APPLICATION NOTE

AT03263: SAM Timer/Counter Driver (TC)

ASF PROGRAMMERS MANUAL

SAM Timer/Counter Driver (TC)

This driver for Atmel® | SMART SAM devices provides an interface for the configuration and management of the timer modules within the device, for waveform generation and timing operations. The following driver API modes are covered by this manual:

- Polled APIs
- Callback APIs

The following peripherals are used by this module:

- TC (Timer/Counter)

The following devices can use this module:

- Atmel | SMART SAM D20/D21
- Atmel | SMART SAM R21
- Atmel | SMART SAM D10/D11

The outline of this documentation is as follows:

- Prerequisites
- Module Overview
- Special Considerations
- Extra Information
- Examples
- API Overview

Table of Contents

Software License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.

4. This software may only be redistributed and used in connection with an Atmel microcontroller product.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1. Prerequisites

There are no prerequisites for this module.

2. Module Overview

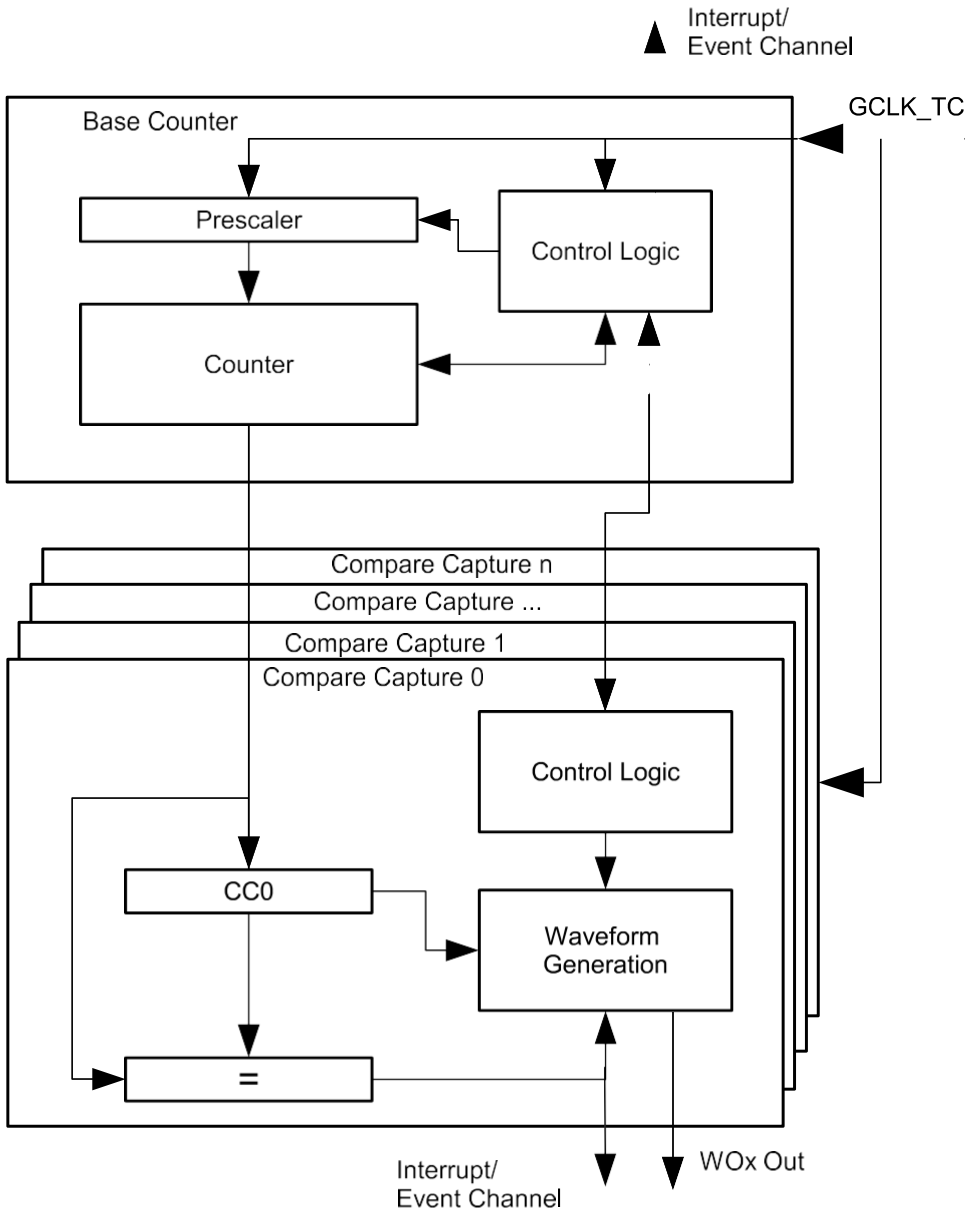
The Timer/Counter (TC) module provides a set of timing and counting related functionality, such as the generation of periodic waveforms, the capturing of a periodic waveform's frequency/duty cycle, and software timekeeping for periodic operations. TC modules can be configured to use an 8-, 16-, or 32-bit counter size.

This TC module for the SAM is capable of the following functions:

- Generation of PWM signals
- Generation of timestamps for events
- General time counting
- Waveform period capture
- Waveform frequency capture

Figure 2-1: Basic Overview of the TC Module on page 7 shows the overview of the TC module design.

Figure 2-1. Basic Overview of the TC Module



2.1 Functional Description

Independent of the configured counter size, each TC module can be set up in one of two different modes; capture and compare.

In capture mode, the counter value is stored when a configurable event occurs. This mode can be used to generate timestamps used in event capture, or it can be used for the measurement of a periodic input signal's frequency/duty cycle.

In compare mode, the counter value is compared against one or more of the configured channel compare values. When the counter value coincides with a compare value an action can be taken automatically by the module, such as generating an output event or toggling a pin when used for frequency or PWM signal generation.

Note The connection of events between modules requires the use of the SAM Event System Driver (EVENTS) to route output event of one module to the the input event of another. For more information on event routing, refer to the event driver documentation.

2.2 Timer/Counter Size

Each timer module can be configured in one of three different counter sizes; 8-, 16-, and 32-bits. The size of the counter determines the maximum value it can count to before an overflow occurs and the count is reset back to zero. Table 2-1: Timer Counter Sizes and Their Maximum Count Values on page 8 shows the maximum values for each of the possible counter sizes.

Table 2-1. Timer Counter Sizes and Their Maximum Count Values

Counter size	Max. (hexadecimal)	Max. (decimal)
8-bit	0xFF	255
16-bit	0xFFFF	65,535
32-bit	0xFFFFFFFF	4,294,967,295

When using the counter in 16- or 32-bit count mode, Compare Capture register 0 (CC0) is used to store the period value when running in PWM generation match mode.

When using 32-bit counter size, two 16-bit counters are chained together in a cascade formation. Except in SAM D10/D11, Even numbered TC modules (e.g. TC0, TC2) can be configured as 32-bit counters. The odd numbered counters will act as slaves to the even numbered masters, and will not be reconfigurable until the master timer is disabled. The pairing of timer modules for 32-bit mode is shown in Table 2-2: TC Master and Slave Module Pairings on page 8.

Table 2-2. TC Master and Slave Module Pairings

Master TC Module	Slave TC Module
TC0	TC1
TC2	TC3
...	...
TCn-1	TCn

In SAMD10/D11, odd numbered TC modules (e.g. TC1) can be configured as 32-bit counters. The even numbered(e.g. TC2) counters will act as slaves to the odd numbered masters.

2.3 Clock Settings

2.3.1 Clock Selection

Each TC peripheral is clocked asynchronously to the system clock by a GCLK (Generic Clock) channel. The GCLK channel connects to any of the GCLK generators. The GCLK generators are configured to use one of the available clock sources on the system such as internal oscillator, external crystals, etc. see the Generic Clock driver for more information.

2.3.2 Prescaler

Each TC module in the SAM has its own individual clock prescaler, which can be used to divide the input clock frequency used in the counter. This prescaler only scales the clock used to provide clock pulses for the counter to count, and does not affect the digital register interface portion of the module, thus the timer registers will synchronize to the raw GCLK frequency input to the module.

As a result of this, when selecting a GCLK frequency and timer prescaler value the user application should consider both the timer resolution required and the synchronization frequency, to avoid lengthy synchronization times of the module if a very slow GCLK frequency is fed into the TC module. It is preferable to use a higher module GCLK frequency as the input to the timer, and prescale this down as much as possible to obtain a suitable counter frequency in latency-sensitive applications.

2.3.3 Reloading

Timer modules also contain a configurable reload action, used when a re-trigger event occurs. Examples of a re-trigger event are the counter reaching the maximum value when counting up, or when an event from the event system tells the counter to re-trigger. The reload action determines if the prescaler should be reset, and when this should happen. The counter will always be reloaded with the value it is set to start counting from. The user can choose between three different reload actions, described in Table 2-3: TC Module Reload Actions on page 9.

Table 2-3. TC Module Reload Actions

Reload action	Description
TC_RELOAD_ACTION_GCLK on page 28	Reload TC counter value on next GCLK cycle. Leave prescaler as-is.
TC_RELOAD_ACTION_PRESC on page 28	Reloads TC counter value on next prescaler clock. Leave prescaler as-is.
TC_RELOAD_ACTION_RESYNC on page 28	Reload TC counter value on next GCLK cycle. Clear prescaler to zero.

The reload action to use will depend on the specific application being implemented. One example is when an external trigger for a reload occurs; if the TC uses the prescaler, the counter in the prescaler should not have a value between zero and the division factor. The TC counter and the counter in the prescaler should both start at zero. When the counter is set to re-trigger when it reaches the maximum value on the other hand, this is not the right option to use. In such a case it would be better if the prescaler is left unaltered when the re-trigger happens, letting the counter reset on the next GCLK cycle.

2.4 Compare Match Operations

In compare match operation, Compare/Capture registers are used in comparison with the counter value. When the timer's count value matches the value of a compare channel, a user defined action can be taken.

2.4.1 Basic Timer

A Basic Timer is a simple application where compare match operations is used to determine when a specific period has elapsed. In Basic Timer operations, one or more values in the module's Compare/Capture registers are used to specify the time (as a number of prescaled GCLK cycles) when an action should be taken by the microcontroller. This can be an Interrupt Service Routine (ISR), event generator via the event system, or a software flag that is polled via the user application.

2.4.2 Waveform Generation

Waveform generation enables the TC module to generate square waves, or if combined with an external passive low-pass filter; analog waveforms.

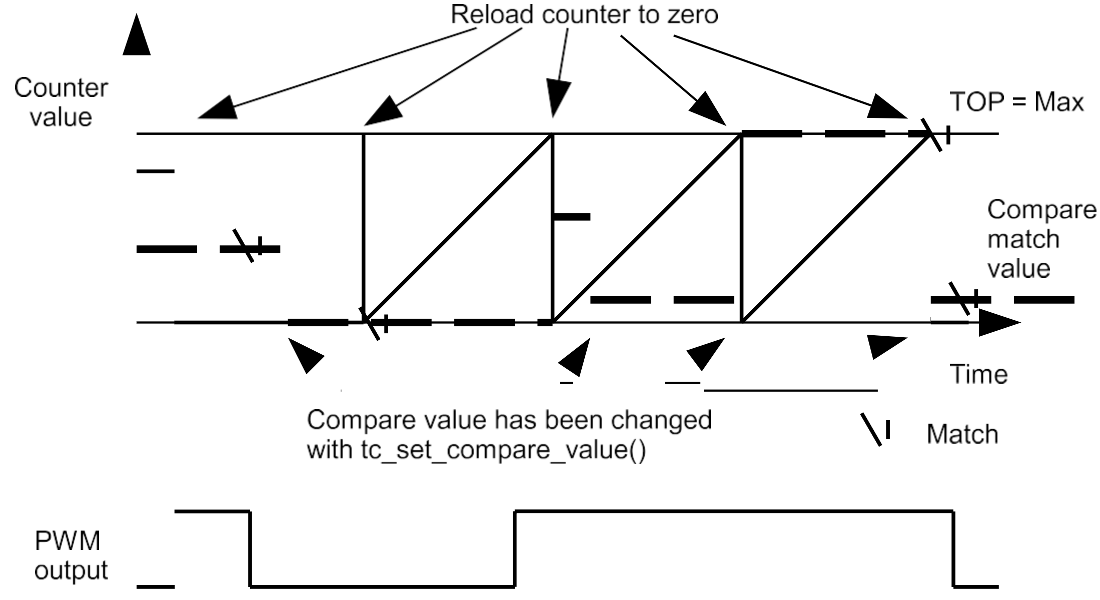
2.4.3 Waveform Generation - PWM

Pulse width modulation is a form of waveform generation and a signalling technique that can be useful in many situations. When PWM mode is used, a digital pulse train with a configurable frequency and duty cycle can be generated by the TC module and output to a GPIO pin of the device.

Often PWM is used to communicate a control or information parameter to an external circuit or component. Differing impedances of the source generator and sink receiver circuits is less of an issue when using PWM compared to using an analog voltage value, as noise will not generally affect the signal's integrity to a meaningful extent.

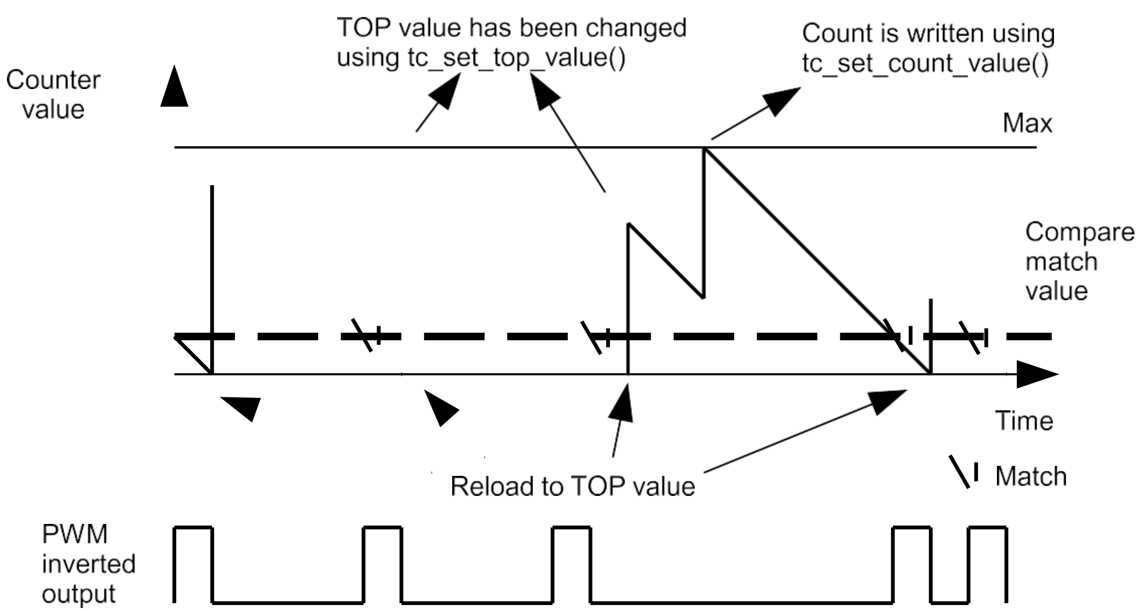
Figure 2-2: Example of PWM in Normal Mode, and Different Counter Operations on page 10 illustrates operations and different states of the counter and its output when running the counter in PWM normal mode. As can be seen, the TOP value is unchanged and is set to MAX. The compare match value is changed at several points to illustrate the resulting waveform output changes. The PWM output is set to normal (i.e. non-inverted) output mode.

Figure 2-2. Example of PWM in Normal Mode, and Different Counter Operations



In Figure 2-3: Example of PWM in Match Mode, and Different Counter Operations on page 10, the counter is set to generate PWM in Match mode. The PWM output is inverted via the appropriate configuration option in the TC driver configuration structure. In this example, the counter value is changed once, but the compare match value is kept unchanged. As can be seen, it is possible to change the TOP value when running in PWM match mode.

Figure 2-3. Example of PWM in Match Mode, and Different Counter Operations



2.4.4 Waveform Generation - Frequency

Frequency Generation mode is in many ways identical to PWM generation. However, in Frequency Generation a toggle only occurs on the output when a match on a capture channels occurs. When the match is made, the timer value is reset, resulting in a variable frequency square wave with a fixed 50% duty cycle.

2.4.5 Capture Operations

In capture operations, any event from the event system or a pin change can trigger a capture of the counter value. This captured counter value can be used as a timestamp for the event, or it can be used in frequency and pulse width capture.

2.4.6 Capture Operations - Event

Event capture is a simple use of the capture functionality, designed to create timestamps for specific events. When the TC module's input capture pin is externally toggled, the current timer count value is copied into a buffered register which can then be read out by the user application.

Note that when performing any capture operation, there is a risk that the counter reaches its top value (MAX) when counting up, or the bottom value (zero) when counting down, before the capture event occurs. This can distort the result, making event timestamps to appear shorter than reality; the user application should check for timer overflow when reading a capture result in order to detect this situation and perform an appropriate adjustment.

Before checking for a new capture, TC_STATUS_COUNT_OVERFLOW should be checked. The response to an overflow error is left to the user application, however it may be necessary to clear both the capture overflow flag and the capture flag upon each capture reading.

2.4.7 Capture Operations - Pulse Width

Pulse Width Capture mode makes it possible to measure the pulse width and period of PWM signals. This mode uses two capture channels of the counter. This means that the counter module used for Pulse Width Capture can not be used for any other purpose. There are two modes for pulse width capture; Pulse Width Period (PWP) and Period Pulse Width (PPW). In PWP mode, capture channel 0 is used for storing the pulse width and capture channel 1 stores the observed period. While in PPW mode, the roles of the two capture channels is reversed.

As in the above example it is necessary to poll on interrupt flags to see if a new capture has happened and check that a capture overflow error has not occurred.

2.5 One-shot Mode

TC modules can be configured into a one-shot mode. When configured in this manner, starting the timer will cause it to count until the next overflow or underflow condition before automatically halting, waiting to be manually triggered by the user application software or an event signal from the event system.

2.5.1 Wave Generation Output Inversion

The output of the wave generation can be inverted by hardware if desired, resulting in the logically inverted value being output to the configured device GPIO pin.

3. Special Considerations

The number of capture compare registers in each TC module is dependent on the specific SAM device being used, and in some cases the counter size.

The maximum amount of capture compare registers available in any SAM device is two when running in 32-bit mode and four in 8- and 16-bit modes.

4. Extra Information

For extra information, see Extra Information for TC Driver. This includes:

- Acronyms
- Dependencies
- Errata
- Module History