when an “End Of Reset” has been detected by the USB controller. This triggers an USB interrupt if EORSTE is set.

Shall be cleared by software. Setting by software has no effect.

- **2 - SOFI - Start Of Frame Interrupt flag**

Set by hardware when an USB “Start Of Frame” PID (SOF) has been detected (every 1ms). This triggers an USB interrupt if SOFE is set.

- **1 - Reserved**

The value read from this bits is always 0. Do not set this bit

- **0 - SUSPI - Suspend Interrupt flag**

Set by hardware when an USB “Suspend” ‘idle bus for three frame periods: a J state for 3ms) is detected. This triggers an USB interrupt if SUSPE is set.
Shall be cleared by software. Setting by software has no effect.

See Section 23.8, page 265 for more details.
The interrupt bits are set even if their corresponding ‘Enable’ bits is not set.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | - | UPRSME | EORSME | WAKEUPE | EORSTE | SOFE | - | SUSPE | UDIEN |
| Read/write | | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **7 - Reserved**

The value read from this bits is always 0. Do not set this bit.

- **6 - UPRSME - Upstream Resume Interrupt Enable bit**

Set to enable the UPRSMI interrupt.

Clear to disable the UPRSMI interrupt.

- **5 - EORSME - End Of Resume Interrupt Enable bit**

Set to enable the EORSMI interrupt.

Clear to disable the EORSMI interrupt.

- **4 - WAKEUPE - Wake-up CPU Interrupt Enable bit**

Set to enable the WAKEUPI interrupt. For correct interrupt handle execution, this interrupt should be enable only before entering power-down mode.

Clear to disable the WAKEUPI interrupt.

- **3 - EORSTE - End Of Reset Interrupt Enable bit**

Set to enable the EORSTI interrupt. This bit is set after a reset.

Clear to disable the EORSTI interrupt.

- **2 - SOFE - Start Of Frame Interrupt Enable bit**

Set to enable the SOFI interrupt.

Clear to disable the SOFI interrupt.

- **1 - Reserved**

The value read from this bits is always 0. Do not set this bit

- **0 - SUSPE - Suspend Interrupt Enable Bit**

Set to enable the SUSPI interrupt.

Clear to disable the SUSPI interrupt.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | ADDEN | UADD6:0 | | | | | | | UDADDR |
| Read/write | 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 | |

- **7 - ADDEN - Address Enable Bit**

Set to activate the UADD (USB address).

Cleared by hardware. Clearing by software has no effect.

See Section 23.7, page 264 for more details.

- **6-0 - UADD6:0 - USB Address Bits**

Load by software to configure the device address.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | FNUM10:8 | | | UDFNUMH |
| Read/write | R | R | R | R | R | R | R | R | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **7-3 - Reserved**

The value read from these bits is always 0. Do not set these bits.

- **2-0 - FNUM10:8 - Frame Number Upper Value**

Set by hardware. These bits are the three MSB of the 11-bits Frame Number information. They are provided in the last received SOF packet. FNUM is updated if a corrupted SOF is received.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | FNUM7:0 | | | | | | | | UDFNUML |
| Read/write | R | R | R | R | R | R | R | R | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Frame Number Lower Value**

Set by hardware. These bits are the eight LSB of the 11-bits Frame Number information.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | FNCERR | - | - | - | - | UDMFN |
| Read/write | | | | R | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **7-5 - Reserved**

The value read from these bits is always 0. Do not set these bits.

- **4 - FNCERR -Frame Number CRC Error flag**

Set by hardware when a corrupted Frame Number in start of frame packet is received.

This bit and the SOFI interrupt are updated at the same time.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | - | EPSIZE2:0 | | | EPBK1:0 | | ALLOC | - | UECFG1X |
| Read/write | R | R/W | R/W | R/W | R/W | R/W | R/W | R | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **7 - Reserved**

The value read from these bits is always 0. Do not set these bits.

- **6-4 - EPSIZE2:0 - Endpoint Size bits**

Set this bit according to the endpoint size:

| | |
|---|---|
| 000b: 8 bytes | 100b: 128 bytes (only for endpoint 1) |
| 001b: 16 bytes | 101b: 256 bytes (only for endpoint 1) |
| 010b: 32 bytes | 110b: Reserved. Do not use this configuration |
| 011b: 64 bytes | 111b: Reserved. Do not use this configuration |

- **3-2 - EPBK1:0 - Endpoint Bank bits**

Set this field according to the endpoint size:

00b: One bank

01b: Double bank

1xb: Reserved. Do not use this configuration

- **1 - ALLOC - Endpoint Allocation bit**

Set this bit to allocate the endpoint memory.

Clear to free the endpoint memory.

See Section 23.6, page 263 for more details.

- **0 - Reserved**

The value read from these bits is always 0. Do not set these bits.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | CFGOK | OVERFI | UNDERFI | - | DTSEQ1:0 | | NBUSYBK1:0 | | UESTA0X |
| Read/write | R | R/W | R/W | R/W | R | R | R | R | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **7 - CFGOK - Configuration Status flag**

Set by hardware when the endpoint X size parameter (EPSIZE) and the bank parametrization (EPBK) are correct compared to the max FIFO capacity and the max number of allowed bank. This bit is updated when the bit ALLOC is set.

If this bit is cleared, the user should reprogram the UECFG1X register with correct EPSIZE and EPBK values.

in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle. See “Differential channels” on page 312 for details on differential conversion timing.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

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, and ADIF is set. In Single Conversion mode, ADSC is cleared simultaneously. The software may then set 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 ADSC remains high. For a summary of conversion times, see Table 26-1 on page 312.

**Figure 26-4.** ADC timing diagram, first conversion (single conversion mode).

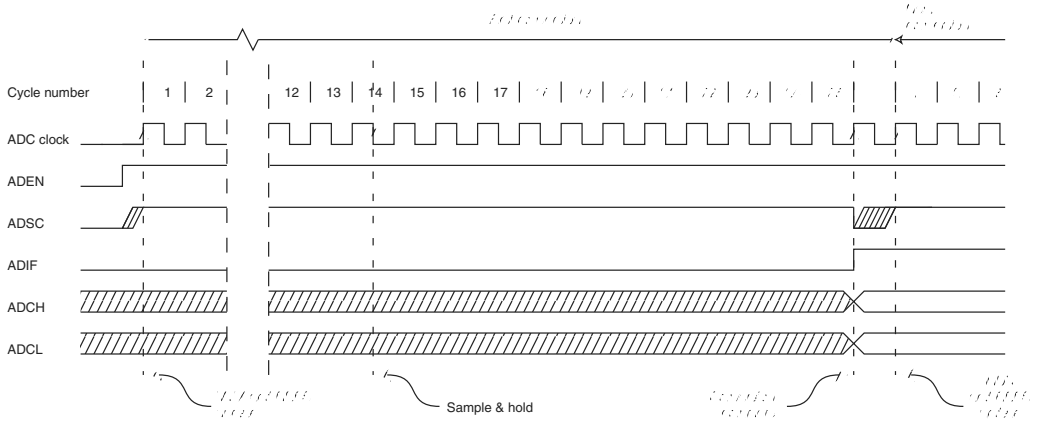



**Figure 26-5.** ADC timing diagram, single conversion.

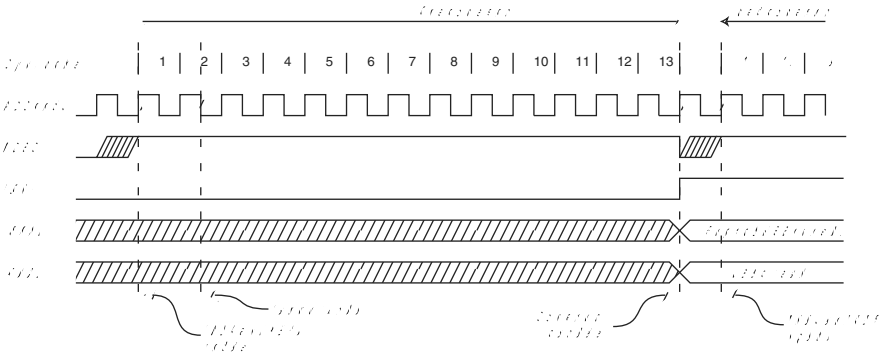
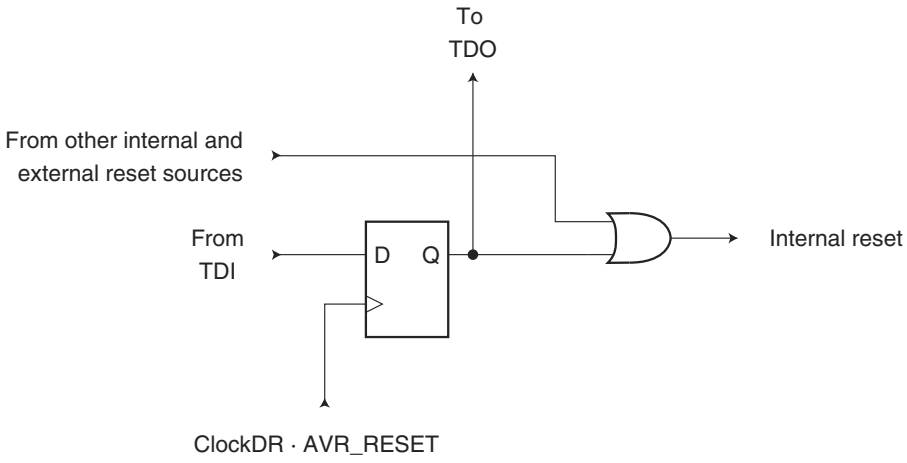


**Figure 28-2.** Reset register.




### 28.3.4 Boundary-scan Chain

The Boundary-scan Chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as the boundary between digital and analog logic for analog circuitry having off-chip connections.

See “Boundary-scan chain” on page 337 for a complete description.

## 28.4 Boundary-scan specific JTAG instructions

The Instruction Register is 4-bit wide, supporting up to 16 instructions. Listed below are the JTAG instructions useful for Boundary-scan operation. Note that the optional HIGHZ instruction is not implemented, but all outputs with tri-state capability can be set in high-impedant state by using the AVR_RESET instruction, since the initial state for all port pins is tri-state.

As a definition in this datasheet, the LSB is shifted in and out first for all Shift Registers.

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which Data Register is selected as path between TDI and TDO for each instruction.

### 28.4.1 EXTEST; 0x0

Mandatory JTAG instruction for selecting the Boundary-scan Chain as Data Register for testing circuitry external to the AVR package. For port-pins, Pull-up Disable, Output Control, Output Data, and Input Data are all accessible in the scan chain. For Analog circuits having off-chip connections, the interface between the analog and the digital logic is in the scan chain. The contents of the latched outputs of the Boundary-scan chain is driven out as soon as the JTAG IR-Register is loaded with the EXTEST instruction.

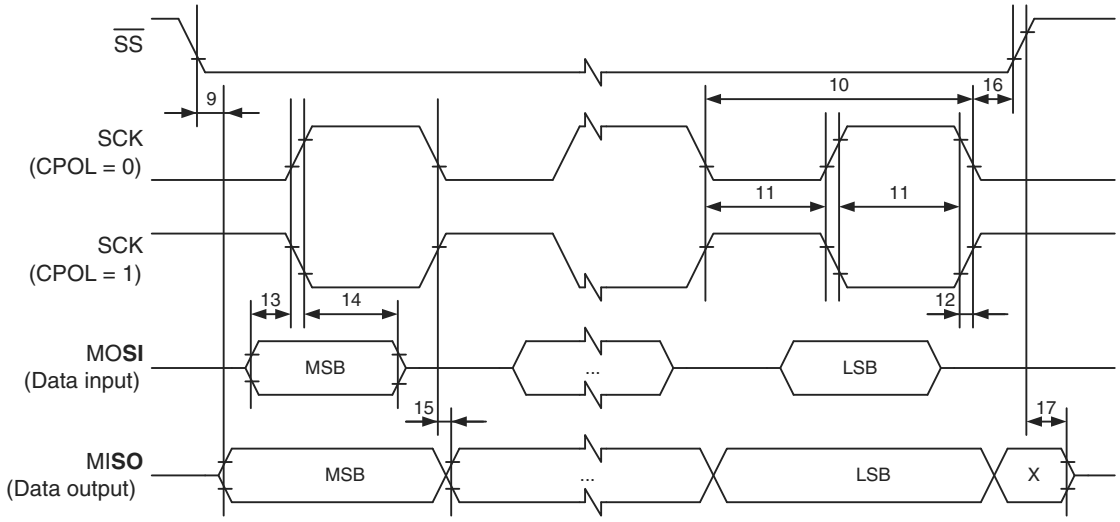
The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain
- Shift-DR: The Internal Scan Chain is shifted by the TCK input
- Update-DR: Data from the scan chain is applied to output pins

### 28.4.2 IDCODE; 0x1

Optional JTAG instruction selecting the 32-bit ID-Register as Data Register. The ID-Register consists of a version number, a device number and the manufacturer code chosen by JEDEC. This is the default instruction after power-up.

**Figure 31-5.** SPI interface timing requirements (slave mode).




## 31.8 Hardware boot entrance timing characteristics

**Figure 31-6.** Hardware boot timing requirements.

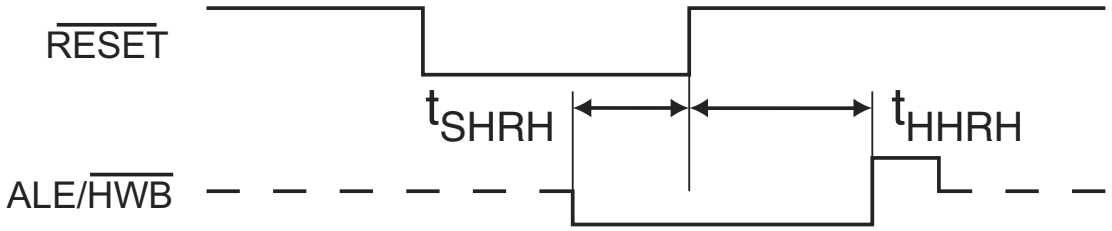
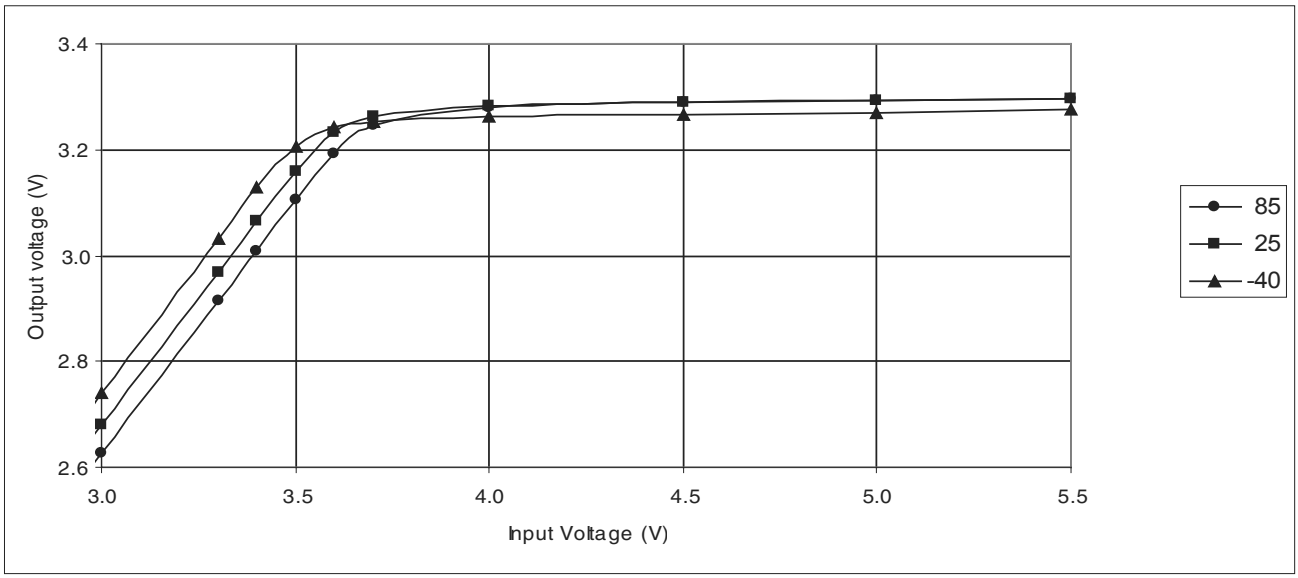



**Table 31-4.** Hardware boot timings.

| Symbol | Parameter | Min. | Max. |
|---|---|---|---|
| tSHRH | HWB low Setup before Reset High | 0 | |
| tHHRH | HWB low Hold after Reset High | StartUpTime (SUT) + Time Out Delay (TOUT) | |

**Figure 32-19.** USB regulator output voltage vs. input voltage, load = 75Ω.




Note: The 75Ω load is equivalent to the maximum average consumption of the USB peripheral in operation (full bus load).

## 32.12 BOD levels

**Figure 32-20.** BOD voltage (2.4V level) vs. temperature.