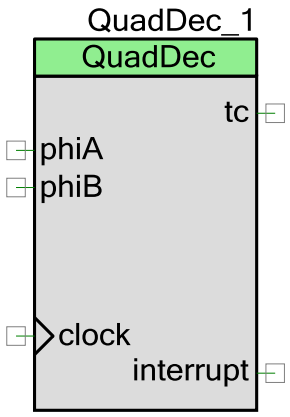
Quadrature Decoder (TCPWM_QuadDec_PDL)

1.0

Features

- 16- or 32-bit Counter
- Counter Resolution of x1, x2, and x4 the frequency of phiA and phiB inputs
- Index Input to determine absolute position
- Peripheral Driver Library (PDL) Component (PDL Application Programming Interface (API) only)



General Description

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

The TCPWM_QuadDec_PDL Component gives you the ability to count transitions on a pair of digital signals. The signals are typically provided by a speed/position feedback system mounted on a motor or trackball.

The signals, typically called phiA and phiB, are positioned 90 degrees out of phase, which results in a Gray code output. A Gray code is a sequence where only one bit changes on each count. This is essential to avoid glitches. It also allows detection of direction and relative position. A third optional signal, named Index, is used as a reference to establish an absolute position once per rotation.

When to Use a TCPWM_QuadDec_PDL Component

A quadrature decoder is used to decode the output of a quadrature encoder. A quadrature encoder senses the current position, velocity, and direction of an object (for example, mouse, trackball, robotic axles, and others).

A quadrature decoder can also be used for precision measurement of speed, acceleration, and position of a motor's rotor and with rotary knobs to determine user input.

Quick Start

1. Drag a TCPWM_QuadDec_PDL Component from the Cypress/Digital/Function/ folder in the Component Catalog onto your schematic (the placed instance takes the name QuadDec_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.
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 QuadDec_1 instance.
5. In *main.c*, initialize the peripheral and start the application using driver APIs and Component Preprocessor Macros.
   If none of the input terminal are used, the software event Cy_TCPWM_TriggerReloadOrIndex must be called to start counting. For example:

```
(void)Cy_TCPWM_QuadDec_Init(QuadDec_1_HW, QuadDec_1_CNT_NUM,
&QuadDec_1_config);
Cy_TCPWM_Enable_Multiple(QuadDec_1_HW, QuadDec_1_CNT_MASK);
Cy_TCPWM_TriggerReloadOrIndex(QuadDec_1_HW, QuadDec_1_CNT_MASK);
```

6. Build and program the device.

Input/Output Connections

This section describes the various input and output connections for the TCPWM_QuadDec_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 IO.

Name	Type	Description
Clock	Digital Input	The clock input defines the operating frequency of this Component.
index *	Digital Input	This input causes an index (reload) event to occur (indicates a completed rotation to determine absolute position). This input is only visible if the IndexInput parameter is set to anything other than disabled.
stop *	Digital Input	This input causes the Quad Dec to stop Counting. This input is only visible if the StopInput parameter is set to anything other than disabled.
phiA	Digital Input	This is the phase A input it is always required.
phiB	Digital Input	This is the phase B input it is always required.
tc	Digital Output	This output goes high on an index event, or when the count equals 0x0000 or 0xffff(ffff) (for 32 bit mode)
ìnterrupt	Digital Output	Selects which events can trigger an interrupt.

Component Parameters

The TCPWM_QuadDec_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
CounterSize	Sets the resolution of the counter
Resolution	Selects the quadrature encoding mode
InterruptSource	Selects which events can trigger an interrupt
IndexInput	Determines if an index is needed and what edge causes an index to occur
StopInput	Determines if a stop input is needed and what edges causes a stop to occur

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the Component using software.

By default, PSoC Creator assigns the instance name "QuadDec_1" to the first instance of a Component in a given design. You can rename it 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.

This Component uses 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.

Global Variables

The TCPWM_QuadDec_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., *QuadDec_1.c*). Each variable is also prefixed with the instance name of the Component.

cy_stc_tcpwm_quaddec_config_t QuadDec_1_config

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

Preprocessor Macros

The TCPWM_QuadDec_PDL Component generates the following preprocessor macro(s). Note that each macro is prefixed with the instance name of the Component (e.g. “QuadDec_1”).

#define QuadDec_1_HW (QuadDec_1_TCPWM__HW)

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

#define QuadDec_1_CNT_HW (QuadDec_1_TCPWM__CNT_HW)

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

#define QuadDec_1_CNT_NUM (QuadDec_1_TCPWM__CNT_IDX)

This is the counter instance number in the selected TCPWM.

#define QuadDec_1_CNT_MASK (1uL << QuadDec_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 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 "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 (PeriClk) dividers should be used as a clock source.

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
QuadDec_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.
QuadDec_1_CNT_HW ->CC	32	Source / Destination	N/A	N/A	Counter capture register.
QuadDec_1_CNT_HW ->CC_BUFF	32	Source / Destination	N/A	N/A	Counter buffered capture register.
QuadDec_1_CNT_HW ->PERIOD	32	Source / Destination	N/A	N/A	Period 0 register. To cause the counter to count for N cycles this register should be written with N-1 (counts from 0 to period inclusive).
QuadDec_1_CNT_HW ->PERIOD_BUFF	32	Source / Destination	N/A	N/A	Period 1register. 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. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The TCPWM_QuadDec_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 Registers section in the chip Technical Reference Manual (TRM) for more information about the registers.

Resources

The TCPWM_QuadDec_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.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.