Renesas Synergy™ Platform

Simple PMOD Display Example for DK-S124

Introduction

This application note describes how to install, build, and run an example PMOD display application on a Renesas Synergy™ development board using the Renesas e² studio ISDE and Renesas Synergy Software Package (SSP).

This example application shows how to create a simple graphics-enabled application using the Okaya PMOD LCD and drivers within the SSP. When the application is running, you can control the displayed content using two on-board pushbuttons and a potentiometer. The example can be adapted to add visual effects to more complex applications.

Note: This application note assumes that as the user, you have some experience with the Renesas e² studio ISDE and SSP. Before performing the procedure in this application note, you should follow the procedure in your board's Quick Start Guide to build and run the Blinky project. This will help familiarize you with e² studio and SSP and also ensure that the debug connection to your board is functioning properly.

Target Device

This example application targets Renesas Synergy S124 devices.

Required Resources

To build and run the application, you will need:

- A Renesas Synergy DK-S124 board v3.0 or later
- An Okaya PMOD Display (bundled with every Renesas Synergy DK-S124 kit)
- A PC running Microsoft® Windows® 8 with the following Renesas software installed:
  - e² studio 7.3.0 or later
  - IAR Embedded Workbench® for Renesas Synergy™ v8.23.3 or later
  - Synergy Software Package (SSP) v1.6.0 or later
  - GCC ARM Embedded Toolchain 7 2017-q4-major (only applicable to e² studio)
  - Synergy Standalone Configurator (SSC) 7.3.0 or later (only applicable to IAR Embedded Workbench).

You can download the required software from the Renesas Synergy Gallery (https://www.renesas.com/us/en/products/synergy/software.html).

Contents

1. Configuring the DK-S124 for a Simple PMOD Display Example

The steps are as follows:

1. Connect the J-Link OB on J18 of the DK-S124 to the PC using a micro USB cable (Figure 1).
2. Verify that J7 (BOOT MODE) is not installed across the pins (Figure 2).
3. Connect Okaya PMOD™ display to J11 (Figure 3).

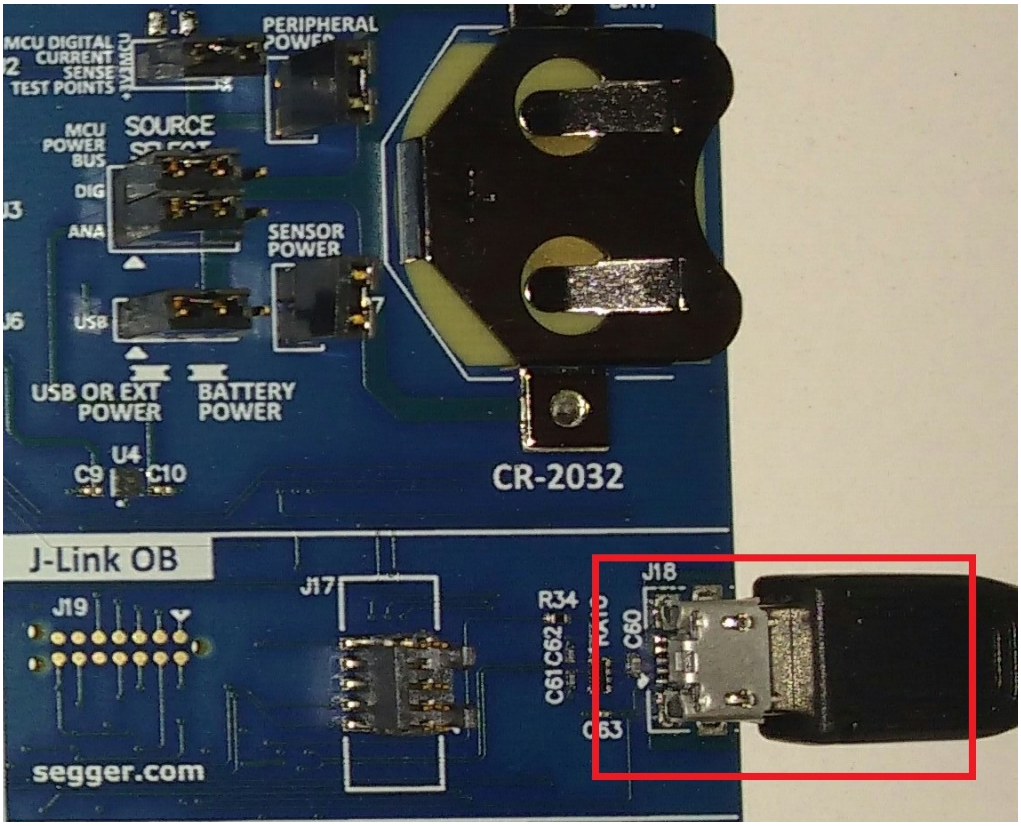

Figure 1. Power setup (top) and J-Link OB connection (bottom)

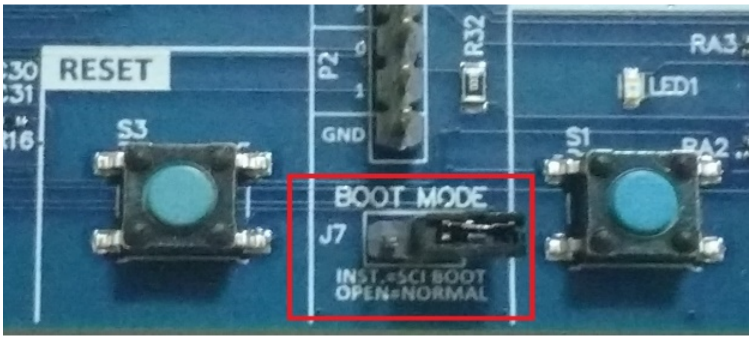

Figure 2. Boot mode configuration

Note: For the purpose of this example application, ensure that J7 is not installed across the pins.

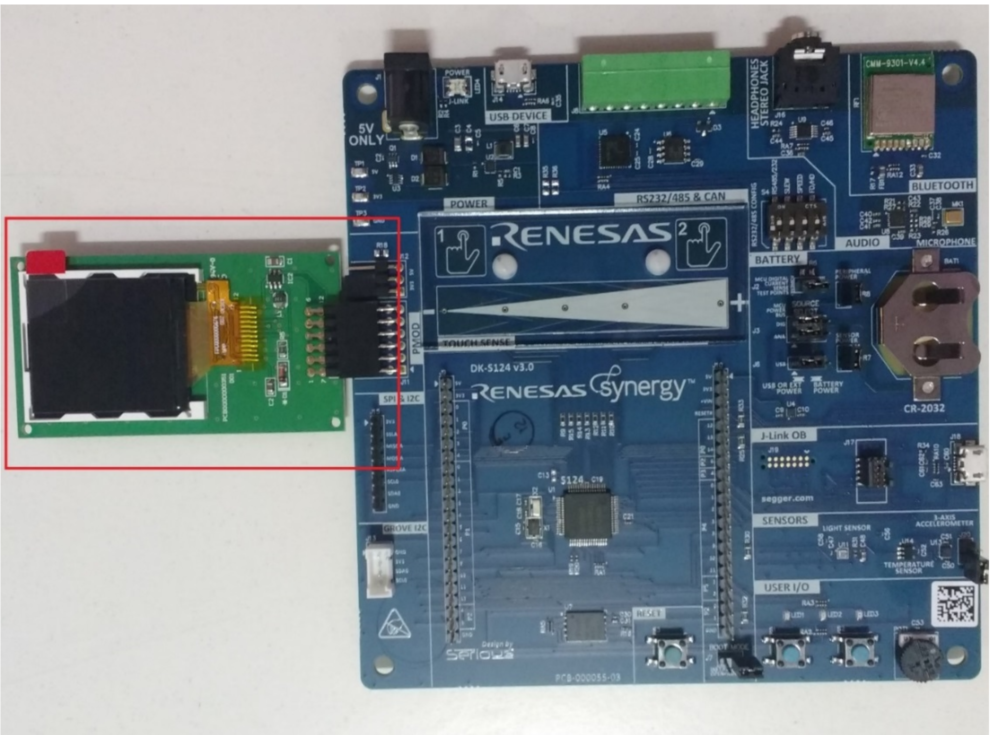

Figure 3. Okaya PMOD display connected to J11 PMOD™ socket

2. Importing and Building the Project

Follow the procedure in the *Importing a Renesas Synergy Project* (r11an0023eu0121-synergy-ssp-import-guide.pdf) to import the project into the e[2] studio ISDE, and to build, and debug the project. For the debug configuration, select **SimplePMODLCDExample Debug** (under **Renesas GDB Hardware Debugging**).

3. Observing the PMOD Display Application Output

When you first press the **F8** key or the **Resume** button to start the application, the application stops at `main()`. Press **F8** or the **Resume** button again to run the code.

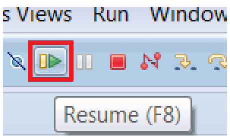

Figure 4. Resume button

Once the application is resumed, you should see a welcome screen (Figure 5, left). After few seconds (or once **S1** is pressed), the program proceeds to the screen with instructions (Figure 5, middle and right). Then, you can use the **S1** pushbutton to go to the next step (provided the application has finished drawing the current screen) and the **S2** pushbutton to change the screen orientation in the clockwise direction. Changing the screen orientation restarts the sequence of drawings, but the instruction screen is now skipped.

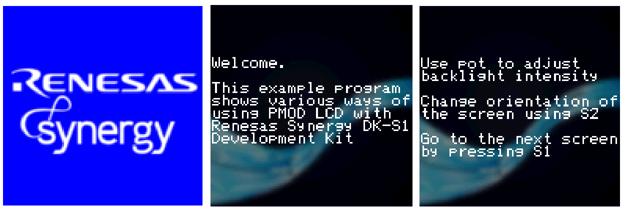

Figure 5. Splash, Welcome, and Instruction screens

While the application is running, you can use the potentiometer to adjust the backlight intensity. If you cannot see any difference in the strength of the backlight, verify that the PMOD™ is outputting 3.3V on the Vcc pin (set on the J12 header). To end the debug session, press `Ctrl + F2` or the **Stop** button.

Figure 6. Stop button

4. How the Application Works

The simple PMOD display example application is controlled by pushbutton switches **S1** and **S2** (found in the lower right corner of the DK-S124 board) and potentiometer **POT1** (located to the right of the two buttons).

Both pushbuttons are connected to hardware interrupt pins, which are controlled by the external IRQ framework. The framework allows you to control thread execution using hardware IRQs. In most cases, the application uses `sf_irq.p_api->wait` with the `TX_WAIT_FOREVER` timeout argument, to block processing in the thread until the given interrupt request is received. In the simple display example, however, the lcd thread scans through both button interrupts with timeout value of zero, and, if neither button is pressed, it waits for 20 milliseconds before repeating the process (IRQs are buffered by the framework before the next `sf_irq.p_api->wait` call). This gives enough processor ticks to lower-priority threads while still providing a responsive interface. The following diagram illustrates a simplified processing flow.

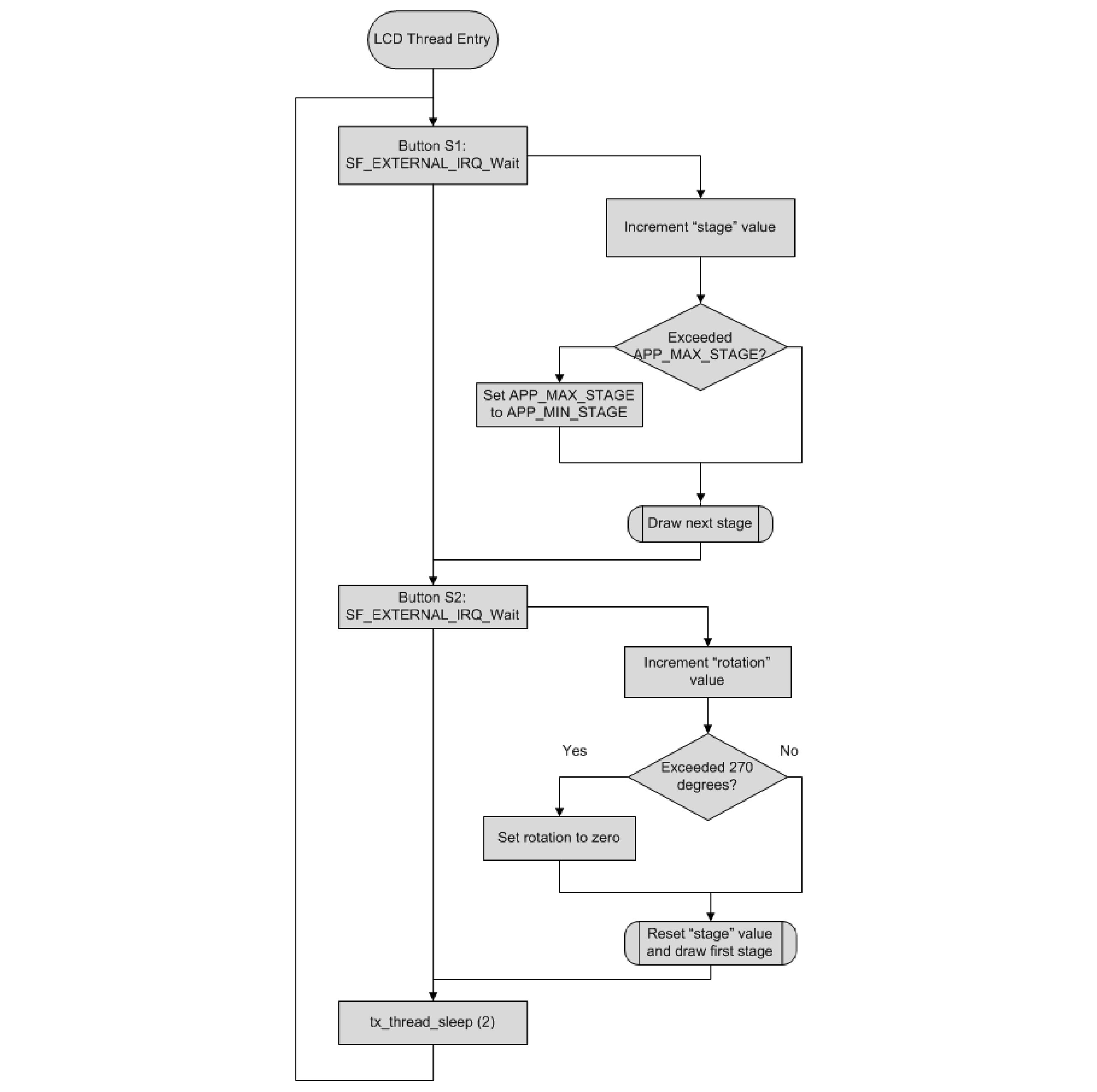

Figure 7. Simplified view of the example display application processing

A more detailed view of the External IRQ Framework processing can be found in the *Synergy Software Package (SSP) User’s Manual*, chapter 4.1.8.

The simple display example uses an Okaya PMOD LCD driver running on the SCI peripheral in Simple SPI mode. All driver files are contained inside the `src/lcd_setup` folder and can be easily copied into another application to enable the use of PMOD display. The `SPILCD_Init` function accepts two arguments: pointer to the SPI instance and initial value for screen rotation. The SPI module needs to be configured for the right channel and bit rate (9 and 2.5 mbit/s, respectively). However, the callback function input in the configurator is discarded as the display driver overrides it automatically with its own implementation, allowing for better data flow control when outputting data through the PMOD. The Okaya PMOD LCD driver works with Simple SPI on SCI as well as RSPI, though care must be taken to remove the transfer drivers for the RSPI interface to allow 8-bit data width.

The driver provides the following features:

- `SPILCD_Init` opens the SPI channel and initializes the PMOD display
- `SPILCD_Rotate` changes screen orientation to one of the enumerations defined in `spilcd_rotation_t`
- `SPILCD_Clear` fills the screen with provided color in RGB565 format
- `SPILCD_DrawPixel` draws a single pixel at a given position and with set color
- `SPILCD_DrawRectangle` draws a rectangle with the position, dimensions and color provided
- `SPILCD_DrawBitmap` renders a bitmap image stored at the address provided. The bitmap must be in RGB565 color format and should be read top-to-bottom with increasing address.
- `SPILCD_DrawText` renders a null-terminated string provided as an argument. Text can be formatted by calling the following commands prior to the `DrawText` function call:
  - SPILCD_SetTextColor sets the color values to be used for the text and its background. If both the values are equal, the background is rendered transparent (that is, the last screen contents are still visible from underneath the text). Text with transparent background is rendered using a slower drawing method and the background color should be provided, when possible.
  - SPILCD_SetTextCursor sets the top-left boundary and a starting point for the DrawText function. When text is wrapped onto the next line, it stays vertically aligned to the starting point and continues directly below it.
  - SPILCD_SetTextMargin sets a rectangular boundary for the text box. This function should be used to wrap text before it reaches the end of a screen. If the current text cursor position is outside of the boundary, the cursor is automatically set to the top-left corner of the rectangle defined by this function.

The simple PMOD display application uses the Okaya LCD driver to draw 15 different screens, each implementing different features of the driver.

In addition to the displayed content, control of backlight intensity is provided through an ADC peripheral running inside the control thread. This task is launched before the LCD thread is initialized, to ensure all setup is complete before displaying any data on the screen. The ADC peripheral running in continuous scan mode takes periodic readings of the channel 7 (connected to the on-board potentiometer POT1) potential difference. Despite the module being configured to read data with 12-bit resolution, software manipulates the data so that the readings become an integer value contained within 0-100 range. While the level of precision required for a smooth backlight control is still sufficient, reducing the effective resolution acts as a jitter-filter to prevent unnecessary PWM duty cycle updates when the ADC readings are rapidly oscillating. If the reading (post-processing) is different from the previous one, the duty cycle of the GPT (connected to the PMOD display backlight-enable pin) is updated using `R_GPT_DutyCycleSet`. ADC sampling frequency is limited to 33 Hz by suspending the thread for 30 ms using `tx_thread_sleep` (3) after each reading.

The following diagram illustrates the processing inside the control thread.

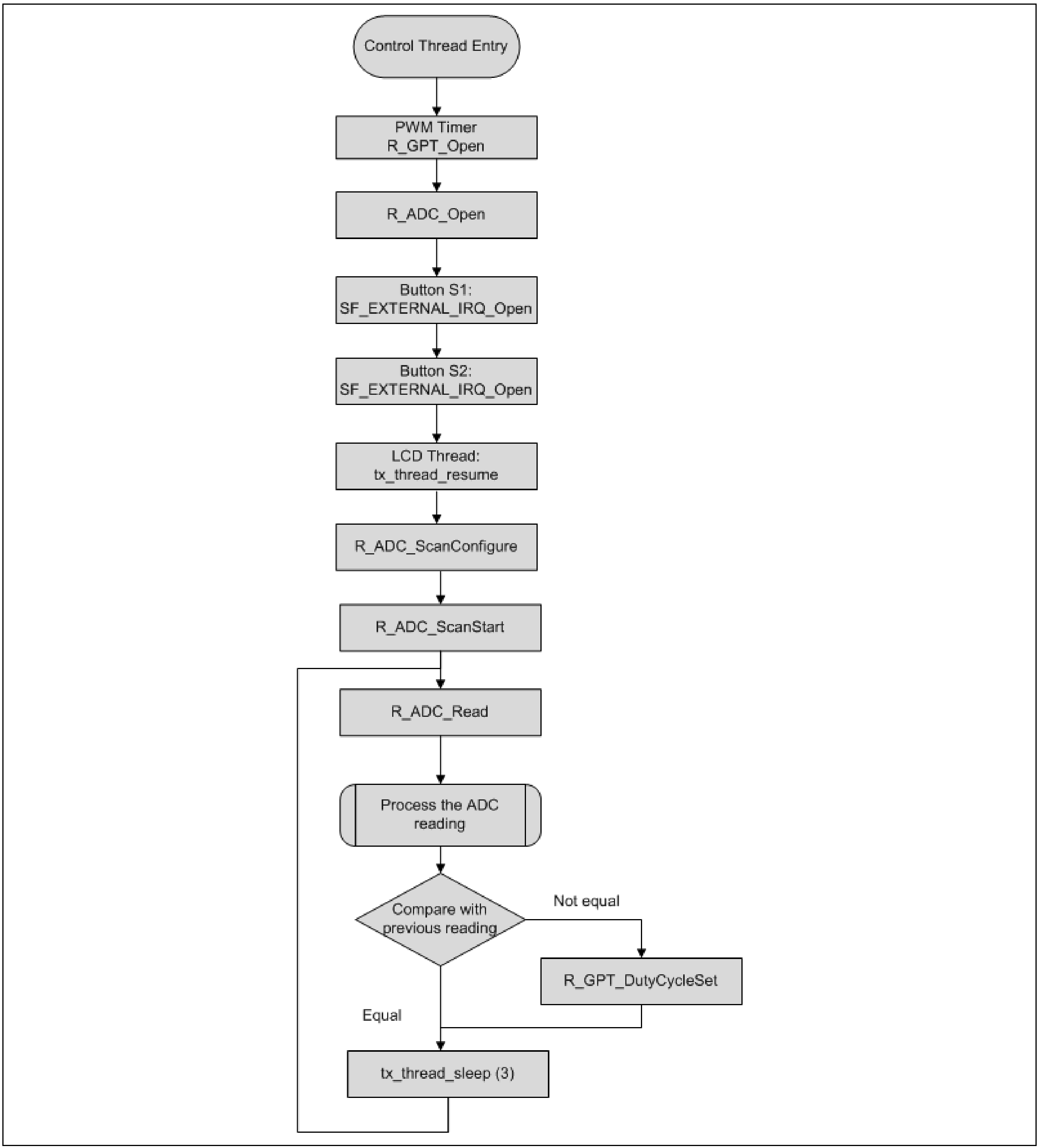

Figure 8. Simplified view of control thread processing

5. 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 also download additional Synergy example application projects from the following URL: www.renesas.com/synergy/applicationprojects.

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	Jan.11.17	—	Initial version
1.01	Feb.17.17	—	Updated for SSP v1.2.0 and DK-S124 v3.0
1.02	Aug.23.17	—	Updated for SSP v1.3.0
1.03	Sep.27.17	1	Required resources of SSP version changed
1.04	Jan.13.18	—	Updated for SSP v1.3.3
1.05	Mar.15.18	—	Updated for SSP v1.4.0
1.06	Mar.04.19	—	Fixed warnings and increased stack size for control thread. Updated for SSP v1.6.0.

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/.