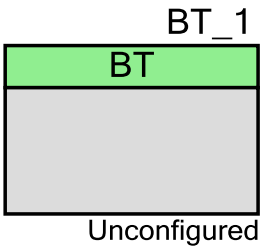
Base Timer Channel (BT)

1.0

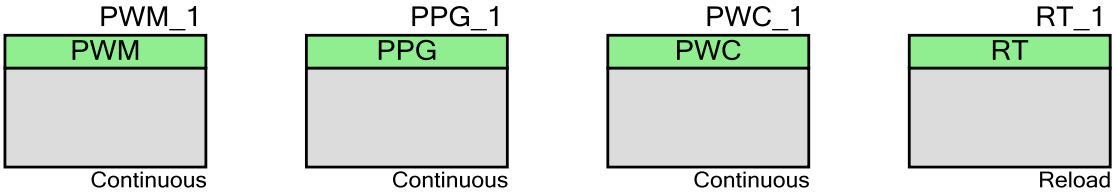


Features

- Four operating modes
  - 16-bit PWM Timer
  - 16-bit PPG Timer
  - 16/32-bit Reload Timer
  - 16/32-bit PWC Timer
- Trigger generation for ADC conversion

General Description

The Peripheral Driver Library (PDL) Base Timer (PDL_BT) component allows configuring a single channel of the BT peripheral block. The BT is a multifunction peripheral block with the following operational modes: PWM, PPG, PWC, RT. Each mode is available as a pre-configured schematic in the PSoC Creator Component Catalog.



There is also unconfigured BT component entry in the Component Catalog.

This component uses firmware drivers from the PDL_BT module, which is automatically added to your project after a successful build.

When to Use a PDL_BT Component

The default use of the PDL_BT component is to generate a periodic event or interrupt signal. However, there are other potential uses, such as creating clock dividers or measuring the length of time between hardware events.

- A Counter mode is better used in situations focused on counting events.
- A PWM mode is better used in situations requiring compare outputs.

- A Timer is typically used to record the number of clock cycles between events. An example of this is measuring the number of clocks between two rising edges as might be generated by a tachometer sensor.
- The PPG is used for arbitrary waveform generation.

Quick Start

Follow the quick start steps below to configure and initialize the BT peripheral block.

1. Drag a PDL_BT component from the Component Catalog FMx/Digital/ folder onto your schematic. The placed instance takes the name BT_1.
2. Double-click to open the component's Configure dialog.
3. On the **Basic** tab set the needed configuration.
4. On the **Interrupts** tab, initialize needed interrupts and their callback functions.
5. Depending on the selected component configuration, there will be added tab(s) with specific parameters.
6. Assign pins in your device using the Pin Editor. If you are creating a design for a development kit, refer the kit User Guide for suitable pin assignments.
7. Build the project to verify the correctness of your design. This will add the required PDL modules to the Workspace Explorer and generate configuration data for the BT_1 instance.
8. In the *main.c* file, initialize the peripheral and start the application. The example assumes the BT configured in PWM mode.

```
BT_1_SetPinFunc_TIOA_OUT();
Bt_Pwm_Init(&BT_1_HW, &BT_1_Config);
Bt_Pwm_WriteCycleVal(&BT_1_HW, val0);/* Set cycle value */
Bt_Pwm_WriteDutyVal(&BT_1_HW, val1);/* Set duty cycle value*/
Bt_Pwm_EnableCount(&BT_1_HW);
```

9. Build and program the device.

Component Parameters

The PDL_BT 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 basic peripheral initialization settings.

Parameter Name	Description
BTConfig	Selects the timer operating mode to one of the following modes: ▪ PWM - configured to be in PWM mode. ▪ PPG - configured to be in PPG mode. ▪ PWC - configured to be in PWC mode. ▪ RT - configured to be in RT mode. ▪ Unconfigured - This is the default mode. The BT component must be configured at run-time when configured in this mode.
ConnectTIOA	Connect the TIOA signal to a pin.
ConnectTIOB	Connect the TIOB signal to a pin.

Interrupts Tab

This tab contains the Interrupts configuration settings.

Parameter Name	Description
bTouchNvic	Update the NVIC with the BT interrupts.
bPpgTrigIrq	Assert an interrupt on a trigger event. Visible when PPG mode selected.
bPpgUnderflowIrq	Assert an interrupt on a timer underflow. Visible when PPG mode selected.
pfnPpgTrigIrqCb	Trigger event interrupt callback. Note: this generates a declaration only - USER must implement the function. Visible when PPG mode selected.
pfnPpgUnderflowIrqCb	Timer underflow event interrupt callback. Note: this generates a declaration only - USER must implement the function. Visible when PPG mode selected.
bPwcMeasureOverflowIrq	Assert an interrupt on a measurement overflow. Visible when PWC mode selected.
pfnPwcMeasureOverflowIrqCb	Measurement overflow event interrupt callback. Note: this generates a declaration only - USER must implement the function. Visible when PWC mode selected.
bPwcMeasureCompleteIrq	Assert an interrupt on a measurement event. Visible when PWC mode selected.
pfnPwcMeasureCompleteIrqCb	Measurement complete interrupt callback. Note: this generates a declaration only - USER must implement the function. Visible when PWC mode selected.

Parameter Name	Description
bPwmDutyMatchIrq	Assert an interrupt when timer matches the duty cycle. Visible when PWM mode selected.
pfnPwmDutyMatchIrqCb	Duty match event interrupt callback. Note: this generates a declaration only - USER must implement the function. Visible when PWM mode selected.
bPwmTrigIrq	Assert an interrupt on a trigger event. Visible when PWM mode selected.
pfnPwmTrigIrqCb	Trigger event interrupt callback. Note: this generates a declaration only - USER must implement the function. Visible when PWM mode selected.
bPwmUnderflowIrq	Assert an interrupt on a timer underflow. Visible when PWM mode selected.
pfnPwmUnderflowIrqCb	Timer underflow event interrupt callback. Note: this generates a declaration only - USER must implement the function. Visible when PWM mode selected.
bRtTrigIrq	Assert an interrupt on a trigger event. Visible when RT mode selected.
pfnRtTrigIrqCb	Trigger event interrupt callback. Note: this generates a declaration only - USER must implement the function. Visible when RT mode selected.
bRtUnderflowIrq	Assert an interrupt on a timer underflow. Visible when RT mode selected.
pfnRtUnderflowIrqCb	Timer underflow event interrupt callback. Note: this generates a declaration only - USER must implement the function. Visible when RT mode selected.

PPG Tab

This tab contains the PPG configuration settings. Visible when PPG mode selected.

Parameter Name	Description
enPpgExtTrig	Select an external trigger and choose the triggering edge.
enPpgMode	PPG mode.
enPpgOutputMask	PPG output mask.
enPpgOutputPolarity	Initial state of PPG output.
enPpgPres	Apply a divider to the internal clock or use an external source.
enPpgRestartEn	PPG restart option.

PWC Tab

This tab contains the PWC configuration settings. Visible when PWC mode selected.

Parameter Name	Description
enPwcMeasureEdge	Select the edges to start and stop the measurements.
enPwcMode	PWC mode.

Parameter Name	Description
enPwcPres	Apply a divider to the internal clock or use an external source.
enPwcSize	PWC counter size.

PWM Tab

This tab contains the PWM configuration settings. Visible when PWM mode selected.

Parameter Name	Description
enPwmExtTrig	Select an external trigger and choose the triggering edge.
enPwmMode	PWM mode.
enPwmOutputMask	PWM output mask.
enPwmOutputPolarity	Initial state of PWM output.
enPwmPres	Apply a divider to the internal clock or use an external source.
enPwmRestartEn	PWM restart option.

RT Tab

This tab contains the RT configuration settings. Visible when RT mode selected.

Parameter Name	Description
enRtExtTrig	Select an external trigger and choose the triggering edge/level.
enRtMode	Reload Timer mode.
enRtOutputPolarity	Initial state of Reload Timer output.
enRtPres	Apply a divider to the internal clock or use an external source.
enRtSize	RT counter size.

Component Usage

After a successful build, firmware drivers from the PDL_BT module are added to your project in the pdl/drivers/bt folder. Pass the generated data structures to the associated PDL functions in your application initialization code to configure the peripheral.

Generated Data

The PDL_BT 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. *BT_1_config.c*). Each variable is also prefixed with the instance name of the component.

Data Structure Type	Name	Description
stc_pwm_irq_en_t	BT_1_IrqEn	Interrupt enable structure. Generated when PWM mode selected.
stc_ppg_irq_en_t	BT_1_IrqEn	Interrupt enable structure. Generated when PPG mode selected.
stc_pwc_irq_en_t	BT_1_IrqEn	Interrupt enable structure. Generated when PWC mode selected.
stc_rt_irq_en_t	BT_1_IrqEn	Interrupt enable structure. Generated when RT mode selected.
stc_pwm_irq_cb_t	BT_1_IrqCb	Interrupt callback function. Generated when PWM mode selected.
stc_ppg_irq_cb_t	BT_1_IrqCb	Interrupt callback function. Generated when PPG mode selected.
stc_pwc_irq_cb_t	BT_1_IrqCb	Interrupt callback function. Generated when PWC mode selected.
stc_rt_irq_cb_t	BT_1_IrqCb	Interrupt callback function. Generated when RT mode selected.
stc_bt_pwm_config_t	BT_1_Config	Configuration structure. Generated when PWM mode selected.
stc_bt_ppg_config_t	BT_1_Config	Configuration structure. Generated when PPG mode selected.
stc_bt_pwc_config_t	BT_1_Config	Configuration structure. Generated when PWC mode selected.
stc_bt_rt_config_t	BT_1_Config	Configuration structure. Generated when RT mode selected.

Once the component is initialized, the application code should use the peripheral functions provided in the referenced PDL files. Refer to the PDL documentation for the list of provided API functions. To access this document, right-click on the component symbol on the schematic and choose “**Open API Documentation…**” option in the drop-down menu.

Preprocessor Macros

The BT component generates the following preprocessor macro(s). Note that each macro is prefixed with the instance name of the component (e.g. “BT_1”).

Macro	Description
BT_1_HW	Hardware pointer to the block instance in the device. This should be used in all API calls to specify the block to access.

Macro	Description
BT_1_SetPinFunc_TIOA()	Macro to assign BT TIOA signal in the device pin. This macro will be generated if the parameter ConnectTIOA was set to *true.*
BT_1_SetPinFunc_TIOB()	Macro to assign BT TIOB signal in the device pin. This macro will be generated if the parameter ConnectTIOB was set to *true.*
BT_1_BT_CONFIG	BT operational mode. Is set to one of the following values based on the BTConfig parameter option: BT_1_BT_UNCONFIGURED , BT_1_BT_PWM, BT_1_BT_PPG, BT_1_BT_RT or BT_1_BT_PWC.

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 CONST_CONFIG to make your selection. The default option is to place the data in flash.

Interrupt Support

If the PDL_BT component is specified to trigger interrupts, it will generate the callback function declaration that will be called from the BT ISR. The user is then required to provide the actual callback code. If a null string is provided the struct is populated with zeroes and the callback declaration is not generated. In that case it is the user's responsibility to modify the struct in firmware.

The component generates the following function declarations.

Function Callback	Description
BT_1_PwmTrigIrqCb	Trigger interrupt callback function. Note: this generates a declaration only - USER must implement the function. This declaration is generated only when PWM mode selected.
BT_1_PwmDutyMatchIrqCb	Duty match interrupt callback function. Note: this generates a declaration only - USER must implement the function. This declaration is generated only when PWM mode selected.
BT_1_PwmUnderflowIrqCb	Underflow interrupt callback function. Note: this generates a declaration only - USER must implement the function. This declaration is generated only when PWM mode selected.
BT_1_PpgTrigIrqCb	Trigger interrupt callback function. Note: this generates a declaration only - USER must implement the function. This declaration is generated only when PPG mode selected.
BT_1_PpgUnderflowIrqCb	Duty match interrupt callback function. Note: this generates a declaration only - USER must implement the function. This declaration is generated only when PPG mode selected.

Function Callback	Description
BT_1_PwcMeasureCompleteIrqCb	Measure completion interrupt callback function. Note: this generates a declaration only - USER must implement the function. This declaration is generated only when PWC mode selected.
BT_1_PwcMeasureOverflowIrqCb	Measure overflow interrupt callback function. Note: this generates a declaration only - USER must implement the function. This declaration is generated only when PWC mode selected.
BT_1_RtTrigIrqCb	Trigger interrupt callback function. Note: this generates a declaration only - USER must implement the function. This declaration is generated only when RT mode selected.
BT_1_RtUnderflowIrqCb	Duty match interrupt callback function. Note: this generates a declaration only - USER must implement the function. This declaration is generated only when RT mode selected.

Code Examples and Application Notes

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

Cypress also provides a number of application notes describing how FMx devices can be integrated into your design. You can access the Cypress Application Notes search web page at www.cypress.com/appnotes.

Resources

The PDL_BT component uses one channel of Base Timer (BT) peripheral block.

References

- FM0+ Family of 32-bit ARM® Cortex®-M0+ Microcontrollers Peripheral Manuals
- Cypress FM0+ Family of 32-bit ARM® Cortex®-M0+ Microcontrollers

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	Initial Version	

© Cypress Semiconductor Corporation, 2016. 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.

