

## Renesas Synergy™ Platform

# DK-S124 Out-of-Box Programming Guide

---

### Introduction

The Out-of-Box (OoB) program shows a multithreaded application design that reads the analog values present on the potentiometer, light sensor, and temperature sensor, and displays those values in the included PMOD LCD screen. The three LEDs on the board either flash or chase to the value read from either the light sensor or the potentiometer. Switch 1 and switch 2 are used to select which sensor is used, and if the LEDs flash or chase.

This application note describes the OoB program of the DK-S124 MCU Group in more detail than the *Quick Start Guide* included in the development kit.

### Prerequisites

This application note assumes that you have some experience with the IAR Embedded Workbench® for Renesas Synergy™ and Synergy Software Package (SSP). Before you perform the procedure in this application note, follow the procedure in the *Quick Start Guide* for your board to build and run the Blinky project. Doing so enables you to become familiar with the Renesas Synergy™ e<sup>2</sup> studio and the SSP, and also ensures that the debug connection to your board functions properly.

### Required Resources

The example application targets Renesas Synergy S124 devices. To build and run the application, you need:

- Renesas Synergy™ DK-S124 MCU Group (v3.0) with included PMOD LCD
- PC running Microsoft® Windows® 7 or above with the following Renesas Synergy™ software installed:
  - e<sup>2</sup> studio ISDE v7.3.0 or later
  - IAR EW for Synergy v8.23.3 or later
  - Synergy Standalone Configurator v7.3.0 or later
  - Synergy Software Package (SSP) v1.6.0 or later.

You can download the required software from the Renesas Synergy™ Gallery at: [www.renesas.com/synergy/software](http://www.renesas.com/synergy/software).

Refer to the included *Renesas Synergy Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf) for instructions on importing the project into e<sup>2</sup> studio, and then building and running the project.

### Time Required and Steps

You should be able to install, build, and run the example application in under 30 minutes following these basic steps:

1. Configure the DK-S124 MCU Group board for the OoB application
2. Import, build, and debug the project
3. Observe the analog sensor values change on the LCD screen
4. Change the mode of operation using the two momentary pushbutton switches
5. Vary the frequency of the LED flashing using either the potentiometer or the light sensor

**Contents**

1. Overview .....	3
2. Synergy Platform Capabilities .....	3
3. OoB Application Design .....	5
3.1 Inputs Thread Logic .....	7
3.2 Accelerometer Initialization .....	8
3.3 LCD Thread Logic .....	8
3.4 LED Thread Logic .....	9
3.5 Threads and Modules .....	9
3.6 PMOD LCD Library .....	10
3.7 Source Code Layout .....	11
3.8 Design Example .....	12
3.9 Configure the DK-S124 Synergy MCU for the Out-of-box Program .....	12
3.10 e <sup>2</sup> studio Tricks .....	13
3.11 Importing and Building the Project .....	16
4. Next Steps .....	16
5. References .....	16

## 1. Overview

This application note describes how the capabilities of the DK-S124 Synergy MCU Group and the challenges faced when trying to design a robust application using a low-cost SPI LCD screen.

The DK-S124 Synergy MCU Group is designed to familiarize developers with the lowest cost processors available in the Renesas Synergy™ Platform line of products. The S1 series processor is a 32-MHz Arm® Cortex-M0+ CPU with 128 KB of code flash.

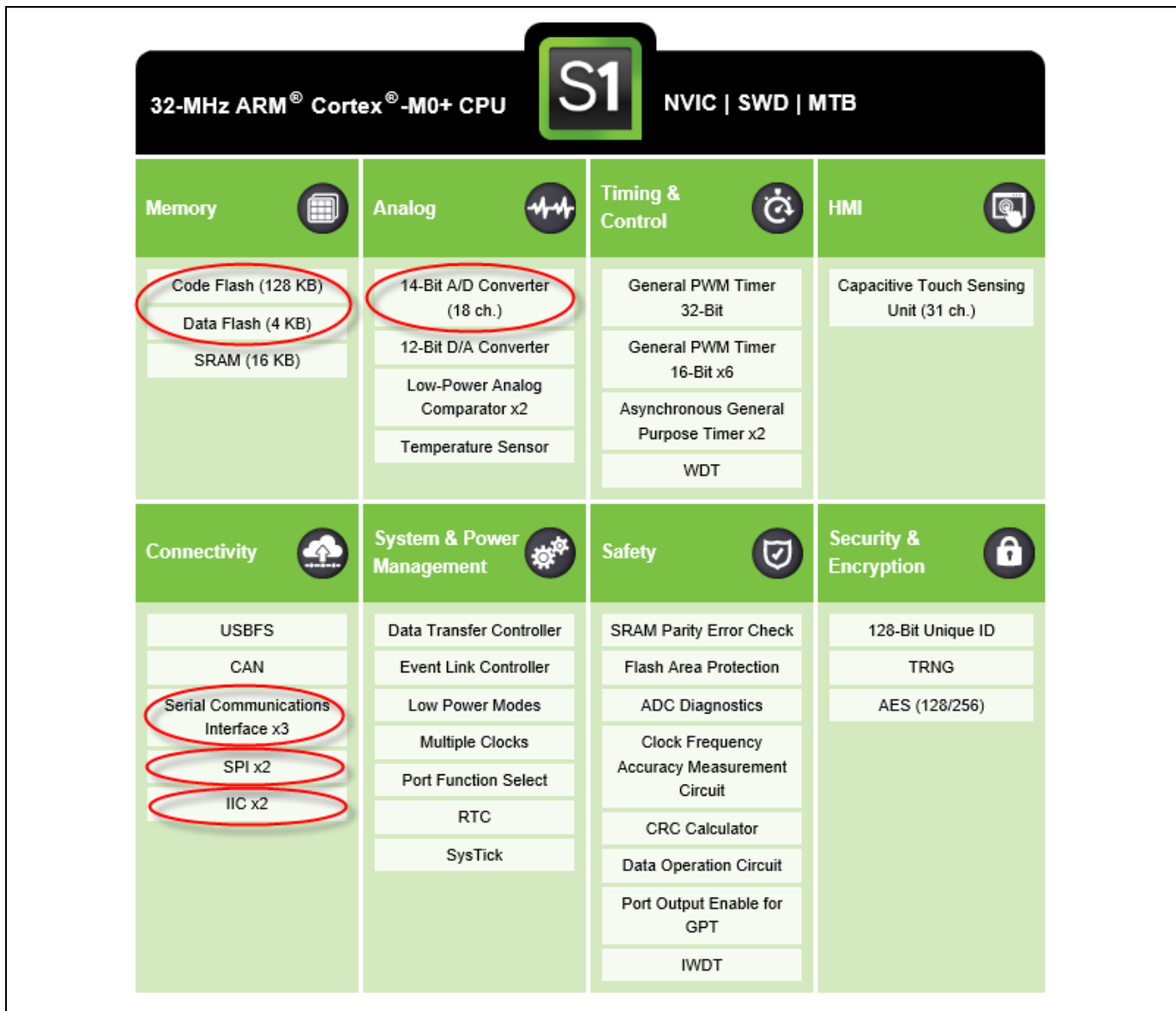
The Renesas Synergy™ Platform also provides a high-level tool, known as GUIX Studio, for rapidly developing graphical user interfaces in a point-and-click environment on these types of higher-end processors. A variety of application notes detail the development of complex graphical interfaces on these higher end boards, and these processors are also available on the Renesas Synergy™ Gallery website:

<https://synergygallery.renesas.com/about/gallery>.

The DK-S124 Synergy MCU is ideally suited for applications requiring a simple human machine interface (HMI) to convey critical information to the user. The development kit ships with a 128x128-pixel PMOD LCD. The DK-S124 Synergy MCU has a single PMOD interface to accommodate this LCD. The DK-S124 Synergy MCU communicates with this LCD over a SPI interface. A principle design consideration for this application note is to provide a small graphical library in order to display characters on the PMOD LCD in a timely fashion.

## 2. Synergy Platform Capabilities

The library driving the PMOD LCD, the Renesas Synergy™ Platform, contains all the remaining resources necessary for quickly prototyping the type of application demonstrated in the DK-S124 Synergy MCU Group OoB program. Figure 1 highlights the DK-S124 MCU Group peripherals used by the OoB application.



**Figure 1. S1 peripherals used by the OoB**

Table 1 defines the processor pin connections used to connect these peripherals to the various external components on the DK-S124 Synergy MCU Group board. For additional pin references, see the DK-S124 Synergy MCU schematics.

**Table 1. Port summary of peripheral connections used by the OoB**

Schematic part	Description	S1 port connection	Pin function
S1	Momentary switch	Port 2 pin 6	I/O port pin
S2	Momentary switch	Port 0 pin 4	I/O port pin
POT1	10K potentiometer	Port 0 pin 12	Analog channel 7
U11	APDS-9005 light sensor	Port 0 pin 0	Analog channel 0
U14	TMP35 temperature sensor	Port 0 pin 1	Analog channel 1
U13	BMA250 3-axis accelerometer	Port 2 pin 6	IRQ
U13	BMA250 3-axis accelerometer	Port 4 pin 1	SDA0 (I2C clock)
U13	BMA250 3-axis accelerometer	Port 4 pin 0	SCL0 (I2C data)
U5	ISL41387 RS-232/RS485 transceiver	Port 4 pin 10	RXD0 (receive data)
U5	ISL41387 RS-232/RS485 transceiver	Port 4 pin 11	TXD0 (transmit data)

### 3. OoB Application Design

The SK-S124 Synergy MCU board contains several analog sensors along with a 3-axis accelerometer connected to the I<sup>2</sup>C bus. The analog sensors include a potentiometer, a temperature sensor, and a light sensor. The OoB application continuously reads the analog values provided by these sensors and displays them on the PMOD LCD screen. It also flashes three on-board LEDs in proportion to the analog voltage produced by the potentiometer or the light sensor. The LEDs may be programmed to flash in unison or to chase each other. The selection of which sensor drives the flashing rate, and whether the LEDs flash or chase, is controlled by the pushbutton switches **S1** and **S2**, found in the lower right corner of the DK-S124 Synergy MCU board.

The application illustrates the use of Synergy peripheral drivers to read the sensor values, while demonstrating how easy it is to setup a multi-threaded application. The application has separate threads to read the sensors, display the values to the LCD and flash the LEDs based on the pushbutton and sensor inputs.

Table 2 summarizes the three user-defined threads and their functions.

**Table 2. Application thread names and functions**

Thread name	Thread function
LED thread	Controls flashing/chasing of the 3 on-board LEDs
LCD thread	Updates the LCD as new data arrives from the three analog sensors
Inputs thread	Periodically polls the switch inputs and reads the analog sensor voltages

Figure 2 conceptually shows the OoB application. The input thread periodically samples the state of the pushbuttons and the voltage values present on the analog sensors. The input thread makes this data available to the other threads through two access functions, eliminating the need for global variables.

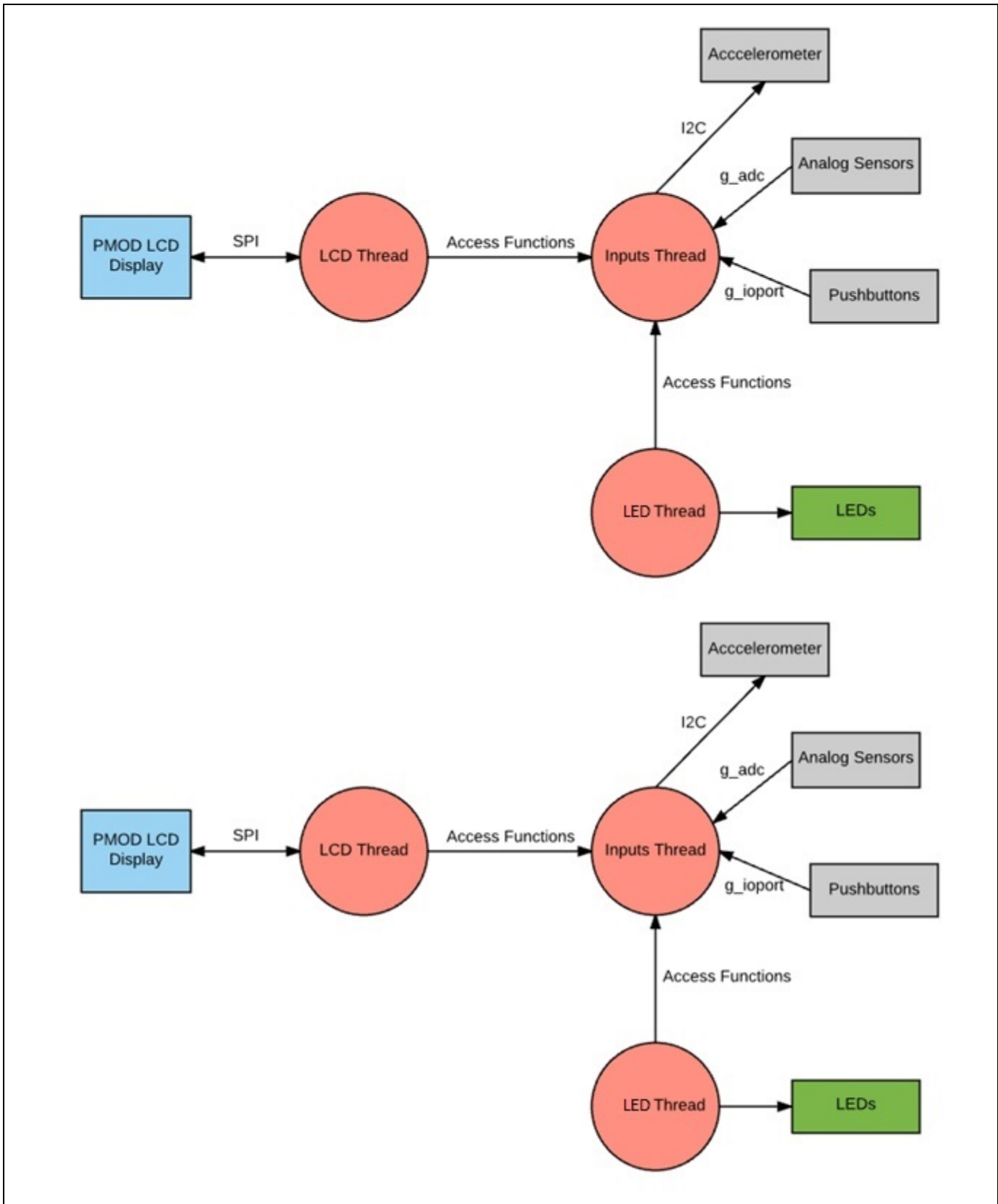


Figure 2. Conceptual diagram of the OoB application

While both pushbuttons are connected to hardware pins with interrupt capability, this application uses a thread to periodically poll the switch states by reading the I/O port pins connected to the switches. This simple technique provides basic switch debouncing by reading the state of the switch at an interval longer than the typical mechanical switch bounce time (50 ms). Other more complex debouncing techniques may be used, such as the excellent switch debouncing algorithms published by Jack Ganssle (<http://www.ganssle.com>).

Other application notes, such as the *Simple Audio Example*, highlight the use of the Synergy IRQ Framework to detect pushbutton presses. There are varying schools of thought about the validity of connecting mechanical switches to interrupts. The Synergy system makes it easy to do it either way.

### 3.1 Inputs Thread Logic

Figure 3 shows the logic implemented in the inputs thread. To be responsive, the inputs thread samples the sensors and switches inputs every 50 ms. One of the advantages of threads is shared memory space. The LCD thread periodically updates the PMOD LCD screen with the current mode, selected by pushbuttons, and the latest sensor values. Subsequent versions of this application note will demonstrate the use of the Synergy Messaging Framework to send messages to the LCD thread as these values change. This allows the LCD thread to be event driven.

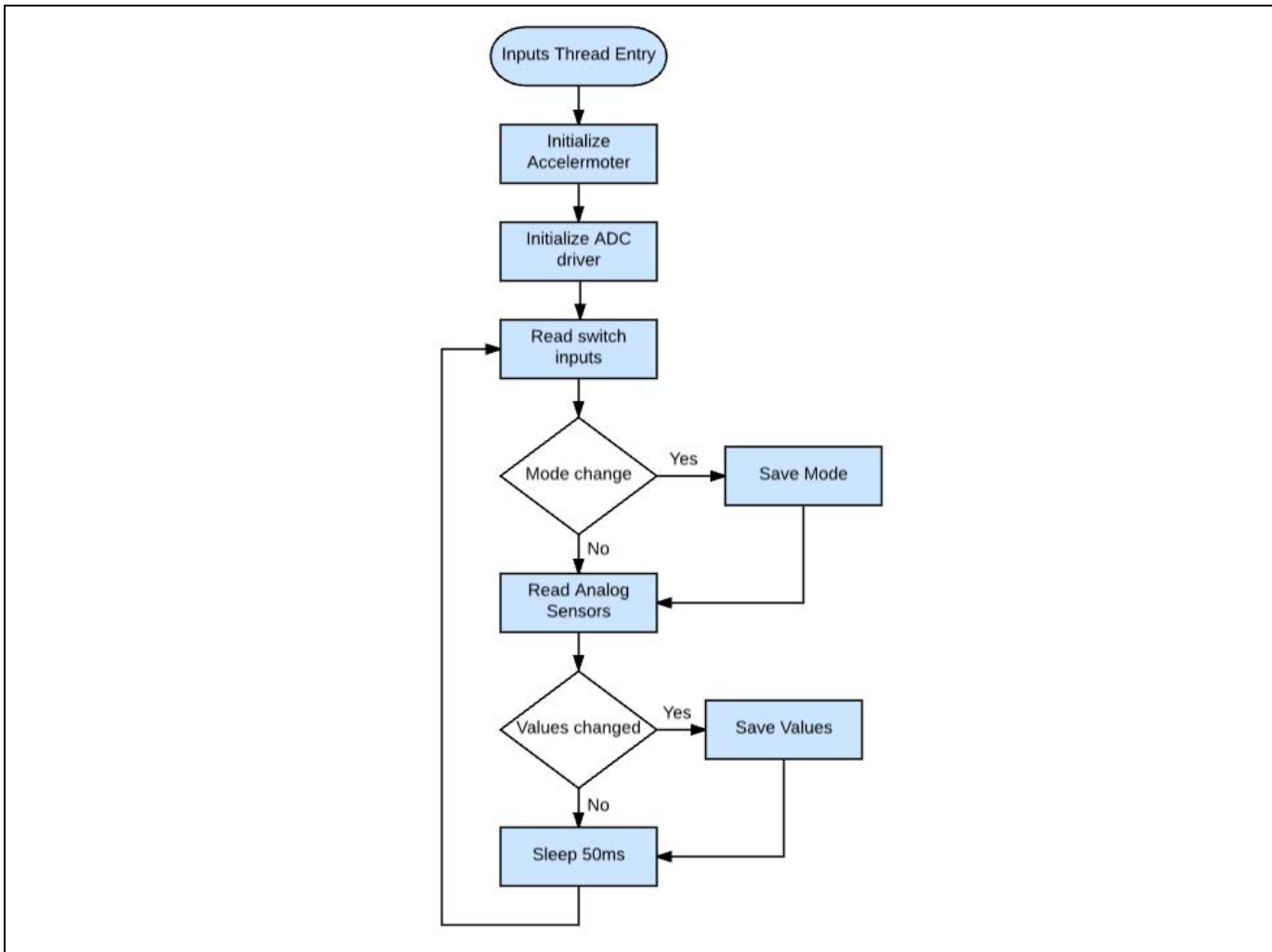


Figure 3. Simplified flowchart of the inputs thread

### 3.2 Accelerometer Initialization

On the SK-S124 Synergy MCU v3.0 board, the accelerometer IRQ line is shared with **S1**, since the accelerometer IRQ pin defaults to a push/pull configuration and works as an active-high IRQ state. This means that it is always pulling down on the line connected to S1, which prevents the use of this line in reading the state of **S1**. To use **S1** to select between the potentiometer and light sensor, it is necessary to change the interrupt configuration of the accelerometer. To implement this change, the I<sup>2</sup>C framework is added to the inputs thread and several I<sup>2</sup>C commands are issued to the accelerometer.

On the DK-S124 Synergy MCU v3.0, the jumper J20 is used to disconnect the accelerometer IRQ pin from S1, and so the accelerometer initialization described above can be skipped.

### 3.3 LCD Thread Logic

The LCD thread periodically wakes up, reads the current mode and sensor values, and displays these values to the PMOD LCD screen. Figure 4 shows the logic implemented in the LCD thread. The mode and sensor values are obtained by calling the two access functions provided by the inputs thread.

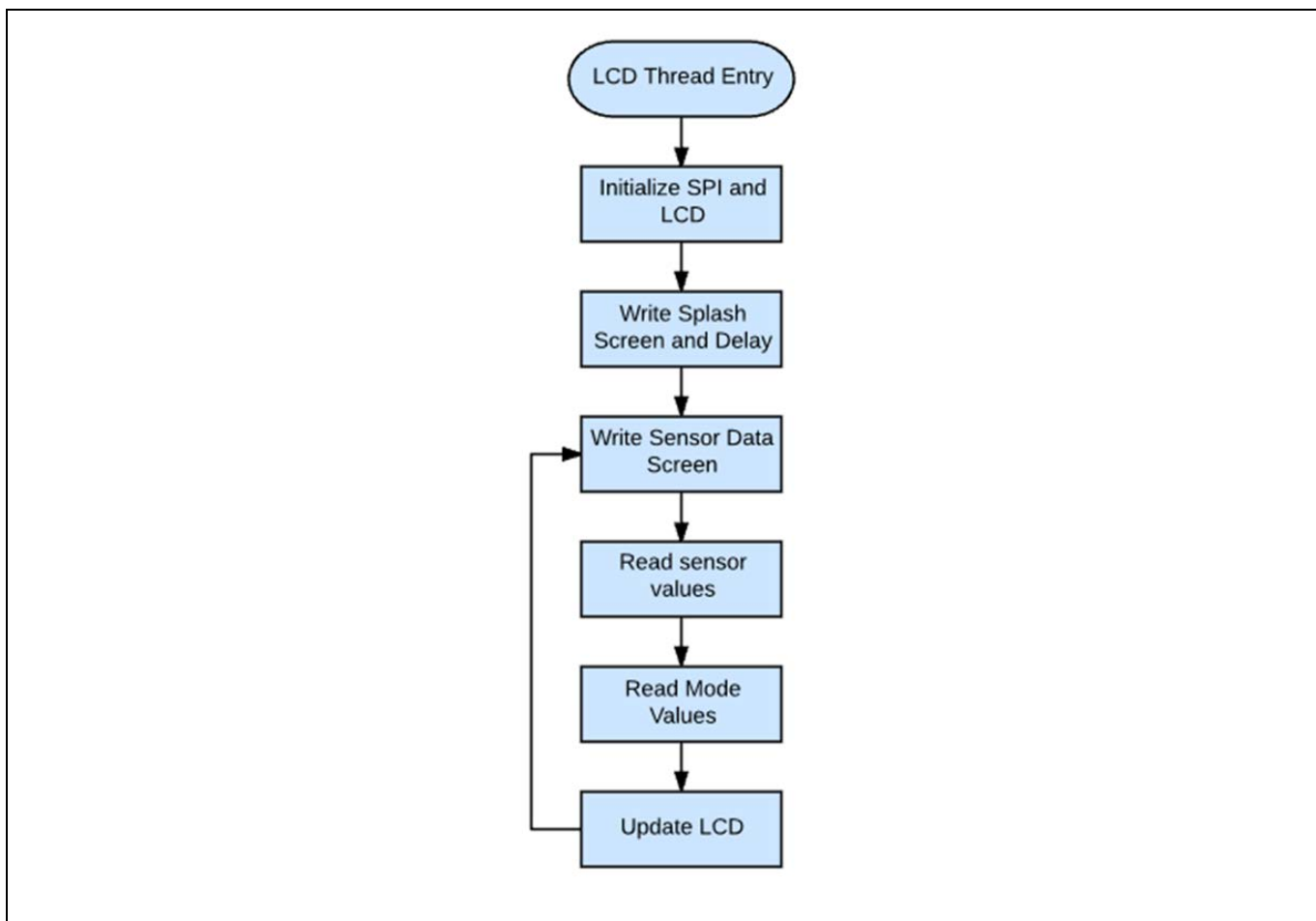


Figure 4. Simplified flowchart of the LCD thread



### 3.4 LED Thread Logic

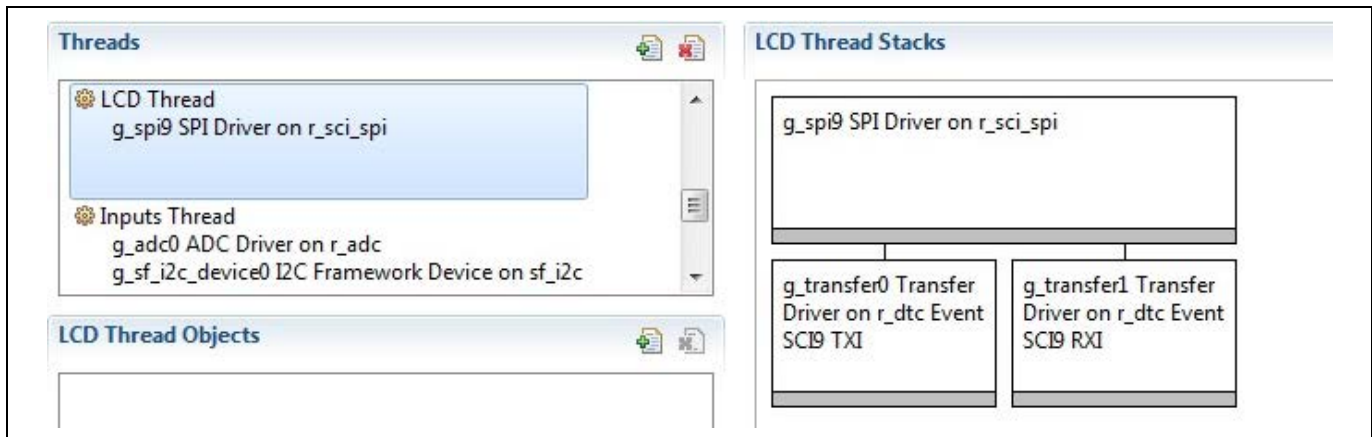
The LED thread must refresh the LEDs at a periodic rate driven by the voltage present on either the light sensor or the potentiometer, depending on the current mode selected by the operator. The standard `tx_thread_sleep()` call is used as the periodic delay.

When the LED thread wakes up, it calls two different access functions, `inputs_get_sensor_values()` and `inputs_get_mode()`, to obtain the current sensor values. This illustrates one of the benefits of threads, which is shared memory space. This allows the inputs thread to maintain a local copy of these variables while still providing access of those variables to the LED thread. The access functions eliminate the use of global variables and allow the sensor variables to have local scope inside the inputs thread.

### 3.5 Threads and Modules

As is the case with all Renesas Synergy™ applications, module drivers are added to threads and their properties are configured using the ISDE configurator. Open the configurator by double clicking on the **configuration.xml** file from inside the project explorer in e<sup>2</sup> studio. This application note does not cover all the driver modules added to the application. As an example, the addition of the SPI driver is discussed; it is necessary for communicating with the PMOD LCD. All other drivers may be inspected through the ISDE in a similar manner. For details on Synergy modules and drivers, refer to the *SSP User's Manual*.

As previously mentioned, the S124 processor communicates with the PMOD LCD through the SPI bus. Therefore, it is necessary to add an SPI driver to the thread that communicates with the display, which in this case, is the LCD thread. Figure 5 shows the ISDE configurator with the LCD thread highlighted. You can see added the `r_sci_spi` driver.



**Figure 5. Threads tab display for the OoB program**

Synergy processors contain two different peripherals for SPI communication — a dedicated SPI peripheral, and the Serial Communication Interface (SCI), that may act either as a simple I<sup>2</sup>C, simple SPI, or UART (synchronous or asynchronous). The SSP supports both peripherals. The same scenario is true for the I<sup>2</sup>C peripheral.

Choosing the peripheral to use is often based on the complexities of your system. You may not need all the power available on the dedicated SPI or I<sup>2</sup>C peripherals. You may need to use the pins assigned to the dedicated I<sup>2</sup>C peripheral for another use. The decision regarding which peripheral to use must typically be made during the hardware design phase, since it affects which physical processor pins are connected to the peripheral.

One consideration when designing the system are the numbers of each type of peripheral. While the S124 processor only has two dedicated SPI and I<sup>2</sup>C peripherals, it has nine SCI peripherals which can be configured as either simple SPI or I<sup>2</sup>C.

The OoB application provides an example for each scenario. The SPI peripheral used to communicate with the PMOD LCD is connected to one of the nine SCI channels, while the accelerometer is connected to one of the two dedicated I<sup>2</sup>C buses.

### 3.6 PMOD LCD Library

The PMOD LCD is a 128x128-pixel LCD that uses an industry standard of the ST7735 graphics controller. Several open source libraries are available from various vendors to communicate with similar screens using the same graphics controller. One of the limitations of using an SPI-based LCD is the slow performance due to addressing each pixel with a separate SPI write. Most of the systems have some latency between SPI writes, and some latency when writing to the other control lines necessary for talking to the display. When writing all the bytes necessary to fill an entire screen, or even writing to output a line of text, these latencies add up to a noticeable delay.

The PMOD LCD library included with the DK-S124 OoB Synergy MCU has been optimized to improve performance by buffering the writes and bursting them out one-at-a-time, eliminating much of the latency that would result from individual SPI write operations. One of the tradeoffs when doing this sort of buffering is that you must allocate buffer space to do it. Currently, the PMOD LCD library allocates a two-dimensional array consuming only 128 bytes for character buffering.

```
static uint16_t      lcd_pmod_buffer[2][64];
```

Figure 6 shows the API calls for the PMOD LCD library at the time this application note was written. The only API calls used in this application are `R_SPILCD_Clear` and `R_SPILCD_DrawText`. The PMOD LCD library undergoes continuous development, so check the Renesas website for updates.

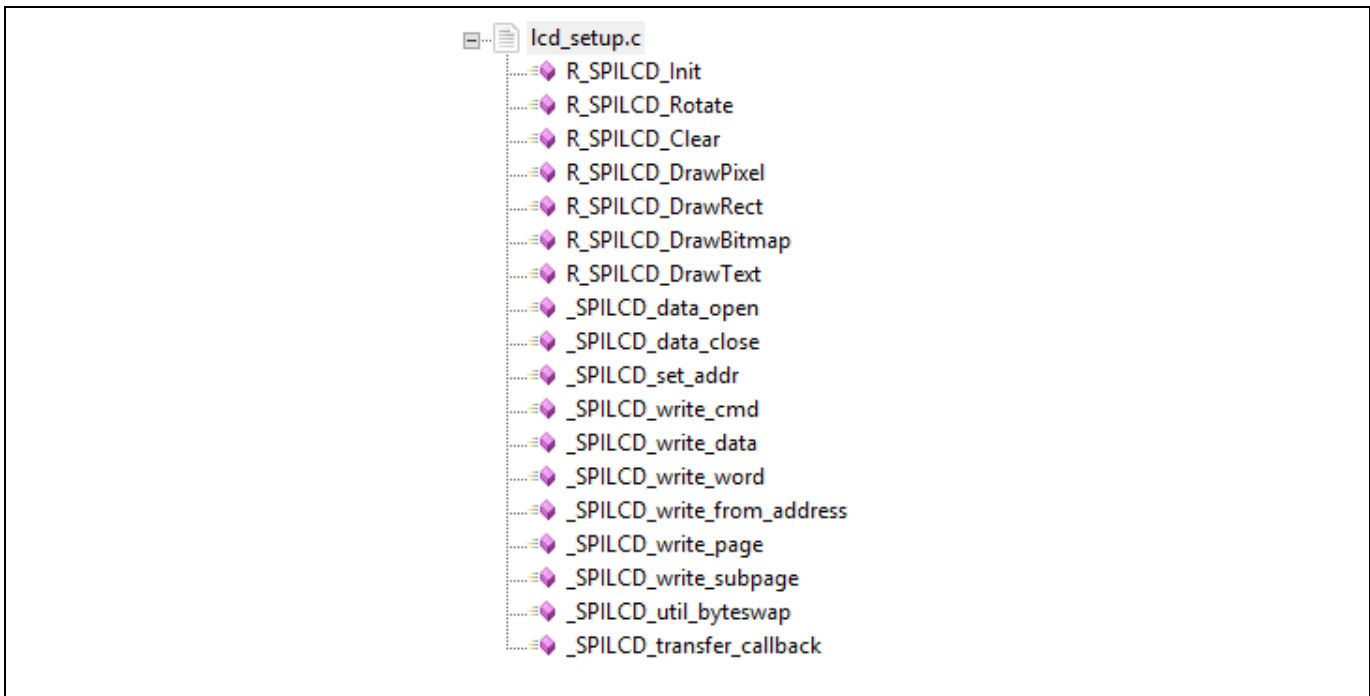


Figure 6. LCD library API calls

Using the PMOD LCD library is easy. Use the following code to display the simple flash screen displayed on powering up.

```

/** Initialize the PMOD LCD */
R_SPILCD_Init((spi_instance_t * const) &g_spi9, LCD_ROTATION_90);
R_SPILCD_Clear(LCD_COL_BLACK);

R_SPILCD_DrawText("DK-S124", -1, TEXT_COLOR_FG, TEXT_COLOR_BG, CURSOR_X(7), CURSOR_Y(5));
R_SPILCD_DrawText("V 1.0", -1, TEXT_COLOR_FG, TEXT_COLOR_BG, CURSOR_X(7), CURSOR_Y(8));

```

In most applications, you would use the Synergy SPI driver `open` call before accessing the SPI driver. This call would normally be done just inside the thread entry call, prior to entering the thread loop. However, in this case, the PMOD LCD library does this for you when you call the `R_SPILCD_Init()` routine to initialize the LCD. As you can see in the code snippet above, you simply pass a pointer to the SPI driver instance to the `R_SPILCD_Init()` call. This instance pointer is automatically created for you when you add the SPI driver to the LCD thread.

### 3.7 Source Code Layout

This section gives a brief overview of the source code layout you expect to see when you build the project with e<sup>2</sup> studio.

If you are already experienced with writing code for Renesas Synergy applications, you are probably familiar with how e<sup>2</sup> studio organizes a standard project. In this case, you can skip to the next section.

After you import the DK-S124 Synergy MCU project into e<sup>2</sup> studio and build the project, you should see a directory structure like the one shown in Figure 7. The first time you import the project, you may notice that the `synergy_gen`, `synergy`, `debug`, `synergy`, and `synergy_cfg` folders do not exist. These folders are built automatically by the framework when you compile the project. To reduce the size, these subfolders are normally not included when programs are distributed. The only files that you typically need to work within the project are files highlighted in green.

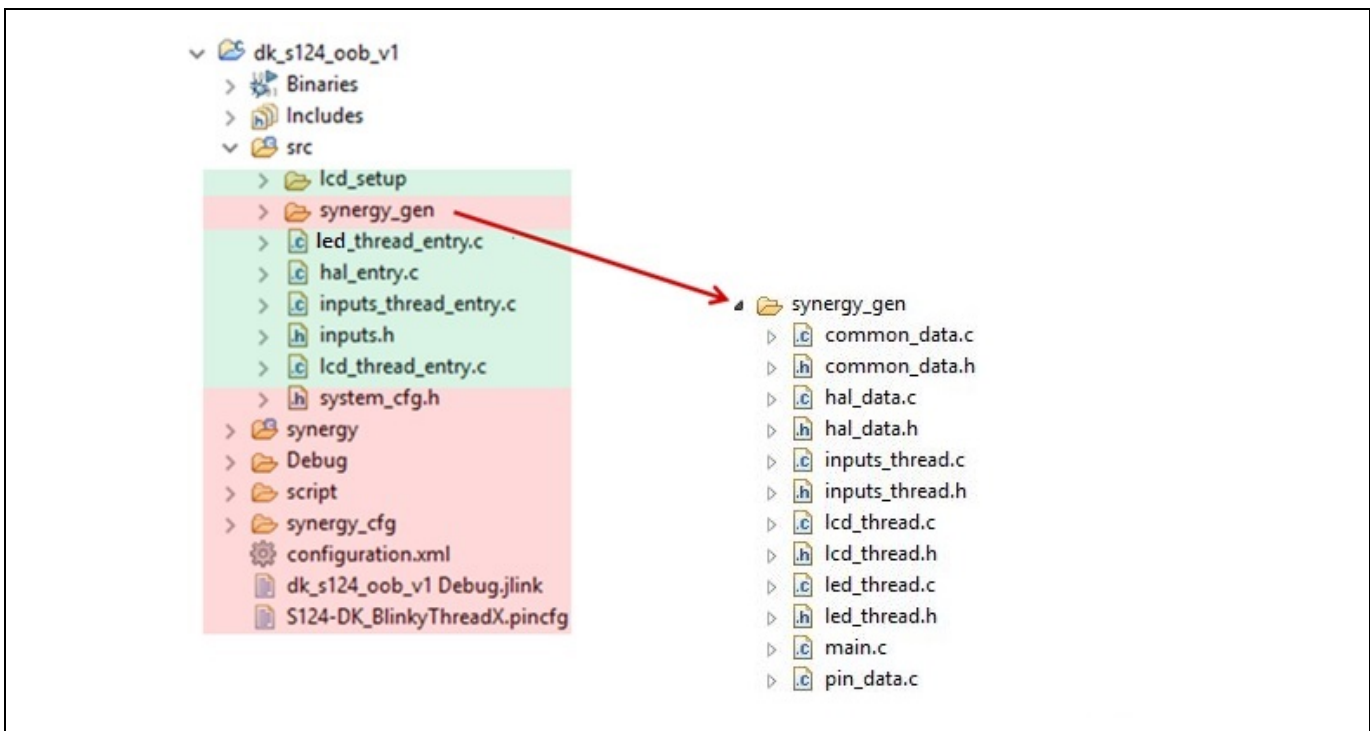


Figure 7. Source code layout for DK-S124 OoB Synergy MCU v1.0 program

### 3.8 Design Example

The DK-S124 Synergy MCU comes preloaded with the out-of-box application software. Prior to running the program or programming the board from e<sup>2</sup> studio ISDE, you must first configure the board.

### 3.9 Configure the DK-S124 Synergy MCU for the Out-of-box Program

The steps to configure the DK-S124 Synergy MCU are:

1. Verify that the J3 header near the battery is configured with two jumpers to make connections 1-3 and 2-4, as shown in Figure 8.
2. Connect the J-Link-OB on DK-S124 connector J18 to the PC using a micro USB cable (Figure 8).

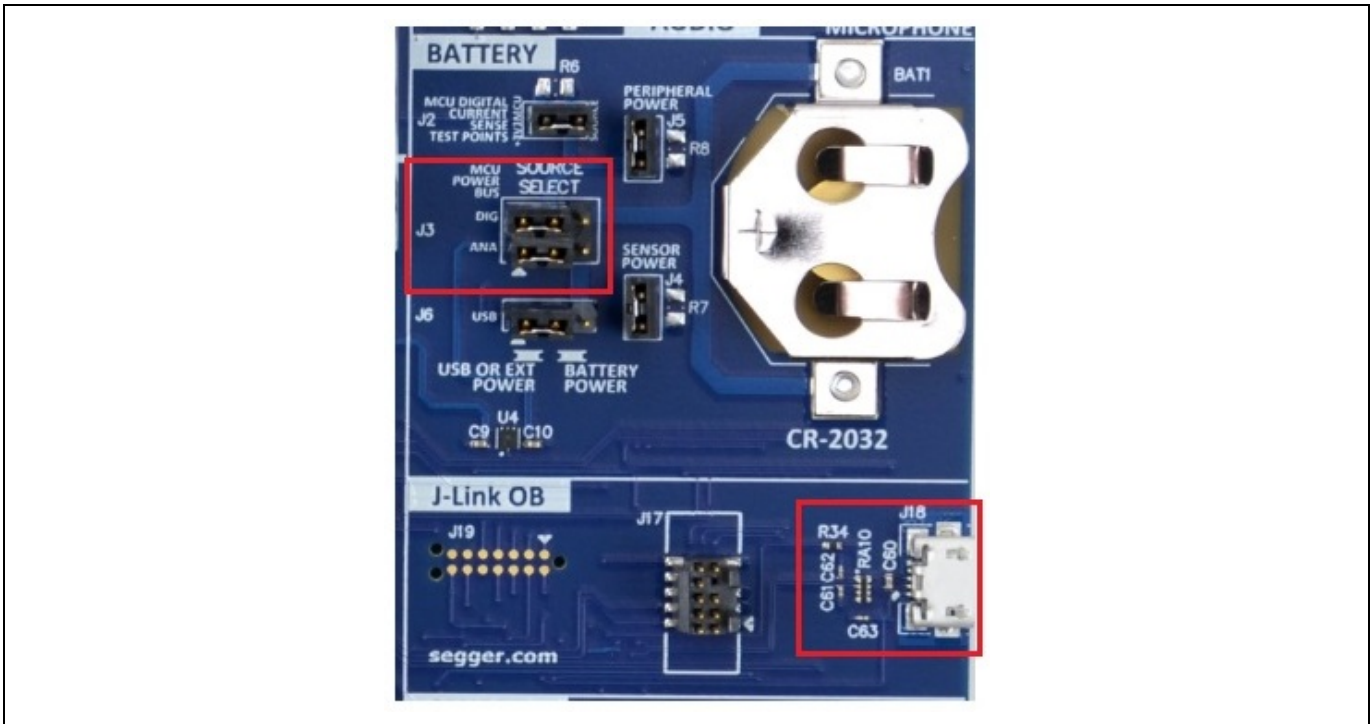


Figure 8. Power setup (top) and SEGGER J-Link® OB connection (bottom)

Once the DK-S124 is plugged in, it powers up and immediately starts flashing the three LEDs: LED1, LED2, and LED3. It displays a simple splash screen on the PMOD LCD for two seconds. This splash screen shows the board number and the version of the out-of-box software that shipped with the board (Figure 9).



Figure 9. OoB splash screen

After 2 seconds, the screen changes to the analog measurement screen (Figure 10). The SSP Out-Of-Box application uses the A/D converter to read the voltage values present on the potentiometer (POT1), the light sensor U11 (APDS-9005), and the temperature sensor U14 (TMP 35). The program displays the raw hex values read from the light sensor and the potentiometer, and it converts the analog value read from the temperature sensor to the equivalent Fahrenheit temperature.

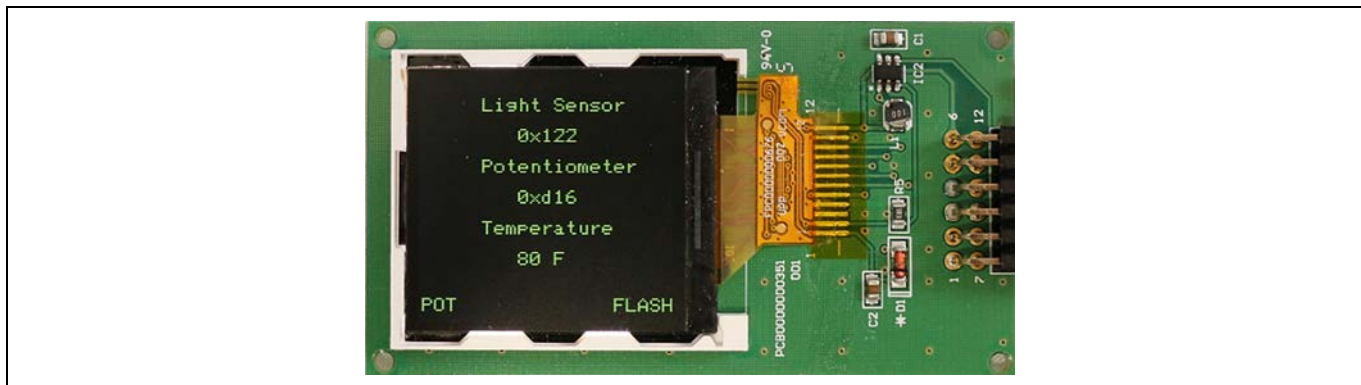


Figure 10. OoB analog sensor screen

You can interact with the Out-of-Box program in the following ways:

1. The three LEDs (LED1, LED2, and LED3), all flash at the same time, or chase each other. Pushing the momentary pushbutton switch **S2** causes the LEDs to alternate between flashing and chasing. The bottom right corner of the screen indicates which mode the LEDs are operating in (Flash/ Chase).
2. The flashing rate of the LEDs is determined by the value read from the light sensor or the potentiometer. You toggle between these two by pushing the momentary pushbutton switch **S1**. The bottom left corner of the screen indicates which sensor is driving the LED flashing rate (POT/Light).
3. With the POT mode selected (push **S1**), rotate POT1 clockwise and counterclockwise. LED flashes increase or decrease appropriately. The screen displays the raw value reported by the A/D converter.
4. With the Light mode selected (push **S1**), move a light source (such as a flashlight) closer and farther away from the light sensor U11. Observe that the flashing rate of the LEDs changes. The raw hex value displayed to the screen also changes.

### 3.10 e<sup>2</sup> studio Tricks

When adding threads to an application, they default to 1 KB for stack space. Reducing unused stack space makes more memory available for the application. It is not easy to determine how much stack space is enough.

ThreadX® initializes all thread stacks with a known value, 0xEF, which makes inspecting each thread stack easier using the memory monitor included with e<sup>2</sup> studio. First, you must determine where the stacks are located in the physical memory. Locating stacks can be done one of two ways. The first is by viewing the map file that is produced when you compile the application, as the following figure shows.

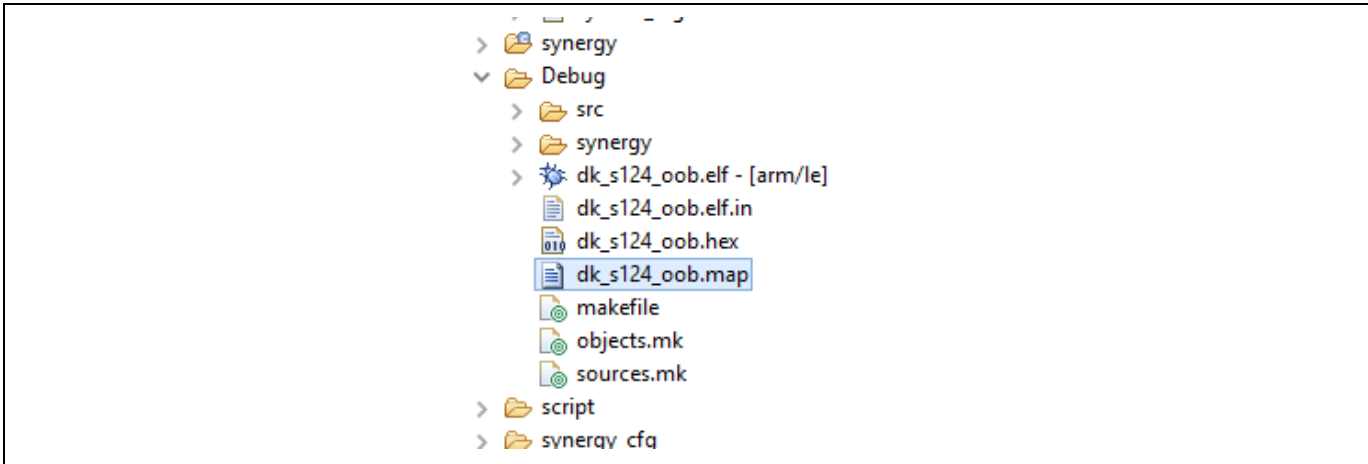


Figure 11. Location of .map file

Depending on the system configuration, e<sup>2</sup> studio may default to opening the file in some other editor. For example, systems with Visual Studio installed may open this file in Visual Studio when you double click on the file. If this happens, you can always right click on the file and select **Open With>Text Editor** to view the file inside the e<sup>2</sup> studio.

Once you have the file open, the fastest way to find the information you are looking for is to search for `stack_dummy`, which brings you to the section in the map file where the thread stack addresses are defined.

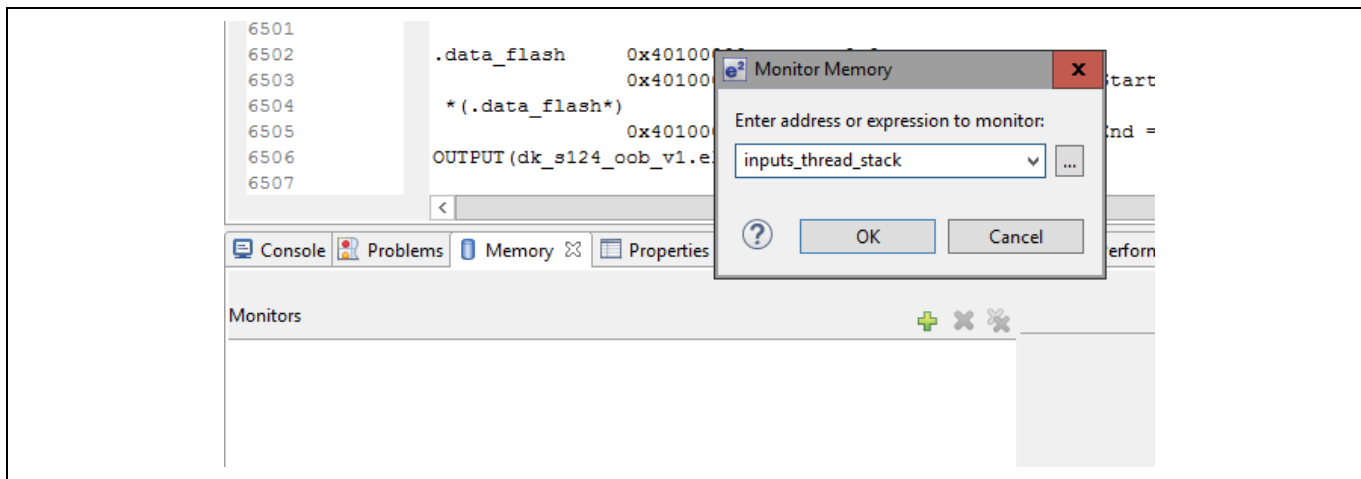
Figure 12 shows an example where the address of the inputs thread stack is 0x20002160, with a size of 0x400 (1024 bytes). To examine the stack memory, you can open the memory monitor. Locate the memory tab along the bottom of e<sup>2</sup> studio. If the tab is not currently visible, open it by going to **Window>Show View>Memory**, using the top tool bar. Click the small green + symbol to add a new memory monitor and type this address into the dialog box.



Figure 12. Locating thread stack addresses using the .map file

While this process works to locate the address of any variable, you must be careful when using the addresses in the memory monitor. As you make the changes during development, the program addresses may change, so it is always better to use symbols if possible.

A shortcut to this process is applying appending `_stack` to the current symbol name you gave to your thread. In this case, the thread name is `inputs_thread`, and so appending `_stack` to this name gives the name `inputs_thread_stack`. You can enter this symbol name into the Memory Monitor dialog box shown in Figure 13. The advantage to using the symbol is that it stays the same as you add/subtract code from your project, unless you explicitly change the thread name.



**Figure 13. Setting up a memory monitor**

Figure 13 shows that there is an amount of unused memory in the accelerometer thread stack. Each row represents 16 bytes. Take the number of rows displaying the EFEFEFEF data and multiply that number by 16 to see how many unused bytes exist in the stack. You can reduce the stack size by half that amount without any problem. You can always monitor the stack usage after the program has run for a while to monitor the free stack space.

Saving a few hundred bytes of stack space might not seem like a lot, but when multiplied by the number of threads in your program, the difference might allow one more feature to be added the application.

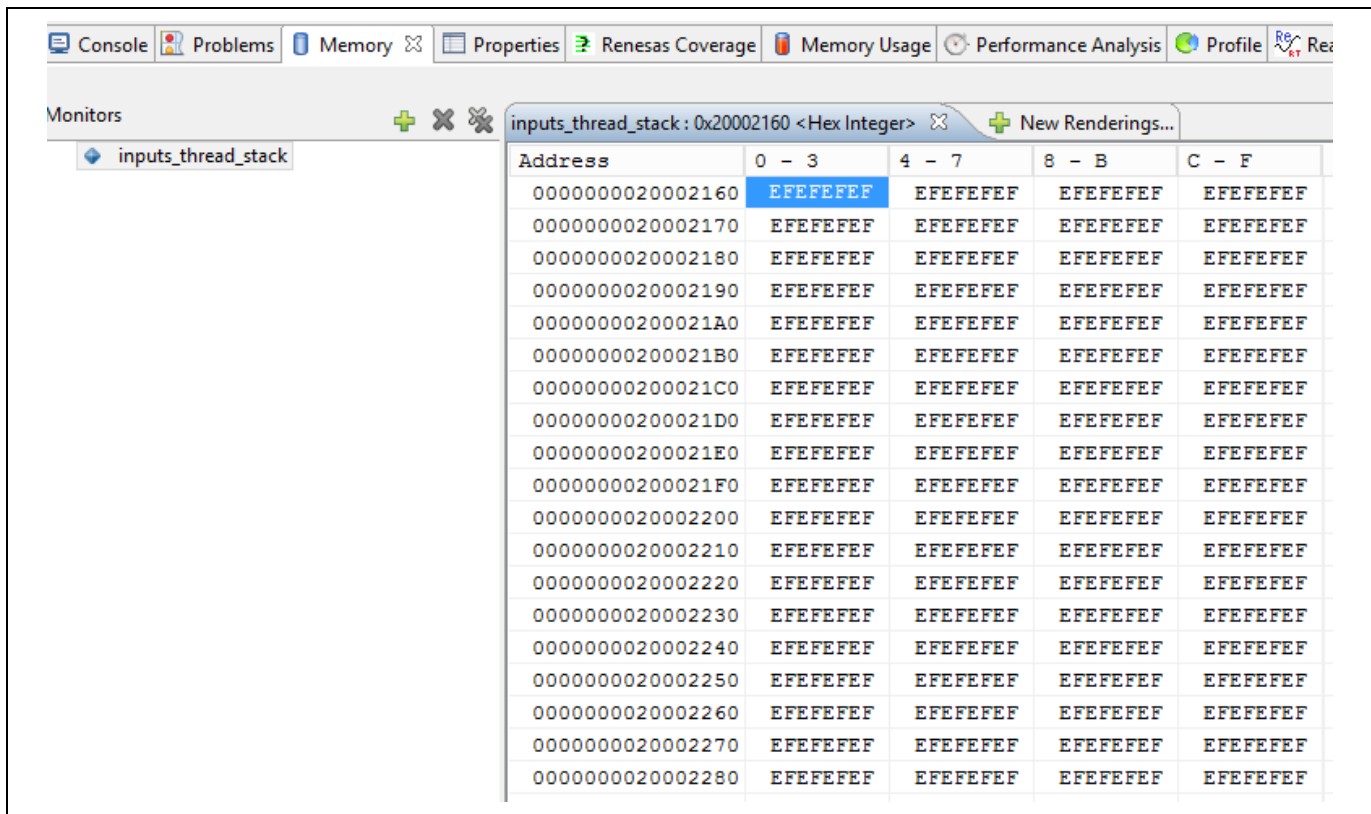


Figure 14. Displaying stack memory with the memory monitor

### 3.11 Importing and Building the Project

Follow the procedure in the included *Renesas Synergy Project Import Guide* to import the project into the e<sup>2</sup> studio ISDE, and then build and debug the project. For the debugging configuration, select **dk\_s124\_oob Debug** (under **Renesas GDB Hardware Debugging**).

## 4. Next Steps

After you run the example application, you can learn more about how the application works and the API calls involved by examining the application source code.

You can download additional Synergy example applications from the following URL:  
[www.renesas.com/products/embedded\\_systems\\_platform/synergy/Application\\_Notes.jsp](http://www.renesas.com/products/embedded_systems_platform/synergy/Application_Notes.jsp)

## 5. References

*Renesas Synergy Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf).



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

**Revision History**

<b>Rev.</b>	<b>Date</b>	<b>Description</b>	
		<b>Page</b>	<b>Summary</b>
1.00	Sep.23.16	-	Initial version
1.10	Oct.20.16	-	Migrated to SSP 1.2.0-b.1
1.11	Nov.30.16	-	Added support for IAR EW
1.12	Feb.21.17	-	Added support for SSP v1.2.0
1.13	Jul.10.17	-	Added support for SSP v1.3.0
1.14	Aug.04.17	-	Initial release
1.15	Feb.09.18	-	Updated for SSP v1.4.0
1.16	Mar.07.19	-	Updated for SSP v1.6.0

## Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.  
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.  
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.  
Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,  
Koto-ku, Tokyo 135-0061, Japan  
[www.renesas.com](http://www.renesas.com)

## Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

## 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/](http://www.renesas.com/contact/).