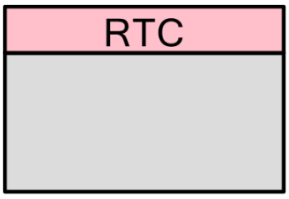
Real-Time Clock (RTC_PDL)
2.0

Features



- Different hour format support
- Multiple alarm function (two-alarms)
- Daylight Savings Time option support
- Automatic leap year compensation
- Option to drive the RTC by an external 50-Hz or 60-Hz clock source

General Description

The Real-Time Clock (RTC_PDL) Component provides an application interface for keeping track of time and date. It provides access to the HW real-time clock. It is driven by the clock that drives the Clk_Bak. You need to choose the source for Clk_Bak as the input clock source in the Design-Wide Resources Clock Editor (Configure System Clocks).

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

The time can be represented in either 12-hour format or 24-hour format. The date representation can be in "MM/DD/YYYY", "DD/MM/YYYY" or "YYYY/MM/DD" format. The RTC_PDL Component keeps track of second, minute, hour, day of the week, day of the month, month, and year. The day of the week is automatically calculated from the day, month, and year. It automatically accounts for leap year changes. Leap year is identified as the year, which is a multiple of 4 or 400 but not 100.

Daylight savings time (DST) may optionally be enabled and supports any start and end date. The start and end dates can be fixed date like 24 March or relative like the second or last Sunday in March.

This Component also has an optional alarm feature, which provides match detection for second, minute, hour, day of week, day of month, and month values. The RTC_PDL Component provides two alarm functions physically, which can trigger interrupts to ask an appropriate action on time. The alarm flexibility supports periodic alarms (such as every minute) or a single alarm (such as 10:45 on 28 September, 2043).

When the external 32.768-kHz Watch-crystal oscillator (WCO) is absent on the board, the RTC_PDL Component can be driven by an external 32.768-kHz square clock source, or by an external 50-Hz or 60-Hz sine-wave clock source; for example, the wall AC frequency.

When to Use the RTC_PDL Component

The RTC_PDL Component can provide an application interface for keeping track of time and date. Optional alarms can be supported as well. The time information and alarm features can be used in a system that requires real-time and alarm functions. It can be used in two main cases:

Display Real-time

Many user applications require displaying a real time with DST option. For example, a smart watch and an activity tracker are required in real time to log human activities. Also, a microwave oven and cook-top stoves show the real time.

Alarm applications

The alarm can be used not only for a real alarm function but also for a reference time to finish an ordered/programmed work.

Quick Start

1. Open the System Clock Editor and make sure Clk_Bak is sourced properly. For more information, refer to the Clock Selection section in this datasheet.
2. Drag a RTC_PDL Component from the Component Catalog System folder onto your schematic (the placed instance takes the name RTC_1).
3. Double-click to open the Configure dialog.
4. Select **Enable Interrupts** check box if you want to work with Alarm 1, Alarm 2, and Century interrupts.
5. In *main.c*, write the Cy_RTC_Alarm1Interrupt(), Cy_RTC_Alarm2Interrupt(), and Cy_RTC_CenturyInterrupt() functions.
6. 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 RTC_1 instance.
7. In *main.c*, initialize the peripheral and start the application

```
/* Set new time and time format in RTC */
Cy_RTC_Init(&RTC_1_config);
Cy_RTC_SetHoursFormat(CY_RTC_12_HOURS);

/* Call these two lines if you enabled "Enable Interrupts" check box */
Cy_SysInt_Init(&RTC_1_RTC_IRQ_cfg, &RTC_1_Interrupt);
NVIC_EnableIRQ(RTC_1_RTC_IRQ_cfg.intrSrc);
```

```
/* Setup the alarm - 12:00:00 20/4 */
Cy_RTC_SetAlarmDateAndTimeDirect(0u, 0u, 12u, 20u, 4u, CY_RTC_ALARM_1);
```

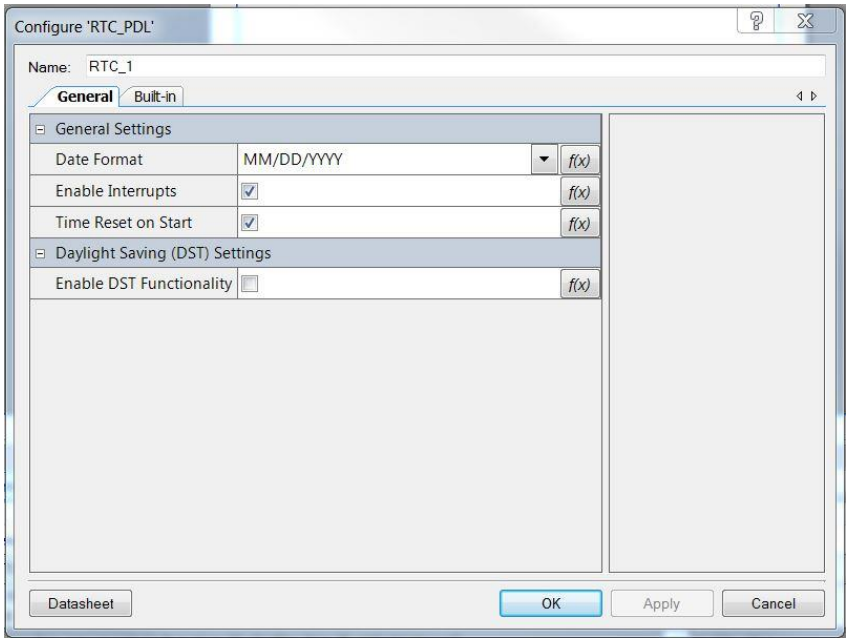
8. Build and program the device.

Component Parameters

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

General Tab

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



Parameter Name	Description
Date Format	This parameter selects how the date can be represented/stored in the single date variable. You can select from one of three standard formats: • "MM/DD/YYYY" (Default) • "DD/MM/YYYY" • "YYYY/MM/DD" Based on this parameter, the next date shifts are generated: day, month, and year. Use date shifts to create a single date value with appropriate date format.

Parameter Name	Description
Enable Interrupts	This parameter allows you to configure the RTC interrupts. It will set the appropriate interrupt vector to the WakeUp interrupt controller (WIC) and the interrupt priority. The interrupt priority can be changed in the “Interrupts” tab of CYDWR page. Disabling the feature will remove the code related to interrupts configuration in the Component the RTC_Start() function. However, this parameter does not enable the RTC interrupts requests to WIC. When needed, the RTC interrupt (Alarm1, Alarm2 or Century Interrupt) should be unmasked using Cy_RTC_1_SetInterruptMask() function. For more details, refer to the function description.
Time Reset on Start	This parameter allows you to skip the time and date setting. If the check box is selected, the RTC_Start() function will set the time and date for the Component when the system starts up due to one of POR, XRES, or BOD. However, the time and date configuration will be skipped after wakeup from Hibernate power mode to avoid unintended time reset. If the check box is not selected, the time and date code will be removed from RTC_Start() function.
Enable DST Functionality	This parameter allows you to choose whether the daylight savings time functionality is enabled in the RTC_PDL Component. Note that DST feature is based on interrupts, as it uses RTC Alarm 2 interrupt. The interrupts should be enabled (“Enable Interrupts” is checked) for DST correct functionality. The DST settings are visible, if “Enable DST Functionality” is checked out These settings are enabled only if the check box is selected. These settings provide two sets of parameters depending on the type of DST format. If the DST format is a ‘Relative date’, then you can set day of week, week of month, and month. If the DST date is a ‘Fixed date’, then you can set day of month and month. The Start/Stop hours setting is available in both the date modes. • DST settings with ‘Relative date’ option This parameter selects "Relative date" format for storing the DST start/stop dates. A relative date can be "Last Sunday of March". • DST settings with ‘Fixed date’ option This parameter selects "Fixed date" format for storing the DST start/stop dates. A fixed date can be "22 March".

Application Programming Interface

The Application Programming Interface (API) is provided by the cy_rtc driver module from the PDL. The driver is copied into the “pdl\drivers\peripheral\rtc\” 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.

The Component generates the configuration structures and base address described in the Global Variables section. Pass the generated data structure and the base address to the associated cy_rtc 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 RTC_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.

Global Variables

The RTC_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. RTC_1.c). Each variable is also prefixed with the instance name of the Component.

bool RTC_1_rtcDstStatus

The RTC_1_rtcDstStatus variable, which is for the DST feature, is called in the Cy_RTC_Interrupt() PDL driver function. This variable is defined as true if DST is enabled in the RTC customizer and as false if DST is disabled

cy_stc_rtc_dst_t const RTC_1_dstConfig

The instance-specific daylight saving time structure. This should be used in the associated DTS functions.

cy_stc_rtc_config_t const RTC_1_config

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

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 Service Routine

The hardware SRSS Backup IP block provides single interrupt line to the WIC. Currently only the RTC interrupts are present in the SRSS Backup IP block. Three RTC interrupts can be asserted by:

- Alarm 1
- Alarm 2
- The Century interrupt

It is required to check which exact interrupt (out of three) was generated by checking the SRSS Backup Interrupt register.

In Active mode, Low-power Active mode, an interrupt request from RTC is sent to the CPU via IRQ 21. In Sleep, Low-power Sleep or Deep-Sleep power mode the CPU is powered-down, so the interrupt request from the RTC is directly sent to the WIC, which will then wake up the CPU. Then, the CPU acknowledges the interrupt request and executes the Interrupt Service Routine.

In the RTC_PDL Component header file are declared the cy_rtc driver functions, which are called in general RTC interrupt function RTC_1_Interrupt(). If needed to have such event, the user should define function implementation in the source code.

The RTC interrupts handling

The RTC_PDL Component provides interrupt handler functions for every RTC interrupt event: Cy_RTC_Alarm1Interrupt(), Cy_RTC_Alarm2Interrupt() and the Cy_RTC_CenturyInterrupt().

All three functions are blank functions with WEAK attribute. The appropriate function(s) should be redefined in user source code in condition that such event(s) is required.

Cy_RTC_Alarm1Interrupt(), Cy_RTC_Alarm2Interrupt() or Cy_RTC_DstInterrupt() and the Cy_RTC_CenturyInterrupt() functions are called in the RTC_Interrupt() function. For more details refer to RTC_1_Interrupt() function description.

Note that DST feature is based on RTC Alarm2 interrupt. It is not allowed to have at the same time the DST feature and Alarm2 interrupt.

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.

Application Notes

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

API Memory Usage

The Component memory usage varies significantly depending on the compiler, device, number of APIs used and Component configuration. The following table provides the memory usage for all APIs available in the given Component configuration.

The measurements have been done with an associated compiler configured in Release mode with optimization set for Size. For a specific design, the map file generated by the compiler can be analyzed to determine the memory usage.

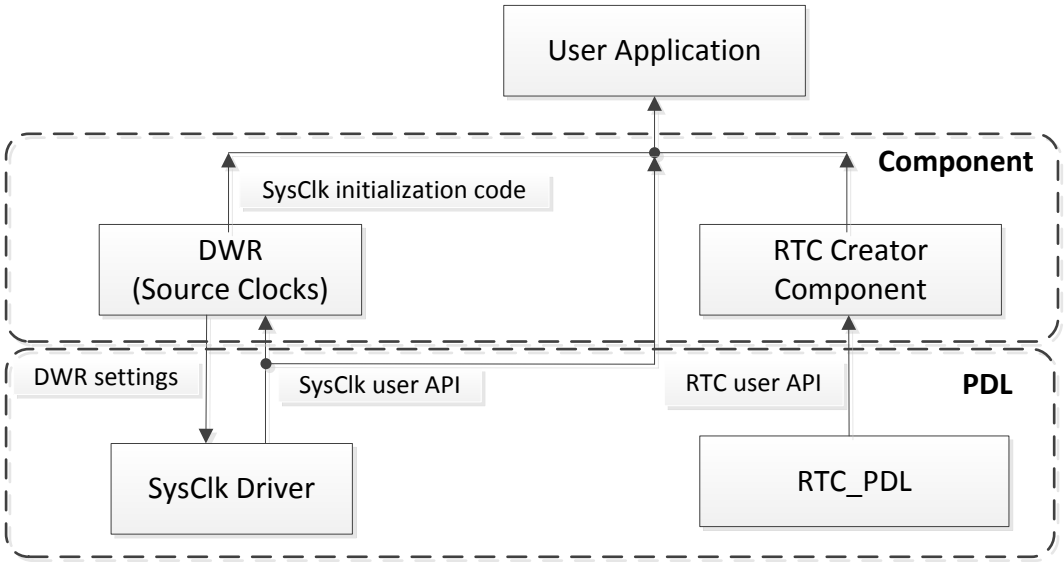
PSoC 6 (GCC)

Configuration	Flash Bytes	SRAM Bytes
Default	1960	2
RTC with interrupts	2646	2
RTC with DST and interrupts	2548	50

Functional Description

Block Diagram and Configuration

The following is a simplified diagram of the RTC hardware:



The RTC solution consists of two major parts: the cy_rtc PDL driver and the PSoC Creator Component:

- The cy_rtc driver (PDL)

The RTC driver wraps the API the Backup System HW domain, providing access to its registers. Most, if not all the RTC core functionality is enclosed in the Backup System HW domain.

- The RTC_PDL Component (PSoC Creator Component)

  The RTC_PDL Component exposes the cy_rtc PDL driver functionality in the PSoC Creator environment, providing the configuration interface (Customizer) and the PSoC Creator-like API. The RTC_PDL Component provides:

  - Initial configuration time-and-date structure
  - Initial configuration DST structure
  - The RTC interrupt handler function for handling DST interrupts.

PSoC Creator (which can be considered as a third part of the solution) provides access to the System Clock (SysClk) driver configuration via the Design-Wide Resources Clock Editor. The Clk_Bak that drives the RTC is configured by the SysClk driver.

Clock Selection

The RTC is driven by Clk_Bak clock source. The RTC_PDL Component does not have an enable/disable configuration. After the RTC is clocked, it automatically starts to tick. The Clk_Bak can be driven by the following:

- Watch-crystal oscillator (WCO)

  The WCO is a high-accuracy clock that is suitable for RTC applications and requires external 32.768-kHz watch crystal oscillator on board. The WCO can be supplied by internal power supply (Vback) and therefore can run without Vddd/Vccd present. This can be used to wake the chip from Hibernate mode.

- Routed from Clk_LF (alternate backup domain clock source) the Internal Low-speed Oscillator (ILO).

  This is a low-accuracy RC-oscillator that does not require any external Components. Its poor accuracy (+/- 30%) means it is not useful as a RTC, but it can be also supplied by internal power supply (Vback) and therefore can run without Vddd/Vccd present. This also can be used to wake the chip from Hibernate mode.

- Routed from Clk_LF (alternate backup domain clock source) the Precision Internal Low-speed Oscillator (PILO).

  This is a RC-oscillator (ILO) that can achieve accuracy (+/- 2%) with periodic calibration. It is not expected to be accurate enough for good RTC capability. The PILO requires Vddd/Vccd present, so it can be used in modes down to Deep Sleep and ceases to function in Hibernate mode.

- External 50-Hz or 60-Hz sine-wave clock source or 32.768-kHz square clock source.

  In addition, the RTC can be sourced by external 50-Hz or 60-Hz sine-wave clock source, for example, the wall AC frequency. In addition, the RTC can be driven by external 32.768-kHz square clock source. Such clock source can be used in condition that the external 32.768-kHz WCO is absent on the board. For more details refer to the Cy_RTC_SelectFrequencyPrescaler() function description.

It is recommended to drive the Clk_Bak by the WCO, if it present in design, instead of the ILO. For setting the Clk_Bak clock source, go to the Configure System Clocks dialog under the **Miscellaneous Clocks** tab.

RTC inaccuracy

The RTC inaccuracy is ±1minutes/year (room temperature) if RTC clock source is ±2ppm calibrated.

Low-Power Modes

Deep Sleep and Hibernate modes are possible for the IP block instances that support Low Power. Refer to the device *Technical Reference Manual (TRM)* for details.

The RTC_PDL Component requires specific processing to support Deep Sleep and Hibernate modes. The cy_rtc PDL module provides callback functions to handle power mode transition. These functions must be registered using the cy_syspm driver before entering Deep Sleep or Hibernate mode appropriately. Refer to the Low Power section of the cy_rtc driver, or the System Power Management section of the PDL Documentation for more details about callback registration.

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

Refer to **PSoC Creator Help > Building a PSoC Creator Project > Generated Files (PSoC 6)** for information on MISRA compliance and deviations of the files generated by PSoC Creator.

This Component has the following embedded Components: SysInt. Refer to the corresponding Component datasheets for information on their MISRA compliance and specific deviations.

The RTC_PDL Component does not have any specific deviations.

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

Registers

Hardware registers

See the Backup System Registers section in the chip *Technical Reference Manual (TRM)* for more information about the RTC hardware registers.

Conditional Compilation Information

The RTC API requires one conditional compile definition to handle initial time, interrupt-based functionality and daylight savings time functionality. The DST related functions are conditionally compiled only if this option is enabled in the Configure dialog. The interrupt-based functionality, such as alarm handler functions are compiled only if this option is enabled in the Configure dialog. The software should never use this parameter directly. Instead, use the symbolic name defined.

- **RTC _INITIAL_DST_STATUS** – The daylight savings time functionality enables define is assigned to be equal to the "Enable DST Functionality" value (from the Configure dialog) at build time. It is used throughout the API to compile data saving time functions.
- **RTC _INITIAL_IRQ_STATUS** – The interrupt-based functionality enables define is assigned to be equal to the "Enable Interrupts" value (from the Configure dialog) at build time. It is used throughout the API to compile data saving time functions.

Date register

This register contains the time value in the format that is selected in customizer with “Date Format” select box. These defines are for creating a date value in a single word with the required date elements sequence. The defines that contains offset to each time value are as follows:

- **RTC_MONTH_OFFSET –** Offset to the field that contains the month value.
- **RTC_DAY_OFFSET –** Offset to the field that contains the day value.
- **RTC_YEAR_OFFSET –** Offset to the field that contains the year value.

Resources

The RTC_PDL Component uses the Backup System peripheral block.

DC and AC Electrical Characteristics

Note Final characterization data for PSoC 6 devices is not available at this time. Once the data is available, the Component datasheet will be updated on the Cypress web site.

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
2.0.d	Minor datasheet edits.	
2.0.c	Update the datasheet.	Added the Low-Power Modes section.
2.0.b	Minor datasheet edits.	
2.0.a	Updated datasheet.	Changed reference for the information on MISRA compliance and deviations of files generated by PSoC Creator.
2.0	Updated the underlying version of PDL driver.	
1.0 b	Updated datasheet.	Minor text update in the General Description. Added information about “Reset on Start” check box in the General tab.
1.0.a	Updated datasheet.	Updated General Description section. Updated MISRA section. Updated API section.
1.0	Initial Version	

© 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. 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.