Renesas Synergy™ Platform

DK-S124 Simple Record/Playback

Introduction

The Out-of-Box (OoB) demonstration program is an example of a simple multithreaded application design that reads the analog value on the microphone and either records it to the available RAM for playback at a later time, or passes it directly to the amplified earphone jack, which is acting as a simple amplifier. It does this at a set sample rate determined by one of the General-Purpose Timers (GPTs).

While the Renesas DK-S124 Synergy™ MCU Kit comes with the Pmod™ LCD display (Figure 1), this program uses a simple two-button interface for control, and on-board LEDs for operator feedback. The Pmod LCD is not used in this example application.

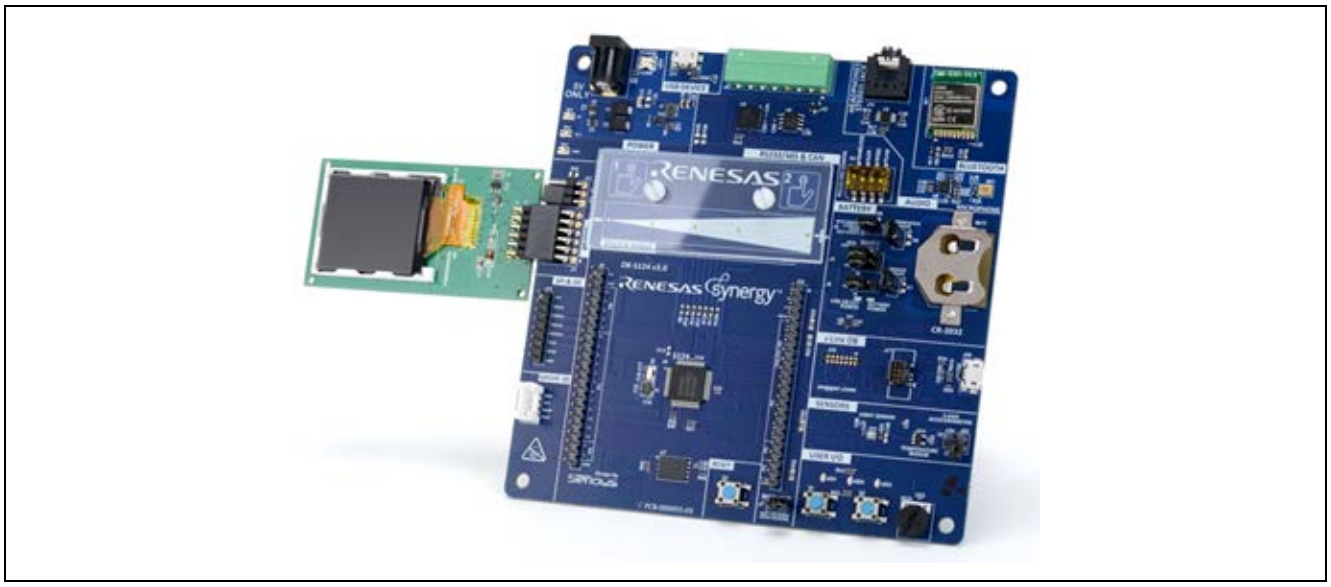

Figure 1. The DK-S124 with included Pmod LCD attached

Prerequisites

You should have some experience with either the Renesas Synergy™ e² studio or the IAR Embedded Workbench® for Renesas Synergy™ (IAR EW for Synergy) Integrated Solutions Development Environment (ISDE). The program may be compiled and run using either tool chain. You should also be familiar with the Synergy Software Package (SSP). Before you perform the procedures in this application note, you may want to follow the procedure referenced in your board's Quick Start Guide to build and run the Blinky project. By doing so, you will become familiar with building and running SSP applications while ensuring that the debug connection to your board is functioning properly.

Required Resources

The example application targets Renesas DK-S124 Synergy MCU Kit. To build and run the application, you need:

- A Renesas DK-S124 v3.0 & v3.1 Synergy MCU board
- A PC running Microsoft® Windows® 7 or later with the following Renesas software installed:
  - e² studio ISDE v7.3.0 or greater or IAR Embedded Workbench® for Renesas Synergy™ v8.23.3 or greater
  - Synergy Standalone Configuration (SSC) v7.3.0
  - GCC toolchain version 7.2.1 or later
  - Synergy Software Package (SSP) v1.6.0 or later

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

Goals and Objectives

The goal of this application note is to familiarize you with the capabilities of the DK-S124 Synergy MCU Kit and the challenges faced when trying to design a robust application.

Time Required

You should be able to install, build, and run the example application in under 30 minutes. The key steps involved are:

1. Configure the DK-S124 MCU board to run the Simple Record/playback application.
2. Import, build, and debug the project.
3. Record and playback recorded sound.
4. Use the board as a simple amplifier and hear the effects of a software IIR filter.
5. Change the mode of operation using the two momentary push button switches.
6. Vary the volume using **POT1**.

Contents

1. Design Considerations

The principle goal of this application is to show how to acquire analog data from the onboard microphone MK1 using the Periodic ADC (Analog-to-Digital Converter) framework in the SSP and use that data to drive the 12-bit DAC (Digital-to-Analog Converter) connected to the amplifier that drives the earphone jack. To keep the application design simple, the record mode records data to the internal SRAM of the S124 processor.

1.1 Synergy Platform capabilities

This application primarily deals with sampling analog data using the on-board ADC and writing that data back to the on-board DAC to drive the earphone amplifier (U9). Both sampling and playback are performed at a controlled rate by the GPT modules. The SSP application provides both higher level framework modules and lower level driver modules to interface with these hardware peripherals without the need to write any low-level driver code.

Figure 2 highlights the S124 hardware peripherals used by the Simple Recorder example application.

Figure 2. S1 Peripherals used by the simple record example

Table 1 defines the processor pin connections provided to connect the peripherals used in this example to the various external components on the DK-S124 board. For additional reference, see the DK-S124 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
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 0 pin 0	SCL0 (I2C data)
U8	Microphone amplifier	Port 0 pin 13	Analog channel 8
U9	Audio power amplifier	Port 0 pin 14	DAC output

2. Design Overview

The DK-S124 board contains a microphone MK1 mounted directly on the PCB along with an earphone jack J16, which accepts the standard 3.5-mm connector found on typical earbuds. While the earphone connection is a typical 3.5mm stereo jack, the audio output is only available as mono since the amplifier connects both channels to the same channel on the DAC. Figure 3 shows a simple block diagram of the microphone and earphone jack connections to the S124 processor.

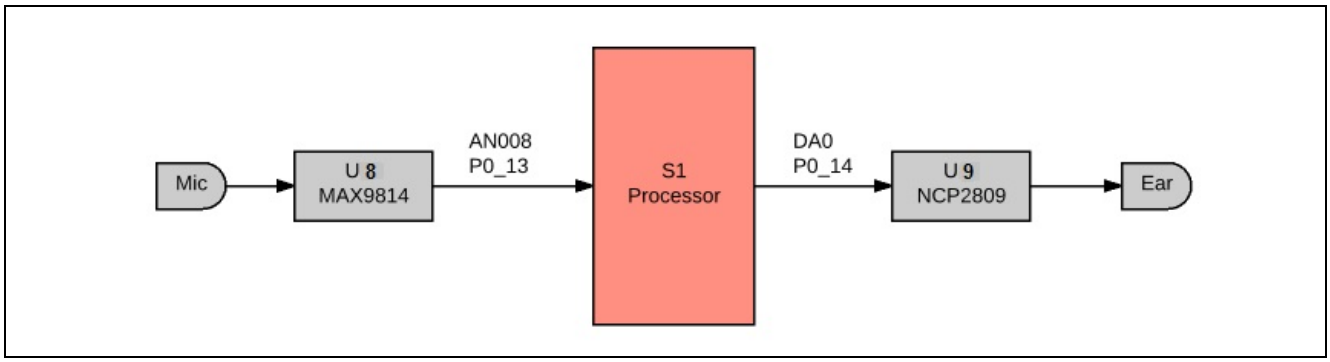

Figure 3. Simple block diagram of hardware connections

The Simple Record Example application has two different operating modes.

- Passthrough mode, which samples the analog value present on the microphone at 16 kHz and simply passes the digital data directly through to the earphone jack. No data is saved.
- Record mode, which samples the analog value present on the microphone, again at 16 kHz, and then stores the data to an internal RAM buffer. The recorded data may be played back at a later time.

Two pushbuttons on the DK-S124, **S1** and **S2**, are used to control the application. Three LEDs on the board are used for operator feedback. Refer to the Design Example section for operator interface details.

Both pushbuttons on the DK-S124 are connected to hardware pins with interrupt capability. Rather than using the interrupt capability, this application shows how to use 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 debounce by reading the state of the switch at an interval longer than the typical mechanical switch bounce time (50 ms). More complex debounce techniques may be employed, such as the excellent switch debounce algorithms published by Jack Ganssle (http://www.ganssle.com). For the purpose of this application note, periodic polling works fine.

You may reference other application notes, such as the Simple Audio Example, which highlights the use of the Synergy IRQ Framework to detect button 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.

2.1 Accelerometer initialization

On the DK-S124 development board v2.0, the Accelerometer IRQ line is shared with button **S1**. Since the accelerometer IRQ pin defaults to a push/pull configuration and works at an active-high IRQ state, an initialization has to pull down on the line. To do this, the I²C framework is added to the Audio thread and several I²C commands are issued to the accelerometer during the thread initialization sequence.

The jumper J20 on the DK-S124 v3.0 board can disconnect the accelerometer IRQ and **S1**, and so the initialization of this accelerometer is unnecessary.

2.2 Maximizing resources

One of the goals of this example is to record audio to the internal RAM of the S124 processor. The amount of available RAM is maximized by reducing its use by other Synergy components to the greatest extent possible. One of the design decisions that resulted from this goal is to not use the SSP audio playback framework component. This framework component provides additional control not found in this example, such as the ability to stop and resume playback. However, it also uses the SSP messaging framework internally and consumes more resources. This application note shows how to mimic some of the functionality of the audio playback framework, through the use of a few lower level drivers, while minimizing resource use.

To ensure simplicity, and to maximize available resources, this application is designed with a single thread as summarized in Table 2.

Table 2. Application thread names and functions

Thread name	Thread function
Audio thread	Handles both record/playback as well as direct Passthrough mode

Since additional threads require additional stack space, reducing the number of threads reduces the amount of RAM used, making more RAM available for the record buffer.

Figure 4 is a conceptual diagram of the Simple Recorder Example application. Peripherals internal to the S124 processor are highlighted in blue, and external peripherals such as the accelerometer are highlighted in gray. The Synergy component used to access each peripheral is shown in text on the connecting line.

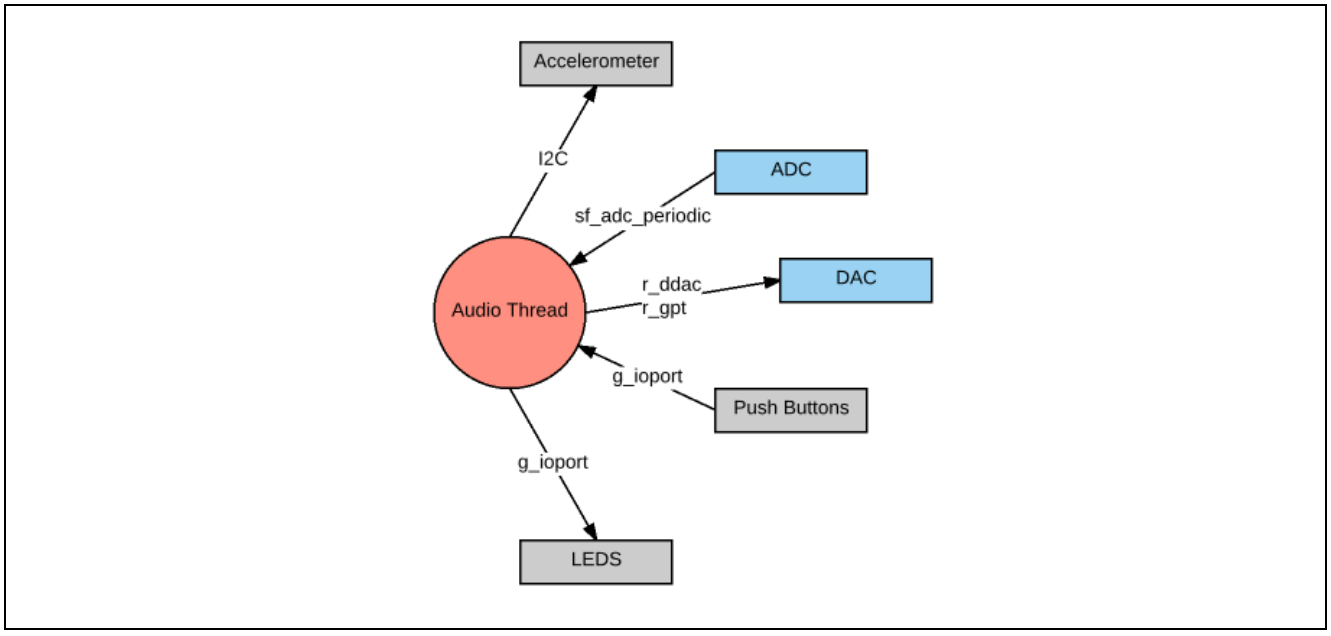

Figure 4. Conceptual diagram of simple recorder application

2.3 Audio Thread logic

The code in the audio thread can be broken into two distinct sections - the code that executes each time the main thread loop executes, and the code that executes inside of one of two callback routines that are executed in response to timer interrupts. The callback routines are discussed in more detail later. There is a callback routine associated with sampling analog data from the microphone and another callback associated with playing back recorded data.

Figure 5 shows the logic implemented in the main audio thread loop. To be responsive, the audio thread samples the button states every 50 ms (5 thread ticks). If a long press of **S1** is detected, the code toggles between the Passthrough and Record modes. Normal button presses result in a few variables, such as recording, playback, filter, amp_enable, being set. These variables are then examined inside one of the two callback routines. The callback routine clears the variables, if appropriate.

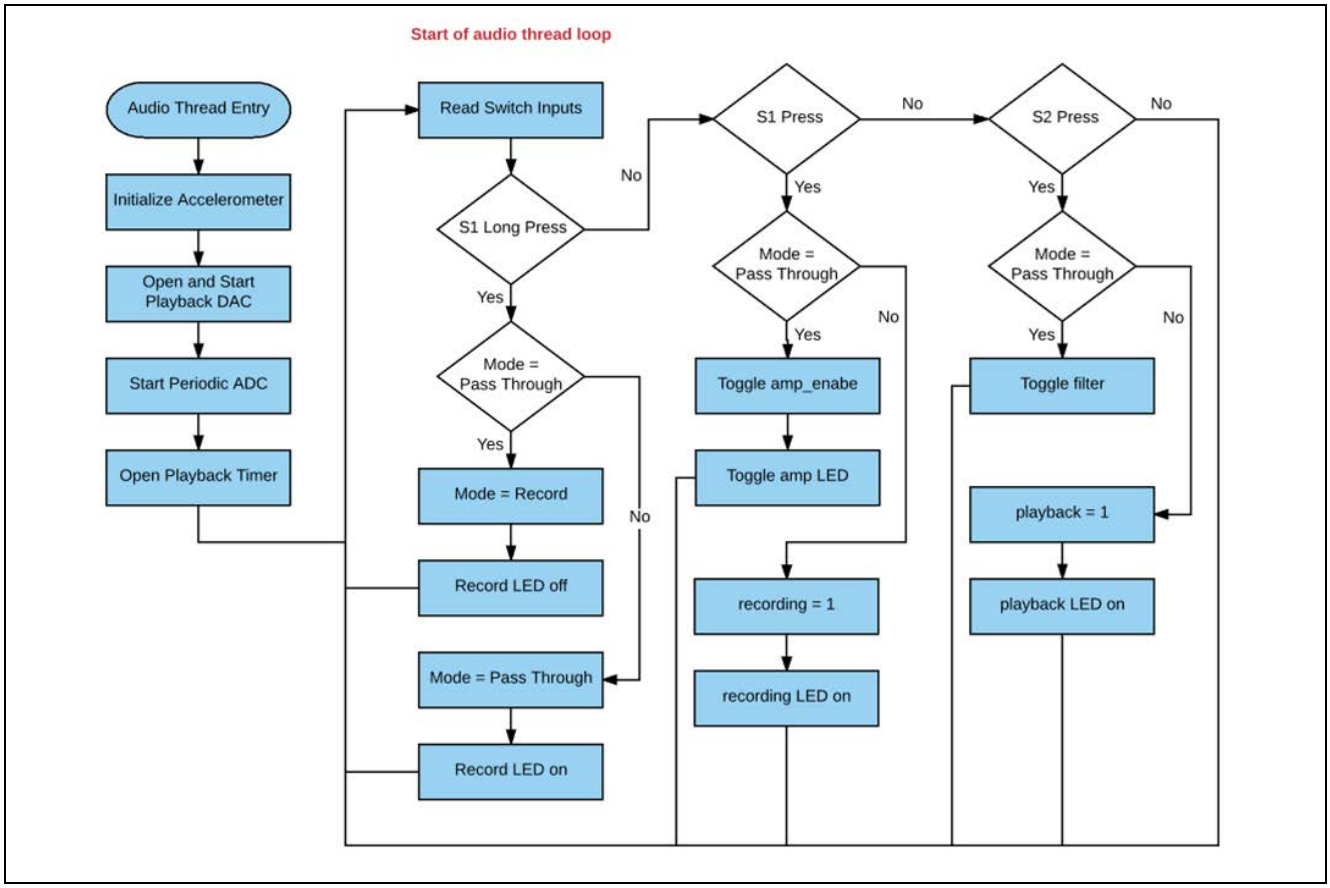

Figure 5. Simplified flowchart of the inputs thread

2.4 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. The configurator is opened by double-clicking on the `configuration.xml` file from inside the project explorer in e^2 studio. For more information on Synergy modules and drivers, refer to the *SSP User's Manual*.

To configure interfaces to external peripherals, you must know how a peripheral is actually connected to the S124 microcontroller. As mentioned, it is necessary to initialize the accelerometer to free up the **S1** for use. The S124 processor communicates with accelerometer over an I^{2}C bus, making it necessary to add an I^{2}C driver to the thread to communicate with the accelerometer. As shown in Figure 6, the `r_riic` driver has been added.

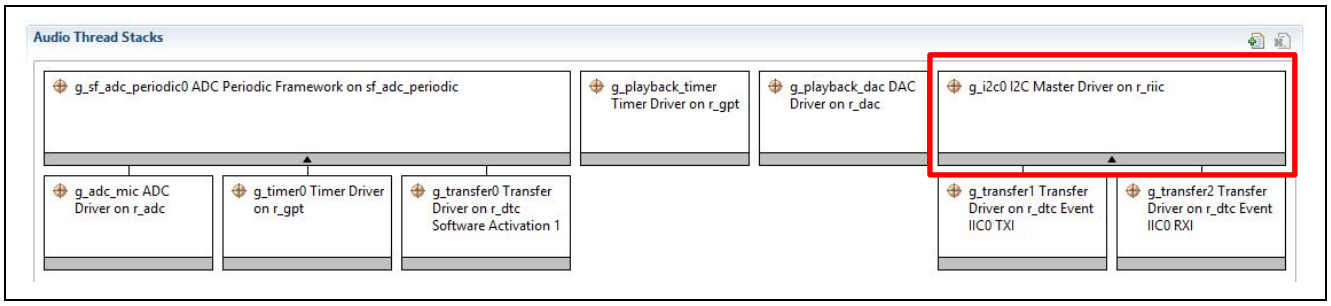

Figure 6. Audio thread stacks

Synergy processors contain two different peripherals for I²C communication, a dedicated I²C peripheral and the Serial Communications Interface (SCI) that may act as either a simple I²C, simple SPI, or UART (Synchronous or Asynchronous). The SSP supports both types of I²C peripherals. The same scenario is true for SPI peripherals.

Choosing which peripheral to use is often based on the complexities of your system. In many cases you may not need all the power available on the dedicated SPI or I²C peripherals. You may need to use the pins assigned to the dedicated I²C 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 to make when designing the system is the numbers of each type of peripheral. While the S124 processor only has two dedicated SPI and I²C peripherals, it has three SCI peripherals which can be configured as either simple SPI or I²C.

The reason that the Simple Record application uses the dedicated I²C peripheral, rather than one of the three SCI peripherals, and this is simply because these are the pins that the hardware designers chose to use to connect the S124 processor to the accelerometer.

2.5 Using the Periodic ADC Framework

Recording digital data is the process of sampling analog data (for example, voltage produced by a microphone) at a fixed rate. The fidelity of the sample is determined by two basic properties - the rate at which the sample is taken, and the accuracy of the sample. The accuracy of the sample is a function of the bit depth of the ADC used. For example, a 12-bit ADC gives better sample accuracy than an 8-bit converter.

One of the cornerstones of this application is the ADC Periodic Framework component. This framework component provides the ability to sample analog data at a prescribed rate. You define a callback function to process the data each time a new sample is complete. The ADC Periodic Framework can be a bit confusing to get working. If you already have experience using this framework component, you can skip ahead to the next section. If not, study this section carefully and review the application code for the callback function associated with it, to understand how to configure and handle scan data in the callback.

One of the key strengths of the ADC Periodic Framework component is the ability to sample multiple analog channels. This feature is used to sample the analog voltage present on the potentiometer POT1 and the microphone MK1 for each sample iteration. The POT1 readings are then used to set the volume level.

Note: You can even configure the ADC Periodic Framework component to take multiple samples of each channel in each iteration. However, this application does not explore that functionality.

While the ability to sample multiple analog channels is convenient, it has a restriction. The ADC can only sample contiguous channels. For example, if you want to sample channels 5, 7 and 8, the returned data also includes channel 6. In the example used, the potentiometer (POT1) is connected to channel 7 and the output of the microphone amplifier is connected to analog channel 8. Since these channels are contiguous, no extra data is returned in the sample.

The ADC Periodic Framework component shown in Figure 7 is essentially a framework layer built on top of the r_adc ADC driver, an r_gpt Timer driver and the r_dtc transfer driver.

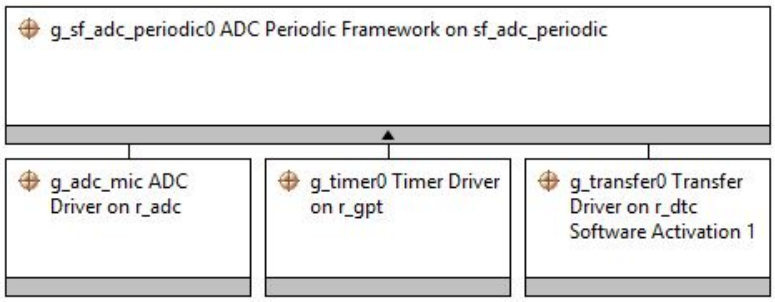

Figure 7. ADC Periodic Framework Stack

As is the case with configuring all SSP components, you click on the appropriate box and then define properties using the **Properties** tab. The ADC Periodic Framework component contains numerous properties, and the defaults are acceptable on many of them. Once you have imported and built the Simple Record application, as outlined in the Design Example section, you can browse all the properties for each of the components used in the applications. For clarity, a few of the properties are reviewed in this section.

As you can see in Figure 8, the buffer used to hold the samples is 128-bytes deep, with the name `g_user_buffer`. A single sample is taken in each scan iteration and the GPT timer channel 0 is used to trigger the scan. The callback name is `g_adc_framework_user_callback`.

◢ Common	
Parameter Checking	Disabled
◢ Module g_sf_adc_periodic0 ADC Periodic Framework on sf_adc_periodic	
Name	g_sf_adc_periodic0
Name of the data-buffer to store samples	g_user_buffer
Length of the data-buffer	128
Number of sampling iterations	1
Callback	g_adc_framework_user_callback
Name of generated initialization function	sf_adc_periodic_init0
Auto Initialization	Enable

Figure 8. Properties for sf_adc_periodic

The data-buffer (`g_user_buffer`) used to store the samples is defined for you in the auto-generated `audio_thread.c` file. This buffer is referenced in the callback routine to read the data associated with each scan. An important consideration here is the GPT channel selection. In this case, we must use channel 0 since the channel used by the GPT timer module is locked at channel 0 (Figure 9).

Property	Value
◢ Common	
Parameter Checking	Disabled
◢ Module g_timer0 Timer Driver on r_gpt	
Name	g_timer0
Channel	0
Mode	Periodic
Period Value	16000
Period Unit	Hertz
Duty Cycle Value	50
Duty Cycle Unit	Unit Raw Counts
Auto Start	False
GTIOCA Output Enabled	False
GTIOCA Stop Level	Pin Level Low
GTIOCB Output Enabled	False
GTIOCB Stop Level	Pin Level Low
Callback	NULL
Interrupt Priority	Priority 2

Figure 9. r_gpt properties for ADC Periodic Framework

Figure 9 displays the key parameters for the r_gpt module. You must enable and pick a priority for the GPT0 COUNTER OVERFLOW IRQ (interrupt) for the scan to work. Additionally, the sample rate of 16 kHz is defined by setting the **Period Value** to 16000 and the **Period Unit** to Hertz. The **Duty Cycle Value** and **Duty Cycle Unit** properties have no effect when the **Mode** is set to periodic or one shot.

The component with the largest number of properties is the `r_adc` component, which is used to configure the ADC unit. Here again, most of the default properties are acceptable. Since no changes are necessary to the lower half of the properties, the focus is only on the highlighted properties as shown in Figure 10. As with many SSP components, you must enable the IRQ associated with it. Since this component is used to scan the live sound coming in through the microphone, the ADC0 SCAN END IRQ is assigned a priority of 1.

Here again, as is the case with many SSP components, a name should be assigned to the module. This name is used by the application code to reference the unit. In this example, the ADC module is named `g_adc_mic`.

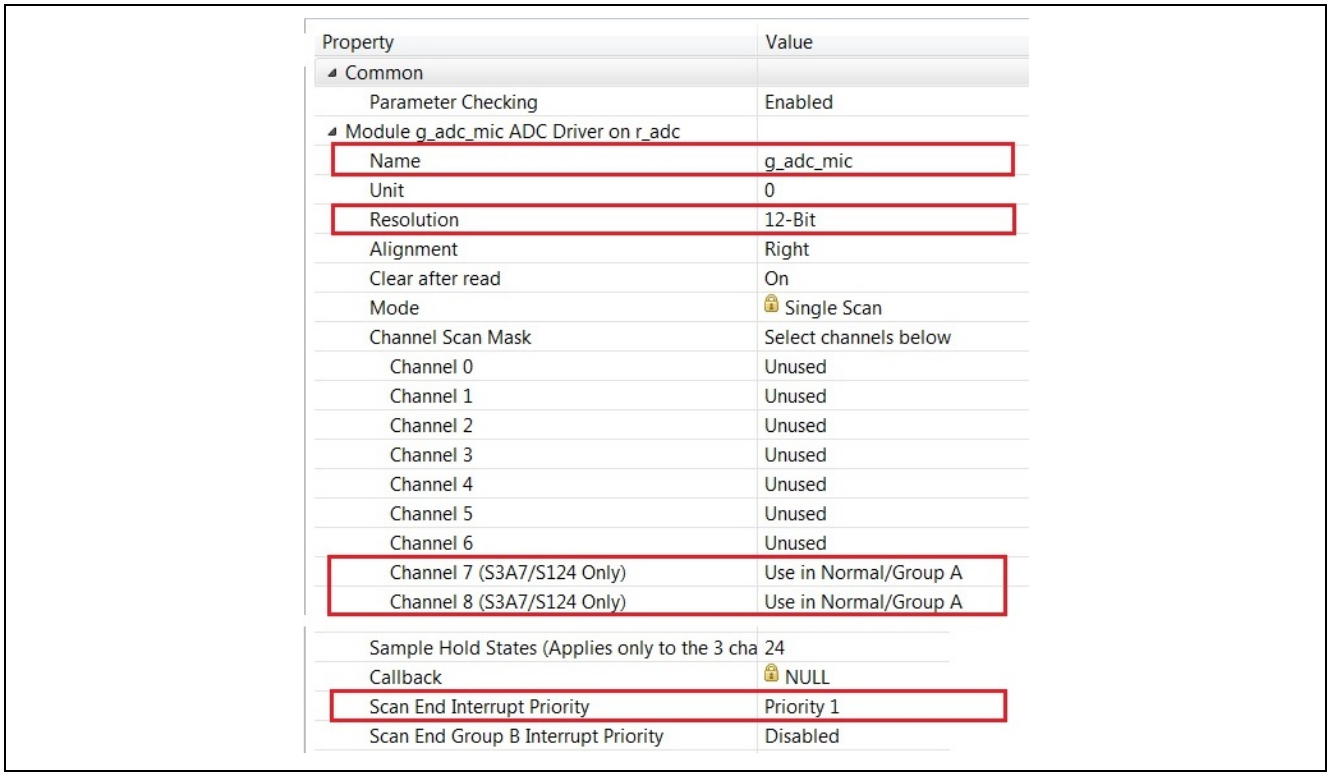

Figure 10. Properties for r_adc component of ADC Periodic Framework

One of the most important properties to set correctly is the **Resolution**. At the time in writing this application note, the SSP defaults this field to 8-bit (S7G2 only), which does not work for the S124 processor (Figure 11). If you are using a S124 Synergy MCU and you forget to select either 12-bit or 14-bit (S3A7/S124 Synergy MCU Groups only), you will receive an invalid parameter run time error. In this example, a 12-bit is used because the DAC output only supports 12-bits, and so the extra 2 bits are wasted if a 14-bit ADC operation is chosen instead.

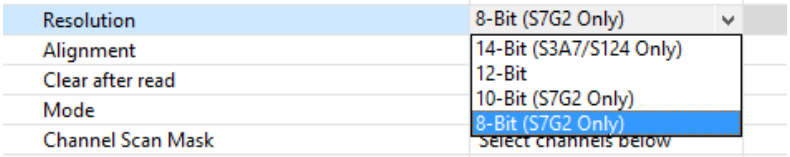

Figure 11. ADC resolution defaults to 8-bit (S7G2 Only)

Finally, the actual channels to be sampled each time an ADC scan occurs, need to be configured. In this example, channel 7 and 8 have been selected.

2.6 Defining the Callback for the ADC Periodic Framework

Once you have defined all the properties and built your code, you see a code section similar to that shown below in the `audio_thread.h` file.

```
/** Buffer where the sampled data will be stored for application usage */
extern uint16_t g_user_buffer[16];
#ifdef g_adc_framework_user_callback
#define ADC_PERIODIC_ON_ADC_PERIODIC_CALLBACK_USED_g_sf_adc_periodic0 (0)
#else
#define ADC_PERIODIC_ON_ADC_PERIODIC_CALLBACK_USED_g_sf_adc_periodic0 (1)
#endif
#if ADC_PERIODIC_ON_ADC_PERIODIC_CALLBACK_USED_g_sf_adc_periodic0
/** Declaration of user callback function. This function MUST be defined in the user application.*/
void g_adc_framework_user_callback(sf_adc_periodic_callback_args_t * p_args);
#endif
```

This code defines the function prototype of the callback for you. This code must be defined in the user application. You will find the code for `g_adc_framework_user_callback()` at the bottom of the `audio_thread_entry.c` file. The flowchart in Figure 12 highlights its functionality.

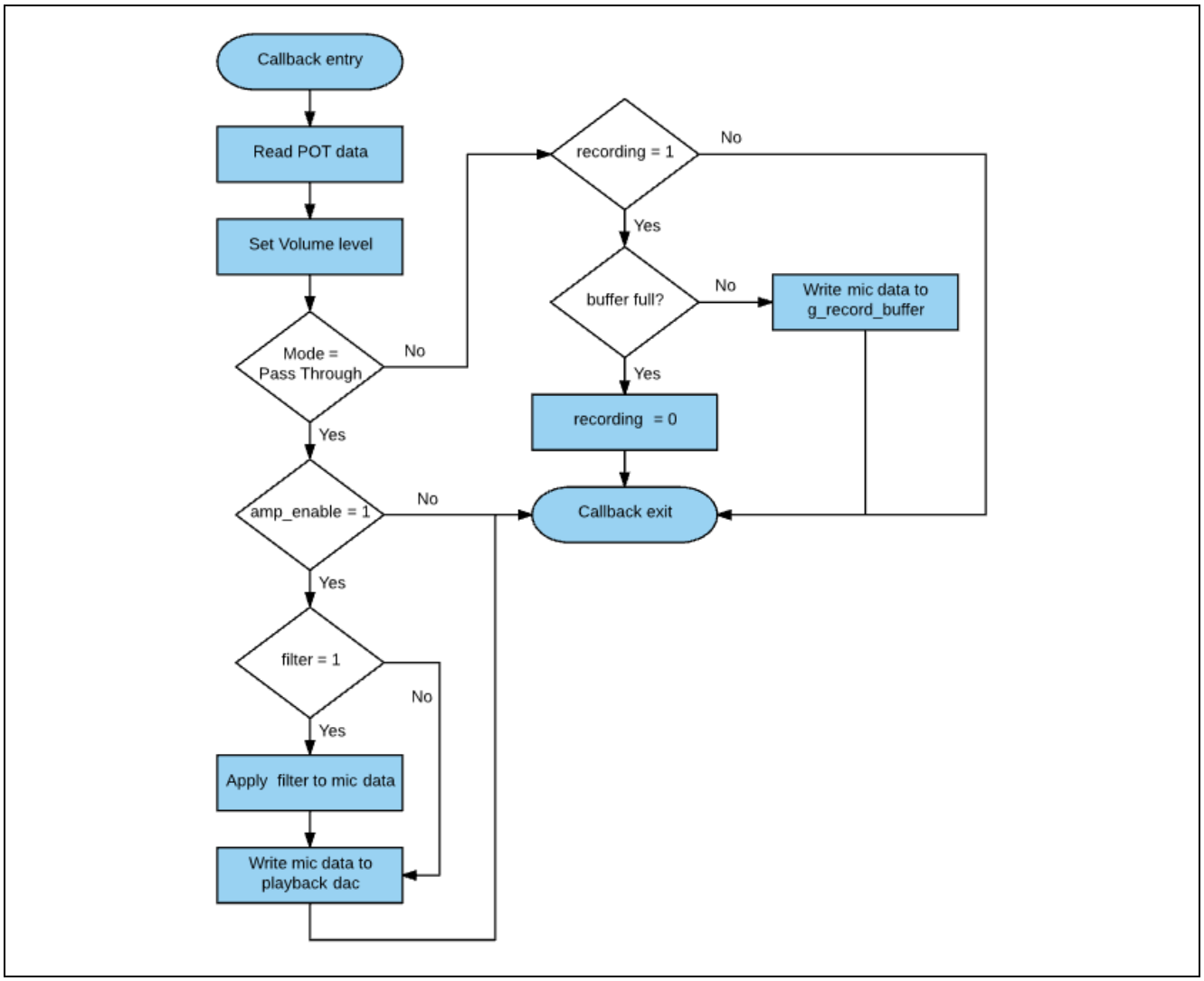

Figure 12. Logic for ADC periodic callback

2.7 Source code layout

This section gives a brief overview of the source code layout that you expect to see when you build the Simple Recorder project in e² studio. If you are experienced at writing Synergy applications, you are probably familiar with how e² studio organizes a standard project, so you can skip to the next section.

After you import the Simple Recorder application into e² studio and built the project, you should see a directory structure similar to the one shown in Figure 13. The first time you import the project, you may notice that the **synergy_gen**, **synergy**, **Debug**, and **synergy_cfg** folders do not exist. These folders are built automatically by the framework when you compile the project, so, to reduce size, these subfolders are normally not included when applications are distributed. The only files you typically need, for example, to add/delete code, in your project are the files highlighted in green. The **synergy_gen** folder is highlighted in Figure 13 to emphasize the fact you may wish to reference these auto-generated files, or include the header files. As an example, the `audio_thread.h` file contains the extern definition for the `g_user_buffer` used to hold sampled data, and so it is included in any of our user defined source files that need to reference this global.

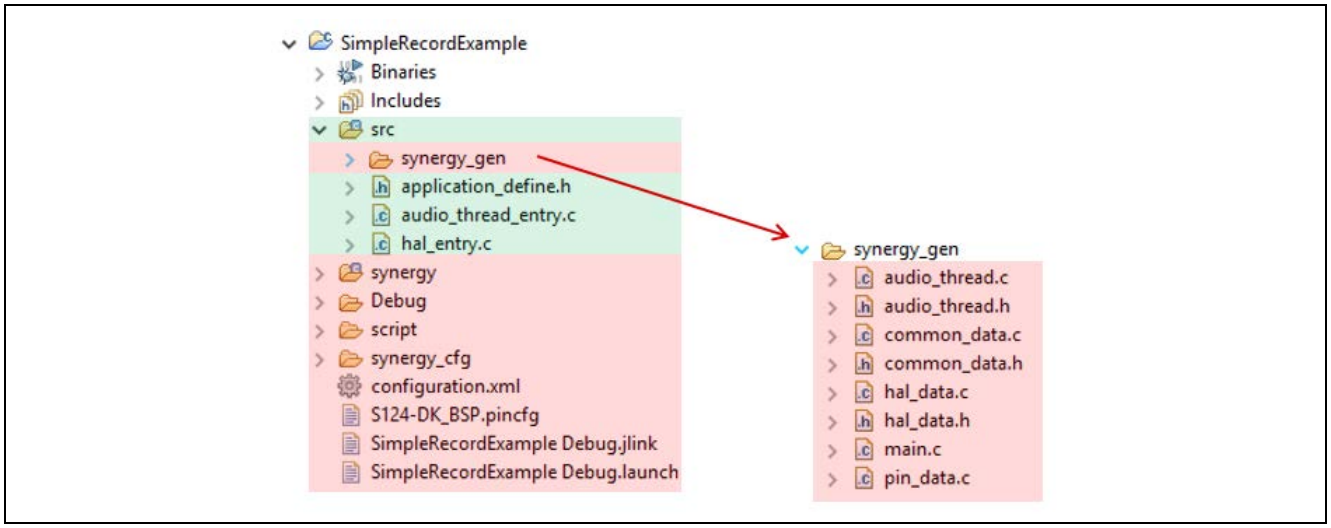

Figure 13. Source code layout for DK-S124 OoB v1.0 program

3. Design Example

The DK-S124 comes preloaded with the OoB application software. Prior to running this OoB program or programming the board from the e² studio ISDE, you must first configure the board according to the following section.

3.1 Configure the DK-S124 Synergy MCU for the Simple Recorder example

To configure the DK-S124 MCU:

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 14.
2. Connect the JLink-OB on J18 of the DK-S124 MCU to the PC using a micro USB cable (Figure 14).
3. Connect a set or earbuds or optionally a small powered speaker to the Headphones Stereo Jack Connector (J16) as shown in Figure 15.

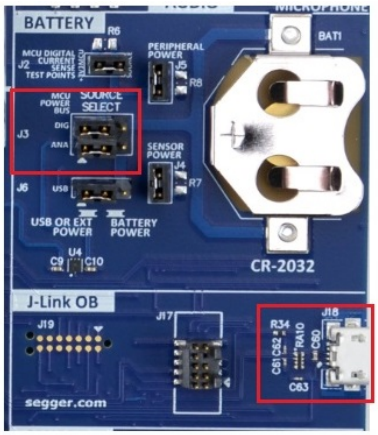

Figure 14. Power setup (top) and J-Link OB connection (bottom)

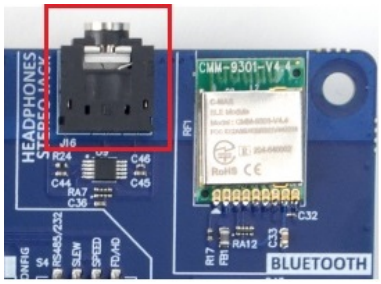

Figure 15. Connecting headphones or external speaker

Now that you have configured the board, you will need to import the Simple Record application into e^2 studio, compile it and load it onto your DK-S124 board.

Upon power up, the Simple Record program defaults to the Passthrough mode, acting as a simple microphone amplifier. You change between the Passthrough mode and the Record/playback mode with a long press of **S1**. The following sections describe how to interact with the program in the two different modes.

Note: Jumper J20 should be removed on both DK-S124 v3.0 and v3.1.

3.1.1 Passthrough mode

In this mode:

1. **S1** enables/disables the output. When enabled, the red LED (LED1) lights up and data is passed directly from the microphone to the earphone jack. When disabled, the red LED turns off and no sound is passed to the earphone jack. Upon power up, the program defaults to disabled.
2. **S2** adds a software filter to the data read from the microphone before passing it to the earphone jack.
3. POT1 acts as a volume control during playback. Rotating clockwise increases the volume.
   CAUTION: Listening to excessive volume levels can damage your hearing.

3.1.2 Record mode

In this mode:

1. A long press of **S1** for 2 seconds, while in the Passthrough mode, switches the application to the Record mode. In the Passthrough mode, the yellow LED (LED2) is lit continuously.
2. Pressing **S1** triggers a recording. The red LED lights up while the board is recording. Once the available buffer space is consumed, the program automatically exits the Record mode and the red LED goes out.
3. Pressing **S2** plays back the recorded data. The green LED (LED 3) is lit for the duration of the playback.
4. POT1 acts as a volume control when recorded data is played back (by pressing **S2**).

4. e² studio Tricks

Memory Usage tab: You may need to monitor the system resources as you implement your design. The e² studio provides a handy tool for examining memory usage. Figure 16 shows how the memory usage display looked during the development of the Simple Audio Recorder example. The **Memory Usage** tab parses the `.map` file that is created when you compile your program and then displays the information in a readable format.

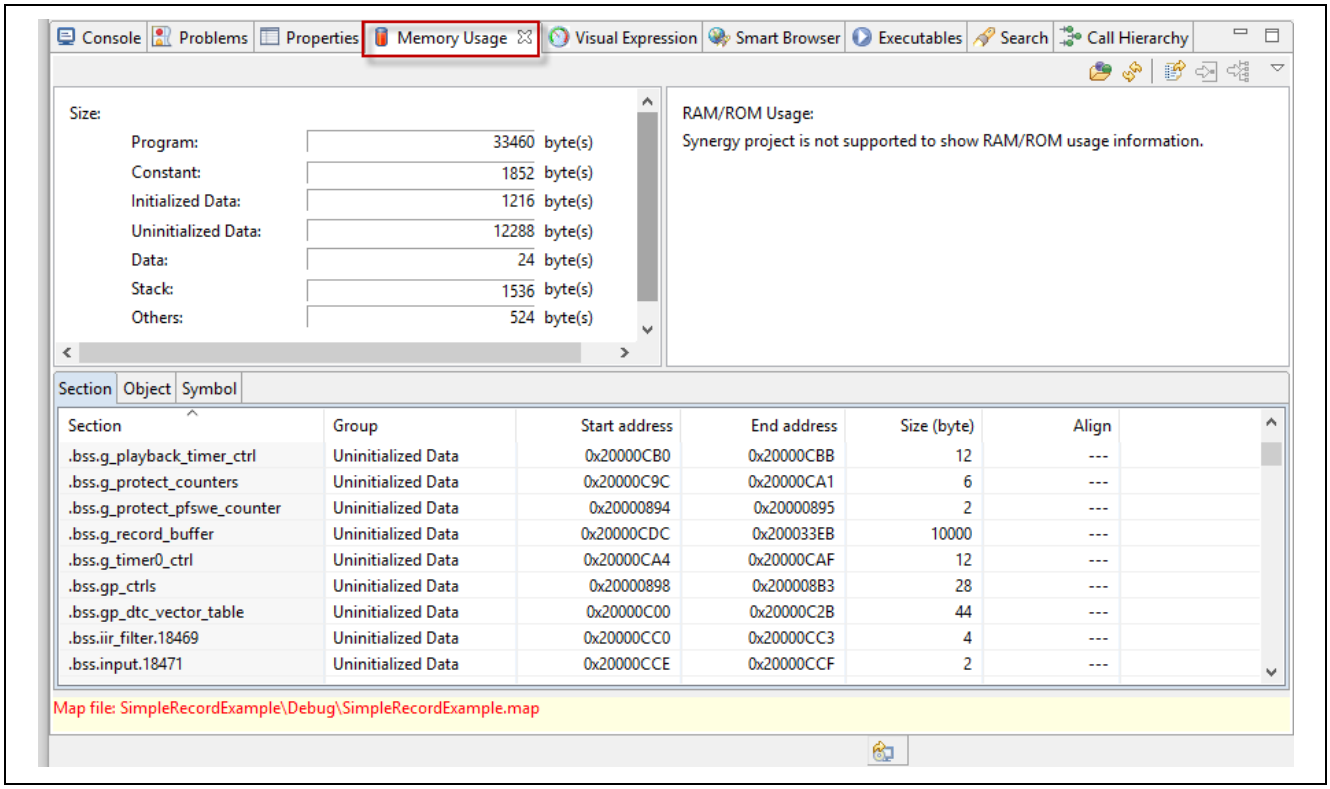

Figure 16. Memory Usage tab

Note: e2studio v6.2.0 is currently not showing any information related to memory in **Memory Usage** tab. This issue will be fixed in a later version.

As you can see in Figure 16, the **Memory Usage** tab contains three panes. The top left pane displays a summary of the memory used by the major sections in the map file. The bottom pane allows you to examine by Section, Object, or Symbol name and sort, based on Group, Start Address, End Address, and Size. The upper right pane, detailing RAM/ROM usage of Synergy projects, is not currently supported in e² studio.

Some knowledge of map file nomenclature is needed to take full advantage of the **Memory Usage** tab but there are a few tricks that can help you use the tool more effectively. As an example, this application defines a static global variable `g_record_buffer` to hold the recorded data. If you look for this symbol, you may not see it since it has a `.bss` prefix. If you pick the **Symbol** tab and double click on any symbol displayed, it displays the `.map` file in e² studio. From here, you can type CTRL+f to bring up the **Find** dialog. Type in the name of the symbol you want to find (for example, `g_record_buffer`) and you will see the actual symbol name in the `.map` file is `.bss.g_record_buffer` as shown in Figure 17.

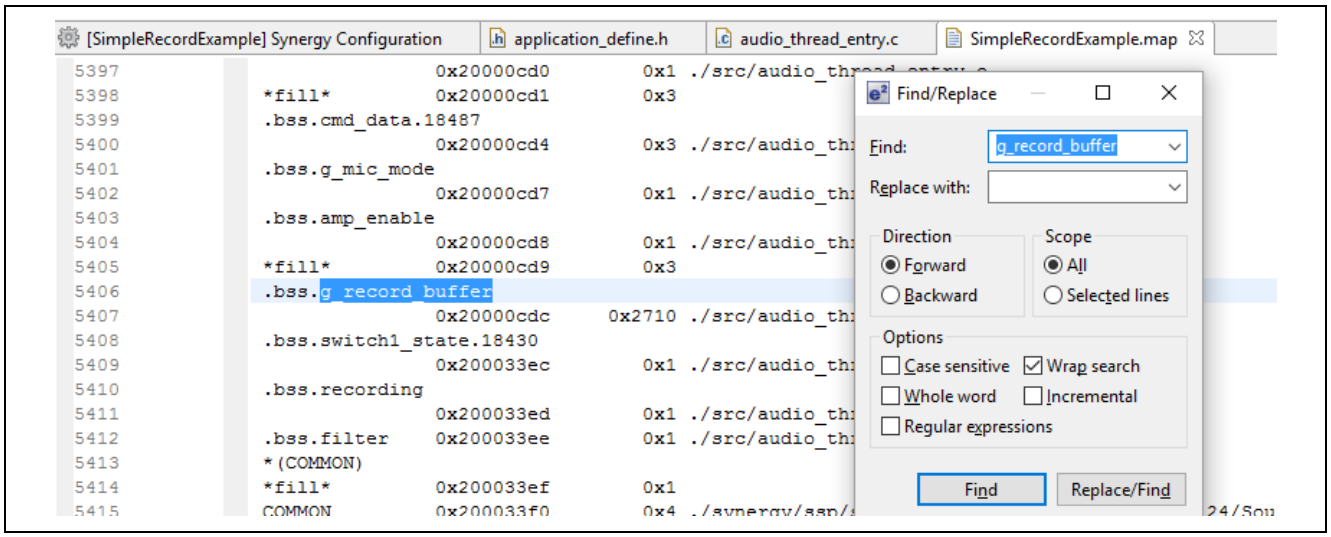

Figure 17. Finding symbols in the .map file

If you do not see the **Memory Usage** tab displayed in your current e² studio setup, you can open it by using the top toolbar and going to **Window**>**Show View**>**Other**, then selecting **Debug** and choosing **Memory Usage** as shown in Figure 18.

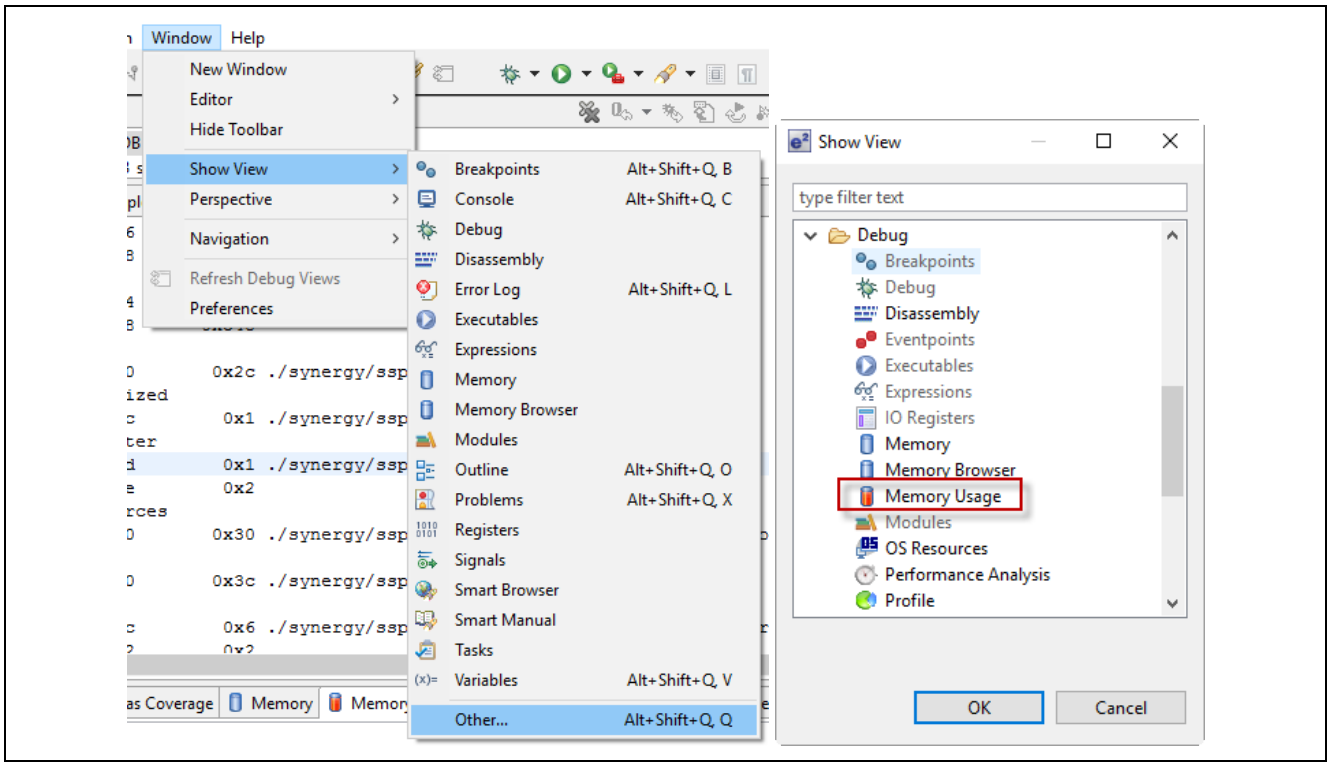

Figure 18. Opening the memory usage tab

5. Importing and Building the Project

Follow the procedure in *Renesas Synergy™ Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide) to import the project into the e² studio ISDE, and to build and debug the project. For the debug configuration, select **SimpleRecordExample Debug** (under **Renesas GDB Hardware Debugging**).

6. 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 applications from the Renesas Gallery site.

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	Nov.30.16	—	Initial version
1.10	Oct.31.16	—	Updated to use e^2 studio v5.2.1.010 and SSP v1.2.0-b.1
1.11	Nov.30.16	—	Added support for IAR EW
1.12	Feb.23.17	—	Updated for SSP v1.2.0
1.13	Aug.16.17	—	Updated for SSP v1.3.0
1.14	Sep.27.17	—	Required resources of SSP version changed
1.15	Mar.13.18	—	Updated for SSP v1.4.0
1.16	Mar.15.19	—	Updated for SSP v1.6.0. Added jumper configuration.

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

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/.