Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE218542 - PSoC 6 MCU Custom Tick Timer Using RTC Alarm

Objective

This example demonstrates a custom tick timer using the periodic Alarm interrupt of the Real-time Clock (RTC).

Overview

This code example demonstrates how to configure the RTC registers for a periodic alarm interrupt using the Peripheral Driver Library (PDL) RTC driver API. A GPIO output is included to toggle the LED to show the period of the interrupt.

Requirements

Tool: PSoC Creator™ 4.2; Peripheral Driver Library (PDL) 3.0.1

Programming Language: C (Arm® GCC 5.4-2016-q2-update)

Associated Parts: All PSoC® 6 MCU parts

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Hardware Setups

This example uses the kit's default configuration. See the kit guide to ensure the kit is configured correctly.

Software Setup

None

Operation

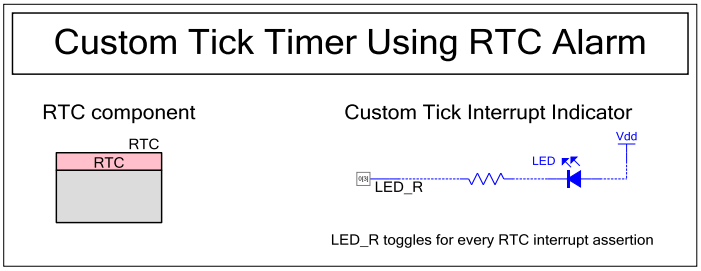
1. Open terminal software such as Tera Term and select the KitProg2's COM port with a baud rate setting of 115200 bps.
2. Build the project and program it into the PSoC 6 MCU device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help. Flash for both CPUs is programmed in a single program operation.
3. Confirm that the red LED (LED_R, P0[3]) toggles every three seconds.
4. Change `TICK_INTERVAL` from (3u) to (1u) in *main_cm4.c* to make a 1-second tick timer.
5. Build the project and program the PSoC 6 MCU device.
6. Confirm that the red LED (P0[3]) toggles every one second.
7. Change the `USE_SECONDS` to (0u) and the `USE_MINUTES` to (1u) to make a 1-minute tick timer.
8. Build the project and program the PSoC 6 MCU device.
9. Confirm that the LED R (P0[3]) toggles every minute.
10. Change `TICK_INTERVAL` from (1u) to (2u) to make a 2-minute tick timer.
11. Build the project and program the PSoC 6 MCU device.
12. Confirm that the LED R (P0[3]) toggles every two minutes.

Design and Implementation

This code example features one RTC and one GPIO for the indicating LED, as shown in Figure 1.

The periodic alarm is a slow timer that can be used as a custom tick timer. The custom timer can wake from either Deep Sleep or Hibernate mode. This is useful for monitoring the status of a peripheral devices over a long period.

Figure 1. Custom Tick Timer Using RTC Alarm



The alarm fields consist of second, minute, hour, day of week, date, and month. An alarm interrupt asserts when the current time fields match the alarm fields. The PDL allows you to disable the individual time fields to ignore them for alarm comparison. This is useful for making a periodic alarm. For example, if the alarm interrupt is required every morning at 7:00 AM, the second, minute, and hour fields are enabled while other time fields are disabled. The shortest interrupt period is one second because the RTC time increases every one second.

As Table 1 shows, you control the interrupt period by enabling or disabling match fields.

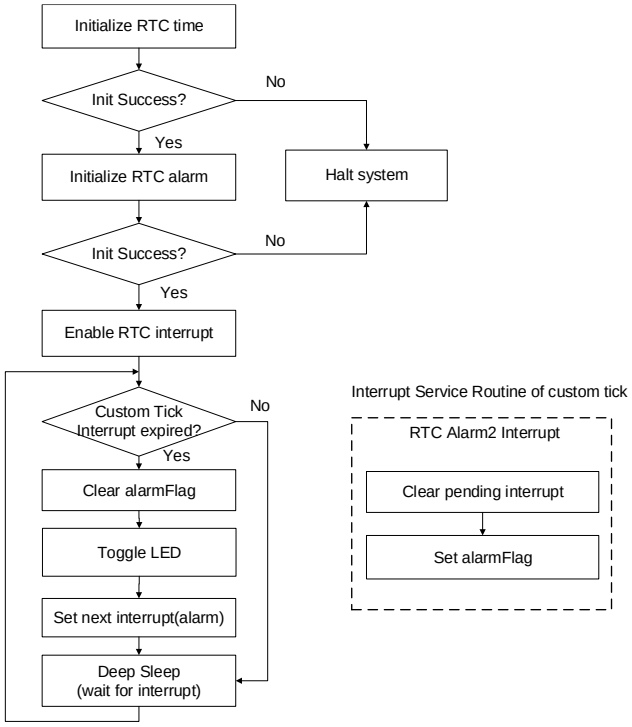
Table 1. Interrupt Period vs. Alarm Field configuration

Interrupt every…	Alarm function	Month	Date	Day of Week	Hour	Minute	Second
Second	**Enabled**	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled
Minute	**Enabled**	Disabled	Disabled	Disabled	Disabled	Disabled	**Enabled**
Hour	**Enabled**	Disabled	Disabled	Disabled	Disabled	**Enabled**	**Enabled**

Although slightly counterintuitive, to have an alarm repeat at any given period, the match for that period is disabled. For an hourly alarm, the alarm goes off when the second and minute fields match, regardless of the hour. So, you do not want to match the hour.

Figure 2 shows the firmware flow of this code example.

Figure 2. Flowchart for the Custom Tick Timer Using RTC Alarm



Design Considerations

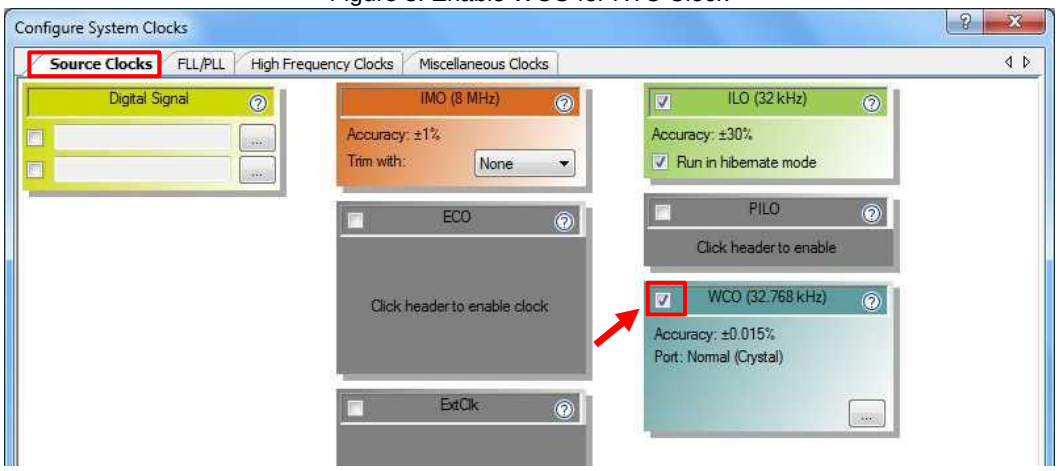
This code example runs on CY8CKIT-062-BLE, which has a PSoC 6 MCU device.

It is necessary to have a 32.768-kHz clock source for the RTC function in the backup power domain. For accurate RTC operation, it is recommended that you use a Watch Crystal Oscillator (WCO).

Perform the following steps to configure the RTC clock source (BakClk) as WCO.

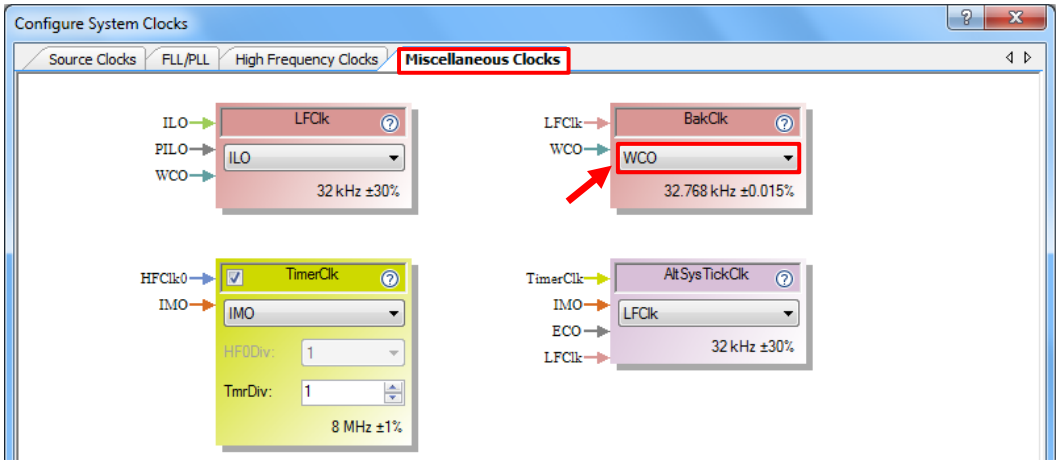
1. Double-click **locks** in Design Wide Resources.
2. Click **Edit Clock…** and open **Configure System Clocks**.
3. Enable WCO clock for the backup clock source in **Source Clocks**, as Figure 3 shows.

Figure 3. Enable WCO for RTC Clock



4. Select **WCO** for **BakClk** in the **Miscellaneous Clocks** tab, as Figure 4 shows.

Figure 4. Set the Backup Clock Source to WCO



Components

Table 2 lists the PSoC Creator Components used in this example, as well as hardware resources used by each.

Table 2. List of PSoC Creator Components

Component	Instance Name	Purpose	Parameter
Real Time Clock (RTC)	RTC	Provide current date and time information, and alarm functionality.	**[General tab]:** Enable Interrupt: Check
Digital Output Pin	LED_R	Provide visual feedback	**[General tab]:** HW connection: Uncheck Drive Mode: Strong Drive Initial drive state: High (1)

Figure 5 shows the pin assignment for the project done through the **Pins** tab in the **Design Wide Resources** window. These assignments are compatible with CY8CKIT-062-BLE.

Figure 5. Pin Assignments



Reusing This Example

This example is designed for the CY8CKIT-062-BLE pioneer kit. To port the design to a different PSoC 6 MCU device, kit or both, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed. For single-core PSoC 6 MCU devices, port the code from *main_cm4.c* to *main.c*.

Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with BLE Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project
AN215656 – PSoC 6 MCU Dual-Core CPU system Design	Describes the dual-core CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-core design
AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project	Describes how to import the code generated by PSoC Creator into your preferred IDE
Code Examples	
CE216825 PSoC 6 MCU Real-Time Clock Basics	
CE218964 PSoC 6 MCU RTC Daily Alarm	
CE219339 PSoC 6 MCU - MCWDT and RTC Interrupts (Dual Core)	
PSoC Creator Component Datasheets	
Pins	Supports connection of hardware resources to physical pins
RTC	Component provides an application interface for keeping track of time and date
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual PSoC 6 MCU: PSoC 63 with BLE Registers Technical Reference Manual
Development Kit (DVK) Documentation	
CY8CKIT-062-BLE Pioneer Kit	

Document History

Document Title: CE218542 - PSoC 6 MCU Custom Tick Timer Using RTC Alarm

Document Number: 002-18542

Revision	ECN	Orig. of Change	Submission Date	Description of Change
*A	5993957	AJYA	12/14/2017	Initial Public Release
*B	6079240	AJYA	04/02/2018	Updated to PSoC Creator 4.2

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2017-2018. 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. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. 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.