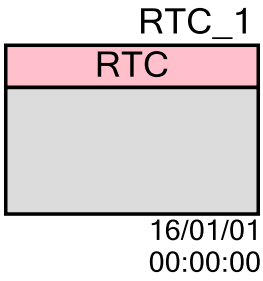
Real-Time Clock (PDL_RTC)
1.0



Features

- Date and time settings
- Alarm settings with a specific date and time
- Leap year compliant

General Description

The Peripheral Driver Library (PDL) Real-Time Clock (PDL_RTC) is composed of the RTC clock control block and the RTC count block. The RTC count block counts years, months, dates, hours, minutes, seconds, and days of the week from 00 to 99 years. Alarm and timer settings are possible as well. An alarm can be set to a specific year, month, date, hour, and minute. It can also be set to a specific year, month, date, hours, or minutes independently. A timer can be set to a period up to one day. It can be set to a desired period (with the hours, minutes, and seconds specified) or in desired intervals (with hours, minutes, and seconds specified).

- It is possible to rewrite the time by resetting the watch count of the RTC count block.
- The following interrupts can be output: alarm (with an interrupt generated at a set date and time), every hour, every minute, every second, every 0.5 seconds, timer, time rewrite error, timer counter read completion, pulse output in 0.5-second intervals.

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

When to Use a PDL_RTC Component

Use the PDL_RTC component when the system requires the current time or date. You can also use the PDL_RTC when you need accurate timing of events with half-second resolution.

Quick Start

1. Drag a PDL_RTC component from the Component Catalog FMx/System/Real-Time Clock folder onto your schematic. The placed instance takes the name RTC_1.
2. Double-click to open the component's Configure dialog.
3. On the **Basic** tab, set the following parameters:
    - select the clock source

- specify clock prescaler
- enable RTCCO, SUBOUT output pins if needed

4. On the **Time** tab, set the following parameters:
   - set the time (hour, minute, seconds)
   - set the date (year, month, day, day of week)
5. To use alarm function, on the **Alarm** tab, set the following parameters:
   - alarm time (hour, minute)
   - alarm date (year, month, day)
6. On the **Interrupts** tab, initialize needed interrupts and their callback functions.
7. To use the timer functions, on the **Timer** tab set the following parameters:
   - timer mode
   - timer cycle
8. To use the frequency correction, on the **Correction** tab set the following parameters:
   - enable frequency correction
   - correction value
   - correction cycle
9. 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 RTC_1 instance.
10. In the *main.c* file, initialize the peripheral and start the application:

```
(void)Rtc_SetDayOfWeek(&RTC_1_TimeDate);
(void)Rtc_Init(&RTC_1_HW, &RTC_1_Config);/* Initialize the RTC */
Rtc_EnableFunc(&RTC_1_HW, RtcCount);/* Start RTC counting */
```

11. Build and program the device.

Component Parameters

The PDL_RTC 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
bEnSuboutDivider	Enable the divider on the SUBOUT clock
enClkSel	Clock source for RTC
enDividerRatio	SUBOUT clock divider value
enRtccoSel	RTCCO output select
u32ClkPrescaler	RTC source clock divider
bEnableRtcco	Generate macro to enable the RTCCO signal to a pin. Note: this parameter controls the presence of the RTCCO pin in the DWR
bEnableSubout	Generate macro to enable the SUBOUT signal to a pin. Note: this parameter controls the presence of the SUBOUT pin in the DWR
bRunNotInit	Disable RTC initialization if it is already running

Alarm Tab

This tab contains the Alarm configuration settings.

Parameter Name	Description
u16AlarmYear	Alarm year (1..99) after year 2000
u8AlarmDay	Alarm day (1…31)
u8AlarmHour	Alarm hour (0…23)
u8AlarmMinute	Alarm minute (1…59)
u8AlarmMonth	Alarm month (1…12)

Correction Tab

This tab contains the Correction configuration settings.

Parameter Name	Description
bFrequencyCorrection	Enable frequency correction. Note: This parameter controls whether the RTC_1_FreqCorrConfig struct exists or is null

Parameter Name	Description
u16FreqCorrValue	Frequency correction value.

Interrupts Tab

This tab contains the Interrupts configuration settings.

Parameter Name	Description
bAlarmIrq	Enable alarm interrupt
pfnAlarmIrqCb	Callback function for alarm interrupts. Note: this generates a declaration only - USER must implement the function
bTouchNvic	Install interrupts in NVIC
bTimeRewriteErrorIrq	Enable rewrite error interrupt
pfnTimeWrtErrIrqCb	Callback function for timer rewrite error interrupts. Note: this generates a declaration only - USER must implement the function
bHalfSecondIrq	Enable half second interrupt
bOneHourIrq	Enable one hour interrupt
bOneMinuteIrq	Enable one minute interrupt
bOneSecondIrq	Enable one second interrupt
pfnHalfSecondIrqCb	Callback function for half second interrupts. Note: this generates a declaration only - USER must implement the function
pfnOneHourIrqCb	Callback function for one hour interrupts. Note: this generates a declaration only - USER must implement the function
pfnOneMinuteIrqCb	Callback function for one minute interrupts. Note: this generates a declaration only - USER must implement the function
pfnOneSecondIrqCb	Callback function for one second interrupts. Note: this generates a declaration only - USER must implement the function
bTimerIrq	Enable timer interrupt
pfnTimerIrqCb	Callback function for timer interrupts. Note: this generates a declaration only - USER must implement the function

Time Tab

This tab contains the Time configuration settings.

Parameter Name	Description
u16Year	Year (1...99) + 2000
u8Day	Day (1...31)

Parameter Name	Description
u8DayOfWeek	Day of the week (0...6)
u8Hour	Hour (0...23)
u8Minute	Minutes (0...59)
u8Month	Month (1...12)
u8Second	Second (0...59)

Timer Tab

This tab contains the Timer configuration settings.

Parameter Name	Description
enMode	RTC timer mode
u32TimerCycle	Count of elapsed seconds before timer interrupt

Component Usage

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

Generated Data

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

Data Structure Type	Name	Description
stc_rtc_irq_en_t	RTC_1_IrqEn	Interrupt enable structure. This is automatically referenced in the RTC_1_Config.
stc_rtc_irq_cb_t	RTC_1_IrqCb	Interrupt callback functions structure. This is automatically referenced in the RTC_1_Config.
stc_rtc_freq_corr_t	RTC_1_FreqCorrConfig	Frequency correction structure
stc_rtc_time_t	RTC_1_TimeDate	Date and time structure
stc_rtc_alarm_t	RTC_1_Alarm	Alarm structure
stc_rtc_timer_t	RTC_1_Timer	Timer configuration structure
stc_rtc_config_t	RTC_1_Config	Configuration structure

Once the component is initialized, the application code should use the peripheral functions provided in the referenced PDL files. Refer to the PDL 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…**" in the drop-down menu.

Preprocessor Macros

The RTC component generates the following preprocessor macro(s). Note that each macro is prefixed with the instance name of the component (e.g. "RTC_1").

Macro	Description
RTC_1_SetPinFunc_RTCCO	Macro to configure RTCCO output. Active when RTCCO output enabled in the configuration window.
RTC_1_SetPinFunc_SUBOUT	Macro to configure SUBOUT output. Active when SUBOUT output enabled in the configuration window.
RTC_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.

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_RTC component is specified to trigger interrupts, it will generate the callback function declaration that will be called from the RTC 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
RTC_1_HalfSecondIrqCb	0.5-second interrupt callback function. Note: this generates a declaration only - USER must implement the function.
RTC_1_OneHourIrqCb	1-hour interrupt callback function. Note: this generates a declaration only - USER must implement the function.
RTC_1_OneMinuteIrqCb	1-minute interrupt callback function. Note: this generates a declaration only - USER must implement the function.

Function Callback	Description
RTC_1_OneSecondIrqCb	1-second interrupt callback function. Note: this generates a declaration only - USER must implement the function.
RTC_1_TimerIrqCb	Timer interrupt callback function. Note: this generates a declaration only - USER must implement the function.
RTC_1_TimeWrtErrIrqCb	Time rewrite error interrupt callback function. Note: this generates a declaration only - USER must implement the function.

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_RTC component uses the Real-Time Clock (RTC) 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 Errata

This section lists known problems with the component.

Cypress ID	Component Version	Problem	Workaround
253036	1.0	RTC is not supported on S6E1C devices. There is a defect in the RTC driver that prevents its use on S6E1C devices. The defect does not impact S6E1A or S6E1B devices.	None. Contact Cypress technical support (http://www.cypress.com/mycases) for possible firmware updates and help with using RTC on S6E1C devices.

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.