
AVR1306: Using the XMEGA Timer/Counter

Features

- True 16 bit operation
- Double buffered timer period setting
- 4 (2) compare or capture channels per timer
 - 4 channels on each Timer 0
 - 2 channels on each Timer 1
 - Double buffered
- 32 bit operation with 32 bit input capture by timer cascading
- Event counter
- Timer overflow and error interrupts (and events)
- Input capture interrupts (and events)

1 Introduction

The XMEGA™ Timer/Counter modules are true 16-bit Timer/Counters with Input Capture and Pulse Width Modulation (PWM) functionality.

This application note gives an introduction on how to use the XMEGA Timer/Counter modules for timing, Input Capture and PWM.



8-bit **AVR**®
Microcontrollers

Application Note

Rev. 8045A-AVR-02/08





2 Definitions

The definitions listed in Table 2-1 will be used throughout this document.

Table 2-1. Definitions used in this document.

Name	Description
BOTTOM	The Timer/Counter reaches the BOTTOM when it becomes zero.
MAX	The Timer/Counter reaches it MAXimum when it becomes 0xFFFF.
TOP	The Timer/Counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be equal to the period register (PER[H:L]) or Compare or Capture register A (CCA[H:L]), depending on the selected Waveform Generation Mode (WGM).
UPDATE	The Timer/Counter signals an update when it reaches BOTTOM or TOP depending on the selected waveform generation mode (WGM).

3 Module Overview

The XMEGA Timer/Counter (TC) modules are 16-bit Timer/Counters with input capture and compare match with optional output to I/O pin. Typical applications include:

- Timing.
- Periodic interrupt/event generation.
- Pulse Width Modulation.
- Event time stamping.
- Event counting.
- Signal parameter measurements (Period, duty cycle, etc.).

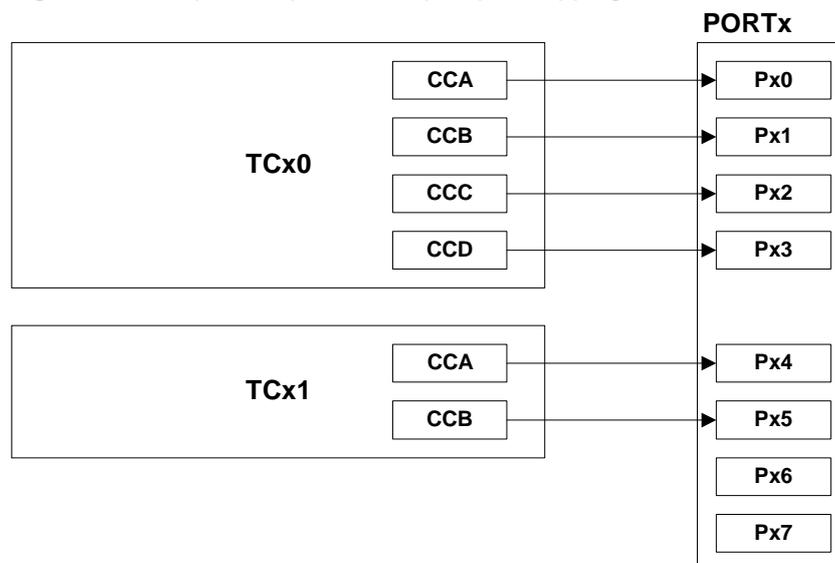
3.1 Timer/Counter Variants

An XMEGA TC module is related to an I/O port through the I/O pins it can control with the output compare module. The naming of the TC modules reflects the I/O port it is connected and the available features for that Timer/Counter. Each TC module has a name that follows the form: TC xn . TC is short for Timer/Counter, x indicates the port it is connected to and n is the TC number within PORT x . As an example, TCD0 is Timer/Counter 0 connected to PORTD.

Timers are available on PORTC, PORTD, PORTE or PORTF, if available on device.

Timer0 on each port has 4 compare or capture channels, while Timer1 has 2 compare or capture channels. Figure 3-1 shows how the output compare channels are connected to the associated I/O port. Timer0 has its outputs connected to pins 0-3 within the port, while Timer1 is mapped to pins 4 and 5.

Figure 3-1. Output compare to I/O port pin mapping



3.2 Clock Sources

The available clock source selections for the XMEGA TC modules are listed in Table 3-1.

Table 3-1. XMEGA Timer/Counter clock sources.

Symbol	Clock selection
TC_CSEL_OFF_gc	TC off (no clock selected)
TC_CSEL_DIV1_gc	$f_{CLK,SYS}$
TC_CSEL_DIV2_gc	$f_{CLK,SYS} / 2$
TC_CSEL_DIV4_gc	$f_{CLK,SYS} / 4$
TC_CSEL_DIV8_gc	$f_{CLK,SYS} / 8$
TC_CSEL_DIV64_gc	$f_{CLK,SYS} / 64$
TC_CSEL_DIV256_gc	$f_{CLK,SYS} / 256$
TC_CSEL_DIV1024_gc	$f_{CLK,SYS} / 1024$
TC_CSEL_EV0_gc	Event channel 0
TC_CSEL_EV1_gc	Event channel 1
TC_CSEL_EV2_gc	Event channel 2
TC_CSEL_EV3_gc	Event channel 3
TC_CSEL_EV4_gc	Event channel 4
TC_CSEL_EV5_gc	Event channel 5
TC_CSEL_EV6_gc	Event channel 6
TC_CSEL_EV7_gc	Event channel 7

3.3 Interrupts

Interrupts can be generated in the following situations:





- Timer overflow.
- Timer error (Input capture data lost because input capture buffer is full).
- Compare or capture. Separate interrupts for each channel.

3.4 Period Setting

Each TC module has a period register (PER[H:L]) that controls the TOP value of the counter (except in frequency generation mode).

The PER[H:L] register is double buffered through the PERBUF[H:L] register. Double buffered registers are covered in section 3.7.

The PER[H:L] register is automatically initialized to 0xFFFF after a reset.

3.5 Capture or Compare Channels

Each TC module has two or four capture or compare channels. The channels can be set up for either input capture or compare match. It is not possible to use the same TC module for both input capture and compare match at the same time. Input capture can be used to time stamp event, measuring waveform parameters like frequency or duty cycle. The compare match feature can be used to generate interrupts or events at certain time points, or for PWM/waveform generation.

The compare or capture channels consist of a set of 16-bit registers named CCx[H:L], where x indicates the channel. Timer0, with 4 channels, has CCA[H:L], CCB[H:L], CCC[H:L] and CCD[H:L], while Timer1 has CCA[H:L] and CCB[H:L]. In addition, each CCx[H:L] register has an associated buffer register CCxBUF[H:L]. Double buffered registers are covered in detail in section 3.7.

When used for input capture, the value of the PER register determines how input capture values are interpreted. If bit 15 of PER is set to one, the entire 16 bit counter value is stored in the capture buffer. If bit 15 of PER is zero, the event polarity is stored in bit 15 in the capture buffer. This can be used to measure frequency parameters like duty cycle using only one input pin and one capture channel.

3.6 Event Input Selection

The TC input capture system uses the XMEGA event system to trigger an input capture. Refer to application note AVR1001 for more information about the XMEGA event system.

When the TC is used for input capture, all enabled input capture channels must be associated with an event channel. The EVSEL[3:0] bits of the CTRLD register select the event channels that are associated with the input capture channels. EVSEL3 must be 1 to select an event source. EVSEL[2:0] selects between the 8 event channels.

The event channels associated with the different channels cannot be selected individually. Table 3-2 shows the mapping between global event channels and input capture channels when EVSEL3 = 1 and EVSEL[2:0] = N. The “%” is the modulo operator. Note that when a set of event channels have been selected for the TC, this does not mean that these event channels are used exclusively by the TC. The TC simply “listens” to the selected channels and is able to receive events on these event channels.

Table 3-2. Event channel to input capture channel mapping.

Input capture channel	Global event channel
A	N
B	$(N + 1) \% 8$
C (If available)	$(N + 2) \% 8$
D (If available)	$(N + 3) \% 8$

Example 1:

EVSEL3 = 1 and EVSEL[2:0] = 0 means that input capture channels A, B, C and D are triggered by event channels 0, 1, 2 and 3 respectively.

Example 2:

EVSEL3 = 1 and EVSEL[2:0] = 6 means that input capture channels A, B, C and D are triggered by event channels 6, 7, 0 and 1 respectively.

If the event source is subject to noise, e.g. caused by the bouncing of an external switch, it is possible to enable digital filtering on the event channel. See application note AVR1001 for more information on the event system and its digital filtering capabilities.

3.7 Double Buffered Registers

The PER[H:L] and CCx[H:L] registers in the TC modules are double buffered. They have dedicated buffer registers named PERBUF[H:L] and CCxBUF[H:L].

Using the PER[H:L] register as an example: when writing to PER[H:L] directly, the value will be updated immediately. If the value is instead written to PERBUF[H:L], the value of PER[H:L] will not be updated instantly, but instead transferred from PERBUF[H:L] on the next UPDATE condition. Each buffer register has a related “buffer valid” flag in the CTRL register. The buffer valid flag is set when a new value is written to the corresponding buffer register. The flag is cleared when the buffer register is copied to its destination. This ensures that only new values are copied.

It is also possible to lock the automatic update from buffer registers. This enables simultaneous updates of several registers at one update condition. Setting the Timer Lock Update (LUPD) bit in the CTRL register enables update locking.

When the TC module is used for PWM, the buffering is typically used to make sure that the duty cycles are not altered in the middle of a PWM period, and to synchronize changes to all PWM channels. New compare/TOP values are typically written to the buffer registers instead of writing directly to the CCx registers. This way, all PWM channels can be updated at the same time (on the UPDATE event).

In Input Capture mode, the input capture values are double-buffered. A new input capture-value is first put in the CCxBUF[H:L] register. Values stored in CCxBUF[H:L] are transferred to CCx[H:L] if the CCIFx flag is not set. The CCIFx flag is cleared automatically when CCx[H:L] is read. In practice, this means that the user only needs to read the CCx[H:L] register, while the rest is handled automatically.

3.8 Modes of Operation

The TC has six different modes of operation:

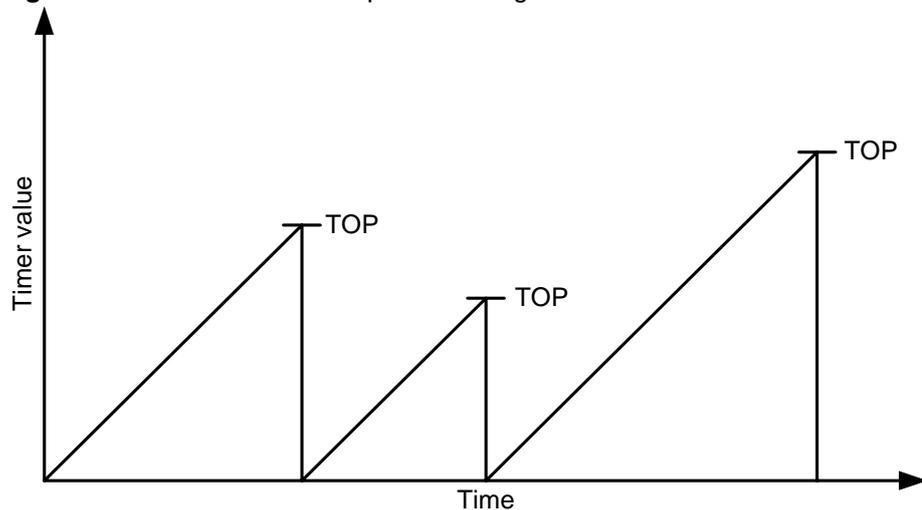


- Normal mode.
- Frequency Generation mode.
- Single Slope PWM.
- Dual Slope PWM, overflow on TOP.
- Dual Slope PWM, overflow on TOP and BOTTOM.
- Dual Slope PWM, overflow on BOTTOM.

3.8.1 Normal Mode

In Normal Mode, the counter will count in the direction set by the DIR bit in CTRLF for each clock until it reaches TOP set by PER[H:L] or BOTTOM (zero). When TOP is reached when up-counting the counter will be set to zero when the next clock is given. If the TC is down-counting the value will wrap around to the value in PER[H:L] after reaching BOTTOM.

Figure 3-2. Normal mode with period setting.



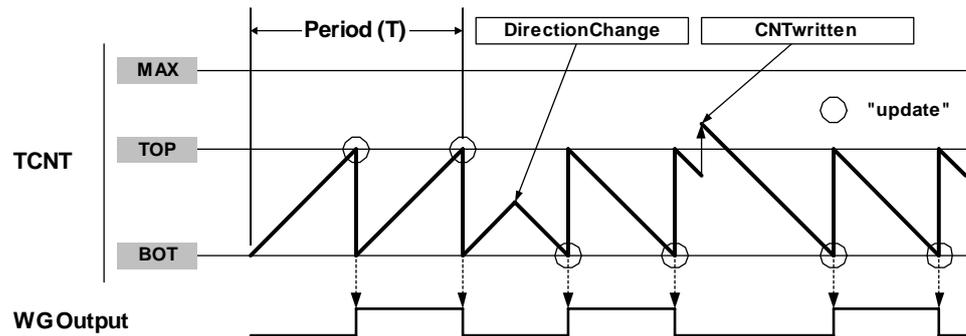
As shown in Figure 3-2, changing the counter value while the counter is running is allowed. The write access has higher priority than count, clear, or reload and will be immediate. However, if the value written is outside the BOTTOM-TOP boundary the counter either has to count down until TOP is reached or count up until wraparound (passing MAX) for the timer to re-stabilize to the period time.

This mode of operation must be used when using the Timer/Counter for Input Capture.

3.8.2 Frequency Generation Mode

There is little difference between the frequency waveform generation mode (FRQ) and the normal mode of operation. For FRQ the period (T) is controlled by the CCA[H:L] register instead of PER[H:L], which in this case is not in use. In addition the Waveform Generation (WG) output is toggled on each compare match between CNT[H:L] and CCA[H:L] as shown in Figure 3-3.

Figure 3-3. Frequency Generation Mode



The waveform generated will have a maximum frequency of $f_{clk}/2$ when CCA[H:L] is set to zero (0x0000). The waveform frequency is defined by Equation 3-1.

Equation 3-1. Frequency Calculation, Frequency generation mode

$$f_{FRQ} = \frac{f_{CLK}}{2 \cdot N(CCA + 1)}$$

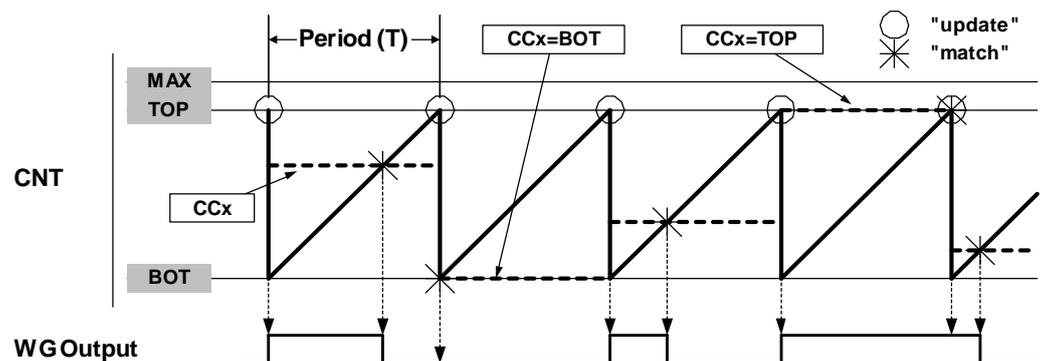
Where N represents the TC clock prescaler.

The Overflow Status Flag (OVIF) or Compare A Flag (CAIF) can be used to generate interrupts. If enabled, the interrupt handler routine can be used for updating CCA[H:L] to change the output frequency.

3.8.3 Single Slope PWM Mode

As Figure 3-4 shows, in this mode of operation the counter counts from BOTTOM to TOP then restarts from BOTTOM. The waveform generator output is set on the compare match between the count and compare registers, and cleared at TOP.

Figure 3-4. Single slope PWM



The single-slope Pulse Width Modulation (PWM) provides twice the PWM frequency compared to dual-slope PWM. The high frequency makes this mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external filtering components, which reduces total system cost.

The period register (PER[H:L]) defines the PWM resolution. The minimum resolution allowed for this module is 2-bit (PER[H:L]=0x0003), and maximum resolution is 16-bit



(PER[H:L]=MAX). Equation 3-2 can be used to calculate the exact bit resolution for single-slope PWM.

Equation 3-2. PWM Resolution, Single Slope PWM Mode

$$R_{PWM_SS} = \log_2(PER + 1)$$

The PWM base frequency depends on the period setting (PER[H:L]), system clock frequency, and clock prescaler. The PWM base frequency can be calculated using Equation 3-3.

Equation 3-3. Frequency calculation, Single Slope PWM Mode

$$f_{PWM_SS} = \frac{f_{CLK}}{N(PER + 1)}$$

Where N represents the TC clock prescaler.

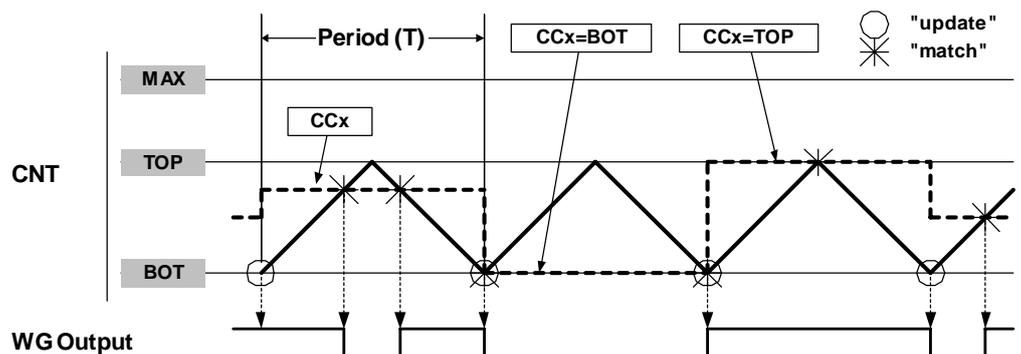
The Overflow Status Flag (OVFIF) or Compare Flag (CCxIF) can be used to generate interrupts. If enabled, the interrupt service routine can be used for updating the period and compare buffer values.

3.8.4 Dual Slope PWM Modes

The counter behavior in dual-slope operation is shown in Figure 3-5. The counter counts repeatedly from BOTTOM to TOP and then to BOTTOM. When the counter hits BOTTOM or TOP, the counter changes direction immediately, holding the BOTTOM and TOP values for only one TC clock cycle. The current counting direction can be determined by inspecting the DIR flag in CTRLF.

The Waveform Generation (WG) output is set on BOTTOM and cleared on compare match. The output duty cycle is thus proportional to the compare value.

Figure 3-5. Dual slope PWM



The dual-slope PWM result in a lower operating frequency at equal resolution compared to the single-slope PWM mode.

The period register (PER[H:L]) defines the PWM resolution. The minimum resolution allowed for this module is 2-bit (PER[H:L]=0x0003), and maximum resolution is 16-bit (PER[H:L]=0xFFFF). Equation 3-4 can be used for calculate the exact resolution for dual-slope PWM.

Equation 3-4. PWM Resolution, Dual Slope PWM Mode

$$R_{PWM_DS} = \log_2(PER + 1)$$

The PWM base frequency depends on the period setting (PER[H:L]), system clock frequency and clock prescaler and can be calculated using Equation 3-5.

Equation 3-5. Frequency calculation, Dual Slope PWM Mode

$$f_{PWM_DS} = \frac{f_{CLK}}{2 \cdot N \cdot PER}$$

Where N represents the TC clock prescaler.

The Overflow Status Flag (OVFIF) or Compare Flag (CCxIF) can be used to generate interrupts. If enabled, the interrupt service routine can be used for updating the period and compare buffer values.

3.9 Timer/Counter Commands

The CMD[1:0] bits in CTRLFSET are used to issue special commands to the TC. These are described below.

3.9.1 Force Update

The “Force update” command is used to force an UPDATE condition. Buffer registers are copied to their destination when the UPDATE conditions are met. Issuing this command forces an UPDATE condition regardless of the counter value. This can be used to update both the period and compare registers at the exact same time. In input capture mode, the “Force update” command only has effect on the PERBUF/PER registers. See section 3.7 for more information about double buffered registers.

3.9.2 Force Restart

The “Force restart” command clears the CNT[H:L] registers and the direction bit to its reset value.

3.9.3 Force Hard Reset

The “Force hard reset” command puts all registers in that TC instance back to their reset state. For safety reasons, the TC clock selection must be set to OFF for this command to have any effect.

4 Accessing 16-bit Registers

Since the XMEGA has a bus width of 8 bits, 16-bit registers are accessed as two 8-bit registers. All accesses to 16-bit registers are performed using a hardware-controlled 8 bit temporary register to ensure concurrent update of the high and low byte registers. Each Timer/Counter has one dedicated temporary register, TEMP, that is used for this purpose. The TEMP register is shared for all 16-bit accesses in that TC module. Usage of the TEMP register is automatically handled when the correct byte access order is used, but it is important to understand the mechanism to avoid corruption of registers.

Note that on the XMEGA, the byte order for multi-byte access is always from the least significant byte to the most significant byte.



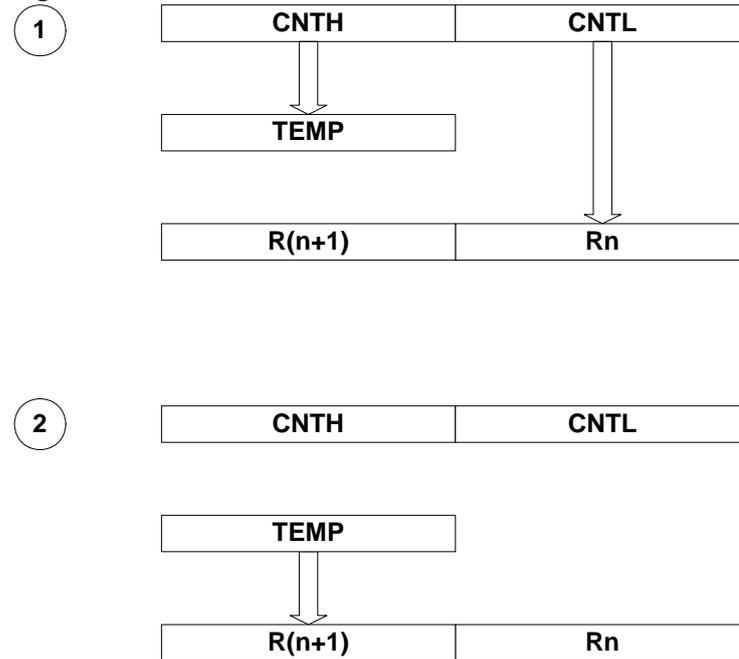
4.1 16-bit Read

Figure 4-1 illustrates how to perform a 16-bit read of CNT, and what happens internally:

1. Read CNTL. This will cause the contents of CNTH to be transferred to TEMP in the same clock cycle.
2. Read CNTH. This will automatically cause the contents of TEMP to be read instead of the current value of CNTH.

The result is a snapshot of the 16-bit CNT[H:L] value at the time CNTL was read.

Figure 4-1. 16-bit read access.



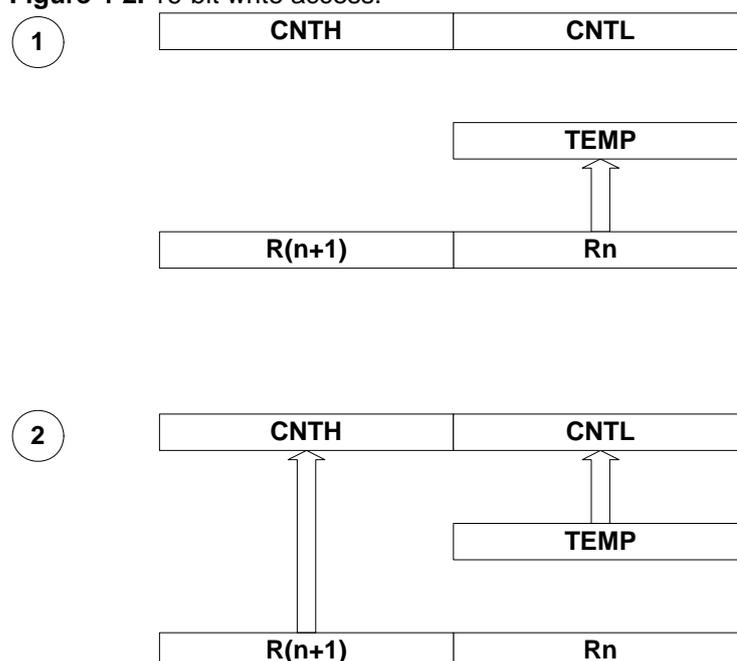
4.2 16-bit Write

Figure 4-2 illustrates how to perform a 16-bit and what happens internally:

1. Write CNTL. This will cause the TEMP register to be updated with the new value for CNTL.
2. Write CNTH. This will cause CNTL to be updated with the contents of the TEMP register at the same time as CNTH is updated.

The result is that the 16-bit CNT[H:L] value is updated in the same clock cycle.

Figure 4-2. 16-bit write access.



4.3 Implications of Having a Shared TEMP Register

Since one TEMP register is shared for all 16-bit registers within one TC module, there is a possibility of register corruption if the necessary measures are not taken.

Consider the following scenario: One part of the program running in the foreground loop is about to perform a write to $CNT[H:L]$. The foreground task has just finished a write to $CNTL$, resulting in the TEMP register holding the low byte temporarily. Before the foreground loop is able to write to $CNTH$, an interrupt is triggered. This interrupt service routine (ISR) performs a write to the 16-bit $PER[H:L]$ register. During this process, the TEMP register is used for the low byte of $PER[H:L]$. When the ISR returns, the contents of TEMP are corrupted. The foreground task completes the write operation by writing to $CNTH$, causing the corrupted low-byte to be transferred from TEMP to $CNTL$.

There are two possible solutions to this problem:

1. Make sure that every 16-bit access is performed as an atomic operation by disabling interrupts during the access. Since the XMEGA is equipped with a multi-level interrupt controller, this does not only apply to the foreground code, but also low and medium level interrupts.
2. If an ISR can interrupt another 16-bit access in the same TC module, it must store the contents of the TEMP register before the 16-bit access is performed, and restore the TEMP register again before leaving the ISR.

4.4 16-bit Access in On-Chip Debug Stopped Mode

When the XMEGA is in On-Chip Debug (OCD) stopped mode ('break' in AVR Studio), the TEMP register is bypassed. It is thus possible to access both the high and low byte of 16-bit registers directly and there will be no automatic transfers of data to the TEMP register. For debugging purposes the TEMP register itself can also be read/written during OCD stopped mode.



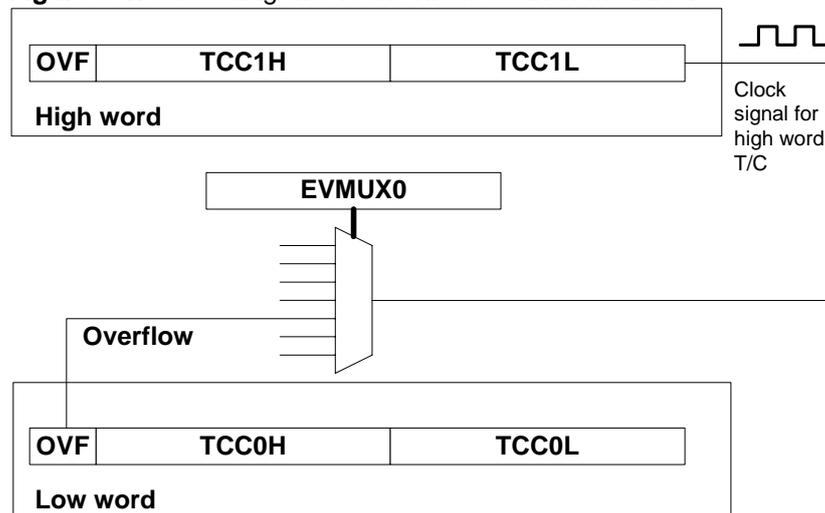
5 Using two 16-bit Timer/Counters as one 32-bit Timer/Counter

It is possible to use the event system to cascade two 16-bit Timer/Counter modules into one 32-bit Timer/Counter with 32-bit input capture functionality.

5.1 Cascading two 16-bit Timer/Counter Modules

All Timer/Counter modules have the option of being clocked by one of the event channels. To cascade two 16-bit TC modules and thus get a 32-bit TC, the overflow event from the low word TC must be routed through the event system and used as the clock for the high word TC. Figure 5-1 shows how this is done using TCC0 as low word and TCC1 as high word, through event channel 0. Event channel 0 is configured through the EVSYS.CH0MUX register to use the TCC0 overflow event as event source. The CLKSEL bits in TCC1.CTRLA are configured to use event channel 0 as clock source. The result is that TCC1 will increase by one each time TCC0 overflows. In effect, this creates a 32-bit TC. Note that any two TC modules can be cascaded in this way.

Figure 5-1. Cascading two 16-bit Timer/Counter modules.



5.2 Accessing the 32-bit Value

When reading the cascaded TC, the 32-bit value must be manually assembled from the two 16-bit parts. However, since there is no temporary register for the 32-bit counter value, the high and low words will not be read in the same clock cycle. The low word TC might overflow just after it has been read. This would lead to an incorrect value being read, since the overflow of the low word would lead the high word to be increased by 1. One way to avoid this is to stop the low word TC while reading the 32-bit count value. If stopping the TC is not acceptable, the best solution is to use a 32-bit input capture.

The same problems can occur when a new value is written to the cascaded TC. The safest way to write a new 32-bit value is to stop the TC, write the value and start the TC again.

5.3 Using 32-bit Input Capture

A 32-bit Timer/Counter can be set up to perform 32-bit input capture. First, a 32-bit Timer/Counter must be set up as described in section 5.1. Both TCs are configured

for input capture, with the same event channels as capture sources. However, there is a one clock-cycle delay in the propagation of the overflow bit from the low word TC to the high word TC. This means that it is necessary to delay the event to the high word TC to ensure the correct input capture value. To delay the event, the EVDLY bit in the CTRLD register must be set.

It is possible to use the EVSYS.STROBE register in the event system to manually trigger an event on any of the available event channels. This can be used as an alternative to reading the TC directly from software.

6 Getting Started

This section describes the basic steps for getting up and running with the Timer/Counters in different configurations. Each of the examples listed here are implemented in the accompanying example source code.

6.1 Basic Timer/Counter Operation

Task: Set up a Timer/Counter for use as a regular timer.

1. Set the PER[H:L] register to control the period/top value of the TC. This sets the point where the TC wraps around to zero and where the TC overflow interrupt/event occurs.
2. Start TC by selecting a clock source (CLKSEL in CTRLA).

In this configuration, the current timer value can be read directly from the CNT[H:L] register. The TC overflow bit indicates whether an overflow has occurred. This bit can be used to generate interrupts at fixed intervals.

6.2 Using the Input Capture Functionality

Task: Configure TCC0 with Input Capture Channel A enabled. The Input Capture is triggered by the falling edge of PC0.

3. Configure PC0 for input, triggered on falling edge.
4. Select PC0 as multiplexer input for event channel 0.
5. Configure TCC0 for input capture by setting event source and event action to "Input capture" in CTRLD.
6. Enable input capture channel A by setting the CCAEN bit in CTRLB.
7. Start the TC by selecting a clock source (CLKSEL in CTRLA).
8. Wait for Input Capture Interrupt Flag A in INTFLAGS to be set.
9. Read input capture value from the CCA[H:L] register.
10. Go to step 6.

6.3 Using Input Capture to Calculate Frequency and Duty Cycle of a Signal

Task: Configure Timer/Counter C0 to measure the frequency and duty cycle of a signal applied to PC0.

1. Select event source(s). See section 3.6 for details on this.
2. Select event action = input capture.
3. Enable input capture channels (CCENx).
4. Set PER[H:L] to 0x7FFF. (MSB must be cleared)



5. Start the TC by selecting a clock source.
6. Wait for Input Capture Interrupt Flag A in INTFLAGS to be set.
7. Read input capture value from the CCA[H:L] register into 'thisCapture'.
8. Check bit 15 of 'thisCapture'.
9. If bit 15 of 'thisCapture' is 1
 - a. Save 'thisCapture' in 'lastRising'
- 10.else
 - a. Calculate total period:

6.4 Using a Timer/Counter for PWM Generation

Task: Configure TCC0 for pulse width modulation output with varying duty cycle on channel A.

1. Configure PC0 for output by setting bit 0 in PORTC.DIR.
2. Select the timer period by setting the PER[H:L] register.
3. Select a waveform generation mode by setting the WGMODE[2:0] bits in
4. Enable Compare Channel A by setting the CCAEN bit in CTRLB.
5. Start the TC by selecting a clock source (CLKSEL[3:0] in CTRLA).
6. Calculate the desired compare value.
7. Write the new compare value to CCA[H:L].
8. Wait for the TC Overflow Flag to be set. (OVFIF in INTFLAGS).
9. Clear the TC Overflow flag.
- 10.Go to step 6.

Using this sequence, the compare value will be updated once every PWM period.

6.5 Event Counting

Task: Configure TCC0 to count the number of switch presses on PC0, using event channel 0. Generate interrupt after 5 key presses that toggle PC1.

1. Configure PC0 as input, sense on falling edge.
2. Configure PC1 as output.
3. Select PC0 as multiplexer input for event channel 0.
4. Enable filtering on event channel 0.
5. Set the period of TCC0 to 4 (to generate overflow interrupt after 5 switch presses).
6. Enable TCC0 overflow interrupt at low level.
7. Select event channel 0 as clock source for the TC.

In the TCC0 overflow interrupt service routine, toggle PC1.

In this configuration, the CNT[H:L] register will contain the number of times a switch connected to PC0 has been pressed. After 5 presses, an overflow interrupt service handler will be triggered, toggling PC1 while CNT[H:L] wraps around to 0.

6.6 Setting up a 32-bit Timer/Counter With Input Capture

Task: Configure TCC0 and TCC1 as one 32-bit TC with input capture channel A triggered by a falling edge on PC0. Event channel 0 is used for overflow propagation, while the input capture signal from PC0 is routed through event channel 1.

1. Configure PC0 for input, triggered on falling edge.
2. Select PC0 as multiplexer input for event channel 1.
3. Select TCC0 overflow as multiplexer input for event channel 0.
4. Configure TCC0 and TCC1 for input capture with event channel 1 as trigger.
5. Enable event delay on TCC1. (This will delay the input capture event by one clock cycle to allow the overflow from TCC0 to propagate to TCC1 before the input capture happens.)
6. Enable Input Capture Channel A on TCC0 and TCC1.
7. Select event channel 0 as clock source for TCC1 (CLKSEL in CTRLA).
8. Start the TCC0 by selecting system clock as clock source (CLKSEL in CTRLA).
9. Wait for TCC0 (or TCC1) input capture interrupt flag A in INTFLAGS to be set.
10. Read low word input capture value from the TCC0.CCA[H:L] register.
11. Read high word input capture value from the TCC1.CCA[H:L] register.
12. Combine low word and high word to a 32-bit input capture value.
13. Go to step 9.

7 Advanced Features

In this application note, the Timer/Counter modules have been used as standalone modules. There are some advanced features on the XMEGA that can be used as “glue” between modules to reduce the amount of code needed and automate parts of the application.

7.1 DMA Controller

Instead of using polling or interrupt handlers to read input capture values, it is possible to use the XMEGA DMA controller to move data from one or more registers to memory buffers or other peripheral modules. This moving of data is done without CPU intervention and leaves the CPU ready for other tasks; even without executing interrupt handlers.

For more information, please refer to the device datasheet or the application note AVR1304.

7.2 Event Generation

The event system has been used in this application note as triggering and clock sources. The XMEGA TCs can be also be used as event sources for other parts of the system. The TC modules can generate the following events

- Timer overflow.
- Timer error. (Input capture data lost because input capture buffer is full).
- Separate events for compare or capture on all channels.





This allows for complex interaction between different parts of the system, independent of any code running. The event system is covered more in-depth in the application note AVR1001.

7.3 Timer/Counter Extension Modules

The Timer/Counter functionality can be extended through Timer/Counter extension modules, such as the Advanced Waveform eXtension (AWeX) and High-Resolution (Hi-Res) extension. The extension modules are not covered in this document. Please refer to application note AVR1311 for more information.

8 Driver Implementation

The included driver has functions that control all the major features of the Timer/Counter modules (including waveform generation). All functions take a pointer to a Timer/Counter module as its first argument, so the same functions can be reused for all Timer/Counter modules on one XMEGA. The driver is written in ANSI® C, and should compile on all compilers with XMEGA support.

Note that this driver is **not** written with high performance in mind. It is designed as a library to get started with the XMEGA Timer/Counters and an easy-to-use framework for rapid prototyping. For time and code space critical application development, consider replacing function calls with macros or direct access to registers.

8.1 Files

The driver package consists of the following files:

- tc_driver.c – Timer/Counter driver source file
- tc_driver.h – Timer/Counter driver header file
- tc_example.c – Examples using the Timer/Counter driver.

8.2 Doxygen Documentation

All source code is prepared for automatic documentation generation using Doxygen. Doxygen is a tool for generating documentation from source code by analyzing the source code and using special keywords. For more details about Doxygen please visit <http://www.doxygen.org>. Precompiled Doxygen documentation is also supplied with the source code accompanying this application note, available from the *readme.html* file in the source code folder.



Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, AVR® and others, are the registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.