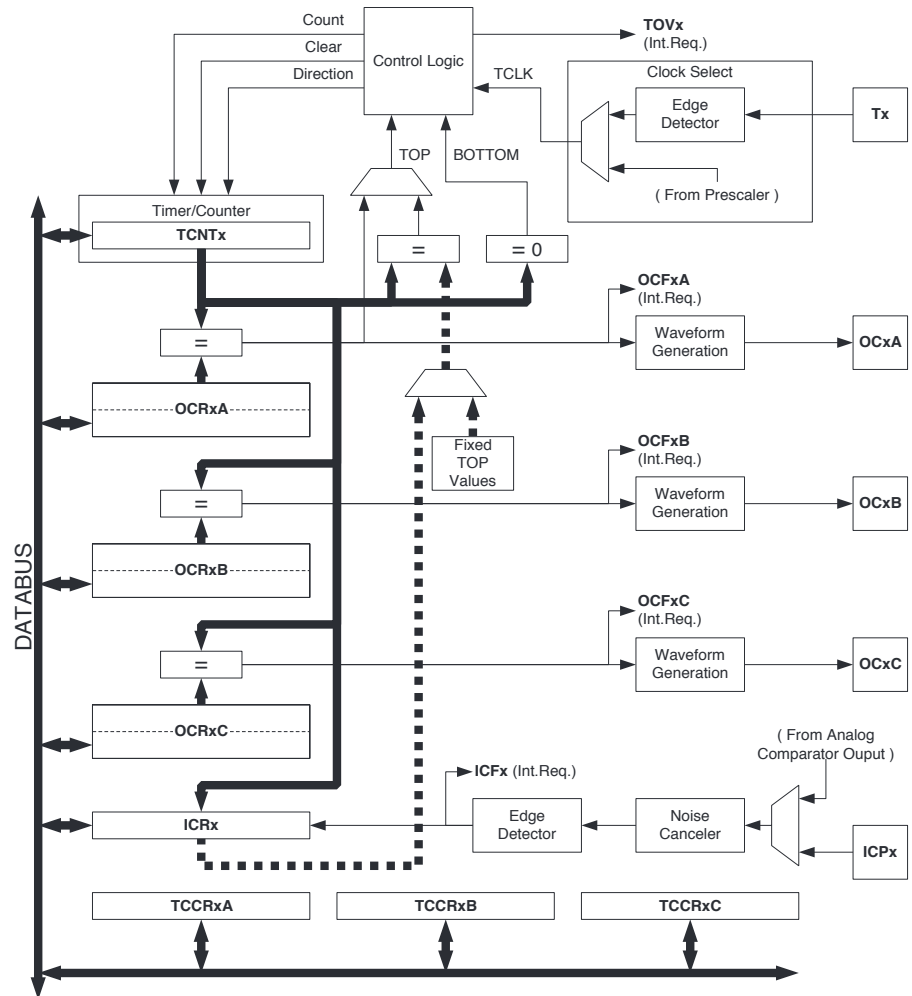


**Figure 46. 16-bit Timer/Counter Block Diagram**



Note: Refer to Figure 1 on page 2, Table 30 on page 71, and Table 39 on page 78 for Timer/Counter1 and 3 pin placement and description.

## Registers

The *Timer/Counter* (TCNTn), *Output Compare Registers* (OCRnA/B/C), and *Input Capture Register* (ICRn) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section “Accessing 16-bit Registers” on page 112. The *Timer/Counter Control Registers* (TCCRnA/B/C) are 8-bit registers and have no CPU access restrictions. Interrupt requests (shortened as Int. Req.) signals are all visible in the *Timer Interrupt Flag Register* (TIFR) and *Extended Timer Interrupt Flag Register* (ETIFR). All interrupts are individually masked with the *Timer Interrupt Mask Register* (TIMSK) and *Extended Timer Interrupt Mask Register* (ETIMSK). (E)TIFR and (E)TIMSK are not shown in the figure since these registers are shared by other timer units.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the Tn pin. The Clock Select logic block controls which clock source and edge 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<sub>Tn</sub>).

The double buffered Output Compare Registers (OCRnA/B/C) are compared with the Timer/Counter value at all time. The result of the compare can be used by the waveform

generator to generate a PWM or variable frequency output on the Output Compare Pin (OCnA/B/C). See “Output Compare Units” on page 118.. The compare match event will also set the compare match flag (OCFnA/B/C) which can be used to generate an output compare interrupt request.

The Input Capture Register can capture the Timer/Counter value at a given external (edge triggered) event on either the Input Capture Pin (ICPn) or on the Analog Comparator pins (See “Analog Comparator” on page 228.) The Input Capture unit includes a digital filtering unit (Noise Canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCRnA Register, the ICRn Register, or by a set of fixed values. When using OCRnA as TOP value in a PWM mode, the OCRnA Register can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICRn Register can be used as an alternative, freeing the OCRnA to be used as PWM output.

## Definitions

The following definitions are used extensively throughout the document:

**Table 57.** Definitions

BOTTOM	The counter reaches the <i>BOTTOM</i> when it becomes 0x0000.
MAX	The counter reaches its <i>MAX</i> imum when it becomes 0xFFFF (decimal 65535).
TOP	The counter reaches the <i>TOP</i> when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be one of the fixed values: 0x00FF, 0x01FF, or 0x03FF, or to the value stored in the OCRnA or ICRn Register. The assignment is dependent of the mode of operation.

## Compatibility

The 16-bit Timer/Counter has been updated and improved from previous versions of the 16-bit AVR Timer/Counter. This 16-bit Timer/Counter is fully compatible with the earlier version regarding:

- All 16-bit Timer/Counter related I/O register address locations, including timer interrupt registers.
- Bit locations inside all 16-bit Timer/Counter Registers, including Timer Interrupt Registers.
- Interrupt vectors.

The following control bits have changed name, but have same functionality and register location:

- PWMn0 is changed to WGMn0.
- PWMn1 is changed to WGMn1.
- CTCn is changed to WGMn2.

The following registers are added to the 16-bit Timer/Counter:

- Timer/Counter Control Register C (TCCRnC).
- Output Compare Register C, OCRnCH and OCRnCL, combined OCRnC.

The following bits are added to the 16-bit Timer/Counter Control Registers:

- COM1C1:0 are added to TCCR1A.
- FOCnA, FOCnB, and FOCnC are added in the new TCCRnC Register.
- WGMn3 is added to TCCRnB.

Interrupt flag and mask bits for output compare unit C are added.

The 16-bit Timer/Counter has improvements that will affect the compatibility in some special cases.

## Accessing 16-bit Registers

The TCNTn, OCRnA/B/C, and ICRn are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same Temporary Register is shared between all 16-bit registers within each 16-bit timer. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the Temporary Register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the Temporary Register in the same clock cycle as the low byte is read.

Not all 16-bit accesses uses the Temporary Register for the high byte. Reading the OCRnA/B/C 16-bit registers does not involve using the Temporary Register.

To do a 16-bit write, *the high byte must be written before the low byte*. For a 16-bit read, *the low byte must be read before the high byte*.

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

Assembly Code Examples <sup>(1)</sup>
<pre> ... ; Set TCNTn to 0x01FF ldi r17,0x01 ldi r16,0xFF out TCNTnH,r17 out TCNTnL,r16 ; Read TCNTn into r17:r16 in r16,TCNTnL in r17,TCNTnH ... </pre>
C Code Examples <sup>(1)</sup>
<pre> unsigned int i; ... /* Set TCNTn to 0x01FF */ TCNTn = 0x1FF; /* Read TCNTn into i */ i = TCNTn; ... </pre>

Note: 1. The example code assumes that the part specific header file is included.  
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

The assembly code example returns the TCNTn value in the r17:r16 register pair.

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

The following code examples show how to do an atomic read of the TCNTn Register contents. Reading any of the OCRnA/B/C or ICRn Registers can be done by using the same principle.

## Assembly Code Example<sup>(1)</sup>

```
TIM16_ReadTCNTn:
; Save global interrupt flag
in r18,SREG
; Disable interrupts
cli
; Read TCNTn into r17:r16
in r16,TCNTnL
in r17,TCNTnH
; Restore global interrupt flag
out SREG,r18
ret
```

## C Code Example<sup>(1)</sup>

```
unsigned int TIM16_ReadTCNTn( void )
{
    unsigned char sreg;
    unsigned int i;
    /* Save global interrupt flag */
    sreg = SREG;
    /* Disable interrupts */
    __disable_interrupt();
    /* Read TCNTn into i */
    i = TCNTn;
    /* Restore global interrupt flag */
    SREG = sreg;
    return i;
}
```

Note: 1. The example code assumes that the part specific header file is included. For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBR”, “SBRC”, “SBR”, and “CBR”.

The assembly code example returns the TCNTn value in the r17:r16 register pair.

The following code examples show how to do an atomic write of the TCNTn Register contents. Writing any of the OCRnA/B/C or ICRn Registers can be done by using the same principle.

#### Assembly Code Example<sup>(1)</sup>

```
TIM16_WriteTCNTn:
    ; Save global interrupt flag
    in r18,SREG
    ; Disable interrupts
    cli
    ; Set TCNTn to r17:r16
    out TCNTnH,r17
    out TCNTnL,r16
    ; Restore global interrupt flag
    out SREG,r18
    ret
```

#### C Code Example<sup>(1)</sup>

```
void TIM16_WriteTCNTn( unsigned int i )
{
    unsigned char sreg;
    unsigned int i;
    /* Save global interrupt flag */
    sreg = SREG;
    /* Disable interrupts */
    __disable_interrupt();
    /* Set TCNTn to i */
    TCNTn = i;
    /* Restore global interrupt flag */
    SREG = sreg;
}
```

Note: 1. The example code assumes that the part specific header file is included. For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNTn.

#### Reusing the Temporary High Byte Register

If writing to more than one 16-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.