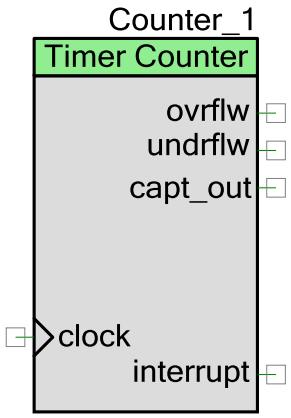
Timer / Counter (TCPWM_Counter_PDL)

1.0

Features



- 16- or 32-bit Timer/Counter
- Programmable Period Register
- Programmable Compare Register
  - Compare value can be swapped with a buffered compare value on comparison event
- Input Capture with buffer register
- Count Up, Count Down, or Count Up and Down Counting Modes
- Continuous or One Shot Run Modes
- Interrupt and Output on Overflow, Underflow, Capture, or Compare
- Start, Reload, Stop, Capture, and Count Inputs

General Description

The TCPWM_Counter_PDL Component is a graphical configuration entity built on top of the cy_tcpwm driver available in the Peripheral Driver Library (PDL). It allows schematic-based connections and hardware configuration as defined by the Component Configure dialog.

The TCPWM_Counter_PDL Component allows rapid configuration of the TCPWM hardware for Timer/Counter functionality. This Component provides a method to measure time intervals or count external events.

When to Use a TCPWM_Counter_PDL Component

This Component should be used whenever a specific timing interval or measurement is needed. Examples include:

- Creating a periodic interrupt for running other system tasks
- Measuring frequency of an input signal
- Measuring pulse width of an input signal

- Measuring time between two external events
- Counting events
- Triggering other system resources after x number events
- Capturing time stamps when events occur

Quick Start

1. Drag a TCPWM_Counter_PDL Component from the Cypress/Digital/Function/ folder in the Component Catalog onto your schematic (the placed instance takes the name Counter_1).
2. Double-click to open the Configure dialog, and configure the Component as required.
3. There is no internal clock in this Component. You must attach a clock source. The clock prescaler functionality is available within the TCPWM_Counter_PDL Component to further divide down the clock.
4. Build the project in order to verify the correctness of your design, add the required PDL modules to the Workspace Explorer, and generate the configuration data for the Counter_1 instance.
5. In *main.c*, initialize the peripheral and start the application using driver APIs and Component Preprocessor Macros.

   If no of the input terminal (start) is used, the software event Cy_TCPWM_TriggerStart or Cy_TCPWM_TriggerReloadOrIndex must be called to start the counting. For example:

   ```
   (void)Cy_TCPWM_Counter_Init(Counter_1_HW, Counter_1_CNT_NUM,
   &Counter_1_config);
   Cy_TCPWM_Enable_Multiple(Counter_1_HW, Counter_1_CNT_MASK);
   Cy_TCPWM_TriggerStart(Counter_1_HW, Counter_1_CNT_MASK);
   ```

6. Build and program the device.

Input/Output Connections

This section describes the various input and output connections for the TCPWM_Counter_PDL Component. An asterisk (*) in the following list indicates that it may not be shown on the Component symbol for the conditions listed in the description of that I/O.

Name	I/O Type	Description
clock	Digital Input	The clock input defines the operating frequency of this Component.
reload* [1]	Digital Input	This input triggers a reload of the counter, and starts the counter. This input is only visible if the ReloadInput parameter is set to anything other than disabled. In up counting mode, the counter is initialized with “0.” For up/down counting modes, the counter is initialized with “1”. In down counting mode, the counter is initialized with the period value. This should only be used when the counter is not running.
count*	Digital Input	This input causes the counter to count depending on how the input is configured. For example, if this input is configured for level sensitive, the counter will change its count on each pre-scaled clock edge. This input is only visible if the CountInput parameter is set to anything other than disabled.
start* [1]	Digital Input	This input starts the counter. This input is only visible if the StartInput parameter is set to anything other than disabled. This should only be used when the counter is not running.
stop* [1]	Digital Input	This input stops the counter. This input is only visible if the StopInput parameter is set to anything other than disabled.
capture* [1]	Digital Input	This input triggers a capture of the current count value. This input is only visible if the CaptureInput parameter is set to anything other than disabled.
ovrflw	Digital Output	This output goes high when the count value overflows from the period to zero. Reload will also generate an Overflow when up counting. See Overflow (OV) in the TRM.
undrflw	Digital Output	This output goes high when the count value rolls over from zero to the period value. Reload will also generate an underflow event in down and up/down counting modes. See Underflow (UN) in the TRM,
capt_out*	Digital Output	This output goes high when a capture occurs. It is only visible if capture mode is selected. See Compare/Capture, cc_match (CC) in the TRM.

1 The input event will take effect on the next counter clock when the count event becomes active (it depends on the edge detection mode of count event). For example, if you are using an external count signal and an external start signal, the first Active count signal won't be counted; it will be used to start the counter, so your counts will be off by one.

Name	I/O Type	Description
compare*	Digital Output	This output goes high when the compare value equals the Count Value. This event is generated when either of the following occurs: • The match is about to occur (COUNTER does not equal COMPARE), and is changed to COMPARE, in the up or down counting modes, or • The match is about to not occur (COUNTER equals COMPARE), and is changed to a different value, in the up/down counting modes. It is only visible if compare mode is selected.
interrupt	Digital Output	This output can only connect to an interrupt.

Component Parameters

The TCPWM_Counter_PDL Component Configure dialog allows you to edit the configuration parameters for the Component instance.

Basic Tab

This tab contains the Component parameters used in the general peripheral initialization settings.

Parameter Name	Description
Resolution	Selects the size of the counter
ClockPrescaler	Divides down the input clock
RunMode	If Continuous is selected counter runs forever. If One Shot is selected counter runs for one period and stops
CountDirection	Selects the direction the counter counts
Period	Sets the period of the Timer/Counter. Range: 0-65535 (for 16 bit resolution) or 0–4294967295 (for 32 bit resolution).
CompareOrCapture	Selects the mode for the compare capture register
CaptureInput	This parameter determines if a Capture terminal is displayed on the schematic
Compare0	Sets the compare value. Range: 0-65535 (for 16 bit resolution) or 0–4294967295 (for 32 bit resolution).
EnableCompareSwap	When selected the compare register is swapped between compare 0 and compare 1 each time the comparison is true
Compare1	Sets the compare value. Range: 0-65535 (for 16 bit resolution) or 0–4294967295 (for 32 bit resolution). COMPARE_BUFF in TRM.
InterruptSource	Selects which events can trigger an interrupt.

Inputs Tab

This tab contains the Input configuration settings.

Parameter Name	Description
ReloadInput	Determines if a reload input is needed and how the reload signal input is registered
StopInput	Determines if a stop input is needed and how that input is registered
StartInput	Determines if a start input is needed and how that input is registered
CountInput	Determines if a count input is needed and how that input is registered

Application Programming Interface

The Application Programming Interface (API) is provided by the cy_tcpwm driver module from the PDL. The driver is copied into the “pdl\drivers\peripheral\tcpwm\” directory of the application project after a successful build.

Refer to the PDL documentation for a detailed description of the complete API. To access this document, right-click on the Component symbol on the schematic and choose the “**Open PDL Documentation…**” option in the drop-down menu.

This Component generates the configuration structures and base address, needed in the PDL API, these are described in the Global variables and Preprocessor Macros sections. Pass the generated data structure and the base address to the associated cy_tcpwm driver function in the application initialization code to configure the peripheral. Once the peripheral is initialized, the application code can perform run-time changes by referencing the provided base address in the driver API functions.

By default, PSoC Creator assigns the instance name "Counter_1" to the first instance of a Component in a given design. It can be renamed to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol.

Global Variables

The TCPWM_Counter_PDL Component populates the following peripheral initialization data structure(s). The generated code is placed in C source and header files that are named after the instance of the Component (e.g. Counter_1.c). Each variable is also prefixed with the instance name of the Component.

cy_stc_tcpwm_counter_config_t Counter_1_config

The instance-specific configuration structure. This should be used in the PDL Init() function.

Preprocessor Macros

The TCPWM_Counter_PDL Component generates the following preprocessor macro(s). Note that each macro is prefixed with the instance name of the Component (e.g. “Counter_1”).

#define Counter_1_HW (Counter_1_TCPWM__HW)

This is a pointer to the base address of the TCPWM instance

#define Counter_1_CNT_HW (Counter_1_TCPWM__CNT_HW)

This is a pointer to the base address of the TCPWM CNT instance

#define Counter_1_CNT_NUM (Counter_1_TCPWM__CNT_IDX)

This is the counter instance number in the selected TCPWM

#define Counter_1_CNT_MASK (1uL << Counter_1_TCPWM__CNT_IDX)

This is the bit field representing the counter instance in the selected TCPWM

Data in RAM

The generated data may be placed in flash memory (const) or RAM. The former is the more memory-efficient choice if you do not wish to modify the configuration data at run-time. Under the Built-In tab of the Configure dialog set the parameter CY_CONST_CONFIG to make your selection. The default option is to place the data in flash.

Code Examples and Application Notes

Code Examples

PSoC Creator provides access to code examples in the Find Code Example dialog. For Component-specific examples, open the dialog from the Component Catalog or an instance of the Component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the "Find Code Example" topic in the PSoC Creator Help for more information.

There are also numerous code examples that include schematics and example code available online at the Cypress Code Examples web page.

API Memory Usage

The Component is designed to use API from the cy_tcpwm PDL module. That is why the Component itself only consumes resources necessary to allocate structures for driver operation and start the Component.

Functional Description

Clock Selection

There is no internal clock in this Component. You must attach a clock source. One of the peripheral clock dividers (PeriClk) should be used as a clock source. The clock prescaler functionality is available within the TCPWM_Counter_PDL Component.

DMA Support

The DMA Component can be used to transfer data from the Component registers to RAM or another Component.

Name of DMA Source / Destination	Width	Direction	DMA Req Signal	DMA Req Type	Description
Counter_1_CNT_HW ->COUNTER	32	Source / Destination	N/A	N/A	Count register. It is not advised to write to this register when the counter is running.
Counter_1_CNT_HW ->CC	32	Source / Destination	N/A	N/A	Counter compare 0/capture register.
Counter_1_CNT_HW ->CC_BUFF	32	Source / Destination	N/A	N/A	Counter buffered capture / (compare 1) register.
Counter_1_CNT_HW ->PERIOD	32	Source / Destination	N/A	N/A	Period register. To cause the counter to count for N cycles this register should be written with N-1 (counts from 0 to period inclusive).
Counter_1_CNT_HW ->PERIOD_BUFF	32	Source / Destination	N/A	N/A	Period buffered register. To cause the counter to count for N cycles this register should be written with N-1 (counts from 0 to period inclusive).

Industry Standards

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the Component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator Components
- specific deviations – deviations that are applicable only for this Component

This section provides information on Component-specific deviations. Refer to PSoC Creator Help > Building a PSoC Creator Project > Generated Files (PSoC 6) for information on MISRA compliance and deviations for files generated by PSoC Creator.

The TCPWM_Counter_PDL Component does not have any specific deviations.

This Component uses firmware drivers from the cy_tcpwm PDL module. Refer to the PDL documentation for information on their MISRA compliance and specific deviations.

Registers

See the Timer, Counter (CNT) Registers section in the chip Technical Reference Manual (TRM) for more information about the registers.

Resources

The TCPWM_Counter_PDL Component uses the Timer Counter Pulse Width Modulation (TCPWM) peripheral block.

DC and AC Electrical Characteristics

Refer to the Digital Peripherals in the Electrical Specifications section of the Device Family Datasheet.

Component Changes

This section lists the major changes in the Component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.0.c	Minor datasheet edits.	
1.0.b	Minor datasheet update.	
1.0.a	Datasheet edits to clarify functionality.	
1.0	Initial Version	

© Cypress Semiconductor Corporation, 2016-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical Components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical Component is any Component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.