Renesas RA2 Series

Getting Started with Low Power Applications for RA2L1/RA2E1 Group

Introduction

This Application Note describes how you can reduce the effective power consumption of the RA Microcontroller using Low Power Modes (LPMs). Two accompanying application projects show common use cases of entering Low Power Modes and configuring the various peripherals to exit the entered mode. Upon completing this guide, you will be able to add an LPM module to your design, configure it correctly for the target application, and write code using the included application project as a reference and efficient starting point.

This application note describes:

- LPM module usage in different modes and supported peripherals.
- Application overview for the different use cases.
- FSP configuration steps for LPM.
- Application design highlights.
- Importing, loading, and running the application project.
- Project migration steps to other RA Kits.

Required Resources

- e^2 studio IDE v2024-01.1
- Flexible Software Package (FSP) v5.2.0
- J-Link RTT viewer V7.94g

Primary target devices

- EK-RA2L1 kit
- EK-RA2E1 kit

Table 1. RA Kits Tested with LPM Application

Kit	Operable Long Timer in LPM	LPM Transition and Clock Changing at Run-Time
EK-RA2E1	Yes	Yes
EK-RA2L1	Yes	Yes

Contents

1. Application Overview

The Application Projects accompanying this document serve as references for operating the microcontroller (MCU) in various Low Power Modes (LPMs), demonstrating different levels of power consumption often required to maximize battery life.

For ease of understanding the LPM, these application projects cover:

- Different Low Power Modes with different clock settings to showcase each mode
- Operation of different peripherals in an LPM
- Required pin configurations
- Trigger/end source configuration
- A user interface to initiate a transition to different LPM states and switch back to Normal mode.

The configuration for each mode is maintained as an independent instance. Users can use these example configurations and change different settings to trigger/end operation as desired.

In addition to the LPM, the application also supports changing the source clock of the MCU dynamically and running LPM for these clocks.

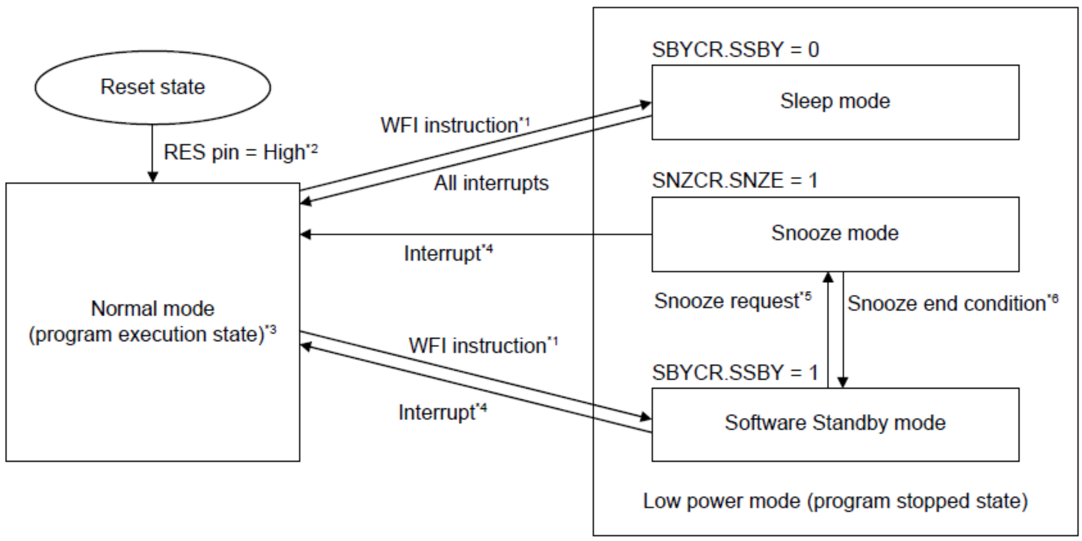
Note: In this application note, the project uses the default power supply source from LDO instead of the optional DC-DC regulator available on the MCU. For more details on using the LPM along with DC-DC regulator, see the MCU Hardware User's Manual.

1.1 Low Power Modes

RA MCUs support four different types of LPM depending on the MCU family. These are:

- Sleep mode
- Software Standby mode
- Snooze mode
- Deep Software Standby mode (Available only in some MCUs).

Low power mode transition and triggering sources for RA MCUs are illustrated in Figure 1. For more details on these transitions, see the User's Manual for the specific MCU.



Note: The MCU cannot transfer to Software Standby mode and Snooze mode in DCDC Power mode.

Note 1. When an interrupt that acts as a trigger for cancel is received during a transition to the program stopped state after the execution of a WFI instruction, the MCU executes interrupt exception handling instead of a transition to low power mode.

Note 2. The MOCO is the source of the operating clock following a transition from the reset state to Normal mode.

Note 3. The transition to Normal mode is made from an interrupt in Sleep mode, Software Standby mode, or Snooze mode. The clock source is the same as before entering the low power mode.

Figure 1. LPM Transition Diagram for RA2L1

In LPM, the CPU stops, but on-chip peripherals and oscillator states may be operational depending on the LPM selected. Therefore, their effects on MCU power consumption are very different. The typical current consumption when the MCU is in a Low Power Mode is found in the MCU Hardware User's Manual section on

Operating and Standby Current. Figure 2 shows the typical power consumption when the MCU is in a Low Power Mode vs throughput.

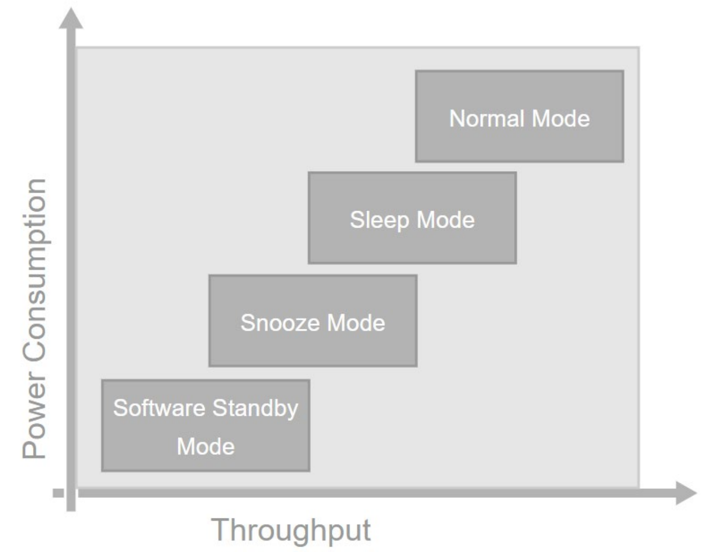


Figure 2. Power Consumption and Throughput of the LPM

In order for the MCU to enter or exit the LPM, associated special function registers need to be configured. This app note does not focus on the bit-level configuration details since the bits can be configured using the API provided by the FSP. The API provided by the FSP is documented in the FSP User's Manual. If you want to explore more details on the LPM and its supported list of peripherals and interrupts, refer to the Low Power Modes section in the RA MCU User's Manual. Low Power Modes commonly available with RA MCUs are described next.

MCU also supports different power control modes to reduce power consumption as part of the LPM. Power consumption can be reduced in Normal, Sleep, and Snooze modes by selecting an appropriate operating power control mode according to the operating frequency and voltage.

Four operating power control modes are available:

- High-speed mode
- Middle-speed mode
- Low-speed mode
- Subosc-speed mode

Note: The Power Control mode is not supported in the bundled application projects. For more details on power control modes, see the User's Manual for the specific MCUs.

1.1.1 Sleep Mode

An operational CPU is typically the primary cause of power consumption. In Sleep mode, the CPU stops operating, but the contents of its internal registers are retained. Other peripheral functions in the MCU do not stop. Available resets or interrupts in Sleep mode can cause the MCU to cancel Sleep mode. All interrupt sources are available in this mode to cancel the Sleep mode. When using an interrupt to successfully cancel Sleep mode, you must set the associated IELSRn register before executing a WFI instruction.

1.1.2 Software Standby Mode

In Software Standby mode, the CPU, most of the on-chip peripheral functions, and the oscillators stop operation. However, the contents of the CPU internal registers and the SRAM data, the states of the on-chip peripheral functions, and the I/O port states are retained. Software Standby mode allows a significant reduction in power consumption since most of the oscillators stop in this mode.

1.1.3 Snooze Mode

The Snooze feature provides operational flexibility to dramatically reduce current consumption. Snooze is an extension to the Software Standby mode, where limited peripheral modules can operate without waking up the CPU. The Snooze mode can be entered through the Software Standby mode via configured interrupt sources. Similarly, the system can be woken up from Snooze mode by interrupts supported in the Snooze mode.

1.2 Activation and Cancel Sources

Low power modes are canceled by various interrupt sources such as RES pin reset, power-on reset, voltage monitor reset, and peripheral interrupts. Refer to the Low Power Modes section in Renesas RA MCU User's Manual for a list of interrupt sources for different LPMs.

Only Snooze mode is triggered by a Snooze request to enter Snooze mode from Software Standby mode. The transitions to other LPMs are done by executing a WFI instruction with appropriate settings in the Standby Control register (SBYCR).

1.3 Peripheral Operation in LPM

Not all the MCU peripherals are available in different LPMs. MCU peripherals also have different retention capabilities for setting up the different LPMs. For example, the contents of the internal registers may be retained in some LPMs, but the contents may be undefined in other modes. Depending on the application's requirements, users are required to choose the peripherals and LPM settings to achieve maximum power savings. Users are also required to turn off/disable oscillators and on-chip peripherals that are not clock-gated or powered off to maximize power savings. Refer to the Low Power Modes section in each RA MCU User's Manual: Hardware to understand different oscillators and peripherals available in a specific LPM.

In the following sections, we will discuss the use case scenarios for the different LPMs with different clock settings and peripherals.

1.4 Use Case: Changing Clocks at Run-Time

This application use case describes how to dynamically change the RA MCU clock and set it to different clock settings supported by the RA MCU using the FSP CGC HAL driver APIs. While the user can configure the Clock Generation Circuits (CGC) within the MCU using the RA FSP Clock Configurator, in many applications where a battery eventually powers the MCU, there is an inherent requirement to change the clock configuration settings as the MCU is running. Based on the desired set of clock sources, the MCU changes to a different clock source and operates normally without rebooting.

Changing the system clock affects the peripherals, which use derivatives of the system clock as a source and other clocks in the system. Users are advised to select the dividers that are applicable to the system. When changing the clock, make sure to allow stabilization with the proper settling time. This stabilization time is designed for the CGC HAL Driver.

This application uses the user switch input to change the MCU clock mode from the previously running clock to the desired clock. The new clock settings are applied and displayed via the RTT interface for the user notification.

Table 2 shows the available user-selectable clock settings in the application.

Table 2. User Selectable Clock on EK-RA2L1 Board

Clock Source	
HOCO	High-Speed On-Chip Oscillator
MOCO	Medium Speed On-Chip Oscillator
LOCO	Low-Speed On-Chip Oscillator

The sequence of the clock being configured is HOCO→MOCO→LOCO →HOCO. The objective of this use case is to show the different clock sources that can be changed during run time without halting the MCU. Changing the clock dynamically is accomplished using the RA CGC HAL driver API `R_CGC_ClocksCfg`. For more details on CGC HAL driver API, refer to the FSP User's Manual.

1.5 Use Case: LPM Transition at Run-Time

This use case shows the different LPMs supported by the MCU for the different clock settings.

The application requires user input from the push button switch to change the LPM available for the MCU and perform transitions as programmed. The supported LPM and its transitions to the different LPMs are displayed using the RTT interface for notifying the user. The application also showcases using a few peripherals, like the AGT timer and RTC operating in different LPMs. It also regularly displays the RTC time information when the MCU transitions to the normal mode from the LPM. The AGT1 timer is used in the Snooze mode to alternate between the Software Standby and Snooze modes. RTC Alarm interrupt is used to cancel the Software Standby mode and enter normal mode. IRQn (User Switch Interrupt) is used to cancel the Sleep mode.

The visual indication of the LPM transition can also be seen with the User LED on the board. When the LED blinks approximately every 1 second, it runs in Normal mode. If the LED is turned OFF, it is in an LPM.

Note: More details on the application are explained in the architecture section 4.1 .The peripherals used in the application are just a few available for the MCU. For the complete list of peripherals supported in the LPM, refer to the LPM section of the MCU User's Hardware Manual.

Different clock sources and LPMs supported for the RA MCUs are shown in Table 3.

Table 3. Clock Sources and Supported LPM for EK-RA2L1 Board

Clock Source Supported	LPM Supported
HOCO	SLEEP, SW_STNDBY, SNOOZE
MOCO	SLEEP, SW_STNDBY, SNOOZE
LOCO	SLEEP, SW_STNDBY, SNOOZE

Transitioning to the different LPMs is accomplished by using the RA LPM HAL driver API `R_LPM_LowPowerModeEnter`. More details of this API can be found in the FSP User's Manual.

1.6 Use Case: Operable Long Timer in Software Standby Mode

The Operable Long Timer in Software Standby mode requires a timer that can operate in a Low Power Mode. The count source of the timer is another element that should also be considered carefully. In Renesas RA MCUs, the 16-bit Asynchronous General-Purpose Timer channel 0 (AGT0) and 16-bit channel 1 (AGT1) can be used in cascade mode to create a 32-bit timer. In the cascade mode, the AGT0 underflow interrupt will trigger the counter of AGT1; the AGT0 count source can be the sub-clock oscillator or LOCO clock, which are available in Software Standby mode. The AGT1 Underflow interrupt is used to wake the MCU up from LPM.

The maximum period of the 16-bit AGT timer channel 0 with the Sub-Clock count source running at 32.768 kHz is approximately 2.0 seconds. The Operable Long Timer with two AGT timer channels in cascade mode will have a maximum period of approximately 2184.5 hours with a timer resolution of 30.517 µs.

Note: If a longer wakeup time is required, the RTC can be used via the RTC alarm, but here the resolution of the timer is limited to 1 second.

Note: With RTC Periodic timer interrupt, the resolution of 1/256 sec can be achieved. However, the RTC periodic timer events cannot be linked to other peripherals with LPM operations.

2. LPM HAL Module

The LPM HAL module in FSP provides a method to include the LPM driver in the application and configure it for different modes. It also configures different trigger/cancel signals required for LPM activation/cancellation. FSP also provides essential APIs to configure and place the MCU in Low Power Modes. It supports the following Low Power Modes:

- Deep Software Standby mode (on supported MCUs)
- Software Standby mode
- Sleep mode
- Snooze mode

3. FSP Configuration

When developing an FSP application in e^2 studio, first configure the FSP using the RA Configurator. To properly configure the FSP, you must have detailed knowledge of both the software design that you will be implementing and the specific hardware it will be running on. For the hardware, this includes the types of peripherals to be used on the hardware and the pins they are mapped to, internal or external to the MCU. From the software perspective, you need to add the HAL modules for the peripherals you use and decide how many threads will be used and what additional software objects like semaphores, queues, and so on each thread will require. Once you have this information, you will be ready to successfully configure the FSP for your specific application needs.

In an application using FSP, the FSP configuration is stored in a file named `configuration.xml`. Double-clicking on this file brings up the **RA Configuration** tab for the project.

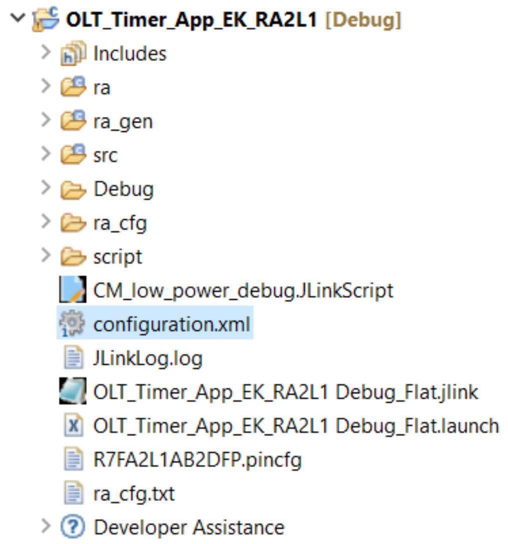


Figure 3. configuration.xml on the Project Explorer

When you build a project from scratch, this configuration tab is where you will perform the initial configuration of the FSP. As you can see in Figure 4, the **RA Configuration** pane contains a **Summary** screen highlighting the items you may configure, along with a scrolling window that lists all the software components currently selected for this project. Below this scrolling window are tabs that allow you to tailor the FSP to the needs of your specific application. More details on using the FSP configurator can be found in the FSP user manual.

For the purposes of this application note, we will highlight a few of the details of the FSP properties such as the r_lpm driver, r_rtc driver, and r_agt driver modules as they are key components operated in the use cases provided in the application.

When you have configured the project appropriately, click the **Generate Project Content**, the green arrow button above the summary screen, to build all the auto-generated files necessary to implement the components you defined.

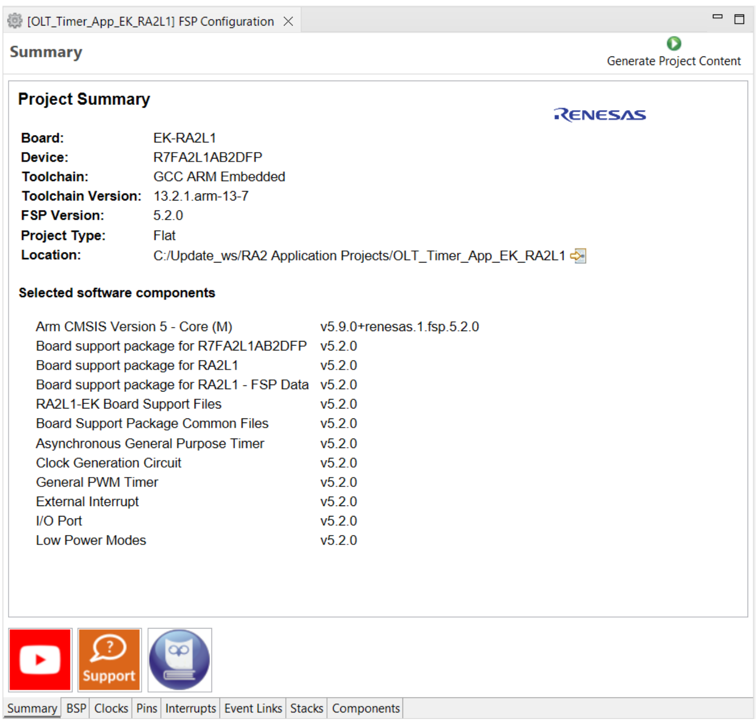


Figure 4. Summary of the Operable Long Timer Configuration

3.1 BSP Tab

For the Operable Long Timer Application, as shown in Figure 5, the RA Configuration pane contains a **Summary** screen highlighting the items you may configure in the BSP tab. **Properties → RA Common → Subclock Populated** should be selected as **Not Populated**.

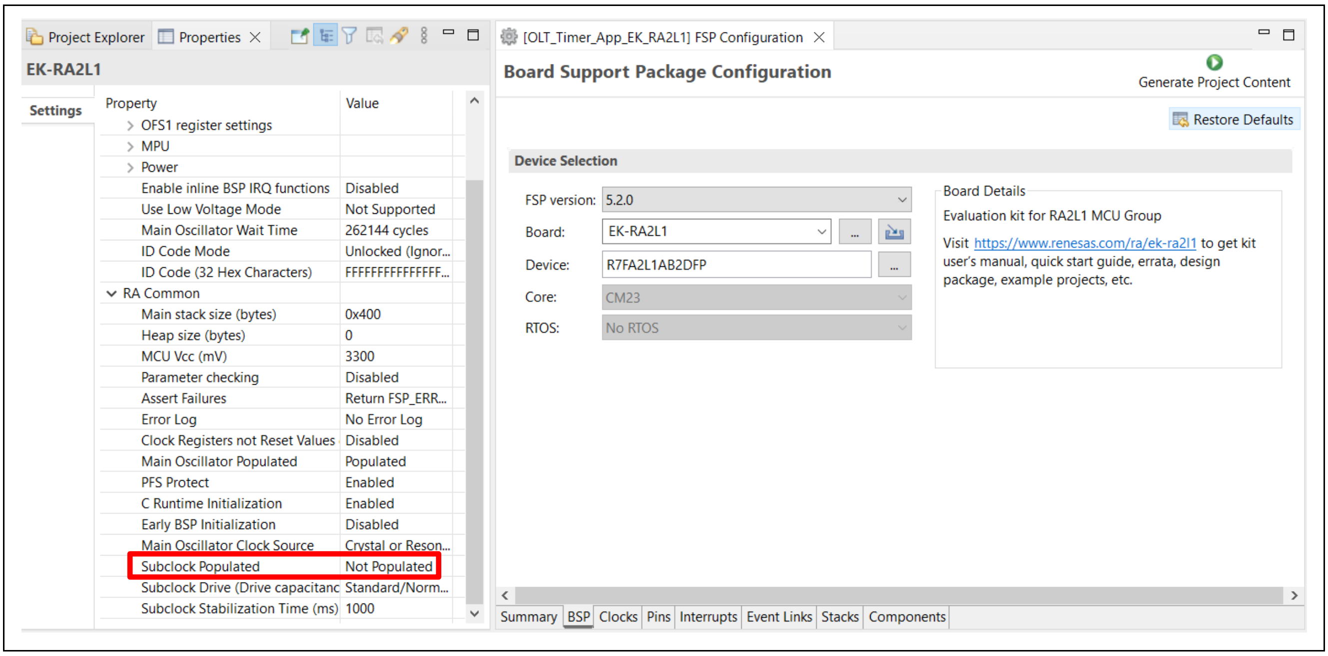


Figure 5. BSP Settings of the Operable Long Timer

Note: This setting is required only for the Operable Long Timer Application.

3.2 Components Tab

Even though the **Components** tab is the last tab showing, it is important to visit and verify that the configured components are checked against the desired FSP version. Components are automatically selected when the modules are added in the **Stack** tab specific to the application. As the final step to verify the chosen components, it is a good practice to confirm that these selections are checked in the **Components** tab. One of the advantages of the FSP is that it will only compile the components you choose, thereby reducing the size of your overall application. As shown in Figure 6, components are broken down into seven categories.

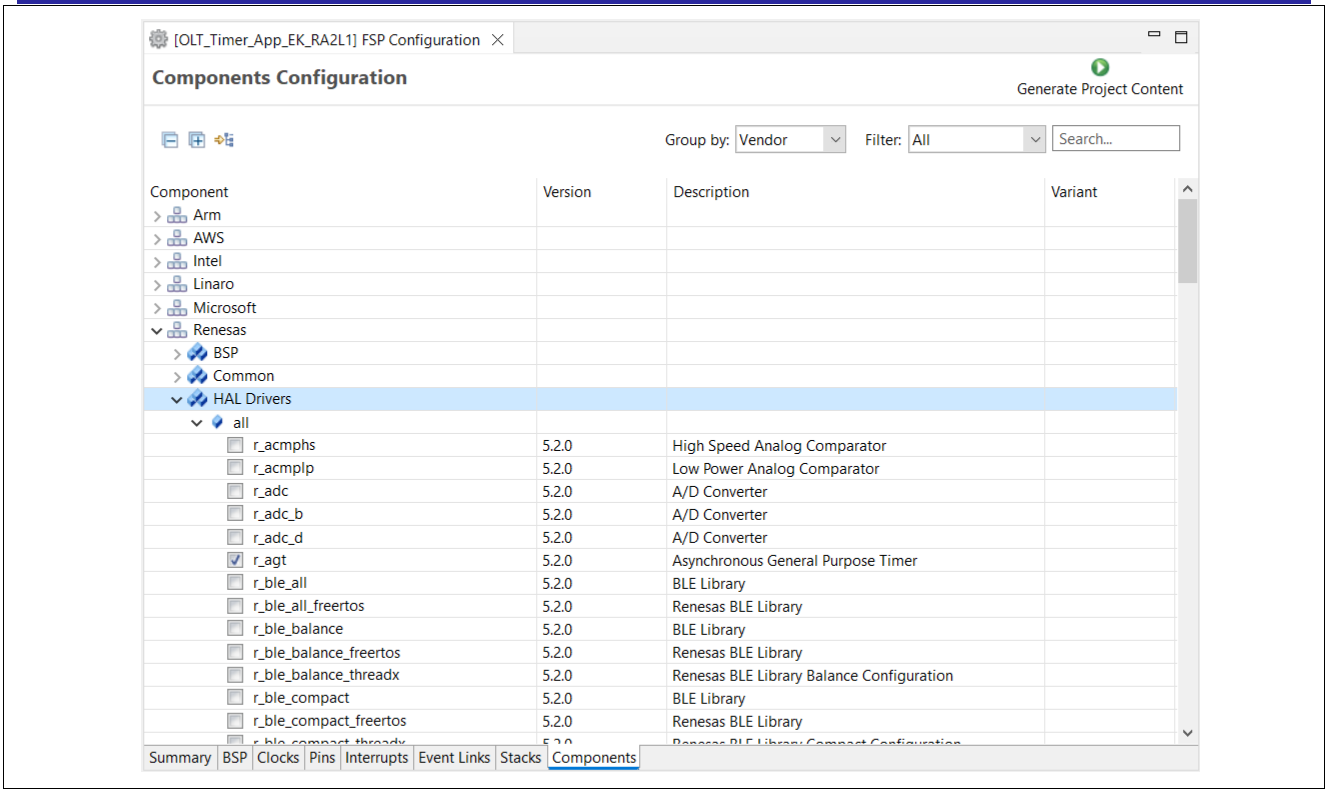


Figure 6. Components Tab Categories

You may expand any categories by clicking the arrow to the left of the category name. The following table highlights the selections used for the LPM applications.

Table 4. Components Used in the LPM Applications

Category	Component	Version	Description
BSP	`ra2l1_ek`	5.2.0	RA2L1-EK Board Support Package Files
	`ra2e1_ek`	5.2.0	RA2E1-EK Board Support Package Files
CMSIS	`CoreM`	5.9.0+renesas.1.fsp.5.2.0	Arm CMSIS Version5 - Core (M)
Common	`fsp_common`	5.2.0	Board Support Package Common Files
HAL Drivers	`r_cgc`	5.2.0	Clock Generation Circuit
	`r_ioport`	5.2.0	I/O Port
	`r_lpm`	5.2.0	Low Power Modes
	`r_icu`	5.2.0	External Interrupt
	`r_gpt`	5.2.0	General PWM Timer
	`r_agt`	5.2.0	Asynchronous General-Purpose Timer
	`r_rtc`	5.2.0	Real Time Clock

Note: This section is for user reference and read-only purposes. Don't select or deselect the options generated options by the FSP configurator.

3.3 Stacks Tab

The **Stacks** tab allows you to add and configure the threads that the FSP automatically creates for your application. You define a new thread by clicking the  button and then entering a unique name for your new thread. Once you add a new thread, you must define the modules that the thread will use along with any thread objects used by your application thread.

As an example, if you click the **Stacks** tab and then single-click on the **HAL/Common** thread, you should see something like the screen capture shown in Figure 7. This shows that the application requires multiple drivers, like the r_lpm driver which is the driver for Low Power Modes of Renesas RA MCU. The LPM applications do not use RTOS, so there is only one HAL/Common thread available for this type of application.

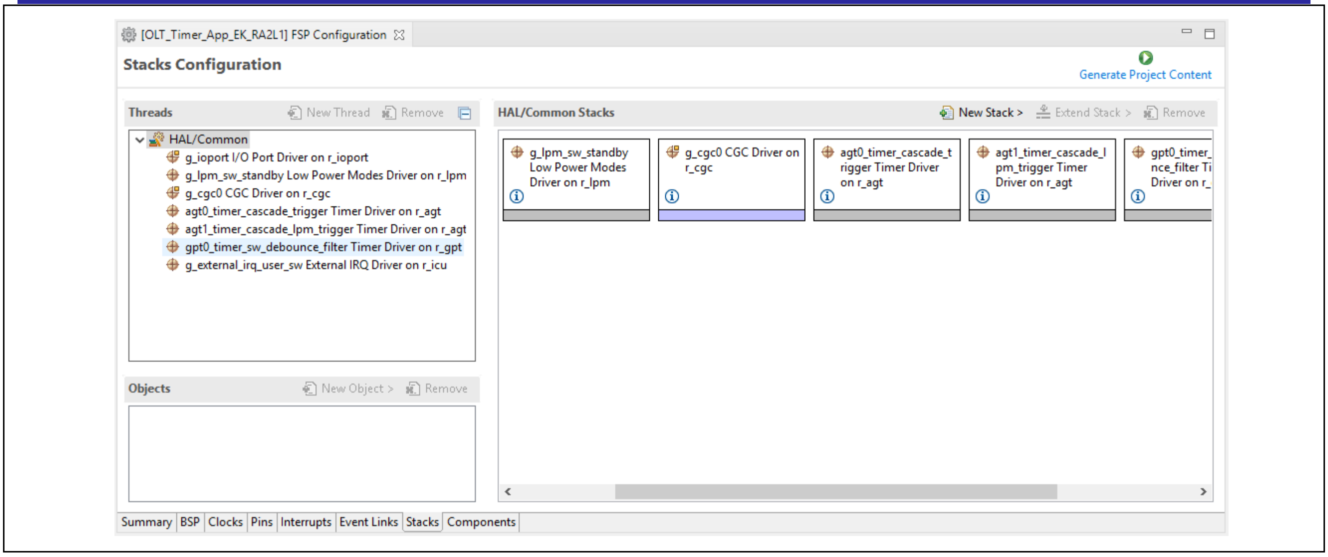


Figure 7. Driver Usage in LPM Application

You can add additional modules to a thread by clicking the button. As an example, Figure 8 shows how to add an AGT timer. The timer is added by choosing New Stack > **Timers** > **Timer, Low Power (r_agt)**.

If you pick a module that you have not preselected, the appropriate component for the module will be automatically selected for you by FSP. If the configurator tool detects errors due to incorrect settings with the module addition, it presents the module with an error. You may examine the errors by hovering over the module name.

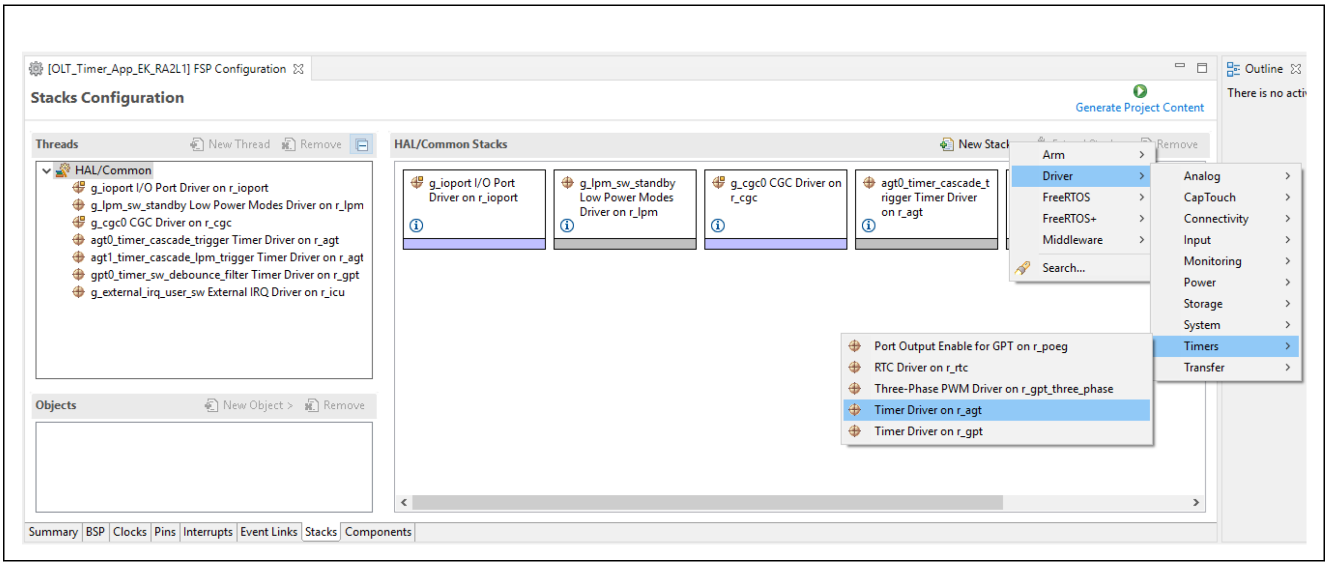


Figure 8. Adding r_agt Module to HAL/Common Thread

3.4 Module Configuration

Once you have added a module to your project, you need to configure its properties. The properties are dependent on the module(s) that you have added. Use the **Properties** tab to configure them.

3.4.1 LPM Configuration

The LPM applications add the r_lpm driver module as the main component to configure the Renesas RA MCUs in Low Power Modes, for Sleep, Software Standby and Snooze. The main settings of Low Power Modes configures the trigger/cancel sources and different modes.

3.4.1.1 Activation and Cancelation Sources

The cancelation source of an LPM will wake up the MCU from the specific LPM. The request of the Snooze mode puts the MCU into Snooze mode from Software Standby mode. These sources are interrupt sources.

Refer to the Low Power Modes section in Renesas RA MCU Hardware Manual for more details on what interrupts are available in the LPM.

Table 5 shows the activation and cancelation sources used in the LPM applications.

Table 5. Activation and Cancelation Source Configuration

Category	Interrupt Source	Application	Description
Request Source	AGT1_AGTI	Clock Changing and LPM Transition	AGT Channel 1 Underflow Interrupt
End Source	AGT1_AGTI	Clock Changing and LPM Transition	AGT Channel 1 Underflow Interrupt
Wake/Cancel Source	AGT1_AGTI	Operable Long Timer	AGT Channel 1 Underflow Interrupt
	PORT_IRQ3	Clock Changing and LPM Transition	External Interrupt 3
	RTC_ALM	Clock Changing and LPM Transition	RTC Alarm Interrupt

3.4.1.2 Sleep Mode Configuration

Since Sleep mode is canceled by any interrupt, there is no need to set up a cancel source for this mode in r_lpm driver configuration as long as there is at least one source of interrupt active in the system. If the LPM application is using RTOS, the Systick timer must be stopped before entering the Sleep mode because the Systick interrupt will wake up the MCU. When you are using the RTOS, if the Systick timer is stopped, it must be restarted after waking up for the proper RTOS kernel operation.

3.4.1.3 Software Standby Mode Configuration

Figure 9 shows how the LPM may be configured to exit Software Standby mode with AGT1 underflow interrupt as the wake source.

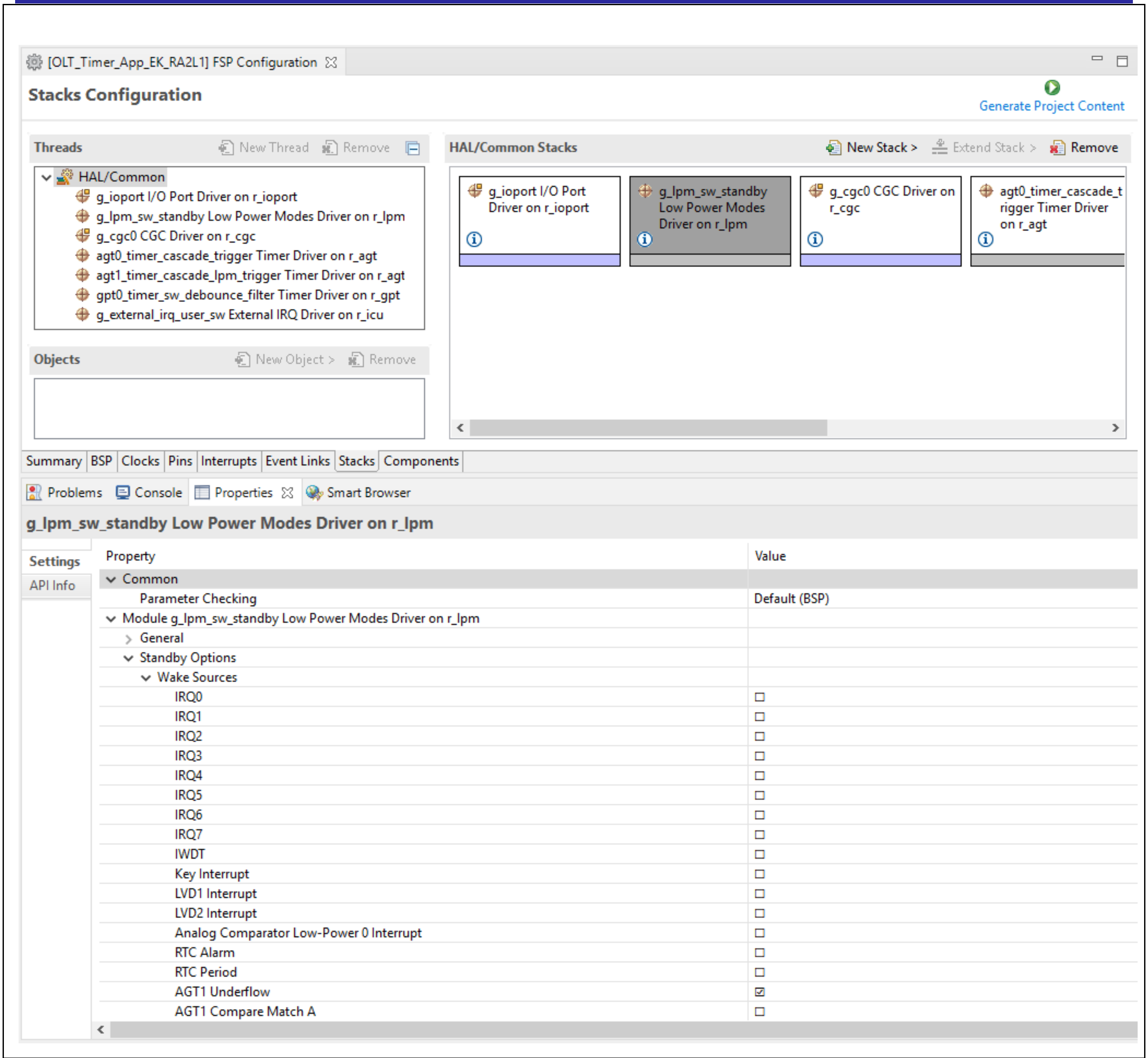


Figure 9. Software Standby Properties Configuration using the Properties Tab

3.4.1.4 Snooze Mode Configuration

The Snooze Request source triggers a transition from Software Standby mode to Snooze mode. The Snooze End source cancels Snooze mode and transitions the MCU back to Software Standby mode.

The wake source of Software Standby mode will wake up the MCU from both Software Standby and Snooze modes.

Figure 10 shows how to configure Snooze mode with AGT1 Underflow as both Snooze Request and End sources.

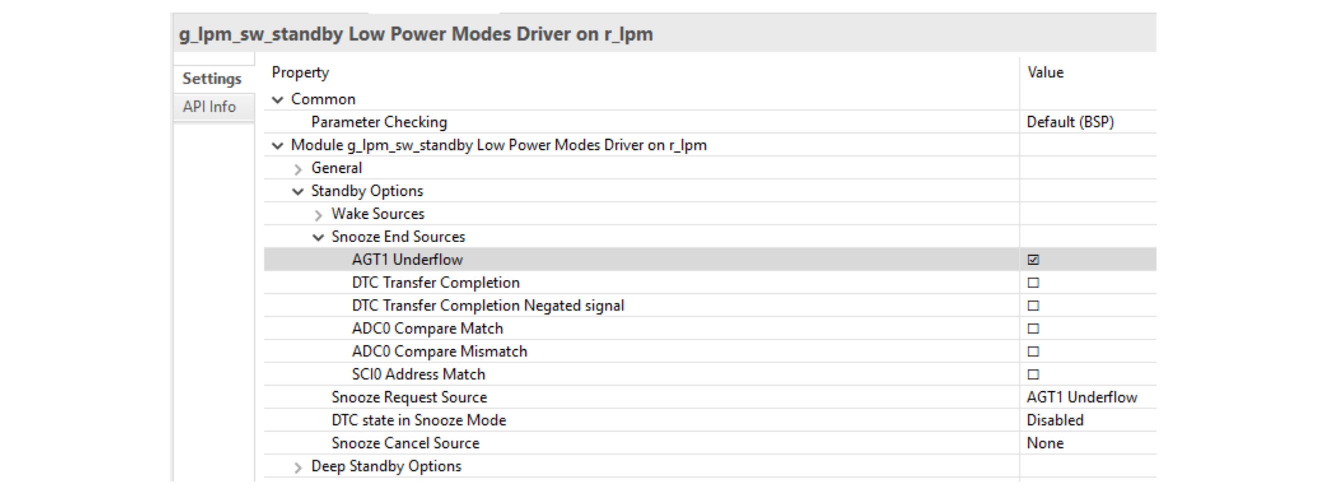


Figure 10. Snooze Properties Configuration using the Properties Tab

3.4.2 Timer Configuration

3.4.2.1 RTC Configuration

The Real Time Clock (RTC) is one of the peripherals that can operate in all the Low Power Modes. The RTC is primarily used for timekeeping, which updates the time independent of MCU in LPM. In this application, the RTC is used for keeping track of the time in the Clock Changing and LPM Transition application and to wake up from Software Standby mode via the RTC alarm interrupt. RTC uses the LOCO as the clock source in the application.

The application displays the RTC time when the MCU transitions from LPM to Normal mode. This indicates how long the MCU was in LPM and the latest time information.

Figure 11 shows the configurations of RTC for Time and Alarm. Alarm is used as a trigger in the LPM. The clock source of RTC is LOCO, which is available in Low Power Modes.

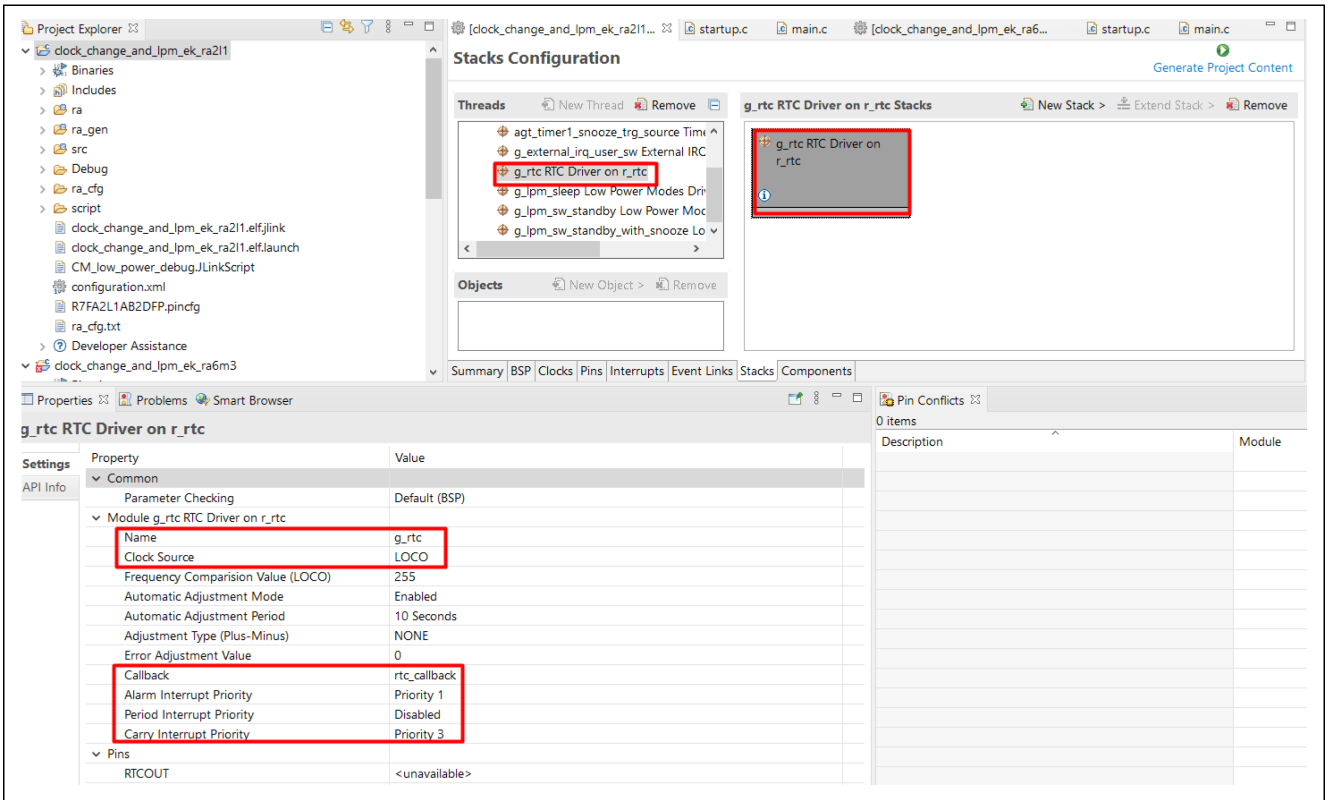


Figure 11. RTC Properties Configuration using the Properties Tab

3.4.2.2 AGT Timer Configuration

As mentioned earlier, the Operable Long Timer application in LPM uses AGT timer channel 0 (AGT0) and AGT timer channel 1 (AGT1) in cascade mode to create a 10-second operable long timer. In this mode, the AGT0 underflow interrupt will trigger the AGT1 counter.

The following figures show the configurations of AGT0 and AGT1. The Count source of AGT0 is the Sub-Clock, which is available in Low Power modes, and the Count source for AGT1 is the AGT0 underflow interrupt.

AGT0 is configured in a periodic mode that generates an underflow interrupt every second, as shown in Figure 12.

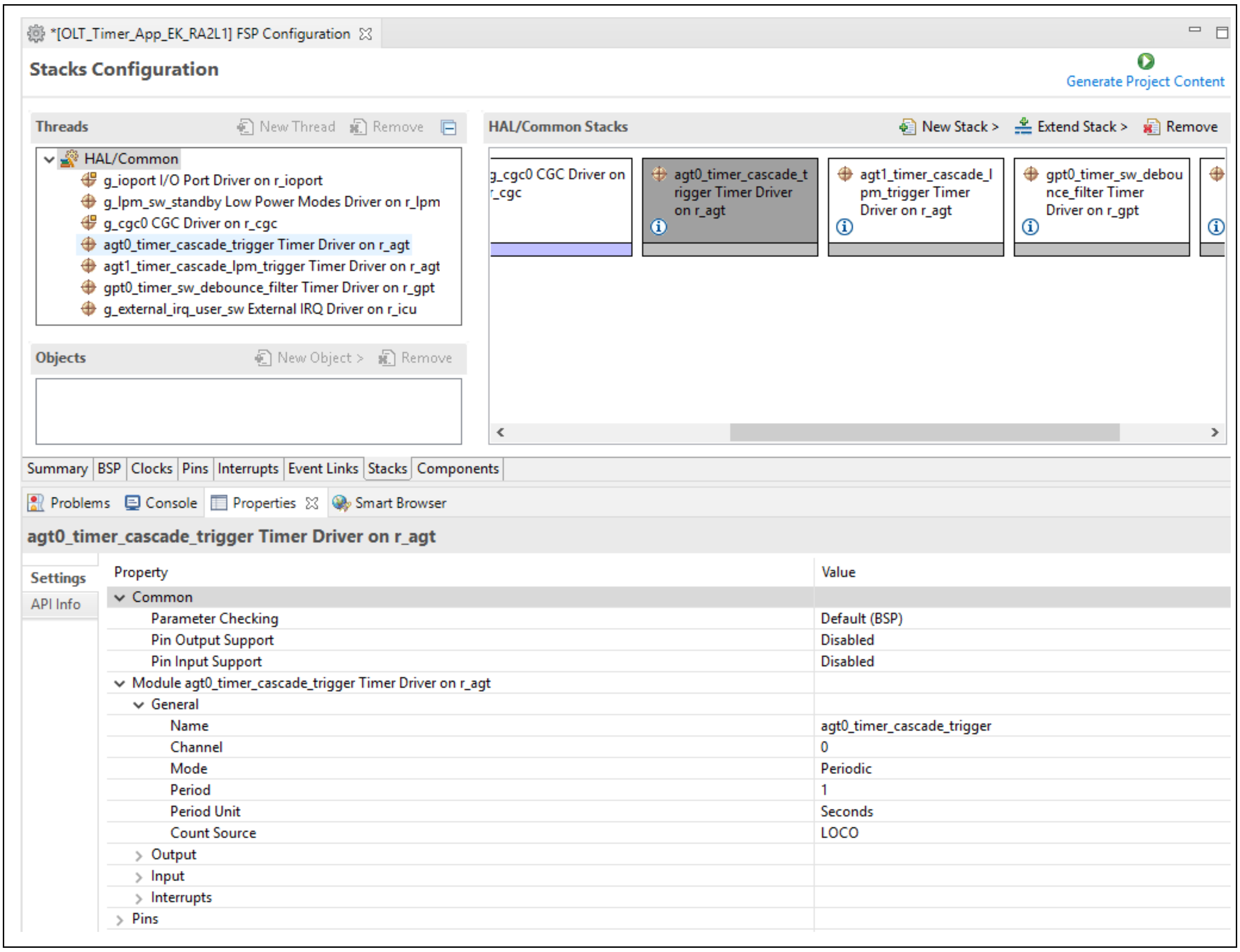


Figure 12. AGT0 Properties Configuration using the Properties Tab

AGT1 is also configured in a Periodic mode with Raw Counts as the period unit shown in Figure 13.

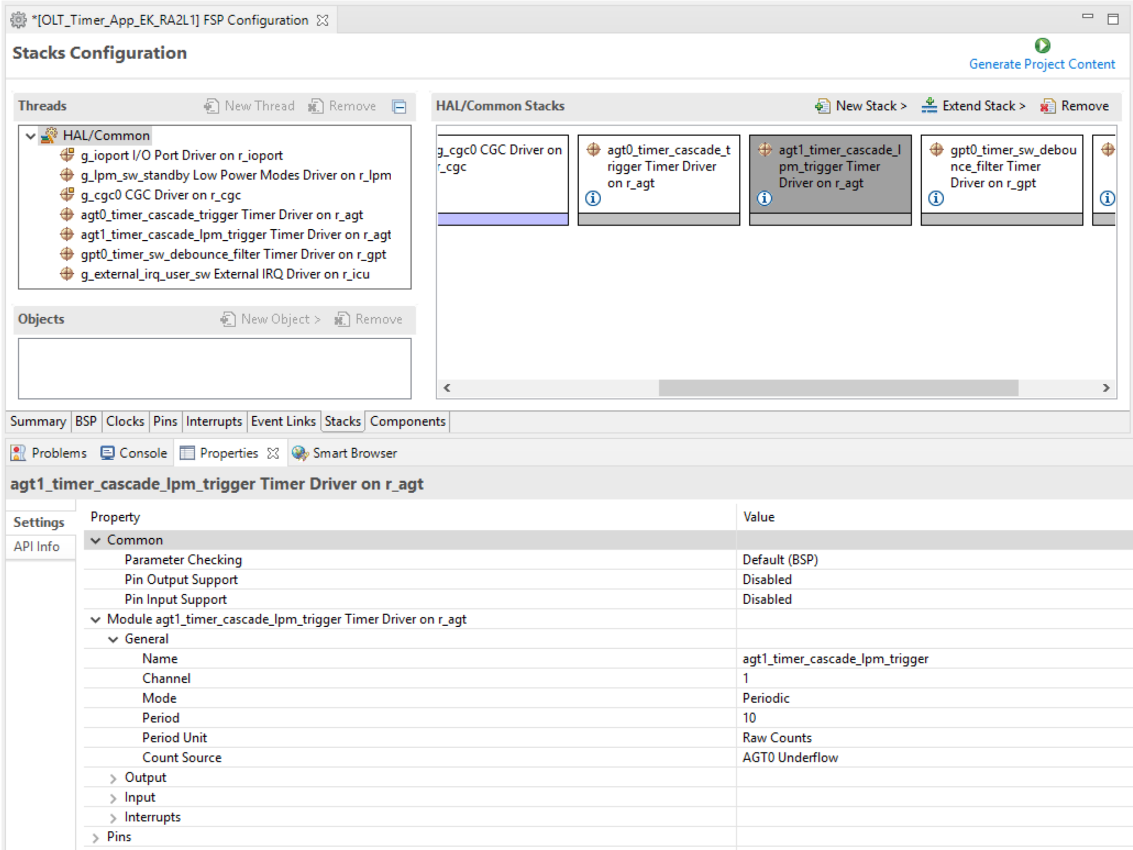


Figure 13. AGT1 Properties Configuration using the Properties Tab

3.5 Pin Configuration

The FSP application can support multiple pin configurations. In this application, we use two different pin configurations, one for active mode of operation and other for power saving mode operation. Refer to the Renesas Flexible Software Package (FSP) User's Manual on how to configure the FSP Pin Configuration.

3.5.1 Pin Configuration in Normal Mode

The pin configuration in normal mode is the MCU pin function that you want to use in normal operating conditions. Figure 14 shows the pin configuration of the EK-RA2L1 kit, which is used in normal mode.

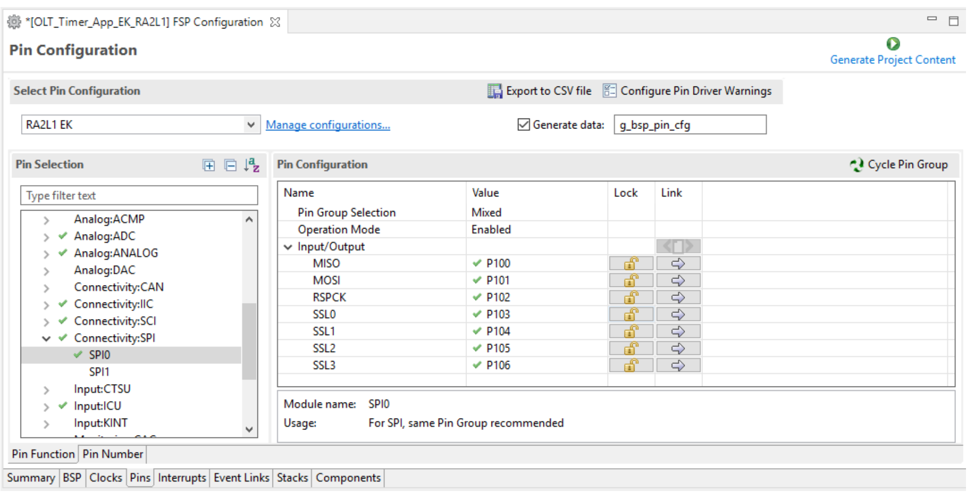


Figure 14. Pin Configuration of EK-RA2L1 Kit in Normal Mode

3.5.2 Pin Configuration in LPM

Use the pin configuration in Low Power Modes to reduce power consumption by disabling unused pins, which puts them in the input mode. Refer to the “Handling of Unused Pins” section of the MCU Hardware user manual for more details.

Figure 15 shows the pin configuration of the EK-RA2L1 kit in LPM named `RA2L1-EK-LPM.pincfg` with unused pins disabled. You may observe that most pins are disabled, except for the IRQ pins and CGC pins, which are used to wake up the MCU and provide clock input through the XTAL and EXTAL pins.

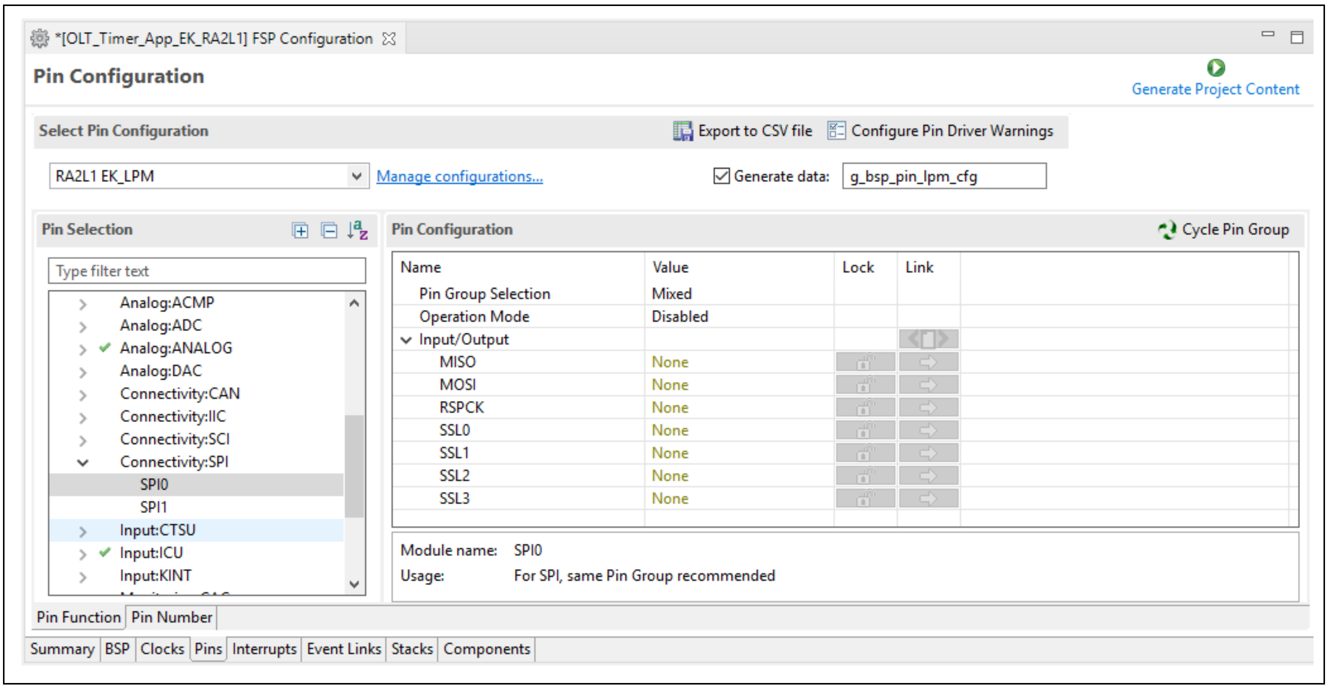


Figure 15. Pin Configuration of EK-RA2L1 Kit, Operating in LPM

4. Application Architectures

The following Figure 16 shows transitions between LPM and normal mode in LPM applications. The WFI instructions activate LPM, and the configured interrupt, such as AGT1 underflow interrupt, cancels LPM and wakes up the MCU.

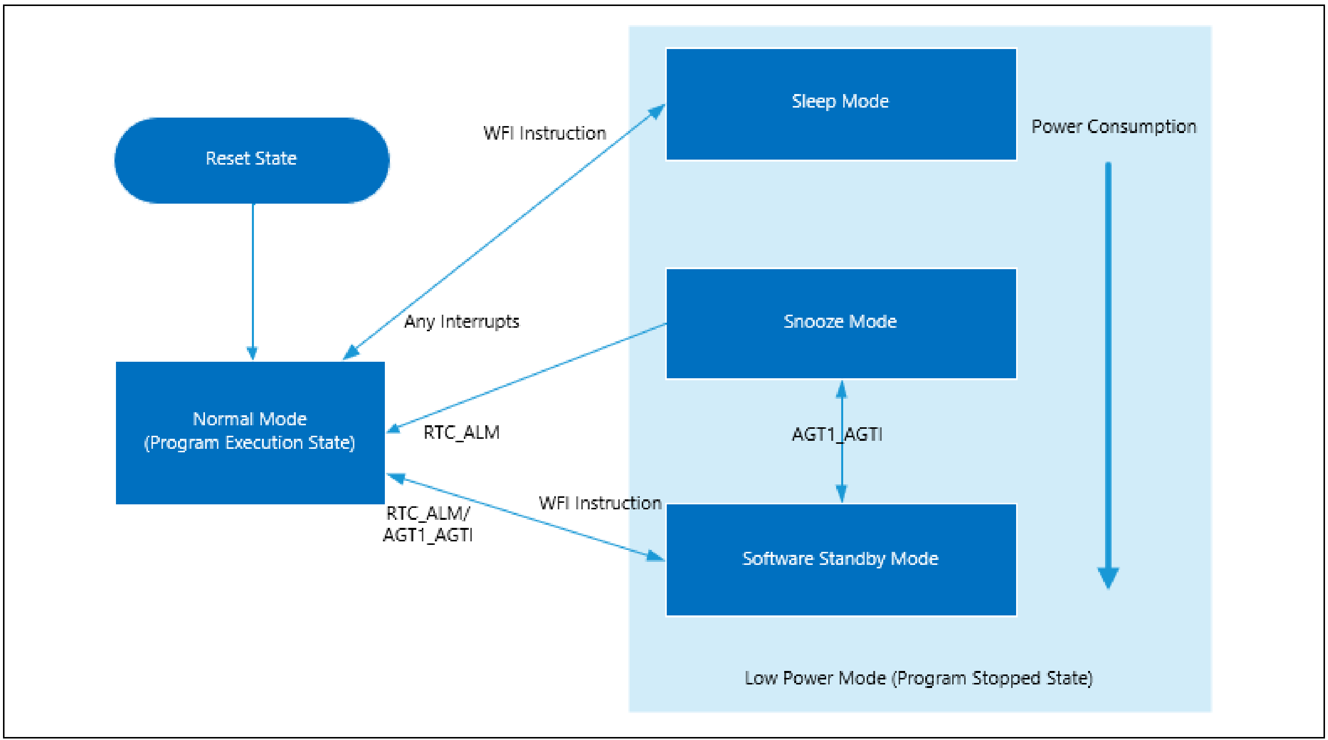


Figure 16. Transition Between LPM and Normal Mode

4.1 Clock Changing and LPM Transition

This application demonstrates the use case where the clock source to the CPU is changed at runtime. It also demonstrates entering and exiting the different LPM using the API provided by the FSP. Without the availability of the APIs, a developer would need to configure the registers related to LPM and CGC manually, thereby adding to the development timeline.

Figure 17 shows the different clock sources and the associated LPM used in the application and different transition states. The application is implemented using the event driven mechanism. In this event driven system, events can be user driven events or system generated events, which are used as the input to the finite state machine. Two separate state transition tables are used here in the application.

- Clock transition table (`clock_transition_table`)
- LPM Transition table (`lpm_transition_table`).

The transition table has the list of actions to be performed based on the events received. For instance, when the user event “Button Press - Long” is received upon power-on reset, the finite state machine will start running and change the clock to HOCO. If user event “Button Press – Short“ is received, the finite state machine will switch to the LPM state machine and start the LPM operation.

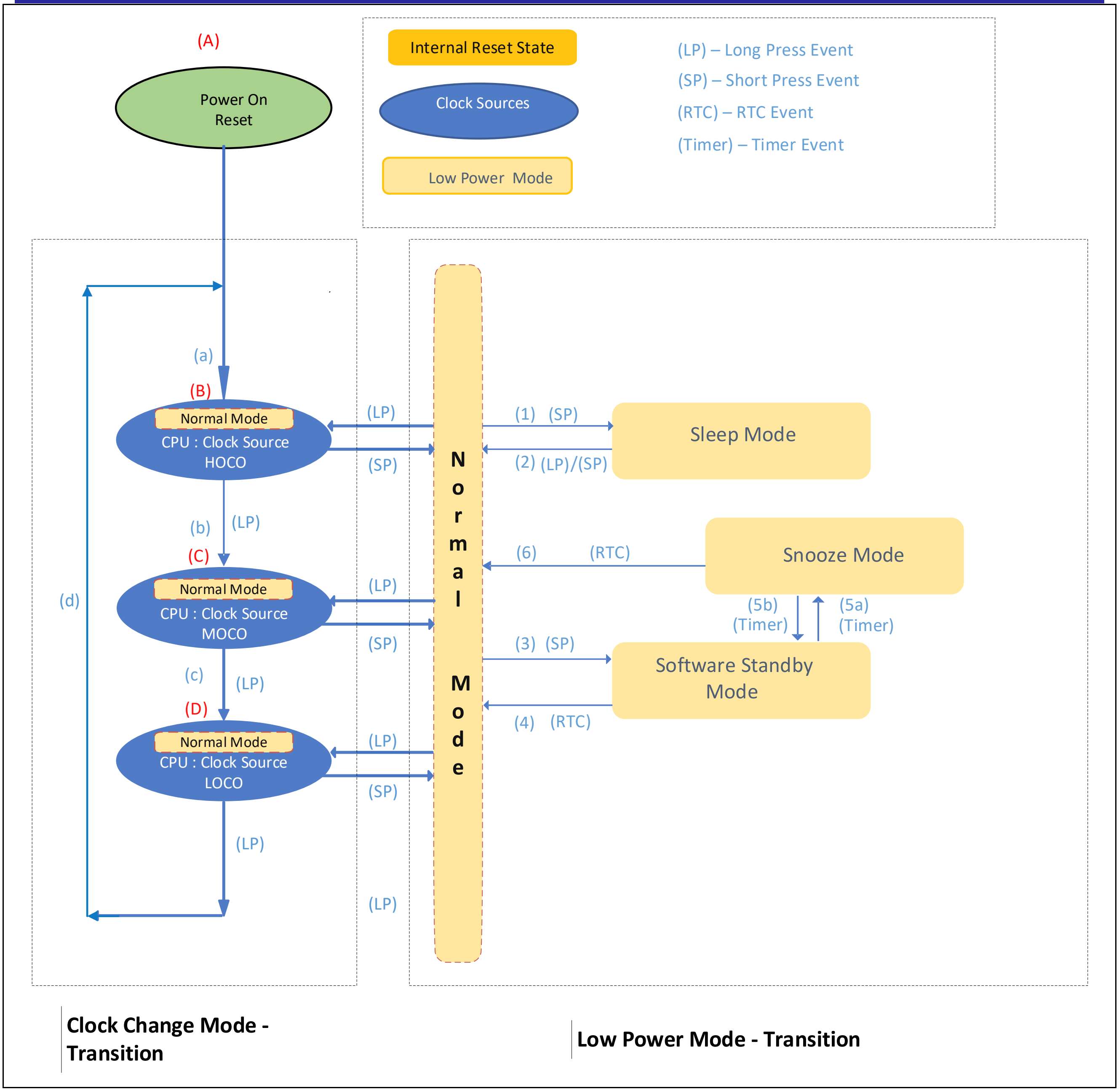


Figure 17. Clock Changing and LPM Transition

In the above Figure 17, the blue-colored oval-shaped blocks (B) through (D) represent the different clock states used in the application, and (a) through (d) labeled arrows are different transition paths it takes when changing the clock.

The yellow blocks are the different LPM states as applicable to the MCU, and (1) through (6) numbered arrows represent the transition path at the different Low Power Modes.

Note: The dotted block represents the Clock Change Mode transition on the left and the LPM Transition on the right, as shown in Figure 17.

Note: The Snooze mode is entered via Software Standby mode. For the Snooze mode, from the application perspective, the MCU and LPM drivers handle the Software Standby mode internally, while it is configured for the Snooze mode.

Note: The clock can only be changed in the Normal mode. Changing clock in the LPM mode is not allowed.

Note: In the above Figure 17, for low-end RA devices (RA2XX), Sleep, Software Standby, and Snooze modes are supported.

Events used in this application are as listed in the Table 6.

Table 6. Events used for the Clock and LPM transitions

List of Event	Description
EV_PB_SHORT_PR	User push button event – "Short Press" – held for 1-2 seconds
EV_PB_LONG_PR	User push button event – "Long Press"– held for 4-6 seconds
EV_PERIODIC_TIMER	AGT1 timer event generated by timer overflow
EV_RTC_ALARM	RTC Alarm Interrupt generated based on the configured time.
EV_POWER_ON_RESET	Power on Reset event,

4.2 RTC Timer Operation in LPM

Realtime clock (RTC) timer operation is an additional feature of the clock changing and LPM transition application, which showcases the running RTC peripheral during the LPM. Even when the CPU and most components in the MCU enter Low Power Modes and cease operation, the RTC clock and its timer operate independently. Updated RTC time information is displayed in the RTT when it transitions back to the Normal mode.

RTC Periodic/Alarm Interrupts can be used as signals for transitioning to different LPMs. In this LPM application project, the RTC Alarm interrupt is used for canceling the Snooze mode/Software Standby mode and revert to the Normal mode.

Note: The RTC Periodic interrupt can also be used for this event. The RTC Periodic interrupt has a maximum period of 2 seconds. For demo purposes, we are avoiding this to showcase the step-by-step transition and not to be limited to the 2 seconds. However, the RTC alarm can be configured to the desired number of seconds.

4.3 Operable Long Timer in Software Standby Mode

The Operable Long Timer Application uses LPM configurations in Software Standby mode. It disables unused clock and IO ports before entering LPM with the WFI instruction, then places IO ports back to normal operating condition after waking up as shown in Figure 18.

LPM is canceled by the Operable Long Timer underflow interrupt, which is created by using AGT0 and AGT1 in cascade mode.

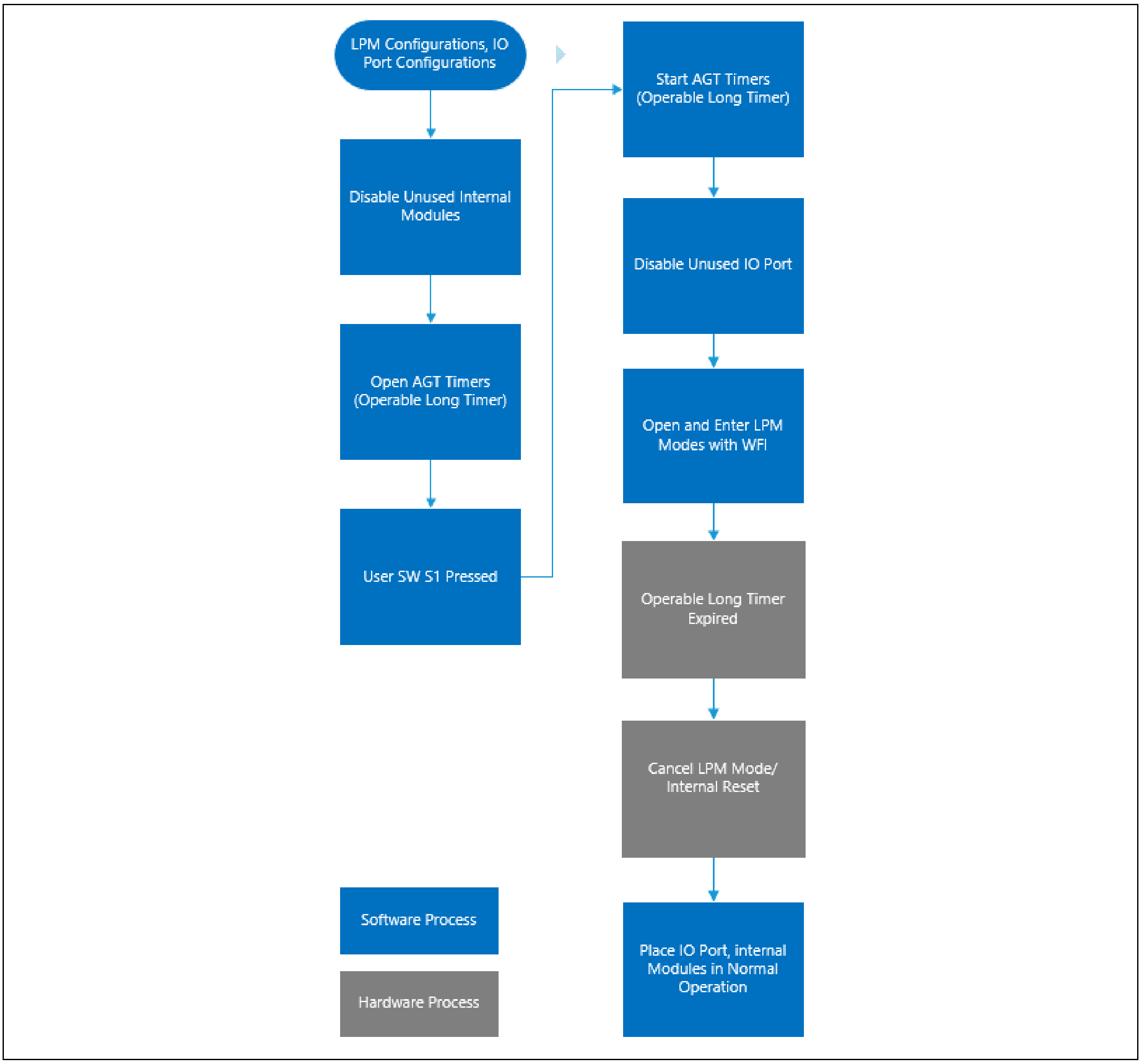


Figure 18. Operable Long Timer Process

5. Application Code Highlights

5.1 Clock Source Setup

5.1.1 Handling On-Chip Modules in LPM to Reduce Power Consumption

Oscillators and on-chip modules may be started automatically after the MCU reset and still be running in LPM modes. Therefore, to reduce MCU power consumption, you should disable these modules before entering LPM.

Unused IO ports in LPM other than in Deep Software Standby mode should be put into input mode before entering the LPM by using the `g_bsp_pin_lpm_cfg`, which is generated from the `RA2L1-EK-LPM.pincfg`. The following code fragments highlight the steps in the application to reduce MCU power consumption before placing the MCU in LPM and restoring the IO port settings after waking it up.

```
/* Disable IO port if it's not in Deep SW Standby mode */
if(APP_LPM_DEEP_SW_STANDBY_STATE != g_lpm_transition_sequence[g_lpm_transition_pos])
{
    /* Disable IO port before going to LPM mode*/
    err = R_IOPORT_PinsCfg(&g_ioport_ctrl, &g_bsp_pin_lpm_cfg);
    /* Handle error */
    if(FSP_SUCCESS != err)
    {
        APP_ERR_TRAP(err);
    }
}
```

Figure 19. Put Unused IO ports in Input Mode

Open and configure the LPM.

```
/* Open LPM instance*/
err = R_LPM_Open(&g_lpm_ctrl_instance_ctrls[g_lpm_transition_pos], &g_lpm_ctrl_instance_cfgs[g_lpm_transition_pos]);
/* Handle error */
if (FSP_SUCCESS != err)
{
    APP_ERR_TRAP(err);
}
```

Figure 20. Open and Configure the LPM

Place the MCU in the LPM.

```
/* Enter LPM mode */
err = lpm_mode_enter(g_lpm_transition_sequence[g_lpm_transition_pos]);
/* Handle error */
if (FSP_SUCCESS != err)
{
    /* Turn on user LED to indicate error occurred*/
    R_IOPORT_PinWrite(&g_ioport_ctrl, leds.p_leds[LED_NO_0], BSP_IO_LEVEL_HIGH);
    APP_ERR_TRAP(err);
}
```

Figure 21. Place the MCU in the LPM

Put IO port in normal mode by configuring it using the `g_bsp_pin_cfg` generated from the `RA2L1-EK.pincfg` after exiting from LPM.

```
/* Put IO port configuration back to user's selections */
err = R_IOPORT_PinsCfg(&g_ioport_ctrl, &g_bsp_pin_cfg);
/* Handle error */
if(FSP_SUCCESS != err)
{
    APP_ERR_TRAP(err);
}
```

Figure 22. Place IO Ports in Normal Mode

The following code stops the LOCO clock when it is not used as a count source for AGT0, as shown in Figure 23.

```
/* Stop LOCO clock if it's unused (not use as AGT1 count source)*/
if(AGT_CLOCK_LOCO != p_agt0_extend->count_source)
{
    err = R_CGC_ClockStop(&g_cgc0_ctrl, CGC_CLOCK_LOCO);
}
```

Figure 23. Stop LOCO Clock when It is Unused

5.1.2 Changing System Clock at Run-Time

When the MCU powers up, the default clock will be the configured clock as part of the BSP. Selecting the clock for the application is done through the FSP configurator using the clock tree, which is available in the clocks tab.

In the changing clock use case application, the current running system clock is read via the `R_CGC_SystemClockGet` API, and a new system clock source is configured. The following code is used for reading the currently running system clock shown in Figure 24.

```
err = R_CGC_SystemClockGet(&g_cgc0_ctrl, &sys_clock_source, &sys_divider_cf);
if (FSP_SUCCESS != err)
{
    APP_ERR_TRAP(err);
}
```

Figure 24. Read the running System Clock

In order to configure the new source for the system clock, proper divisors are required so that the peripherals get the permitted range of frequencies. Users need to calculate this based on the peripherals used in the application. The new system clock is configured via the API `R_CGC_ClocksCfg`.

Also, the clock sources need to be started if they are not already running. This is done using the API `R_CGC_ClockStart` as shown in the Figure 25.

```
if (CGC_CLOCK_SUBCLOCK == sys_clock_source)
{
    new_clk.source_clock = CGC_CLOCK_SUBCLOCK;
    err = R_CGC_ClockStart(&g_cgc0_ctrl, CGC_CLOCK_SUBCLOCK, &new_clk);
    if (FSP_SUCCESS != err)
    {
        APP_ERR_TRAP(err);
    }
}
```

Figure 25. Starting the Clock Source

6. Importing and Building the Project

To bring the applications into the e² studio ISDE, follow these steps:

1. Launch e^2 studio ISDE.
2. In the workspace launcher, browse to the workspace location of your choice.
3. Close the **Welcome** window.
4. In the ISDE, go to **File** > **Import.**
5. In the Import Dialog Box, pick **Existing Projects into Workspace.**
6. Select the root directory of your workspace (where you placed the project).
7. Select the project you wish to import and click **Finish**.
8. Click on **Generate Project Content** on the FSP configurator window.
9. Now build the project.

7. Running Applications

To connect and run the code, follow these steps:

7.1 Board Setups

The EK-RA2L1 kit contains a few switch settings that must be configured prior to running the application associated with this application note. In addition to these switch settings, it has a USB debug port and connectors to access the J-Link® programming interface.

Table 7. Switch Settings for EK-RA2L1

Switch	Setting
J8	Jumper on pins 1-2
J9	Open
J29	Jumper on pins 5-6
J29	Jumper on pins 7-8
J6	Open

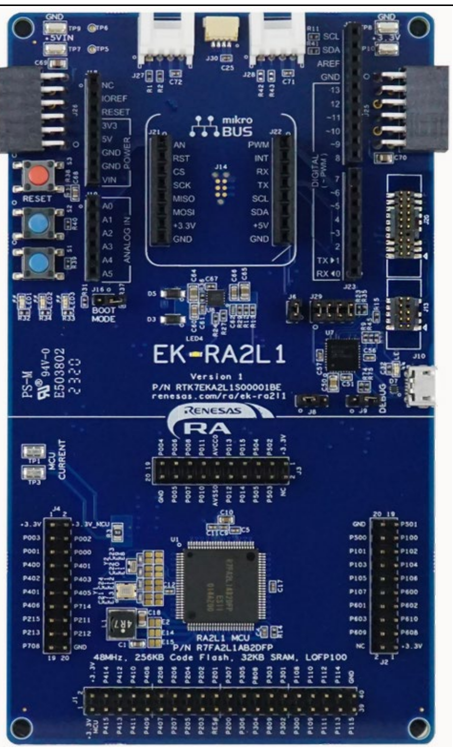


Figure 26. EK-RA2L1 Kit

7.2 Downloading the Executables

The executable file may be programmed into the target MCU through any one of three means.

7.2.1 Using a debugging interface with e^2 studio

Instructions to program the executable binary are found in the latest RA FSP User Manual. See Section *Starting Development > e^2 studio ISDE User Guide > Tutorial: Your First RA MCU Project > Debug the Blinky Project.*

This is the preferred method for programming as it allows for additional debugging functionality available through the on-chip debugger.

7.2.2 Using J-Link tools

SEGGER J-Link Tools such as J-Flash, J-Flash Lite, and J-Link Commander can be used to program the executable binary into the target MCU. Refer to User Manuals UM08001 and UM08003 on www.segger.com.

7.2.3 Using Renesas Flash Programmer

The Renesas Flash Programmer provides usable and functional support for programming the on-chip flash memory of Renesas microcontrollers in each development and mass production phase. The software supports all RA MCUs, and the software user's manual is available online: https://www.renesas.com/us/en/software-tool/renesas-flash-programmer-programming-gui#documents.

7.3 User Interface

The user interface used to interact with the application is shown below. The Clock Changing and LPM Transition application uses the RTT interface in addition to the LED and push button switch, while the Operable Long Timer application uses the LED and user push button switch.

7.3.1 LED Indication

7.3.1.1 Clock Changing and LPM Transition

The Clock Changing and LPM Transition application uses LED1 to indicate the board initialization status, error condition, and normal mode operation. In the Sleep, Software Standby, and Snooze modes, LED1 will be turned off. In the Normal mode, this LED1 will blink every second. If any error condition occurs, the LED1 will be turned ON.

7.3.1.2 Operable Long Timer

The Operable Long Timer application uses LED1 to indicate the Normal mode. In the Normal mode, this LED1 will blink. In the Software Standby mode, LED1 will be turned off.

7.3.2 User Push Button Input

7.3.2.1 Clock Changing and LPM Transition

Push button switch S1 input is mainly used for transitioning to different MCU clocks and transitioning to different LPMs. For the Clock Changing and LPM Transition application, the same switch has dual functionality. If the switch is held under 1–2 seconds, it is considered a short press. If the switch is held for 3–6 seconds, it is considered a long press. A long press event is used to change the clock source dynamically, and a short press event is used to transition the LPM mode.

Note: A long press event during the LPM will not change the system clock source to a different clock but instead has a different role: to exit the LPM and go back to the Normal mode.

7.3.2.2 Operable Long Timer

In the application, pressing the push button switch S1 will set the Software Standby mode. The AGT1 underflow interrupt will cancel the LPM in 10 seconds.

7.3.3 RTT Console

The RTT console comes in handy to view the application messages while running/debugging the application. While you are using the RTT console, the debugger script for the LPM must be selected as shown in Figure 27.

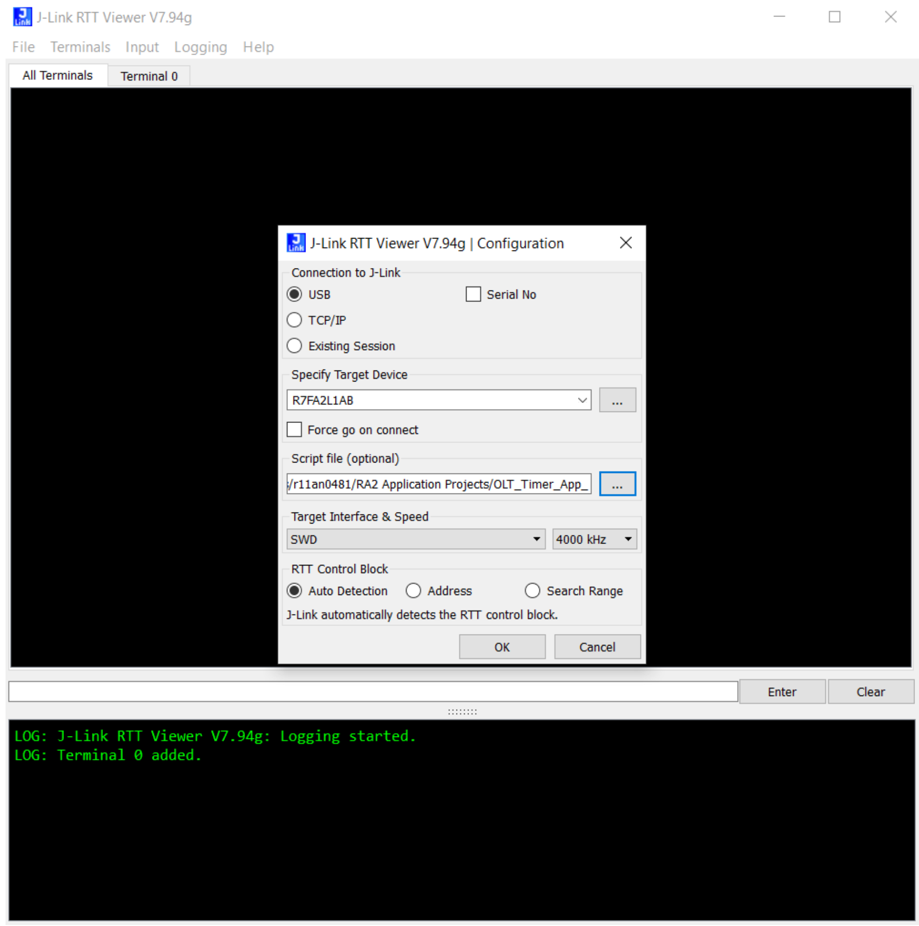


Figure 27. RTT Console for User Print Messages

7.4 Debugging Low Power Modes

By default, it is not possible to debug the low power modes of an RA device. If an application tries to enter Sleep mode, pending a peripheral interrupt to wake it, this will not happen as it will be woken almost immediately by a debug interrupt.

If the application tries to enter software mode, then the connection between the CPU and the IDE will be lost, closing the debug session within the IDE.

However, if the supplied debug script is specified, then it will be possible to debug the low power modes.

These scripts are used for the purpose of demonstrating the LPM. Note that even though this will allow you to develop your application, it will not allow you to measure accurate Icc figures, as you will be measuring the Icc of the on-chip debug circuit. Once you have created your low power application, accurate Icc figures can be measured with the OCD disabled.

The low power script also allows the RTT application to be used. While debugging the application, configure the debugger as shown in Figure 28. With these modifications, one can use the RTT without getting disconnected during the LPM.

Note: The script is attached as part of this project, and the debugger points to the same location.

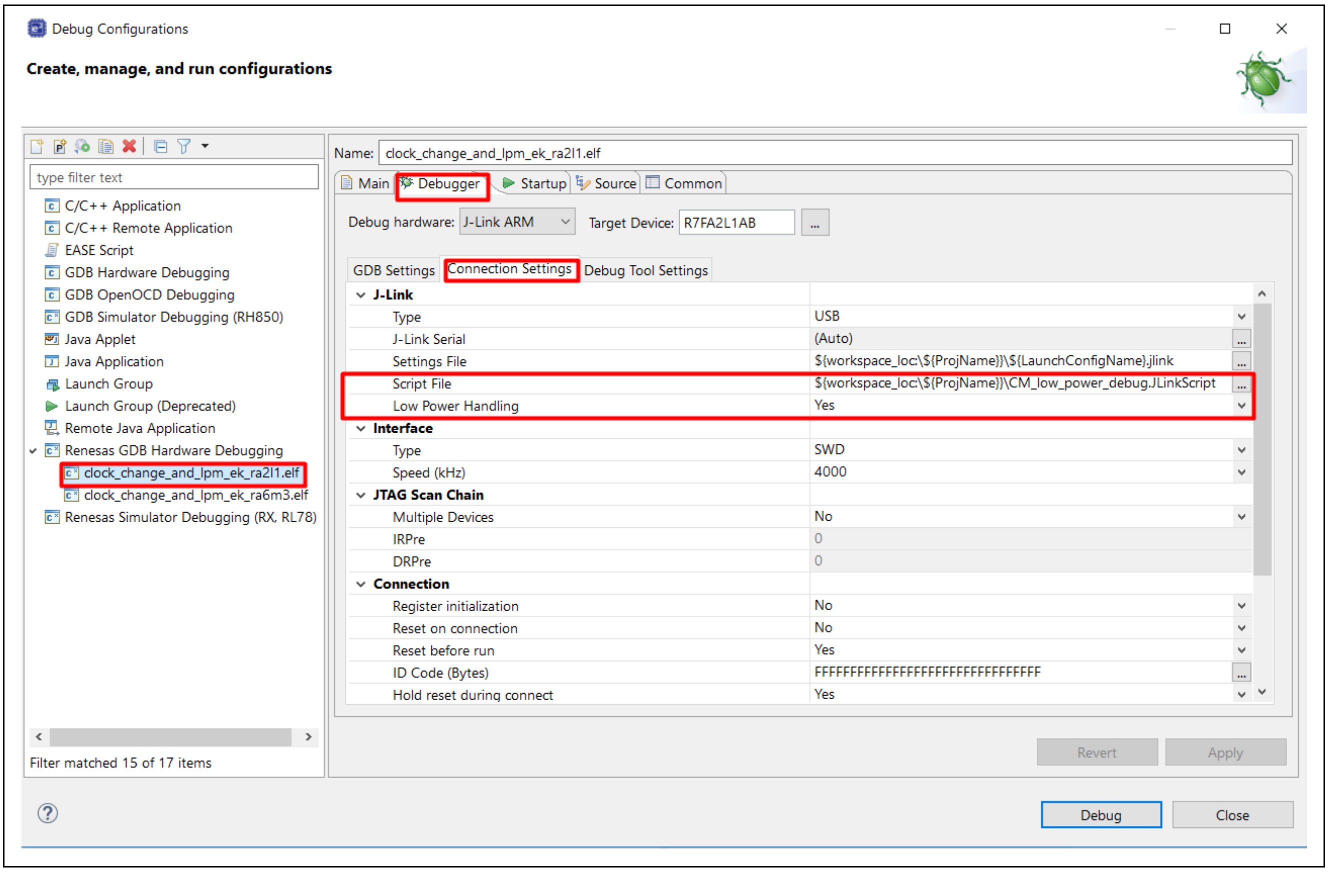


Figure 28. Debugger Settings for LPM Application Debugging

7.5 Steps to Run the Application

The following table shows the steps to run the Clock Changing and LPM Transition application. When the board is connected to RTT on power up, it will display the welcome message. To change the clock/LPM and run through the different use cases in this application, use the following tables as references. These tables have the list of events and current modes and refer to the new transition states and the expected outcomes.

7.5.1 Clock Changing:

The user button has two functions based on how long you press and hold it. In the clock changing mode, the user button pressed and held for 3-6 seconds selects the different system clocks for the MCU, whereas the user button pressed and held for 1-2 seconds exits the clock changing mode and enters into LPM for the configured clock. The details are as shown in Table 8.

Table 8. Clock Mode Transition Table

	Events	
System Clock	**User Button – Long Press**	**User Button – Short Press**
HOCO (High-speed on-chip oscillator)	Changes the System Clock to MOCO	Exits the Clock State Machine and enters the LPM State Machine for the configured clock HOCO.
MOCO (Medium speed on-chip oscillator)	Changes the System Clock to LOCO	Exits the Clock State Machine and enters the LPM State Machine for the configured clock MOCO.
LOCO (Low-speed on-chip oscillator)	Changes the System Clock to HOCO	Exits the Clock State Machine and enters the LPM State Machine for the configured clock LOCO.

7.5.2 LPM Transition

In the LPM application, the user switch has two functionalities based on how long you press, hold, and release it. If the MCU is in the LPM mode (Sleep) and the user button is pressed, held, and released, the MCU exits the LPM mode and enters the Normal mode, whether it is a short press or long press. The behavior is different when the MCU is in Normal mode.

When the MCU is in Normal mode if the user button is pressed, held for 1-2 seconds, and released, the MCU exits the Normal mode and enters (Sleep or Software Standby or Snooze) modes depending on the previous transition states. If the switch is pressed, held, and released for 3-6 seconds, MCU exits the LPM transition setting mode and enters the Clock setting mode. Also, the user switch has no effect during the LPM modes (Software Standby or Snooze).

Timer events cancel the Software Standby and Snooze modes and put the MCU back to Normal mode.

The details of the switch events and timer events are tabulated in Table 9.

Table 9. LPM Transition Table

	Events		
Low Power Modes	**User Button – Long Press**	**User Button – Short Press**	**Timer Event**
Normal	Exits the LPM transition State Machine and enters the Clock Mode State Machine	Enters the Sleep mode	Not Applicable
Sleep	Exits the Sleep mode and enters the Normal mode	Exits the Sleep mode and enters the Normal mode	Not Applicable
Normal (from Sleep)	Exits the LPM transition State Machine and enters the Clock Mode State Machine	Exits the Normal mode and enters the Software Standby mode	Not Applicable
Software Standby	Not Applicable	Not Applicable	Exits the Software Standby mode and Enters the Normal mode
Normal (from Software Standby)	Exits the LPM transition State Machine and enters the Clock Mode State Machine	Exits the Normal mode and enters the Snooze with Software Standby mode	Not Applicable
Snooze with Software Standby	Not Applicable	Not Applicable	Exits the Snooze with Software Standby mode and Enters the Normal mode

7.5.3 Operable Long Timer

In the Operable Long Timer application, the user button is used to enter Software Standby mode from Normal mode. AGT1 underflow interrupt is used to exit the LPM modes.

Table 10 shows the transition sequence and associated events used in the Operable Long Timer application.

Table 10. LPM Transition Table in Operable Long Timer Application

	Events	
Low Power Modes	**User Button – Press**	**Timer Event – AGT1_AGTI**
Normal	Enters Software Standby Mode	Not Applicable
Software Standby	Not Applicable	Exits the Software Standby mode and enters the Normal Mode

7.6 Measure MCU Current

The following steps are required to measure MCU current on EK-RA2L1, which is supported by the LPM applications:

- Power cycle the board after downloading the LPM projects.
- Measure the voltage drops across R3 and calculate the MCU's operating and standby current (ICC). Replace R3 with a bigger resistor if needed.

To measure the ICC current directly, connect a multimeter between the +3V3 and +3V3_MCU pins on the kit connectors after removing the R3 resistor.

Note: To measure MCU current in total, which includes ICC and analog power supply current (IAVCC0), follow the below steps on the EK-RA2L1 board

- Cut the trace-cut jumper E3.
- Connect J4-2 to J3-11.

8. Migrating LPM Applications to Different MCU/Kit

Even though the LPM applications are created for the Renesas EK-RA2L1 kit, they are designed to easily migrate to other Renesas RA Kits. For more details, refer to section 8, “Migrating LPM Applications to Different MCU/Kit” in the Getting Started with LPM Application Note for the EK-RA6M3 Board included as part of the LPM bundle.

9. References

- Renesas FSP User’s Manual: https://renesas.github.io/fsp
- Renesas RA MCU Datasheets: See http://renesas.com/ra and select the relevant MCU
- LPM Example Projects on Renesas RA GitHub: https://github.com/renesas/ra-fsp-examples

Website and Support

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

RA Product Information	www.renesas.com/ra
RA Product Support Forum	www.renesas.com/ra/forum
RA Flexible Software Package	www.renesas.com/FSP
Renesas Support	www.renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec.3.20	-	Initial version supporting EK-RA2L1
1.01	Jan.25.21	-	Updated for EK-RA2E1
1.02	Sep.24.21	-	Minor changes for readability
1.03	Jun.30.23	-	Document updated as per FSP v4.4.0
1.04	Jul.21.24	-	Document updated as per FSP v5.2.0

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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/.