Renesas Synergy™ Platform

Audio Record ADC Framework Module Guide

Introduction

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

The Audio Record ADC Framework module is a high-level API for audio recording applications and is implemented on `sf_adc_periodic`. The Audio Record ADC Framework module uses the ADC, GPT, and DTC peripherals on the Synergy MCU. A user-defined callback can be created to indicate that the sample count has been completed.

Contents

1. Audio Record ADC Framework Module Features

- Records data in 8-bit or 12-bit PCM
- Uses ADC Periodic Framework to simplify configuration and integration
- Uses ThreadX® object, like mutex, to protect hardware from improper access
- APIs for high-level functions simplify coding:
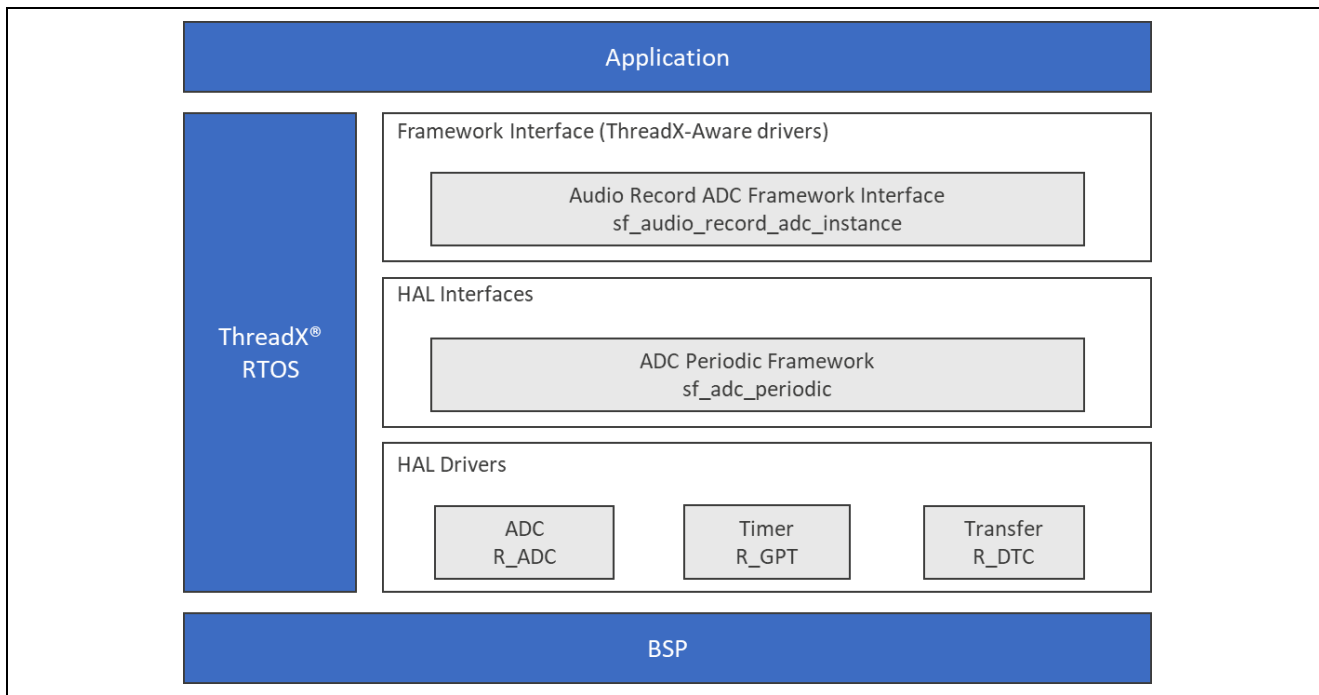 —— Open, start
  —— Stop, infoGet
  —— Close

Figure 1.   Audio Record ADC Framework Module Organization, Options and Stack Implementations

2. Audio Record ADC Framework Module APIs Overview

The Audio Record ADC Framework module defines APIs for opening, closing, starting, and stopping the record process. A complete list of the available APIs, an example API call, and a short description of each can be found in the following table. A table of status return values follows.

Table 1.   Audio Record ADC Framework Module API Summary

Function Name	Example API Call and Description
.open	g_sf_audio_record_adc.p_api->open(g_sf_audio_record_adc.p_ctrl, g_sf_audio_record_adc.p_cfg); Initialize the module.
.start	g_sf_audio_record_adc.p_api->start(g_sf_audio_record_adc.p_ctrl); Start audio recording.
.stop	g_sf_audio_record_adc.p_api->stop(g_sf_audio_record_adc.p_ctrl); Stop audio recording.
.infoGet	g_sf_audio_record_adc.p_api->infoGet(g_sf_audio_record_adc.p_api.p_ctrl); Get the channel information (mono or Stereo).
.close	g_sf_audio_record_adc.p_api->close(g_sf_audio_record_adc.p_ctrl); Close the module.

.versionGet	`g_sf_audio_record_adc.p_api->versionGet(&version);` Retrieve the API version with the version pointer.

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

Table 2. Status Return Values

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value
SSP_ERR_IN_USE	The adc periodic framework mutex may be unavailable for the unit requested. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred. This is typically a failure to create/use a mutex or to create an internal thread.
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	The parameter `p_ctrl` or `p_sample` is NULL
SSP_ERR_UNSUPPORTED	This function is not supported by the HAL driver (`p_ctrl > p_api > close` is NULL)

Note: Lower level drivers may return Common Error Codes. Refer to the *SSP User's Manual* API References for the associated module for a definition of all relevant status return values.

3. Audio Record ADC Framework Module Operational Overview

The Audio Record ADC Framework Module samples audio analog data using the ADC Periodic Framework and the data samples captured are stored in the user buffer. The data is made available for further processing as needed by the application. The Audio Record ADC Framework Module has a configuration parameter that is initialized during the framework initialization, which also initializes the underlying ADC periodic framework for data capture.

The data is captured in a user defined buffer and this is done in the callback function as illustrated as follows, assuming that the name of the callback has been configured to be `sf_audio_record_user_callback`:

```
uint16_t * audio_record_buffer;
void sf_audio_record_user_callback (sf_audio_record_callback_args_t *p_args)
{
audio_record_buffer = ((uint16_t *)g_sf_audio_record_adc.p_cfg->
                       p_capture_data_buffer + (p_args->buffer_index/2));
```

3.1 Audio Record ADC Framework Module Operational Notes

The Audio Record ADC Framework Module configuration data can specify the length of the data buffer, data width, sampling rate, and the number of sampling iterations.

3.2 Audio Record ADC Framework Module Limitations

- Currently the Audio Record ADC only supports the ADC Periodic Framework as the lower level and thus recording via I2S is not supported with the framework.
- The framework currently supports recording 8-bit or 12-bit PCM data.
- Refer to the most recent *SSP Release Note* for any additional operational limitations for this module.

4.   Including the Audio Record ADC Framework Module in an Application

This section describes how to include the Audio Record ADC Framework Module in an application using the SSP Configurator.

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

To add the Audio Record ADC Framework Module to an application, simply add it to a thread using the Stacks Selection Sequence given in the following table. (The default name for the Audio Record ADC Framework Module is `g_audio_record_adc0`.) This name can be changed in the associated Properties window.

Table 3.   Audio Record ADC Framework Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_audio_record_adc0 Audio Record ADC Framework on `sf_audio_record_adc`	Threads	New Stack> Framework> Audio> Audio Record ADC Framework on `sf_audio_record_adc`

When the Audio Record ADC on `sf_audio_record_adc` is added to the Thread Stack as shown in the following figure, the configurator automatically adds any needed lower level drivers.
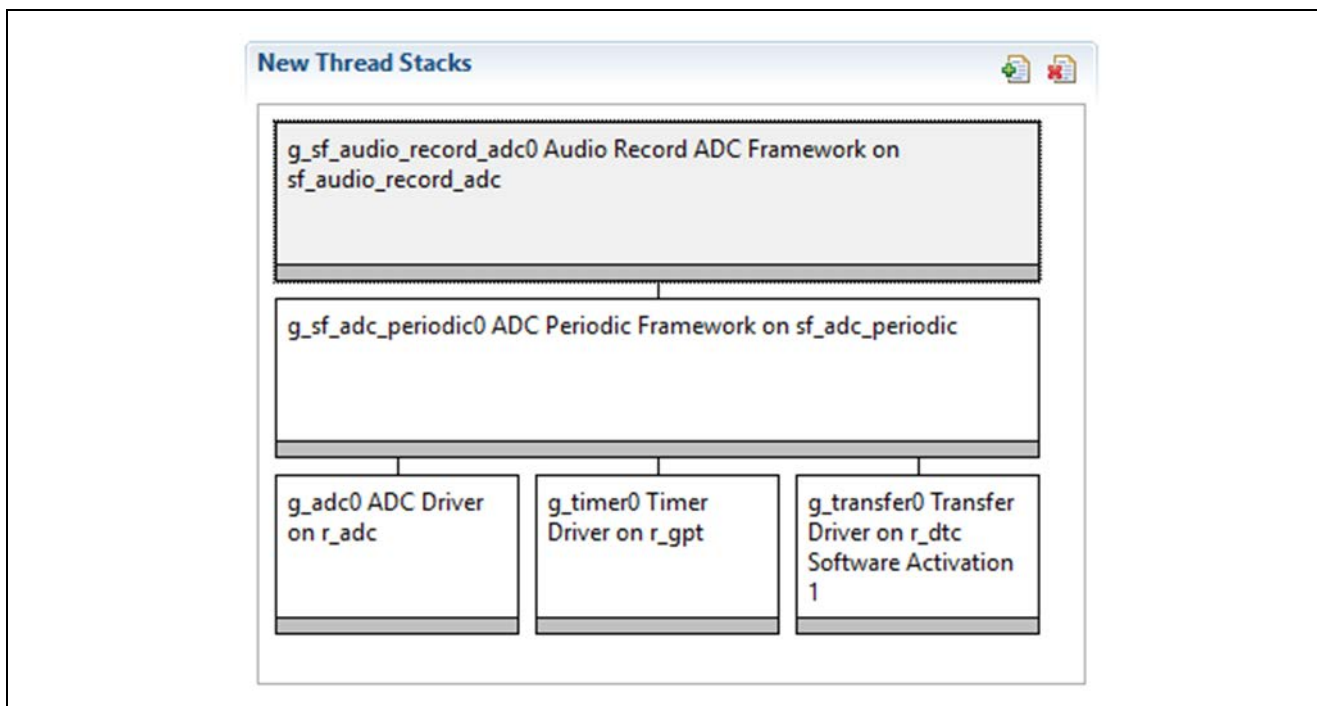
Figure 2.   Audio Record ADC Framework Module Stack

5.   Configuring the Audio Record ADC Framework Module

The Audio Record ADC Framework Module must be configured by you for the desired operation. The SSP configuration window will automatically identify, by highlighting the block in red, any required configuration selections, such as Interrupts or Operating modes, which must be configured for lower level modules, for successful operation. Only those properties that can be changed without causing conflicts are available for modification. Other properties are **locked** and not available for changes and are identified with a **lock icon** for the **locked** property in the Property window in the ISDE. This approach simplifies the configuration process and makes it much less error prone than previous manual approaches to configuration. The

available configuration settings and defaults for all the user accessible properties are given in the properties tab within the SSP Configurator, and are shown in the following tables for easy reference.

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

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

Table 4. Configuration Settings for Audio Record ADC Framework Module on sf_audio_record_adc

ISDE Property	Value	Description
Parameter Checking	Enabled, Disabled, BSP Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_sf_audio_record_adc0	Module name
Name of the data-buffer to store samples	P_capture_data_buffer	Name of the data buffer to store samples
Length of the data-buffer	2048	Length of the buffer to which data is to be stored
Audio Record Data Size	8-bit, 16-bit Default: 8-bit	Data width
Sampling Rate in HZ	8000	Sampling rate
Number of sampling iterations	Default: 256	Number of samples captured per iteration
Callback	g_audio_redord_framework_ user_callback	User function that will be called once the number of sampling iterations of data has been buffered.

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

In some cases, settings other than the defaults for stack modules can be desirable. For example, it might be useful to select different buffer sizes or sample rates. The configurable properties for the lower level stack modules are given in the following sections for completeness and as a reference.

Note: Most of the property settings for modules are fairly intuitive and usually can be determined by inspection of the associated properties window from the SSP Configurator.

5.1 Configuration Settings for the Audio Record ADC Framework Module Low Level Drivers

Typically, only a small number of settings must be modified from the default for lower level drivers and these are indicated with red text in the Thread Stack block. Notice that some of the configuration properties must be set to a certain value for proper framework operation and will be locked to prevent user modification. The following tables identify all the settings within the properties section for the module.

Table 5. Configuration Settings for ADC Periodic Framework on sf_adc_periodic

ISDE Property	Value	Description
Parameter Checking	Enabled, Disabled, BSP Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_sf_adc_periodic0	Module name

ISDE Property	Value	Description
Name of the data-buffer to store samples	g_user_buffer	Name of the data buffer to store samples
Length of the data-buffer	2048	Length of the buffer to which data is to be stored
Number of sampling iterations	256	Number of samples captured per iteration
GPT Timer channel used to trigger scan	Channel 0-12	Channel number
Callback	NULL	User function that will be called once the number of sampling iterations of data has been buffered

Table 6. Configuration Settings for the ADC HAL Module on r_adc

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: Enabled	If selected code for parameter checking is included in the build
Name	g_adc0	Module name
Unit	0, 1 (S7G2 Only) Default: 0	Specify the ADC Unit to be used. The S7G2 has two units; 0 and 1.
Resolution	14-Bit (S3A7/S124 Only), 12-Bit, 10-Bit (S7G2 Only), 8-Bit (S7G2 Only) Default: 8-Bit (S7G2 Only)	Specify the conversion resolution for this unit.
Alignment	Right, Left Default: Right	Specify the conversion result alignment.
Clear after read	Off, On Default: On	Specify if the result register must be automatically cleared after the conversion result is read. Note: If this is enabled, then watching the result register using a debugger always results in a 0.
Mode	Single Scan	The ADC Framework preconfigures and locks this field.
Channels 0-6	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 7-10 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 11-15 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.

ISDE Property	Value	Description
Channels 16-20	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channel 21 (Unit 0 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channel 22 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 23-27 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Temperature Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	Temperature sensor use selection for Channel Scan Mask
Voltage Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	Voltage sensor use selection for Channel Scan Mask
Scan Mask Group B	Use #define ADC_MASK_xxx which are defined in r_adc.h. Use (ADC_MASK_xxx|ADC_MASK_xxx) for multiple channels.	Do not use with ADC Framework since the mode is locked to Single Scan Mode.
Normal/Group A Trigger	ELC Event	The ADC Framework preconfigures and locks this field.
Group B Trigger (Valid Only in Group Scan Mode)	ELC Event (The only valid trigger for either group in Group Scan Mode)	The ADC Framework preconfigures and locks this field.
Group Priority (Valid only in Group Scan Mode)	Group A cannot interrupt Group B, Group A can interrupt Group B; Group B scan restarts at next trigger, Group A can interrupt Group B; Group B scan restarts immediately, Group A can interrupt Group B; Group B scan restarts immediately and scans continuously Default: Group A cannot interrupt Group B	Do not use with ADC Framework since the mode is locked to Single Scan Mode.

ISDE Property	Value	Description
Add/Average Count	Disabled, Add two samples, Add three samples, Add four samples, Add sixteen samples, Average two samples, Average four samples Default: Disabled	Specify if addition or averaging needs to be done for any of the channels in this unit. The actual channels are specified by using a channel mask adc_channel_cfg_t::add_mask.
Channels 0-27	Disabled, Enabled Default: Disabled	This field is valid only if adc_cfg_t::add_average_count is enabled. This field determines what channel results are to be averaged or summed.
Temperature Sensor	Disabled, Enabled Default: Disabled	Temperature sensor use selection for Addition/Averaging Mask
Voltage Sensor	Disabled, Enabled Default: Disabled	Voltage sensor use selection for Addition/Averaging Mask
Channels 0-2	Disabled, Enabled Default: Disabled	Determines which of channels 0, 1, and 2 are using the updated sample-and-hold states value specified in adc_channel_cfg_t::sample_hold_states. This field must only be set if it is desired to modify the default sample and hold count value for channels 0, 1, and 2.
Sample Hold States (Applies only to the 3 channels selected above)	24	If selected code for parameter checking is included in the build
Callback	NULL	Module name
Scan End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	Specify the ADC Unit to be used. The S7G2 has two units; 0 and 1.
Scan End Group B Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	Specify the conversion resolution for this unit.

Table 7. Configuration for the GPT HAL Module on r_gpt

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_timer0	Module name.
Channel	0	The ADC Framework preconfigures and locks this field based on channel selected in the ADC Framework.

ISDE Property	Value	Description
Mode	Periodic	The ADC Framework preconfigures and locks this field.
Period Value	10	Configure timer period to trigger ADC scans.
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz Default: Milliseconds	Configure units of the timer period set above.
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	Duty cycle unit selection
Auto Start	False	The ADC Framework preconfigures and locks this field.
GTIOCA Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
GTIOCB Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
Callback	NULL	The ADC Framework preconfigures and locks this field.
Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	Interrupt priority selection

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

5.2   Audio Record ADC Framework Module Clock Configuration

The ADC peripheral module uses PCLKC as its clock source.

5.3   Audio Record ADC Framework Module Pin Configuration

The ADC peripheral module uses pins on the MCU to communicate to external devices. I/O pins must be selected and configured as required by the external device. ADC pins must be configured as analog pins. The first table that follows illustrates the method for selecting the pins within the SSP configuration window and the next table illustrates an example selection for the pins.

Note: For some peripherals, the Operation Mode selection mode determines what peripheral signals are available and therefore what MCU pins are required.

Table 8. Pin Selection Sequence for ADC

Resource	ISDE Tab	Pin selection Sequence
ADC	Pins	Select Peripherals > Analog:ADC > ADC0

Note: The above selection sequence assumes ADC0 is the desired hardware target for the driver.

Table 9. Pin Configuration Settings for ADC

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Custom Default: Custom	Select operating mode for ADC
ADTRG	None, P407, P102 Default: None	ADTRG Pin
AN00-19	None, Pnnn, Pmmm Default: None	Analog input pins
PGAVSS0	None, P003 Default: None	PGAVSS pin

Note: The preceding example settings are for a project using the S7G2 Synergy MCU Group and the SK-
S7G2 Kit. Other Synergy Kits and other Synergy MCUs may have different available pin configuration
settings.

6.   Using the Audio Record ADC Framework Module in an Application

The typical steps in using the Audio Record ADC Framework module in an application are:
1. Open the module using the `open` API.
2. Start the recording using the `start` API.
3. Buffer data in a user buffer with the callback.
4. Operate on data as needed.
5. Close the module using the `close` API.

The above common steps are illustrated in a typical operational flow diagram in the following figure.
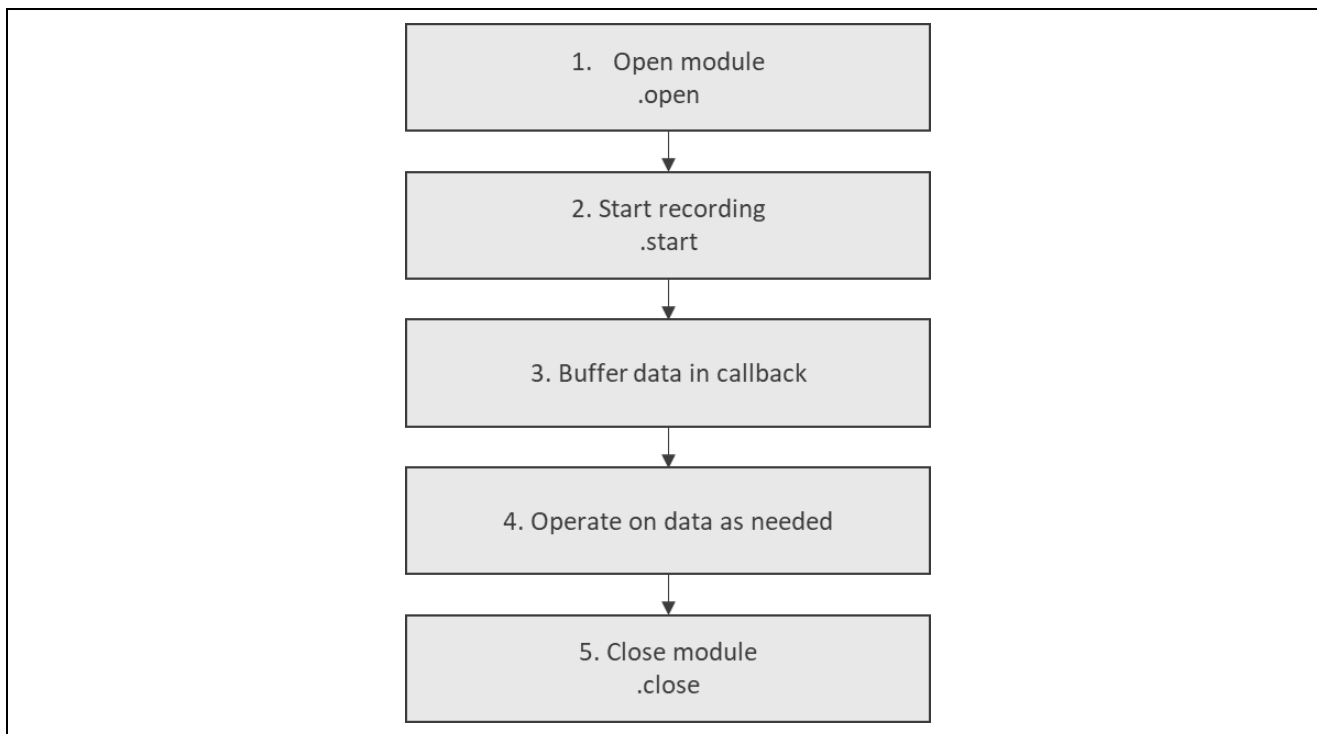
Figure 3.   Flow Diagram of a Typical Audio Record ADC Framework Module Application

7. Audio Record ADC Framework Module Application Project

The Application Project demonstrates the above steps in an example application. You may want to import and open the Application Project within ISDE and view the configuration settings for the Audio Record ADC Framework Module. You can also read over the code in `audio_record_thread_entry.c`, which is used to illustrate the Audio Record ADC Framework Module APIs in a complete design.

The Application Project demonstrates the typical use of the Audio Record ADC Framework Module APIs. The application contains two threads, which are named Audio Playback Thread and Audio Record Thread. The Audio Playback Thread is responsible for providing audio data. This will not be described in detail in this document; see the *Audio Playback Framework Module Guide* for details. The Audio Record Thread is responsible for recording the audio data on the USB mass storage device. Once the thread is started, the 3 LEDs are switched off, IRQ driver opens, and an infinite loop is started. In the loop, the program checks if the S4 button is pressed – the button starts and stops the recording.

Once the recording is started, the program deletes the **audio.wav** file on the USB mass storage device, which should be inserted in USB_H port. Then the new file is created, opened, and a WAV file header is written in the file. The red LED is switched on to indicate that the recording is on. If the recording is stopped, the LED is switched off, the data length is written to the appropriate bytes of the header file and the file itself is closed.

In the loop, the program also checks if any new data appeared in the buffer. If so, it is saved in the file on the USB mass storage device.

Table 10. Software and Hardware Resources Used by the Application Project

e² studio ISDE	v7.3.0 or greater	Integrated Solution Development Environment
SSP	v1.6.0 or greater	Synergy Software Platform
IAR EW for Renesas Synergy	v8.23.3 or greater	IAR Embedded Workbench for Renesas Synergy
SSC	v7.3.0 or greater	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit
USB mass storage device		A mass storage device to save the audio file on
Female to female jumper cable		To connect ADC to DAC pins on the SK-S7G2 board
Headphones or speakers with 3.5 mm audio connector		Optional – used to play the audio sample

A simple flow diagram of the Application project is given in the following figure.
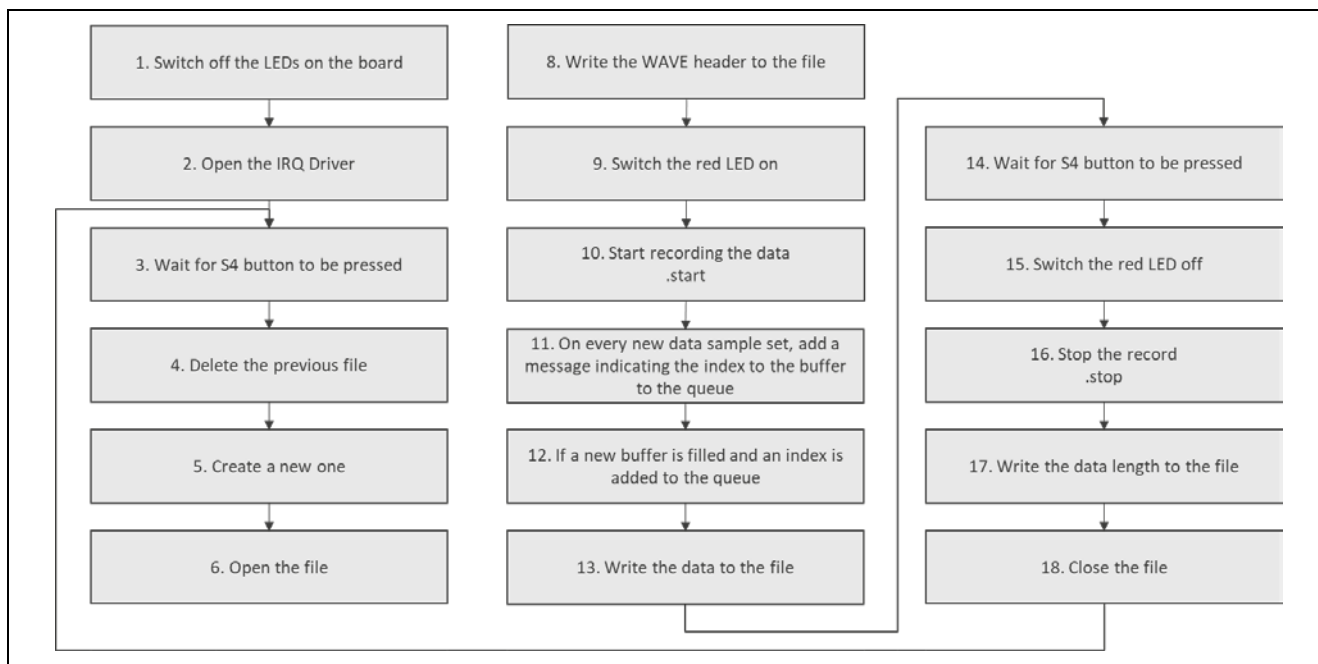
Figure 4. Framework Module Application Project Flow Diagram

The `audio_record.c` file is located in the project once it has been imported into the ISDE. You can open this file, within the ISDE, and follow along with the following description, to help identify key uses of APIs.

Use the following paragraph as a starting point for a specific description. It provides a guide for flow and level of detail. Make sure that the description matches the flow diagram and the code snippets above; ideally the main block of code map easily to the flow diagram. Refer to code snippets where they are used.

The first section of `audio_record.c` has the header files which reference the Audio Record ADC Framework instance structure and other modules. After that, a macro `APP_ERR_TRAP` is defined and is used to handle errors. Then, a file buffer is declared.

The next section is an entry of the main thread function. In this function, the LEDs are switched off and the IRQ driver is opened. After that, the infinite loop is started. Inside that loop, the program checks if you pressed the S4 button. Then the program starts or stops the recording. In the first situation, the **audio.wav** file is deleted and a new one is created and opened. In the file, the wav header that defines the properties of the file (like bitrate, number of channels and so on) is saved. After that, the red LED is switched on to indicate to that the record is on. In case the S4 button is pressed again, the program stops recording, switches off the red LED and saves the file length to the WAVE file header. Then the file is closed. In case the new data is recorded by the framework, which is indicated by the message queue, the data is saved in the file.

The last section includes the interrupt callbacks. The IRQ Driver callback function indicates the S4 button being pressed. The Audio Record ADC Framework is triggered when a set of audio data is saved in the buffer. The data buffers store 16-bit unsigned integers even though the desired ADC resolution is 8 bits; the data is copied to 8 bits unsigned integer buffer that will then be used to write the audio record in the file. The message queue is populated with an integer number indicating the buffer index that begins the latest recorded data.

Keep in mind that the project requires usage of USB Mass Storage device connected to USB_H port and connecting the P000 and P014 ports with the jumper cable.

A few key properties are configured in this Application Project to support the required operations and the physical properties of the target board and MCU. Following are the properties with the values set for this specific project. You can also open the Application Project and view these settings in the property window as a hands-on exercise.

Tables of settings required to configure the AP for the target MCU and Kit are provided. Use the existing table for an example of standard format and content.

Table 11.   Audio Record Framework ADC Threads List and Parameters

Symbol	Name	Stack Size (bytes)	Priority
audio_record_thread	Audio Record Thread	2048	3
audio_playback_thread	Audio Playback Thread	2048	6

Table 12.   Audio Record Thread Objects

Object type	Name	Symbol	Properties
Semaphore	Audio Record Semaphore	g_audio_record_semaphore	Initial count: 0
Queue	Buffer Index Queue	g_buffer_index_queue	Message Size (Words): 1 Queue Size (Bytes): 64

Table 13.   Audio Playback Thread Objects

Object type	Name	Symbol	Properties
Semaphore	Audio Playback Semaphore	g_audio_playback_semaphore	Initial count: 0

Table 14.   g_sf_audio_record_adc0 Audio Record ADC Framework on sf_audio_record_adc Module Configuration Settings

ISDE Property	Value Set
Length of the data-buffer	16385
Sampling Rate	16000
Number of sampling iterations	1024

Table 15.   g_adc0 ADC Driver on r_adc Module Configuration Settings for the Application Project

ISDE Property	Value Set
Channel 0	Use in Normal/Group A
Scan End Interrupt Priority	Priority 13 (CM4: valid, CM0+: invalid)

Table 16.   g_timer0 Timer Driver on r_gpt Module Configuration Settings

ISDE Property	Value Set
Period Value	16000
Period Unit	Hertz
Interrupt Priority	Priority 10 (CM4: valid, CM0+: invalid)

This project requires additional modules to be added to the Audio Record Thread. These are IRQ Driver on `r_icu` and FileX® on USB Mass Storage. You should add USBX Port HCD to the USBX Host Configuration `g_ux_host_0` and choose new USBX Port HCD on `sf_el_ux` for USBHS. The proper configuration of those modules is shown in the following tables.

Table 17.   g_external irq0 External IRQ Driver on r_icu Module Configuration Settings

ISDE Property	Value Set
Channel	11
Digital Filtering	Enabled
Callback	g_irq_callback
Interrupt Priority	Priority 4 (CM4: valid, CM0+: invalid)

Table 18.   g_fx_media0 FileX on USB Mass Storage Module Configuration Settings

ISDE Property	Value Set
Auto Media Initialization	Enabled

Table 19.   g_sf_el_ux_hcd_hs_0 USBX Port HCD on fs_el_ux for HSBHS Module Configuration Settings

ISDE Property	Value Set
High Speed Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)

Table 20.   USBX on ux Module Configuration Settings

ISDE Property	Value Set
USBX Pool Memory Size	65536

The elements in the second thread, Audio Playback Thread, must be added and configured. To do that, you should add Audio Playback Framework on `sf_audio_playback` to this thread. The new Audio Playback Hardware Framework shared on `sf_audio_playback_hw_dac` should be added under Add Audio

Playback Hardware. Next, add a new Transfer Driver on `r_dmac` and a new Timer Driver on `r_gpt`. The proper configuration of those parameters is shown in the following tables.

Table 21. g_sf_audio_playback0 Audio Playback Framework on sf_audio_playback Module Configuration Settings

ISDE Property	Value Set
Buffer Size Bytes	8192
Callback	g_audio_playback_callback

Table 22. g_sf_audio_playback_common0 Audio Playback Framework Shared on sf_audio_playback Module Configuration Settings

ISDE Property	Value Set
Audio Message Queue Name	audio_playback_thread_message_queue

Table 23. g_sf_audio_playback_hw0 Audio Playback Hardware Framework Shared on sf_audio_playback_hw_dac Module Configuration Settings

ISDE Property	Value Set
DMAC Support	Enabled

Table 24. g_transfer1 Transfer Driver on r_dmac Module Configuration Settings

ISDE Property	Value Set
Channel	1
Destination Pointer	&R_DAC->DADRn[0]
Activation Source	Event GP1 COUNTER OVERFLOW
Interrupt Priority	Priority 9 (CM4: valid, CM0+: invalid)

Table 25. g_timer1 Timer Driver on r_gpt Module Configuration Settings

ISDE Property	Value Set
Channel	1
Period Value	16000
Interrupt Priority	Priority 9 (CM4: valid, CM0+: invalid)

The Audio Playback Thread must be added as a message subscriber to the Audio Playback Event Class. It can be done in the **Messaging** tab in the SSP Configurator.

8. Customizing the Audio Record ADC Framework Module for a Target Application

Provide a description of common configuration settings that a developer might typically change for a target application. Use the following paragraph as a starting point.

Some configuration settings will normally be changed by the developer from those shown in the Application Project. For example, you can easily change the length of the data buffer, sampling rate or audio data resolution. You can also change the ADC port pins to select the desired analog input. This can be done using the **Pins** tab in the configurator. You can also change the timer that triggers the ADC scanning.

9. Running the Audio Record ADC Framework Module Application Project

To run the Audio Record ADC Framework Module Application Project and to see it executing on a target kit, you can simply import it into your ISDE, compile and run debug. Refer to the *Renesas Synergy™ Project Import Guide* (11an0023eu0121-synergy-ssp-import-guide.pdf, included in this package) for instructions on importing the project into e² studio or IAR EW for Synergy, and build and run the application.

An optional description for building the project from scratch is provided here. Use this if you believe it will be helpful to the reader. It is recommended for simpler projects.

To implement the Audio Record ADC Framework Module Application in a new project, follow the steps below for defining, configuring, auto-generating files, adding code, compiling, and debugging on the target kit. Following these steps is a hands-on approach that can help make the development process with SSP more practical.

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

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

1. Create a new Renesas Synergy project for the S7G2-SK called `AUDIO_RECORD_ADC_MG_AP`.
2. Select the **Threads** tab.
3. Add two threads to the application; their parameters are presented in the Table 11. Add proper objects to the threads, according to Table 12 and Table 13.
4. To the Audio Record Thread, add Audio Record ADC Framework on `sf_audio_record_adc`, External IRQ Driver on `r_icu,` and FileX on the USB Mass Storage. To USBX Host Configuration `g_ux_host_0,` add USB Port HCD on `fs_el_ux` for USBHS. Adjust the parameters of the modules shown in Table 14 to Table 20.
5. To the Audio Playback Thread add Audio Playback Framework on `sf_audio_playback`. Under `g_sf_audio_playback_common0`, Audio Playback Framework shared on `sf_audio_playback`, add new Audio Playback Hardware Framework shared on `sf_audio_playback_hw_dac`. To this module, you should add a new Transfer Driver on `r_dmac` and a new Timer Driver on `r_gpt`. Table 21 to Table 25 are shown in the parameters of those modules.
6. Add the Audio Playback Thread as a message subscriber to the Audio Playback Event Class in the **Messaging** tab in SSP configuration.
7. Click on the **Generate Project Content** button.
8. Copy the files `audio_playback.c`, `audio_record.c` and `audio_data.c` to the project `src` directory. Modify the code in the `audio_playback_thread_entry.c` file so the `audio_playback()` function is called. Also modify the `audio_record_thread_entry.c` file so the function `audio_record()` is called.
9. Connect to the host PC via a micro USB cable to J19 on the SK-S7G2.
10. Connect P000 and P014 with the jumper cable.
11. Connect as USB Mass Storage device to the USB_H port J6. If there already exists a file named **audio.wav** in the main directory of the device, it will be overwritten.
12. Connect headphones to the audio J16 connector, if you wish to hear the audio data.
13. Start to debug the application.
14. When 3 LEDs are switched off, press the S4 button to start recording the audio data. After a while, press the S4 button again. A red LED will indicate the recording.
15. The recorded audio is saved in the USB Mass Storage device in its main directory.
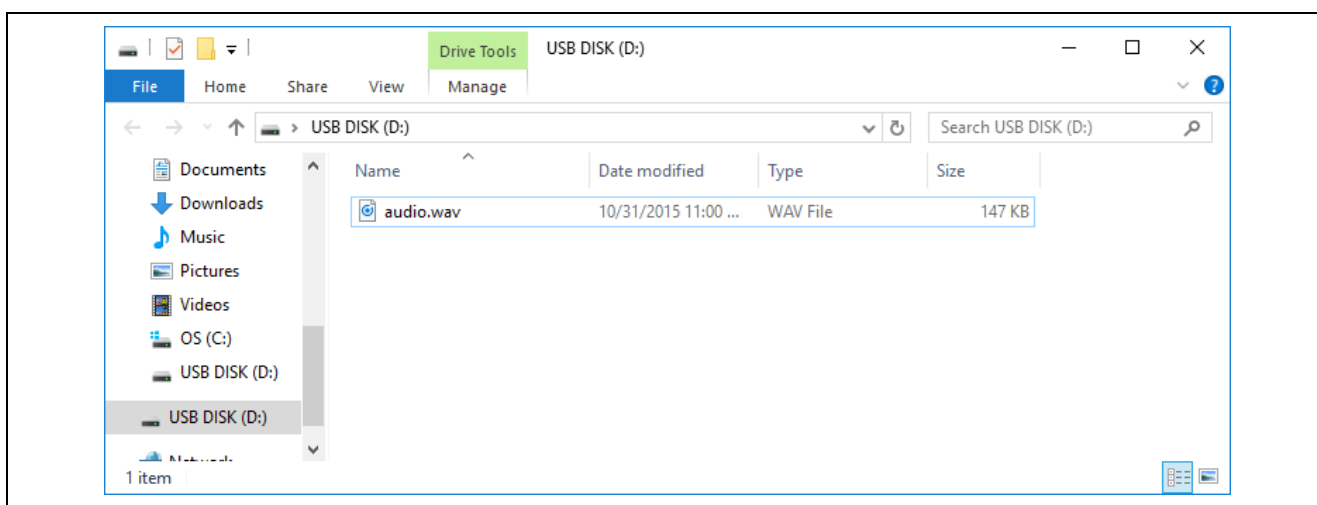
Figure 5.   Example Output from Audio Record ADC Framework Application Project

10. Audio Record ADC Framework Module Conclusion

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

11. Audio Record ADC Module Next Steps

After you have mastered a simple Audio Record ADC Framework module project, you may want to review a more complex example. You may modify the ADC Driver so multiple channels are used to record the audio data to provide stereo or multi-channel audio recording.

12. Audio Record ADC Module Reference Information

SSP User's Manual: Available in html format in the SSP distribution package and as a pdf from the Renesas Synergy Gallery. Links to all the most up-to-date `sf_adc_periodic` module resource materials are available on:

https://en-us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Base/sf_adc_periodic_Module_Guide_Resources

Website and Support

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

Synergy Software   www.renesas.com/synergy/software
    Synergy Software Package   www.renesas.com/synergy/ssp
    Software add-ons   www.renesas.com/synergy/addons
    Software glossary   www.renesas.com/synergy/softwareglossary
    Development tools   www.renesas.com/synergy/tools

Synergy Hardware   www.renesas.com/synergy/hardware
    Microcontrollers   www.renesas.com/synergy/mcus
    MCU glossary   www.renesas.com/synergy/mcuglossary
    Parametric search   www.renesas.com/synergy/parametric
    Kits   www.renesas.com/synergy/kits

Synergy Solutions Gallery   www.renesas.com/synergy/solutionsgallery
    Partner projects   www.renesas.com/synergy/partnerprojects
    Application projects   www.renesas.com/synergy/applicationprojects

Self-service support resources:
    Documentation   www.renesas.com/synergy/docs
    Knowledgebase   www.renesas.com/synergy/knowledgebase
    Forums   www.renesas.com/synergy/forum
    Training   www.renesas.com/synergy/training
    Videos   www.renesas.com/synergy/videos
    Chat and web ticket   www.renesas.com/synergy/resourcelibrary

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep.12.18	-	Initial release
1.10	Nov.14.18	-	Updated for SSP v1.5.0
1.20	Apr.29.19	-	Updated for SSP v1.6.0

Notice

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/.