Renesas Synergy™ Platform

RTC HAL Module Guide

Introduction

This module guide will enable you to effectively use a module in your own design. Upon completion of this guide, you will be able to add this module to your own design, configure it correctly for the target application and write code, using the included application project code as a reference and efficient starting point. References to more detailed API descriptions and suggestions of other application projects that illustrate more advanced uses of the module are available in the Renesas Synergy Knowledge Base (as described in the References section at the end of this document), and should be valuable resources for creating more complex designs.

The Real-Time Clock (RTC) HAL module is a high-level API for RTC applications and is implemented on `r_rtc`. The RTC HAL module configures the RTC module and controls clock, calendar, and alarm functions. The RTC uses the real-time clock module on the Synergy MCU. A user-defined callback can be created to respond to any of the three supported interrupt types: alarm, periodic, or carry.

Contents

1. RTC HAL Module Features

The RTC HAL module supports the following functions of the real-time clock:

- RTC peripheral configuration
- RTC time and date get and set
- RTC time and date alarm get and set
- RTC time counter start and stop
- RTC alarm, periodic and carry event notification
- RTC event type enable and disable
- RTC event rate configuration
- RTC clock source get and set

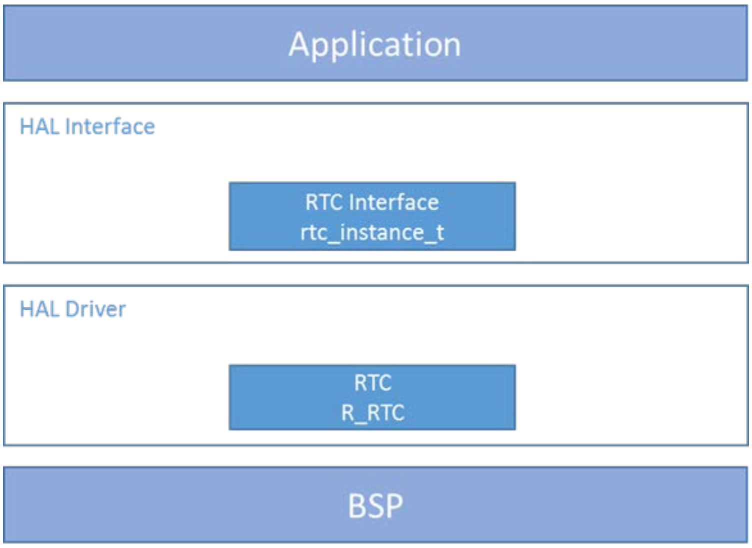

Figure 1. RTC HAL Module Block Diagram

2. RTC APIs Overview

The RTC HAL module defines APIs for opening, closing, setting alarms, starting, and stopping RTC operations. A complete list of the available APIs, an example API call, and a short description of each can be found in the following table. A table of all applicable status return values follows the API summary table.

Table 1. RTC HAL Module API Summary

Function Name	Example API Call and Description
.open	`g_rtc0.p_api->open(g_rtc0.p_ctrl, g_rtc0.p_cfg);` Open the RTC HAL.
.close	`g_rtc0.p_api->close(g_rtc0.p_ctrl);` Close the RTC HAL.
.calendarTimeSet	`g_rtc0.p_api->calendarTimeSet(g_rtc0.p_ctrl, &start_time_struct_in, true);` Set the calendar time.
.calendarTimeGet	`g_rtc0.p_api->calendarTimeGet(g_rtc0.p_ctrl, &current_time_struct_out);` Get the calendar time.
.calendarAlarmSet	`g_rtc0.p_api->calendarAlarmSet(g_rtc0.p_ctrl, &in_alarm_time_struct_in, true);` Set the calendar alarm time.
.calendarAlarmGet	`g_rtc0.p_api->calendarAlarmGet(g_rtc0.p_ctrl, &get_alarm_time_struct_out);` Get the calendar alarm time.

.calendarCounterStart	`g_rtc0.p_api->calendarCounterStart(g_rtc0.p_ctrl);` Start the calendar counter.
.calendarCounterStop	`g_rtc0.p_api->calendarCounterStop(g_rtc0.p_ctrl);` Stop the calendar counter.
.irqEnable	`g_rtc0.p_api->irqEnable(g_rtc0.p_ctrl, CALLBACK);` Enable the alarm irq.
.irqDisable	`g_rtc0.p_api->irqDisable(g_rtc0.p_ctrl, CALLBACK);` Disable the alarm irq.
.periodicIrqRateSet	`g_rtc0.p_api->periodicIrqRateSet(g_rtc0.p_ctrl, Rate);` Set the periodic irq rate.
.infoGet	`g_rtc0.p_api->infoGet(g_rtc0.p_ctrl, clk_src);` Return the currently configure clock source for the RTC.
.versionGet	`g_rtc0.p_api->versionGet(&version);` Retrieve the API version with the version pointer.

Note: For more complete descriptions of operation and definitions for the function data structures, typedefs, defines, API data, API structures, and function variables, review the SSP User's Manual API References for the associated module.

Table 2. Status Return Values

Name	Description
SSP_SUCCESS	Function executed successfully
SSP_ERR_ASSERTION	API dependent error
SSP_ERR_INVALID MODE	Invalid mode
SSP_ERR_INVALID_PTR	Invalid parameter

Note: Lower-level drivers may return common error codes. Refer to the SSP User's Manual API References for the associated module for a definition of all relevant status return values.

3. RTC HAL Module Operational Overview

The RTC HAL module controls the operation of the real-time clock module on a Synergy MCU. The typical RTC application configures the real-time clock controller periodically based on a system configuration driven by the user. Common operations include setting the time, setting an alarm, configuring a periodic interrupt and starting or stopping operation. An RTC application usually consists of calls to the RTC HAL module and an optional callback from the ISR handler.

- The RTC HAL module can use two main clock sources:
  - A Low Speed On-Chip Oscillator (LOCO) with lower power but less accuracy
  - A sub-clock oscillator with higher power, increased accuracy and more cost (external crystal required)
- The RTC HAL module supports three different interrupt types:
  - An alarm interrupt generated on a match of any combination of year, month, day, day of the week, hour, minute and second
  - A periodic interrupt generated every 2, 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, or 1/256 second(s)
  - A carry interrupt when either a carry to the second counter occurs or when a carry to the R64CNT counter occurs during a read access to the 64-Hz counter
- A user-defined callback function can be registered (in the `open` API call) and will be called from the interrupt service routine (ISR) for any supported interrupt type. When called, it is passed a pointer to a structure (`rtc_callback_args_t`) that holds a user-defined context pointer and an indication of which type of interrupt was fired.

3.1 RTC HAL Module Important Operational Notes and Limitations

3.1.1 RTC HAL Module Operational Notes

The RTC HAL module must be opened before any of the other RTC module APIs can be called. A configuration structure is passed to the `open` call which specifies the clock source, the name of the user callback from the ISR handler, and a user-specified context for the callback. Configuration structures can be either manually defined or generated by the ISDE based on user input during the configuration process.

Functions in the driver can be accessed by either making direct calls to the HAL layer or by using the RTC interface structure. The name of this interface structure is based on the name setting entered in the module configuration. For example, if the name is `g_rtc`, then the interface structure is called `g_rtc_api`.

3.1.2 RTC HAL Module Limitations

This module has no support for the following functions:

- Binary-count mode
- Binary alarm get and set
- Binary time get and set
- Clock-error correction
- 1-Hz/64-Hz Clock output

Refer to the most recent SSP Release Notes for any additional operational limitations for this module.

4. Including the RTC HAL Module in an Application

This section describes how to include the RTC HAL module in an application using the SSP configurator.

Note: It is assumed that you are familiar with creating a project, adding threads, adding a stack to a thread and configuring a block within the stack. If you are unfamiliar with any of these items, refer to the first few chapters of the *SSP User's Manual* to learn how to manage each of these important steps in creating SSP-based applications.

To add the RTC Driver to an application, simply add it to a thread using the stacks selection sequence given in the following table. (The default name for the RTC is r_rtc0. This name can be changed in the associated **Properties** window.)

Table 3. RTC Driver Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
r_rtc0 RTC HAL on r_rtc	Threads	New Stack> Driver> Timers> RTC HAL on r_rtc

When the RTC HAL module on r_rtc is added to the thread stack as shown in the following, the configurator automatically adds any needed lower-level modules. Modules with a Gray band are individual modules that stand alone.

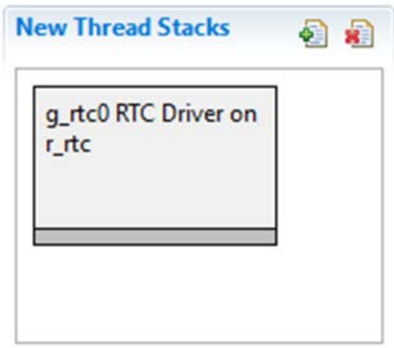

Figure 2. RTC HAL Module Stack

5. Configuring the RTC HAL Module

The RTC HAL module must be configured for the desired operation. The SSP configuration window automatically identifies (by highlighting the block in red) any required configuration selections, such as interrupts or operating modes, which must be configured for lower-level modules for successful operation. Also, only those properties that can be changed without causing conflicts are available for modification. Other properties are 'locked' and are not available for changes, and are identified with a lock icon for the 'locked' property in the Properties window in the ISDE. This approach simplifies the configuration process and makes it much less error-prone than previous 'manual' approaches to configuration. The available configuration settings and defaults for all the user-accessible properties are given in the Properties tab within the SSP configurator and are shown in the following tables for easy reference.

One of the properties most often identified as requiring a change is the interrupt priority. This configuration setting is available within the **Properties** window of the associated module. Simply select the indicated module and then view the **Properties** window. The interrupt settings are often toward the bottom of the properties list, so scroll down until they become available. Also, note that the interrupt priorities listed in the **Properties** window in the ISDE include an indication of the validity of the setting based on the targeted MCU (CM4 or CM0+). This level of detail is not included in the following configuration properties tables, but is easily visible with the ISDE when configuring interrupt-priority levels.

Note: You may want to open your ISDE, create the module, and explore the property settings in parallel while looking over the following configuration table settings. This will help orient you and can be a useful hands-on approach to learning the ins and outs of developing with SSP.

Table 4. Configuration Settings for the RTC HAL Module on r_rtc

Parameter	Value	Description
Parameter Checking Enable	BSP, Enabled, Disabled (Default: BSP)	Enable or disable parameter error checking
Name	g_rtc0	The name to be used for the RTC module control block instance. This name is also used as the prefix of the other variable instances. See the example code below.
Clock Source	LOCO, Sub-clock (Default: LOCO)	Clock source for the RTC block
Error Adjustment Value	0	**Important:** Deprecated configuration field. Must be 0.
Error Adjustment Type	None, Add prescaler, Subtract prescaler (Default: None)	**Important:** Deprecated configuration field. Must be None.
Callback	NULL	The name of the ISR that is called when one of the three interrupts fire. The argument passed into this ISR has an indication of which interrupt caused it to be called.
RTC ALARM	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Priority 5)	Priority level for RTC alarm interrupt
RTC PERIOD	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Priority 6)	Priority level for RTC period interrupt
RTC CARRY	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Priority 7)	Priority level for RTC carry interrupt

Note: The example values and defaults are for a project using the Synergy S7G2 MCU. Other MCUs may have different default values and available configuration settings.

In some cases, settings other than the defaults for a module can be desirable. For example, it might be useful to select a different clock source than the default. The configurable properties for the lower-level stack modules are given in the following sections for completeness and as a reference.

5.1 RTC HAL Module Clock Configuration

The RTC HAL module can use the following clock sources:

- LOCO (Low Speed On-Chip Oscillator)
  - Lower power consumption
  - Less accurate
- Sub-clock oscillator
  - Higher power consumption
  - More accurate
  - More cost (requires a crystal)

The LOCO is the default selection during configuration.

5.2 RTC HAL Module Pin Configuration

The RTC does not currently support outputs, so no output pin selections are available.

6. Using the RTC HAL Module in an Application

The typical RTC application configures the real-time clock controller periodically based on a system configuration driven by the user. Examples include setting the time, setting an alarm, configuring a periodic interrupt, and so forth. An RTC application consists of calls to the RTC module and optional ISR callbacks.

The RTC module must be opened before any of the other APIs can be called. A configuration structure is passed to the `open` call which specifies the clock source, the names of the ISR callbacks, and the user-specified context for the handler. Configuration structures can be either manually defined or generated by the ISDE based on user input during the configuration process. Functions in the module can be accessed by using the RTC interface structure. The name of this interface structure is based on the name setting entered in the module configuration.

The typical steps in using the RTC periodic IRQ in an application are:

1. Initialize the RTC using the `open` API.
2. Set periodic IRQ rate using the `periodicIrqRateSet` API.
3. Start calendar counter using the `calendarCounterStart` API.
4. Enable interrupt using the `irqEnable` API.

These common steps are illustrated in a typical operational flow diagram in the figure below:

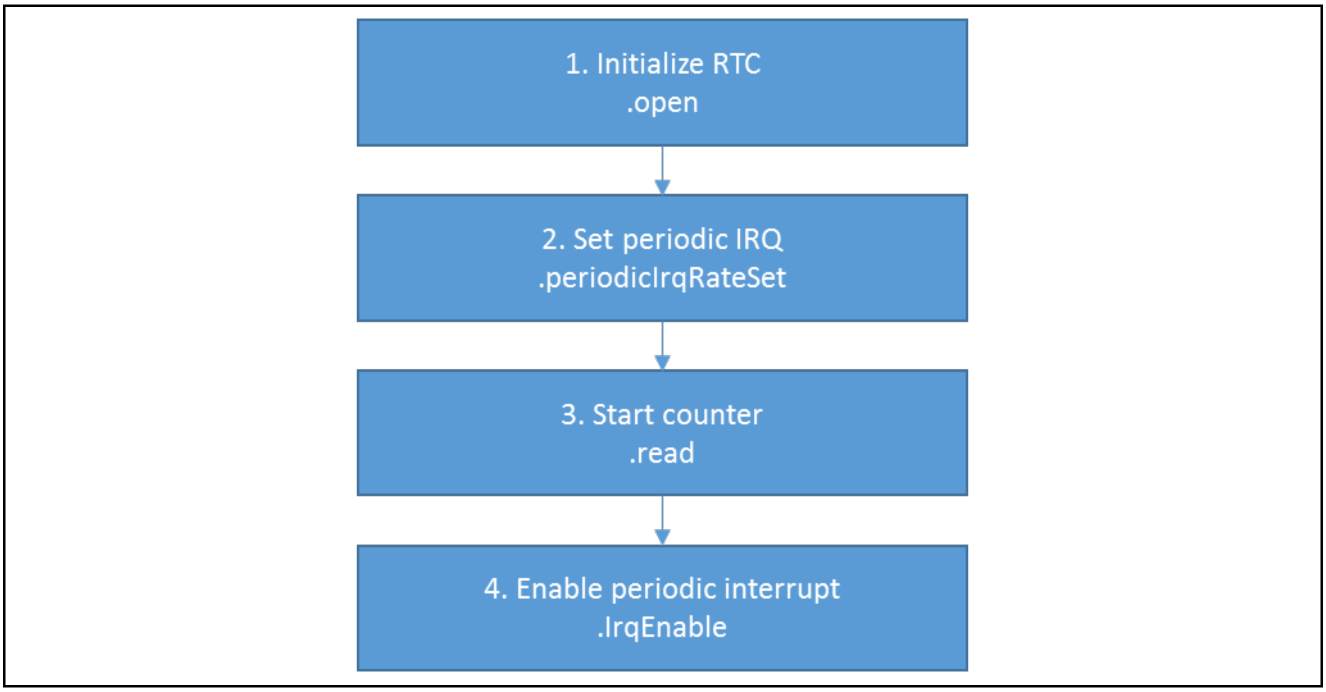

Figure 3. Flow Diagram of a Typical RTC Periodic Interrupt Application

The typical steps in using the RTC Alarm IRQ in an application are:

1. Initialize the RTC using the `open` API.
2. Set calendar time using the `calendarTimeSet` API.
3. Set alarm time using the `calandarAlarmSet` API.
4. Start calendar counter using the `calendarCounterStart` API.

These common steps are illustrated in a typical operational flow diagram in the following figure:

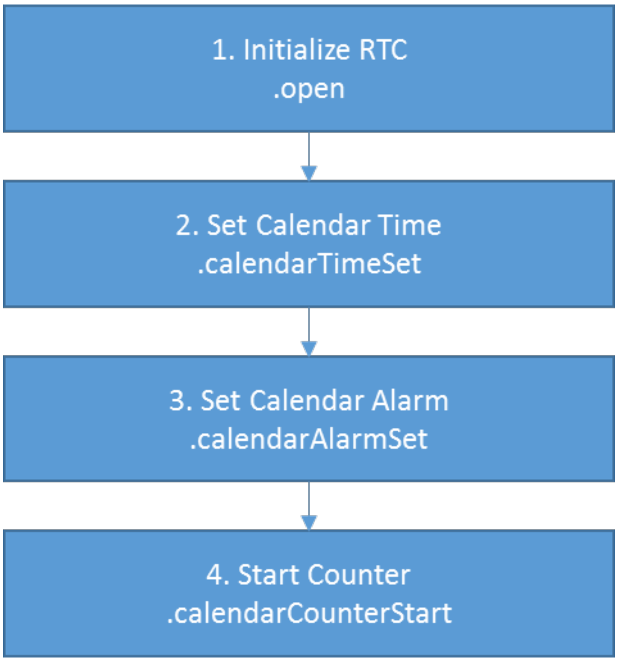

Figure 4. Flow Diagram of a Typical RTC Alarm Interrupt Application

7. The RTC HAL Module Application Project

The application project associated with this module guide demonstrates the previously mentioned steps in a full design. You may want to import and open the application project within the ISDE and view the configuration settings for the RTC HAL module. You can also read over the code (`in rtc_hal_api_mg.c` and `rtc_hal_mg.c`), which illustrates the RTC APIs in a complete design.

The application project demonstrates the typical use of the RTC HAL module APIs. The application project HAL entry initializes the RTC HAL modules. In the application project, the initialized RTC HAL module generates an interrupt periodically at every two seconds. Additionally, the application project generates an alarm interrupt, when the **Alarm Second** value matches the value of **Clock Second**. A user-callback function is entered when either of these interrupts is generated. If the callback function is called due to periodic interrupt, the function toggles LED2 on the board. In case a callback function call was made due to an alarm interrupt, the function toggles LED3 on the board. An alarm interrupt generates every minute starting from 03:05:05 (HH:MM:SS). LED1 shows activity on the board and toggles every second. Also, the clock time is printed at every one second using semi-hosting.

The following table identifies the target versions for the associated software and hardware used by the application project:

Table 5 Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e² studio	5.3.1 or later	Integrated Solution Development Environment
SSP	1.2.0 or later	Synergy Software Platform
IAR EW for Synergy	7.71.2 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	5.3.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1 or later	Starter Kit

A simple flow diagram of the application project is given in the following figure:

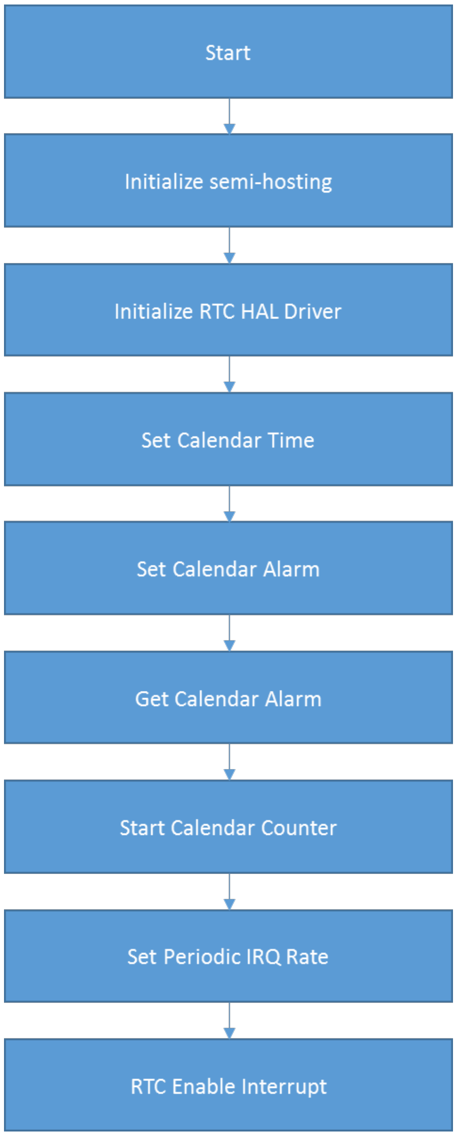

Figure 5 RTC Application Project Flow Diagram

The `rtc_hal_mg.c/.h` files are in the project once it has been imported into the ISDE. You can open these files within the ISDE and follow along with the description provided to help identify key uses of APIs.

All the configuration and initialization steps are in the `rtc_hal_mg.c` file. This file also uses semi-hosting to display results using `printf()` if #define `SEMI_HOSTING` is uncommented in the `rtc_hal_mg.h` file.

The first section in the `rtc_hal_mg.h` file includes #defines for semi-hosting, periodic interrupt, alarm interrupt, and periodic-interrupt rate. Other #defines in this header file initialize the alarm and calendar structures to set/get values to/from the RTC timer. The last section in this file has the prototype for the functions used in the application.

The associated source file, `rtc_hal_mg.c`, defines the entry function `rtc_IRQ_init()` for the RTC timer that initializes and configures the RTC HAL module. This file includes a user-callback function that toggles the respective LEDs based on a generated interrupt.

Note: This description assumes that you are familiar with using `printf()` with the Debug Console in the Synergy Software Package. If you are unfamiliar with this, refer to the "*How do I Use Printf() with the Debug Console in the Synergy Software Package*" Knowledge Base article, available as described in the References section at the end of this document. Alternatively, the user can see results via the watch variables in the debug mode.

A few key properties are configured in this application project. These properties support the required operations and the physical properties of the target board and the MCU device. The following table lists properties with the values set for this specific project. You can also open the application project and view these settings in the **Properties** window as a hands-on exercise.

Table 6 RTC Configuration Settings for the Application Project

ISDE Property	Value Set
Parameter Checking Enable	Enabled
Name	g_rtc0
Clock Source	LOCO
Configure RTC hardware in open() call	Yes
Error Adjustment Value	0
Error Adjustment Type	None
Callback	rtc_irq_callback
Alarm Interrupt Priority	Priority 3
Period Interrupt Priority	Priority 3
Carry Interrupt Priority	Priority 12

8. Customizing the RTC HAL Module for a Target Application

8.1 Change interrupt type

To change the interrupt type or to support more than one interrupt type, enable interrupt types in the configuration table. For the periodic interrupt, add the periodic interrupt event in the interrupt enable API.

8.2 Set periodic interrupt rate

A periodic interrupt rate can be configured to generate an interrupt at every 2, 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, or 1/256 second(s) using the periodic IRQ rate set API.

8.3 Set Alarm interrupt

To set an alarm interrupt, configure the RTC alarm time structure to enable the alarm for various entity matches and set the time in the RTC time sub-structure. Be sure to initialize the match variables with 1 or true in the RTC alarm time structure to generate an alarm interrupt when the RTC time matches the RTC alarm time.

9. Running the RTC HAL Module Application Project

To run the RTC HAL module application project and see it executed on a target kit, you import the application code into your ISDE, compile, and run debug. Refer to *Importing a Renesas Synergy Project* (`11an0023eu0116-synergy-ssp-import-guide.pdf`, included in this package) for instructions on importing the project into e^2 studio or IAR EW for Synergy and building/running the application.

To implement the RTC HAL module application in a new project, follow the steps for defining, configuring, auto-generating files, adding code, compiling, and debugging on the target kit. Following these steps is a hands-on approach that can help make the development process with SSP more practical, while just reading over this guide tends to be more theoretical.

Note: The following steps are described in sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are not familiar, refer to the first few chapters of the *SSP User's Manual* for a description of how to accomplish these steps.

To create and run the RTC application project, simply follow these steps:

1. Create a new Renesas Synergy project for the SK-S7G2 board called RTC_HAL.
2. Select the BSP or Blinky project template.
3. Open `Configuration.xml` file from the project.
4. In the **Threads** tab, select HAL/Common and add the RTC HAL on `r_rtc` stack from the HAL/Common Stacks Drivers Timers.
5. Change the configuration settings for the RTC HAL stack from the configuration properties window.
   a. Set Alarm interrupt priority
   b. Set Period Interrupt priority
   c. Set user-callback function name
6. Click on the **Generate Project Content** button.
7. Add the code from the supplied project file `rtc_hal_mg.c/.h` and add an entry function in `hal_entry.c` that calls `rtc_initiate()`.
8. Compile the application without errors and warnings.
9. Connect to the USB micro cable at J19 on SK-S7G2 board and connect other end of USB cable to the Host.
10. Start to debug the application.
11. As an output, LED1, LED2 and LED3 will toggle based on various events as follows:
    a. LED1: Toggles every 1 second to show activity on board.
    b. LED2: Toggles every 2 seconds from Periodic interrupt callback function.
    c. LED3: Toggles every minute after 03:05:05 (HH:MM:SS) when clock second value matches with the Alarm second value 05.
    d. If the user enables the semi-hosting option from `rtc_hal_mg.h` file, the output on console will look like the following figure:

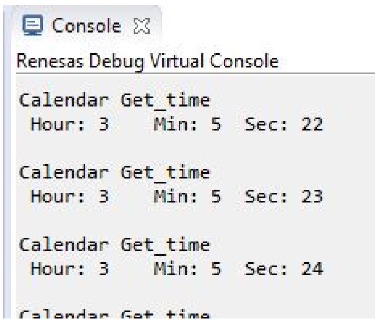

Figure 6 Example Output from RTC HAL Module Application Project

10. RTC HAL Module Conclusion

This module guide has provided all the background information needed to select, add, configure, and use the module in an example project. Many of these steps were time consuming and error-prone activities in previous generations of embedded systems. The Renesas Synergy™ Platform makes these steps much less time consuming and removes the common errors, like conflicting configuration settings or incorrect selection of lower-level drivers. The use of high-level APIs (as demonstrated in the application project) illustrates additional development time savings by allowing work to begin at a high level and avoiding the time required in older development environments to use or, in some cases, create, lower-level drivers.

11. RTC HAL Module Next Steps

After you have mastered a simple RTC HAL module project, you may want to review a more complex example.

The Developer Examples template, as described in section 12, has an example of the RTC HAL module-use that complements the application project described in this document. Other application projects and application notes that demonstrate RTC HAL module-use are available as described in section 12.

12. RTC HAL Module Reference Information

SSP User Manual: Available in html format in the SSP distribution package and as pdf from the Synergy Gallery.

Links to all the most up-to-date r_rtc module reference materials and resources are available on the Synergy Knowledge Base: https://en-us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Base/r_rtc_Module_Guide_Resources.

Website and Support

Visit the following vanity URLs to learn about key elements of the Synergy Platform, download components and related documentation, and get support.

Synergy Software	www.renesas.com/synergy/software
Synergy Software Package	www.renesas.com/synergy/ssp
Software add-ons	www.renesas.com/synergy/addons
Software glossary	www.renesas.com/synergy/softwareglossary
Development tools	www.renesas.com/synergy/tools
Synergy Hardware	www.renesas.com/synergy/hardware
Microcontrollers	www.renesas.com/synergy/mcus
MCU glossary	www.renesas.com/synergy/mcuglossary
Parametric search	www.renesas.com/synergy/parametric
Kits	www.renesas.com/synergy/kits
Synergy Solutions Gallery	www.renesas.com/synergy/solutionsgallery
Partner projects	www.renesas.com/synergy/partnerprojects
Application projects	www.renesas.com/synergy/applicationprojects
Self-service support resources:	
Documentation	www.renesas.com/synergy/docs
Knowledgebase	www.renesas.com/synergy/knowledgebase
Forums	www.renesas.com/synergy/forum
Training	www.renesas.com/synergy/training
Videos	www.renesas.com/synergy/videos
Chat and web ticket	www.renesas.com/synergy/resourcelibrary

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May.15.17	-	Initial Release
1.01	Aug.3.17	7	Update to Hardware and Software Resources Table
1.02	Sep.12.17	-	Fixed a bug in example code that surfaced with 1.3.0 where the initial year value was too low. It was set to 100. Minor formatting and language edits to the document.
1.03	Mar.20.19	9, 11	Updated files required for operation

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.