Renesas Synergy™ Platform

Audio Playback I²S 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 an efficient starting point. References to more detailed API descriptions and suggestions of other application projects that illustrate more advanced uses of the module are included in this document and should be valuable resources for creating more complex designs.

The Audio Playback Framework handles synchronization to play mono 16-bit Pulse-code modulation (PCM) samples. It uses a hardware port DAC Audio Playback Framework or I²S Audio Playback Framework for hardware access.

This module guide focuses on the I²S Audio Playback Framework hardware port. The module, Audio Playback I²S Framework, is a high-level API for Audio Playback application and is implemented on `sf_audio_playback_hw_I2S`. It handles the synchronization needed to play 16-bit PCM samples. The Audio Playback Framework uses the I²S, Timer (AGT or GPT), and Data Transfer (DMA or DTC) peripherals on a Renesas Synergy™ MCU. A user-defined callback can be created to respond to additional data needs.

Contents

1. Features

The Audio Playback Hardware I²S Framework module supports the following features:

- Plays long buffers by splitting the data into manageable chunks.
- Repeats playback until a ThreadX® timeout (for repeated audio like sine wave tones or looped background music).
- Requests next data using callback after last buffer playback begins.
- Software volume control.
- Pauses and resumes functions.
- Basic mixing for multiple streams.

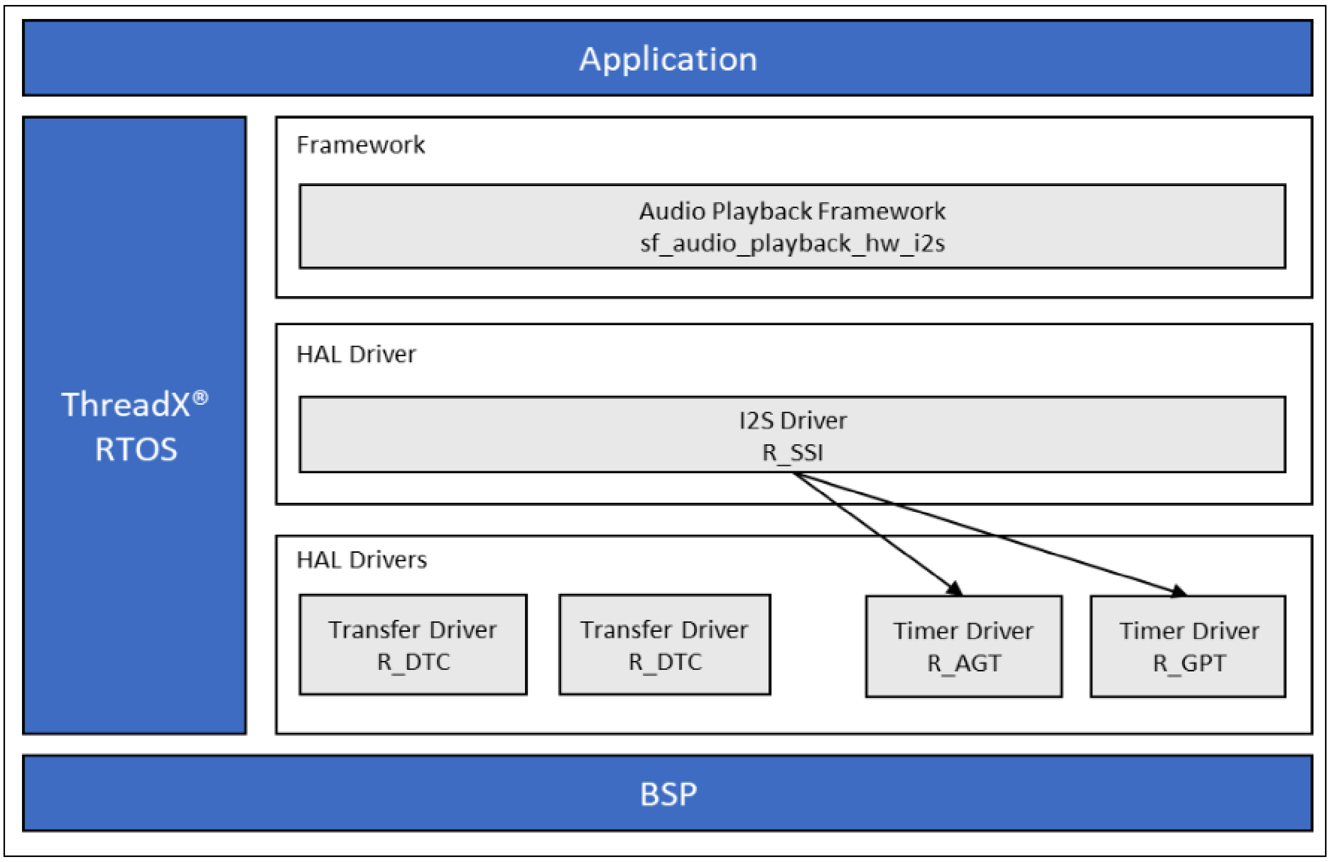


Figure 1. Audio Playback I²S Framework Module Block Diagram

2. Overview

The APIs from the Audio Playback I²S Framework are abstracted by the APIs for the Audio Playback Framework. The Audio Playback I²S Framework module defines APIs for operations such as opening, starting, playing, and stopping.

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 the API summary table.

Table 1. Audio Playback Framework Module API Summary

Function Name	Example API Call and Description
.open	`g_sf_audio_playback0.p_api->open(g_sf_audio_playback0.p_ctrl, g_sf_audio_playback0.p_cfg);` Configure the audio framework by creating a thread for audio playback and configuring HAL layer drivers used.
.start	`g_sf_audio_playback0.p_api->start(g_sf_audio_playback0.p_ctrl, &p_data, 100);` Play audio.
.stop	`g_sf_audio_playback0.p_api->stop(g_sf_audio_playback0.p_ctrl);` Stop audio playback.
.pause	`g_sf_audio_playback0.p_api->pause(g_sf_audio_playback0.p_ctrl);` Pause audio playback.
.resume	`g_sf_audio_playback0.p_api->resume(g_sf_audio_playback0.p_ctrl);` Resume audio playback.
.volumeSet	`g_sf_audio_playback0.p_api->volumeSet(g_sf_audio_playback0.p_ctrl, 255);` Sets software volume control. Software volume control is applied globally to all streams on the hardware.
.close	`g_sf_audio_playback0.p_api->close(g_sf_audio_playback0.p_ctrl);` The `close` API handles the cleanup of internal driver data.
.versionGet	`g_sf_audio_playback0.p_api->versionGet(&version);` Return the version of the driver.

Table 2. Status Return Values

Name	Description
SSP_SUCCESS	Function successful
SSP_ERR_ASSERTION	A pointer is NULL or a parameter is invalid
SSP_ERR_OUT_OF_MEMORY	The number of streams open at once is limited to SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS. If this number is exceeded, an out of memory error occurs.
SSP_ERR_TIMEOUT	Timeout occurred before playback finished
SSP_ERR_NOT_OPEN	The stream control block p_ctrl is not initialized

Note: Lower-level drivers may return common error codes. See the *SSP User's Manua*l, API References for the associated module for a definition of all relevant status return values.

3. Operational Overview

The Audio Playback Framework module creates a thread internally to support audio playback. The following flowchart shows the audio playback framework thread and its interactions with public Audio Playback Framework APIs.

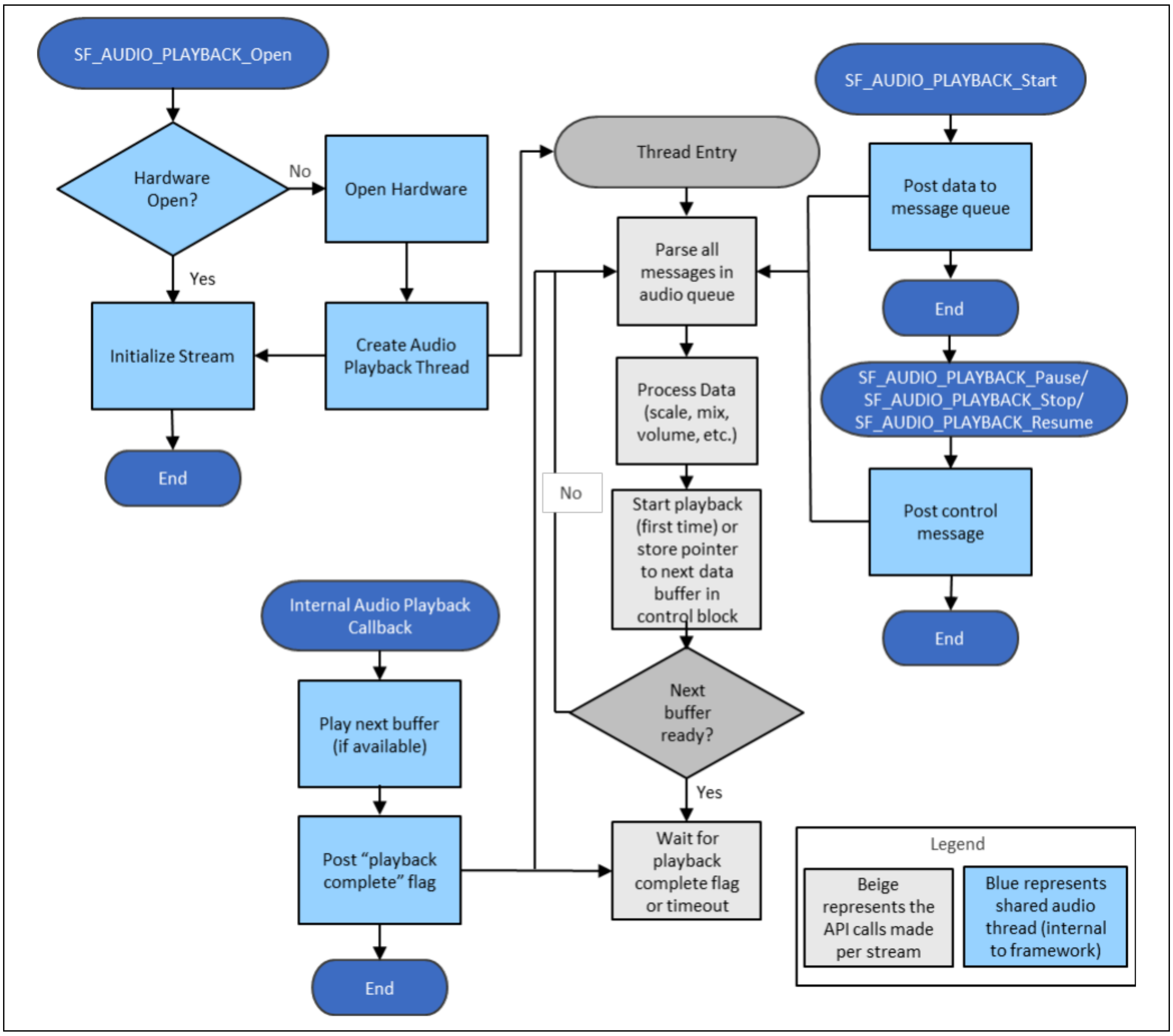

SF_AUDIO_PLAYBACK_Open
Hardware Open?
No
Open Hardware
Yes
Initialize Stream
Create Audio Playback Thread
End
Thread Entry
Parse all messages in audio queue
Process Data (scale, mix, volume, etc.)
No
Start playback (first time) or store pointer to next data buffer in control block
Next buffer ready?
Yes
Wait for playback complete flag or timeout
SF_AUDIO_PLAYBACK_Start
Post data to message queue
End
SF_AUDIO_PLAYBACK_Pause/ SF_AUDIO_PLAYBACK_Stop/ SF_AUDIO_PLAYBACK_Resume
Post control message
End
Internal Audio Playback Callback
Play next buffer (if available)
Post "playback complete" flag
End
Legend
Beige represents the API calls made per stream
Blue represents shared audio thread (internal to framework)

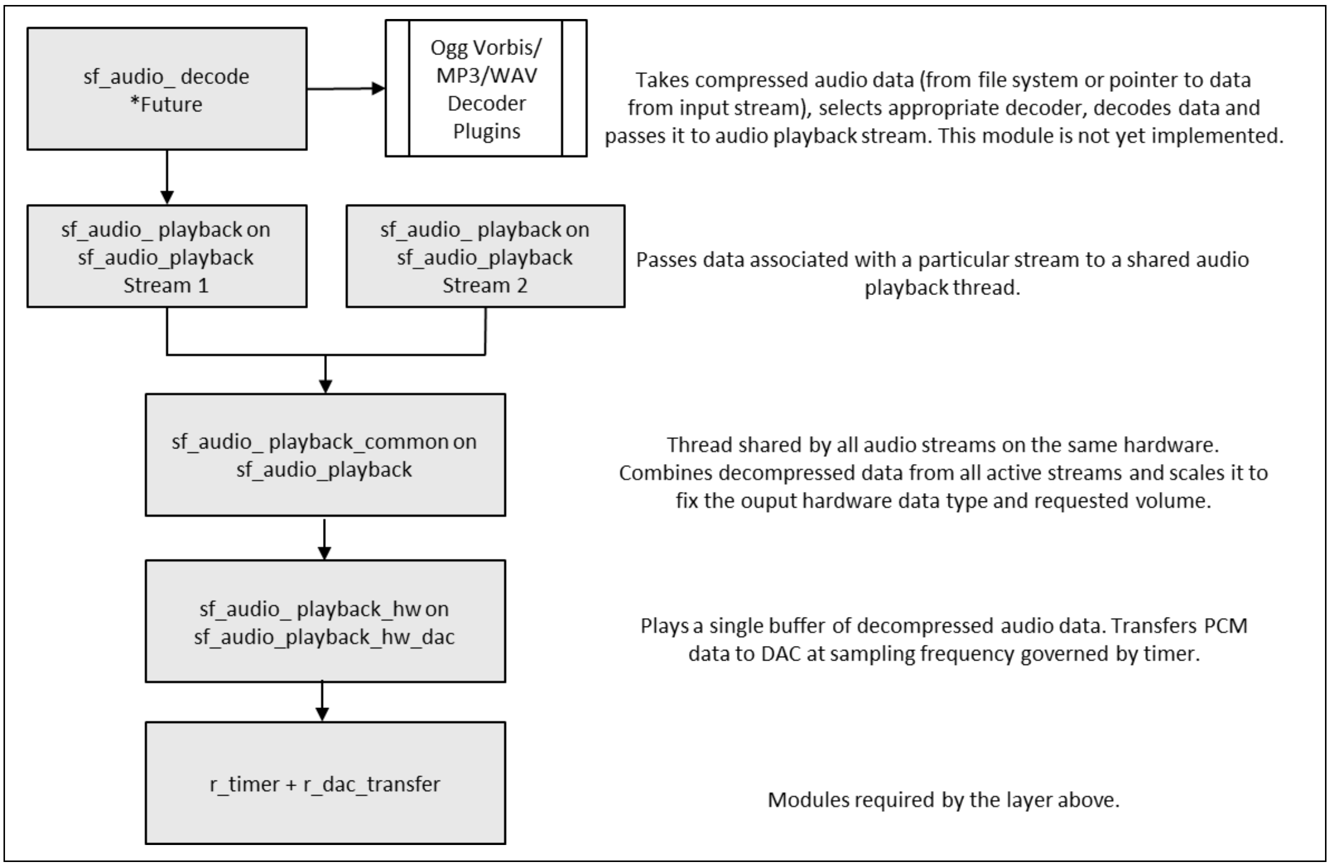
Figure 2. Audio Playback Framework Flowchart

Suggested use of the audio playback framework:

- Create a semaphore (for example, `g_sf_audio_playback_semaphore`). This can be done in the **Threads** tab. Set the initial value to 2 (the audio playback framework can store up to two data messages per stream).
- Create a callback function (for example, `sf_audio_playback_callback`). Enter the name of your callback function in the Audio Playback Framework instance. The callback function is called when the audio playback framework is done with the data. In the callback, put the semaphore created above.
- In your main loop, get the semaphore before playing data. To play data, first acquire a buffer from the messaging framework, then create your audio playback data structure inside the buffer.

The Audio Playback Framework supports multiple audio streams on a single hardware port.

The following block diagram shows the modules required if two streams are used.



sf_audio_ decode *Future
Ogg Vorbis/ MP3/WAV Decoder Plugins
Takes compressed audio data (from file system or pointer to data from input stream), selects appropriate decoder, decodes data and passes it to audio playback stream. This module is not yet implemented.
sf_audio_ playback on sf_audio_playback Stream 1
sf_audio_ playback on sf_audio_playback Stream 2
Passes data associated with a particular stream to a shared audio playback thread.
sf_audio_ playback_common on sf_audio_playback
Thread shared by all audio streams on the same hardware. Combines decompressed data from all active streams and scales it to fix the ouput hardware data type and requested volume.
sf_audio_ playback_hw on sf_audio_playback_hw_dac
Plays a single buffer of decompressed audio data. Transfers PCM data to DAC at sampling frequency governed by timer.
r_timer + r_dac_transfer
Modules required by the layer above.

Figure 3. Implementing Multiple Audio Streams

3.1 Important Operational Notes and Limitations

3.1.1 Operational Notes

- The Queue used must match the name specified in **Properties** for Audio Playback Framework shared on sf_audio_playback (default is `g_sf_audio_playback_queue`).
- The audio framework I²S hardware port has dependencies on the I²S driver module. The I²S driver module can be accelerated with DTC.
- I²S driver module
  - Set the Audio Clock Frequency in Hz to the frequency of the input audio clock used.
  - Set the Sampling Frequency in Hz to the sampling frequency of the audio data.
  - Set the Data Bits and Word Length to 16 bits (audio framework accepts 16-bit samples only).
  - Enable the SSIn TXI and SSIn INT interrupts.
- Transfer module on r_dtc
  - Set the Activation Source to the SSIn TXI interrupt.
- The Audio Playback I²S Framework is designed to support the following Synergy MCU families with no changes to the API:
  - S7G2 MCU Group
  - S3A7, S5D9, S3A3 MCU Groups

3.1.2 Limitations

Refer to the latest *SSP Release Notes* for any additional operational limitations for this module.

4. Including Audio Playback I2S Framework Module in an Application

The following instructions tell you how to include the Audio Playback I²S Framework Module in an application using the SSP configurator.

Note: It is assumed 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 *Getting Started Guide for SSP* given in the References section at the end of this document to learn how to manage each of these important steps in creating SSP-based applications.

To use the Audio Playback Framework with I²S hardware port, simply add the following three modules: Audio Playback Framework on `sf_audio_playback`, Audio playback Framework Shared on `sf_audio_playback`, and Audio Playback Hardware Framework Shared on `sf_audio_playback_hw_i2s`. Add these modules to a project thread using the stacks selection sequence in the following table. (The default name for the Audio Playback I²S Framework is `g_sf_audio_playback_hw0`. This name can be changed in the associated **Properties** window.)

Table 3. Audio Playback I²S Framework Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_audio_playback Audio Playback Framework on sf_audio_playback	Threads	New Stack> Framework> Audio> Audio Playback Framework on sf_audio_playback
g_sf_audio_playback_hw0 Audio Playback Hardware Framework on sf_audio_playback_hw_i2s	Threads	Add Audio Playback Hardware> New > Audio Playback Hardware Framework on g_sf_audio_playback_hw_i2s

The following figure shows that when the Audio Playback I²S Framework module on `sf_audio_playback_hw_I2S` is added to the thread stack, the configurator automatically adds the needed lower-level drivers. Any drivers that need additional configuration information are highlighted in red. Modules with a gray band are individual modules that stand alone. Modules with a blue band are shared or common and need only be added once and can be used by multiple stacks. Modules with a pink band can require the selection of lower- level drivers. Sometimes these modules are either optional or recommended, and these modules are indicated in the block with the inclusion of this text. If the addition of lower-level drivers is required, the module description includes **Add** in the text. Clicking on any pink banded modules brings up the **New** icon and then shows the possible choices.

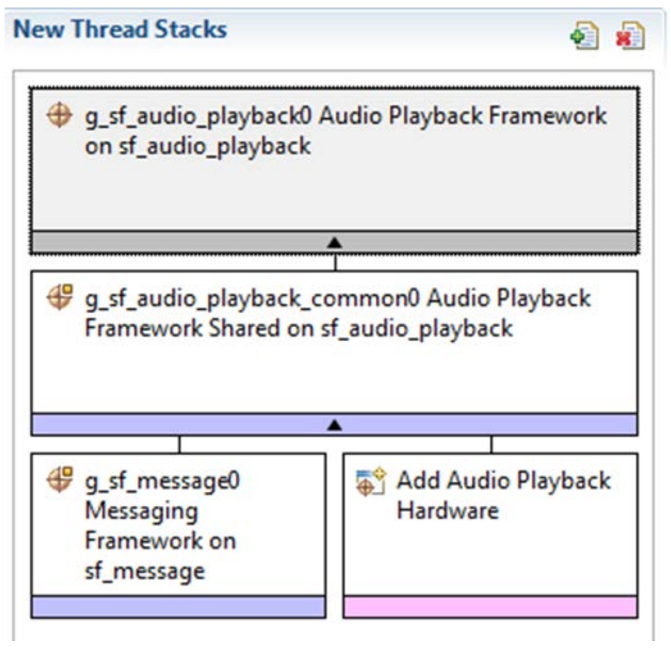

New Thread Stacks
g_sf_audio_playback0 Audio Playback Framework on sf_audio_playback
g_sf_audio_playback_common0 Audio Playback Framework Shared on sf_audio_playback
g_sf_message0 Messaging Framework on sf_message
Add Audio Playback Hardware

Figure 4. Audio Playback I²S Framework Module Stack

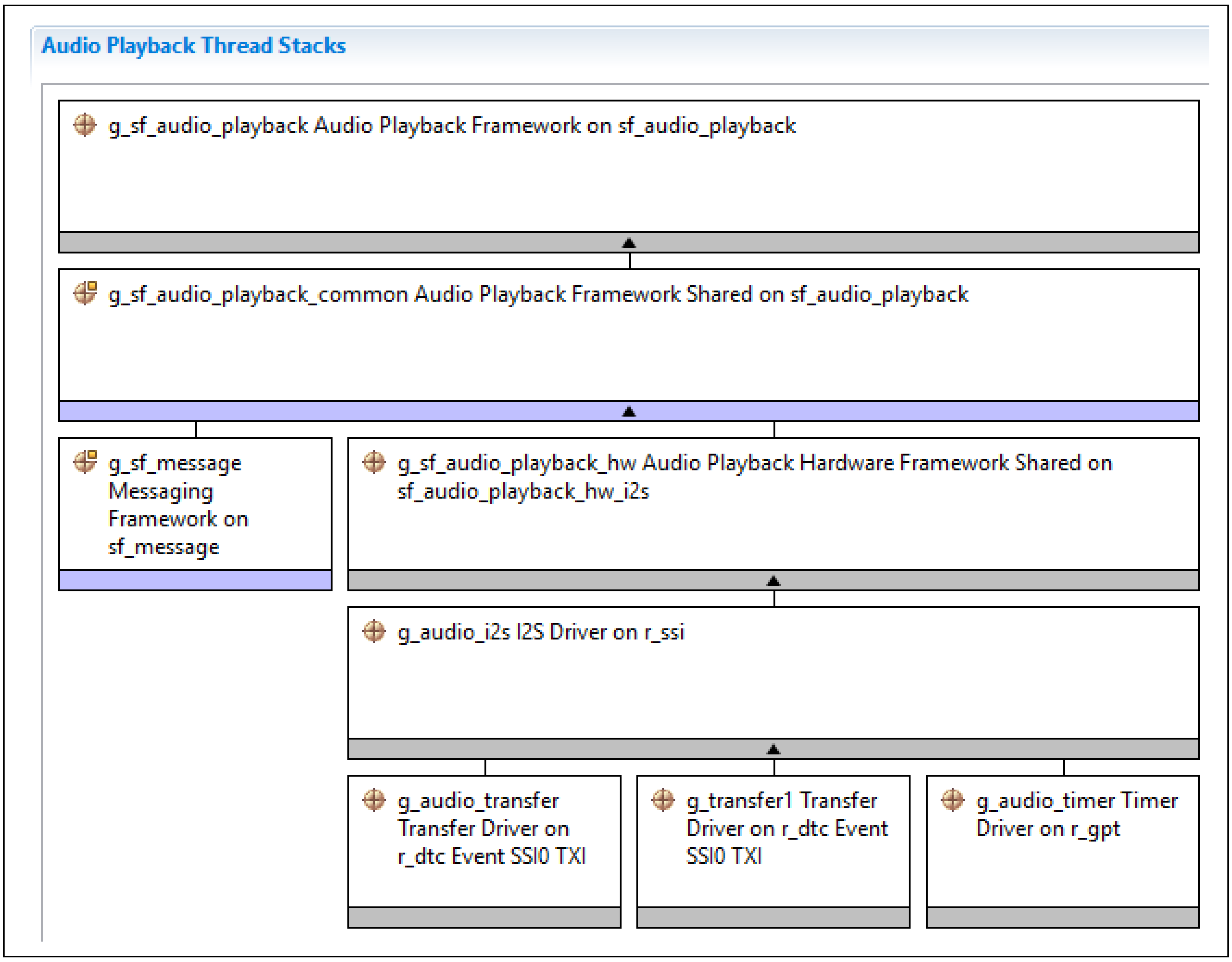

Audio Playback Thread Stacks
g_sf_audio_playback Audio Playback Framework on sf_audio_playback
g_sf_audio_playback_common Audio Playback Framework Shared on sf_audio_playback
g_sf_message Messaging Framework on sf_message
g_sf_audio_playback_hw Audio Playback Hardware Framework Shared on sf_audio_playback_hw_i2s
g_audio_i2s I2S Driver on r_ssi
g_audio_transfer Transfer Driver on r_dtc Event SSI0 TXI
g_transfer1 Transfer Driver on r_dtc Event SSI0 TXI
g_audio_timer Timer Driver on r_gpt

Figure 5. Audio Playback I²S Framework Module Stack

5. Configuring the Audio Playback I2S Framework Module

The Audio Playback I²S Framework module must be configured for the desired operation. The SSP configuration window automatically identifies (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. Locked properties are identified with a lock icon in the **Properties** 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 listed 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 within 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. Note that the interrupt priorities listed in the **Properties** window in the ISDE indicate 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 within the ISDE when configuring interrupt-priority levels.

Note: You may want to open your ISDE, create the module and explore the property settings in parallel with looking over the following configuration table settings. This helps to orient you and can be a useful hands-on approach to learning the ins and outs of developing with SSP.

Table 4. Configuration Settings for the Audio Playback I²S Framework Module on sf_audio_playback_hw_i2s

Parameter	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking
Name	g_sf_audio_playback_hw0	Module name

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

In some cases, settings other than the defaults for lower-level modules can be desirable. For example, it might be useful to select the SSI Channel based on the target hardware implementation. The configurable properties for the lower-level stack modules are given in the following sections as a complete reference.

5.1 Configuration settings for the Low-Level Modules

Typically, only a small number of settings must be modified from the default for lower-level drivers as indicated with the red text in the thread stack block. Note that some of the configuration properties must be set to a certain value for proper framework operation and are locked to prevent user modification. The following table identifies all the settings within the properties section for the module.

Table 5. Configuration Settings for the I²S HAL Module on r_ssi

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking
Name	g_i2s0	Module name
Channel	0	Physical hardware channel
Audio Clock Frequency (Hertz)	2822400	Input audio clock frequency is used to generate the I²S clock. Must be a multiple between 1 and 128 of: (sampling_freq_hz * word_length_in_bits)
Sampling Frequency (Hertz)	44100	Sampling frequency of audio data
Data Bits	8, 16, 18, 20, 22, 24 bits Default: 16 bits	Bit depth of audio data, which is the size in bits of one sample of audio data
Word Length	8 bits, 16, 24, 32 Default: 16 bits	Word length of audio data, must be at least the same size as the bit depth (Data Bits field)
WS Continue Mode	Enabled, Disabled Default: Disabled	Enable WS continue mode to continue to output the word select line when the peripheral is idle. Disable to stop outputting the word select line when the peripheral is idle.
Name of I²S callback function to be defined by user	NULL	A user callback function must be registered in open. The callback will be called from the interrupt service routine (ISR) when the transmission FIFO reaches the high watermark point after all data for transmission is transmitted or when reception is complete (the requested number of bytes have been received). Warning: Since the callback is called from an ISR, care should be taken

ISDE Property	Value	Description
		not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system
Transmit 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	Transmit interrupt priority selection
Receive 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	Receive interrupt priority selection
Idle/Error 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	Idle/error interrupt priority selection

Note: The example values and defaults are for a project using the SK-S7G2 Kit. Other MCUs may have different default values and available configuration settings.

Table 6. Configuration Settings for the DTC HAL Module on r_dtc Software Activation 1

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	4 Bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	Activation source selection
Auto Enable	False	Auto enable selection

ISDE Property	Value	Description
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event 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	ELC Software Event interrupt priority selection

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

Table 7. Configuration Settings for the DTC HAL Module on r_dtc Software Activation 1

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table
Name	g_transfer1	Driver name
Mode	Normal	Mode selection
Transfer Size	4 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1, Software Activiation 2, Peripheral Events Default: Software Activation 1	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event 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	ELC software event interrupt priority selection

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

Table 8. Configuration Settings for the AGT HAL Module on r_agt

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection
Name	g_timer0	Module name
Channel	0	Channel selection
Mode	Periodic	Mode selection
Period Value	2822400 * 2	Period value selection
Period Unit	Hertz	Period unit selection
Auto Start	FALSE	Auto start selection
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	Count source selection
AGTO Output Enabled	True, False Default: False	AGTO output selection
AGTIO Output Enabled	True, False Default: False	AGTIO output selection
Output Inverted	True, False Default: False	Output inverted selection
Enable comparator A output pin	True, False Default: False	Enable comparator A output pin selection
Enable comparator B output pin	True, False Default: False	Enable comparator B output pin selection
Callback	NULL	Callback selection
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 example values and defaults are for a project using the Synergy S7G2 MCU. Other MCUs may have different default values and available configuration settings.

Table 9. Configuration Settings for the GPT HAL Module on r_gpt

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection
Name	g_timer0	Module name
Channel	0	Channel selection
Mode	Periodic	Mode selection
Period Value	2822400 * 2	Period value selection
Period Unit	Hertz	Period unit selection
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	Auto start selection
GTIOCA Output Enabled	True, False Default: False	GTIOCA output enabled selection
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCA stop level selection

ISDE Property	Value	Description
GTIOCB Output Enabled	True, False Default: False	GTIOCB output enabled selection
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCB stop level selection
Callback	NULL	Callback selection
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 example values and defaults are for a project using the Synergy S7G2 MCU Group. Other MCUs may have different default values and available configuration settings.

5.2 Clock configuration

The Audio Playback Framework hardware modules (I²S) use the peripheral clocks in the Clocks configuration window.

5.3 Pin configuration

The SSI 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. The following table lists this method to select pins within the SSP configuration window and the subsequent table lists an example selection for the associated pins.

Note: For some peripherals, the operation mode selected determines the peripheral signals available and the MCU pins required.

Table 10. Pin Selection for I²S

Resource	ISDE Tab	Pin Selection Sequence
I²S	Pins	Select **Peripherals** > **Connectivity:SSI** > **SSI0/SSI1**

Note: The selection sequence assumes SSI0 or SSI1 is the desired hardware target for the driver.

Table 11. Pin Configuration Settings for the I²S driver on SSI

Pin Configuration Property	Value	Description
Pin Group Selection	A only, _B only, Mixed (Default: _A only)	Pin group for I²S port
Operation Mode	Enabled, Custom, Disabled (Default: Disabled)	Operation selection
SSISCK	None, P403, P112 (Default: None)	SSI Serial Clock
SSIWS	None, P404, P113 (Default: None)	SSI Stereo pin selection
SSITXD	None, P405, P115 (Default: None)	SSI Transmit pin selection
SSIRXD	None, P406, P114 (Default: None)	SSI Receive pin selection

Note: The example values are for a project using the Synergy S3A7 MCU Group and the DK-S3A7 Kit. Other Synergy MCUs and other Synergy Kits may have different pin configuration settings available.

6. Using the Audio Playback I2S Framework Module in an Application

The typical steps in using the Audio Playback I²S Framework module in an application are:

- Initialize the Audio Framework using the `open` API.
- Use the callback function to post the semaphore while the main thread is waiting on the same semaphore.
- Acquire the buffer from the Messaging Framework.
- Create the Audio Framework Data Structure inside the buffer.
- Start the Audio Playback Framework using the `start` API.

The following diagram illustrates the common steps in a typical operational flow:

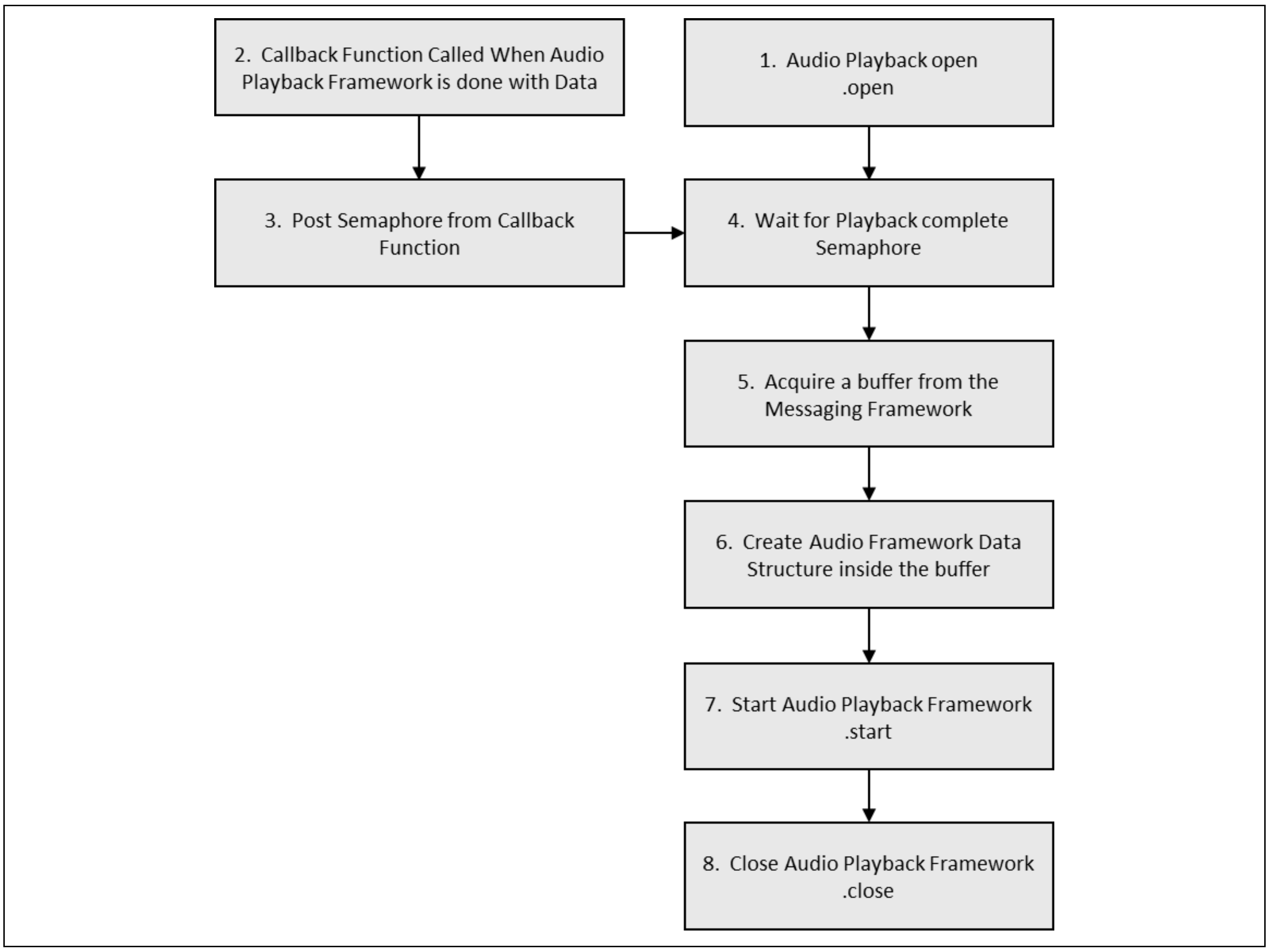

2. Callback Function Called When Audio Playback Framework is done with Data
1. Audio Playback open .open
3. Post Semaphore from Callback Function
4. Wait for Playback complete Semaphore
5. Acquire a buffer from the Messaging Framework
6. Create Audio Framework Data Structure inside the buffer
7. Start Audio Playback Framework .start
8. Close Audio Playback Framework .close

Figure 6. Flow Diagram of a Typical Audio Playback I²S Framework Module Application

7. The Audio Playback I2S Framework Module Application Project

The application project associated with this module guide demonstrates the operational flow steps in an example application. In ISDE, you may want to import and open the application project to view the configuration settings for the Audio Playback I²S Framework module. The project can be found using the link provided in the References section at the end of this document. You can also read over the code in `audio_playback_thread_entry.c` which is used to demonstrate the Audio Playback APIs in a complete design.

The application project shows how to use the audio framework I²S hardware port with the Audio Playback Framework APIs. The application project main thread entry initializes the Audio Playback Framework and plays the RAW PCM file on flash memory. The playback framework gets information from the Messaging Framework and stores this in the audio buffer. Stream related information can be found in the `audio_data.c` file. The following table identifies the target versions for the associated software and hardware used by the application project:

Table 12. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e^2 studio	v5.4.0.023 or later	Integrated Solution Development Environment
SSP	v1.3.0 or later	Synergy Software Platform
IAR EW for Synergy	v7.71.3	IAR Embedded Workbench® for Renesas Synergy™
SSC	v5.4.0.023 or later	Synergy Standalone Configurator
DK-S3A7	v2.0	Development Kit

The following figure shows the application project flow.

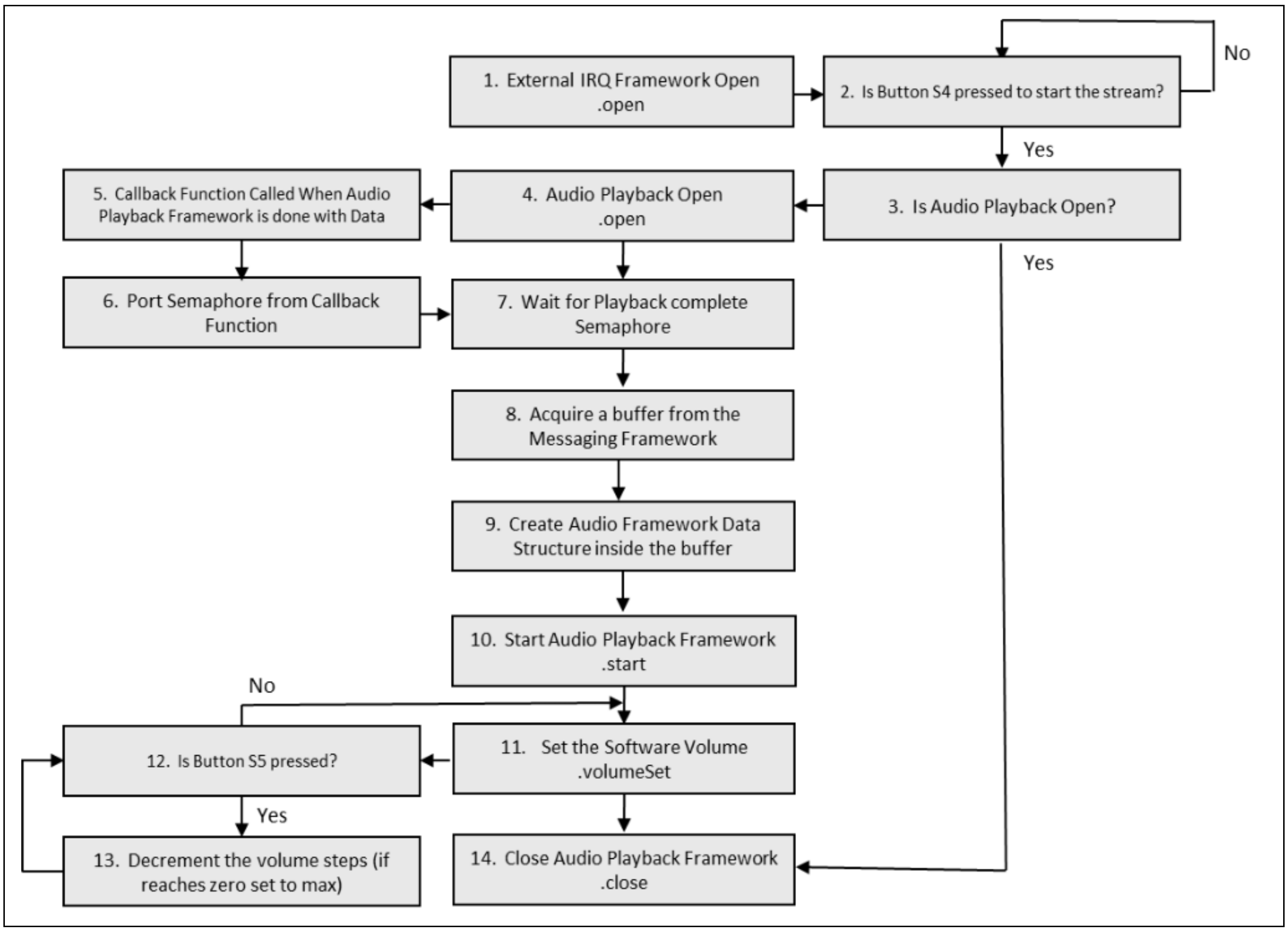

1. External IRQ Framework Open .open
2. Is Button S4 pressed to start the stream?
No
Yes
3. Is Audio Playback Open?
Yes
4. Audio Playback Open .open
5. Callback Function Called When Audio Playback Framework is done with Data
6. Port Semaphore from Callback Function
7. Wait for Playback complete Semaphore
8. Acquire a buffer from the Messaging Framework
9. Create Audio Framework Data Structure inside the buffer
10. Start Audio Playback Framework .start
No
11. Set the Software Volume .volumeSet
12. Is Button S5 pressed?
Yes
13. Decrement the volume steps (if reaches zero set to max)
14. Close Audio Playback Framework .close

Figure 7. Audio Playback I²S Framework Module Application Project Flow

The complete application project can be found using the link provided in the References section at the end of this document.

The `audio_playback_hw_i2s_control.c` file resides in the project once it has been imported into the ISDE. The project also includes the `application_define.h` and `audio_data.c` files. You can open these files from within the ISDE and follow along with the description provided to help identify key uses of the APIs.

The first section of `audio_playback_hw_i2s_control.c` has the header files which references the audio playback framework instance structure and structure for basic information about PCM stream and payload for sf_audio_playback. An audio callback function which executes a post of the previously created semaphore (`g_sf_audio_playback_sem`) is defined next. The callback function is called when the audio playback framework is done with the data.

The next section of the `audio_playback_hw_i2s_control.c` file is the entry function for the main program control section. The application uses the on-board push-buttons S1 and S2 for controlling the audio playback. The external IRQ is first opened for both S1 and S2. Next `audio_chip_configure()` is called to configure the audio codec chip, MAX98089, on the DK-S3A7 to output playback. The file `iic_max_98089.c` contains the `audio_chip_configure`.

The application waits for a user input – external IRQ which is generated by pressing the on-board push-button S1, before playing the audio data. After successfully receiving the external IRQ event, a local variable is updated with the stream related information. Thereafter, it acquires the semaphore `g_sf_audio_playback_sem` before starting to play data from `data_stream_0`. After acquiring this semaphore, the main program control acquires a buffer from the messaging framework and copies the audio playback data into the buffer. Playback is started by calling the `.start` API.

The on-board pushbutton S1 is used to start and stop playback. The `.stop` API is used to stop the playback. Pushbutton S2 is used to change the volume of the playback. Volume is changed by using the `.volumeSet` API. Each press of S2 decreases the volume by 50. If the volume is currently 0, the volume is set to 255, the maximum value.

Note: This description assumes you are familiar with using External IRQ Framework. If you are unfamiliar with this, see the *Synergy Software Package (SSP) User's Manual*. External IRQ Framework has been used to implement S4 and S5 button press event notification and wait to execute other API functionalities. Currently in this application project, button S4 on SK-S7G2 starts and stops the playback stream. During playback, button S5 press decreases the playback volume because normal playback starts with full volume in this application project.

Remember to add the subscriber thread for Audio event in the **Messaging** tab. Highlight the new Subscriber in the Audio Playback Subscribers. Record the **Symbol** name. Highlight the **Audio Playback Framework Shared** module in the HAL/Common in **Thread** tab and set the **Audio Message Queue** name to the **Symbol** name from the Audio Playback Subscriber.

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

Table 13. Audio Playback Thread Properties

ISDE Property	Value Set
Symbol	audio_playback_thread
Name	Audio Playback Thread
Stack size (bytes)	2048
Priority	5
Auto start	Enabled
Time slicing interval (ticks)	10

Table 14. Audio Playback Framework Module Configuration Settings for Application Project

ISDE Property	Value Set
Buffer Size Bytes	512
Maximum Number of Streams	1
Thread Stack Size	512
Name	g_sf_audio_playback
Message Class Instance	0
Callback	g_sf_audio_playback_callback

Table 15. Audio Playback Framework Shared Module Configuration Settings for Application Project

ISDE Property	Value Set
Name	g_sf_audio_playback_common
Thread Priority	3
Audio Message Queue Name	audio_playback_thread_message_queue

Table 16. I²S Driver Module Configuration Settings for Application Project

ISDE Property	Value Set
Name	g_i2s0
Channel	0
Audio Clock Frequency (Hertz)	11289600
Sampling Frequency (Hertz)	22050
Data Bits	16 bits
Word Length	16 bits
WS Continue Mode	Enabled
Transmit Interrupt Priority	Priority 2
Receive Interrupt Priority	Priority 1
Idle/Error Interrupt Priority	Priority 2

Table 17. I²C Master Driver Module Configuration Settings for Application Project

ISDE Property	Value Set
Name	g_i2c_codec
Channel	2
Rate	Fast-mode
Slave Address	0x10
Address Mode	7-bit
Timeout Mode	Short Mode
Callback	NULL
Receive Interrupt Priority	Priority 2
Transmit Interrupt Priority	Priority 2
Transmit End Interrupt Priority	Priority 2
Error Interrupt Priority	Priority 2

Table 18. External IRQ Framework Module Configuration Settings for Application Project

ISDE Property	Value Set
Name	g_sf_irq_button1
Event	Semaphore Put

Table 19. Transfer Driver Module Configuration Settings for Application Project

ISDE Property	Value Set
Name	g_audio_transfer
Activation Source (Must Enable IRQ)	Event SSI0 TXI

Table 20. Timer Driver Module Configuration Settings for Application Project

ISDE Property	Value Set
Name	g_audio_timer
Channel	6

Table 21. Semaphore Configuration Settings for Application Project

ISDE Property	Value Set
Name	Audio Playback Framework Semaphore
Symbol	g_sf_audio_playback_sem
Channel	2

Table 22. Messaging Framework Settings for g_sf_message

ISDE Property	Value Set
Message Queue Depth	16
Name	g_sf_message
Work memory size in bytes	2048
Pointer to subscriber list array	p_subscriber_lists
Name of the block pool internally used in the messaging framework	sf_msg_blk_pool
Name of generated initialization function	sf_message_init
Auto Initialization	Enable

Table 23. Pin Configuration Settings for the Application Project

Pin Selection Sequence	Pin Configuration Property	Setting
Peripherals > Connectivity: SSI > SSI	Pin Group Selection	A only
	Operation Mode	Custom
	AUDIO_CLK	P400
Peripherals > Connectivity: SSI > SSI0	Pin Group Selection	A only
	Operation Mode	Enabled
Peripherals > Timer: GPT > GPT6	Pin Group Selection	Mixed
	Operation Mode	GTIOCA or GTIOCB
	GTIOCB	P401

8. Customizing for a Target Application

You can change the configuration settings in the Application Project to suit your requirements. For example, you can change the sampling frequency and transfer driver through timer module and transfer driver, respectively. For the transfer driver, you have a choice between the DTC and DMAC.

This Audio Playback HW I²S Framework Application Project uses on-board pushbuttons S1 and S2 to control playback, but you can change these.

9. Running the Audio Playback I2S Framework Module Application Project

To run the Audio Playback I²S Framework module application project and to see it executed on a target kit, you can simply import it into your ISDE, compile, and run debug.

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 unfamiliar, refer to the *Getting Started with SSP* chapter in the *SSP User's Manual* listed in the References section at the end of this document.

Use the following steps to create and run the Audio Playback I²S Framework module application project:

1. Import and build the example project included with this module guide by using the Renesas *Synergy Project (SSP) Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf).
2. Apply power to the board through the 5V barrel connector (J1). Connect a USB cable to the J-Link OB (J15) on the Main Board to the USB port of your PC work station.
3. Connect a headphone or speaker to the green Audio_Out connector (J110).
4. Short DK-S3A7 pins P4_0 and P4_1.

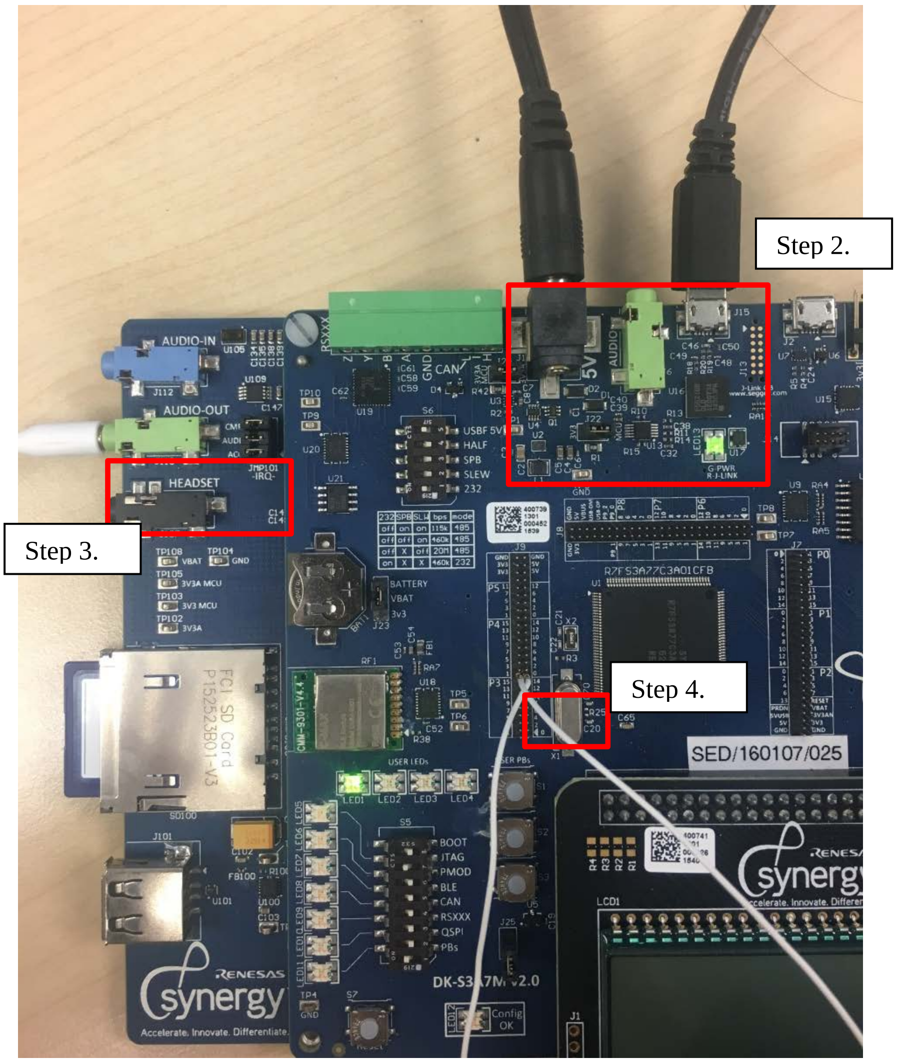


Figure 8. Connections required for board

5. Execute the application program.
6. The output can be heard on a speaker or a headphone connected to the 3.5 mm audio jack on the DK-S3A7 board. Playback is started when the on-board pushbutton S1 is pressed for the first time. Volume can be set by pressing pushbutton S2 in decrement steps of 50 units.
7. Press S1 again to stop the playback. To restart the playback, press S1 once more.

10. 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 the incorrect selection of lower- level drivers. The use of high-level APIs (in the application project) demonstrates the development time savings in allowing work to begin at a high level and avoiding the time required in older development environments to use, or, in some cases, create, lower-level drivers.

11. Next Steps

After you have mastered a simple Audio Playback I²S Framework project, you may want to review a more complex example. You may find that the Audio Playback Framework is a better fit for your target application. The *Audio Playback I2S Framework Module Guide* demonstrates the use of the Audio Framework within a ThreadX-based implementation. This guide is available at the link shown in the References section at the end of this document.

12. Reference Information

SSP User Manual: Available in html format in the SSP distribution package and as a pdf from the Synergy Gallery.

Links to all the most up-to-date Audio Playback I²S Framework module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/search/sf_audio_playback_hw_i2s%20Module%20Guide%20Resources.

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	Jun.15.18	–	Initial release
1.01	Jan.07.19	15 to 18	Added tables 13, 17, 18, 19 & 22. Updated Table 14 and corrected labels to Figure 8.

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/.

Trademarks