# RX Family

## Audio Input/Output Program Using SSIE

### Introduction

This application note describes the operation examples of the audio Input/Output program using the Firmware Integration Technology (FIT) compatible SSI module and the RX671 SSIE function.

### Target Device

RX671
Evaluation board EK-RX671 (RTK5EK6710S00001BE)

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

## Contents

## 1. Overview

This application note describes operation examples of the audio Input/Output program using the Firmware Integration Technology (FIT) compatible SSI module and the RX671 SSIE function.

Figure 1.1 shows the circuit configuration of the evaluation board EK-RX671, on which we tested the operation of this program.

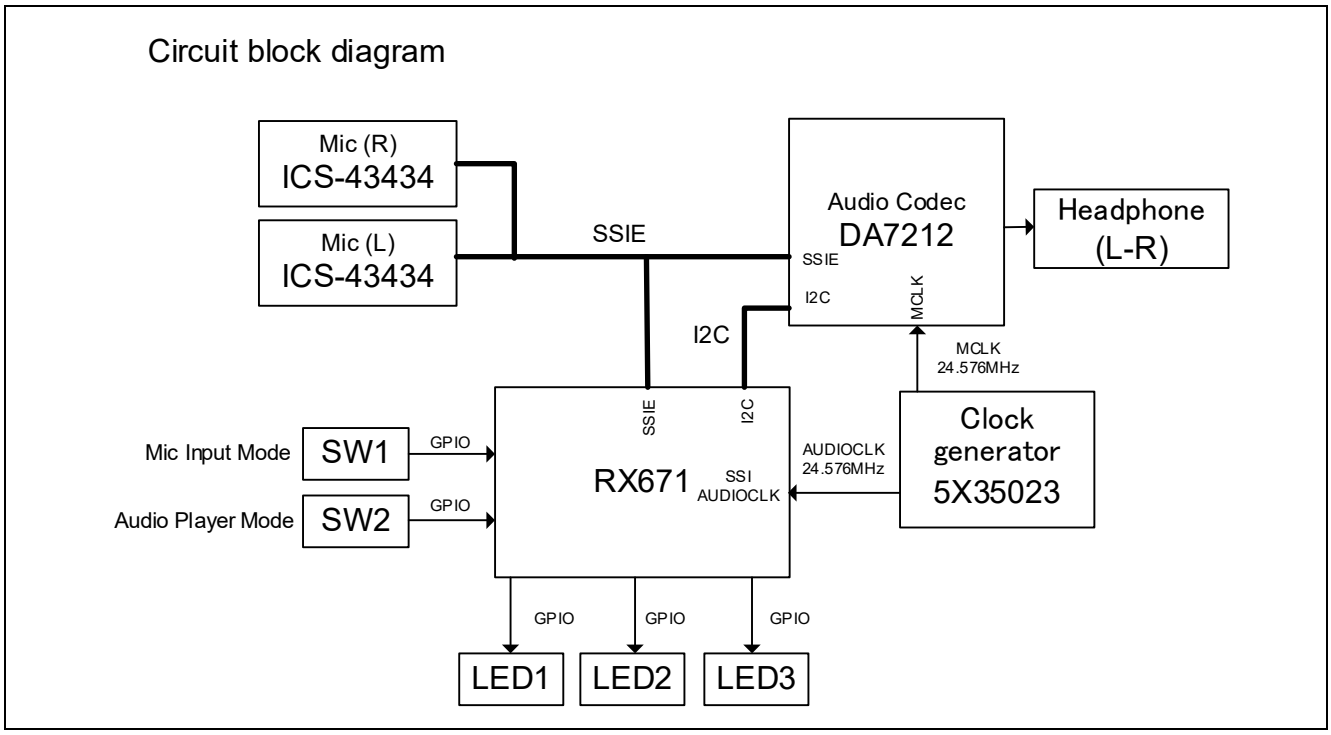


**Figure 1.1 Block diagram**

This configuration diagram illustrates the circuit components related to the operation of this program.

For the overall configuration of the evaluation board EK-RX671, refer to the manual for that evaluation board.

By using the EK-RX671 and this program, audio data stored in RAM or flash memory or acquired from the on-board MEMS Mic can be output to headphones via AudioCodec.

## 1.1 System function

This program can operate in the following three modes using the SSIE function.

- Sine wave output mode
- On-board microphone audio Input/Output mode
- Audio Player mode

### 1.1.1 Sine wave output mode

In this mode, a set-frequency sine wave is generated. The waveform data is recorded in internal RAM and then output to the Audio Codec device via SSIE communication through DMAC.

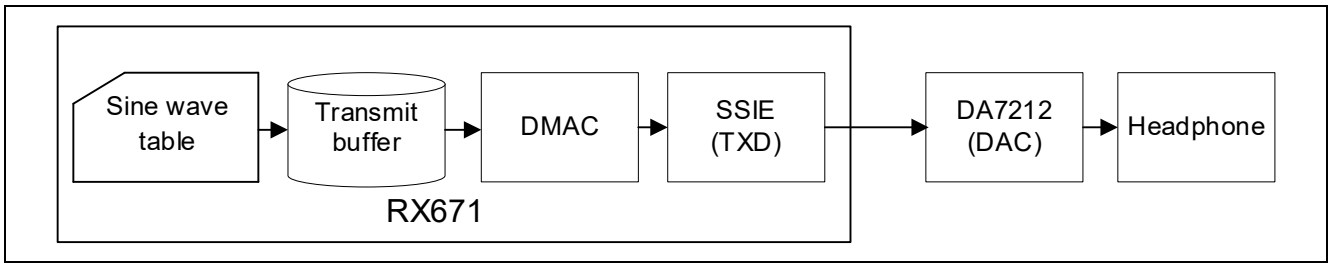


**Figure 1.2 Sine wave generation block diagram**

PCM data transfer format

- PCM data transfer format: I2S
- PCM data: 8/16/18/20/22/24/32 bit stereo
- Sampling rate: RX671 master 96/48/32/24/16/12/8 kHz
  RX671 slave 96/48/44.1/32/24/22.05/16/12/11.025/8 kHz

Note: This sample program only supports multiples of 25 Hz (due to limitations in generating a sine wave table).

Application examples: Various alarms, equipment operation sound generation, audio signal generator for adjustment

### 1.1.2 On-board microphone audio Input/Output mode

In this mode, audio data from the microphone on the evaluation board is captured via SSIE communication and transferred to the receive buffer through DMAC. The data is then copied to the transmit buffer and output to the Audio Codec device via SSIE communication through DMAC.

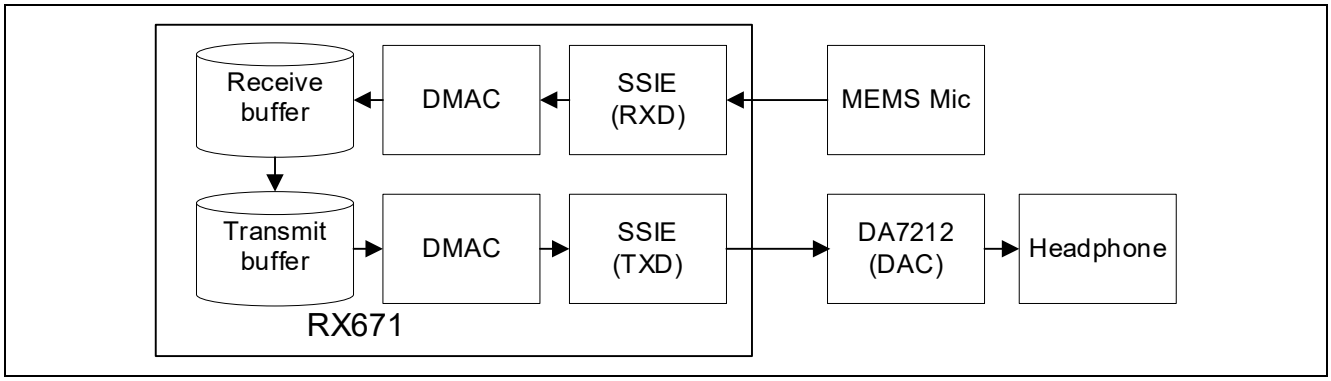


**Figure 1.3 Microphone audio I/O block diagram**

PCM data transfer format

- PCM data transfer format: I2S
- PCM data: 8/16/18/20/22/24/32 bit stereo
- Sampling rate: RX671 master 96/48/32/24/16/12/8 kHz
  RX671 slave 96/48/44.1/32/24/22.05/16/12/11.025/8 kHz

Note: This system limits PCM data to 24 bits and sampling rate to 48 kHz (due to limitations of the MEMS Mic (ICS-43434) on the EK board).

Application examples: Voice recorder, speech recognition system

### 1.1.3 Audio Player mode

This mode reads the PCM or IMA ADPCM WAV file recorded in the internal flash memory and outputs it to the Audio Codec device via SSIE communication.

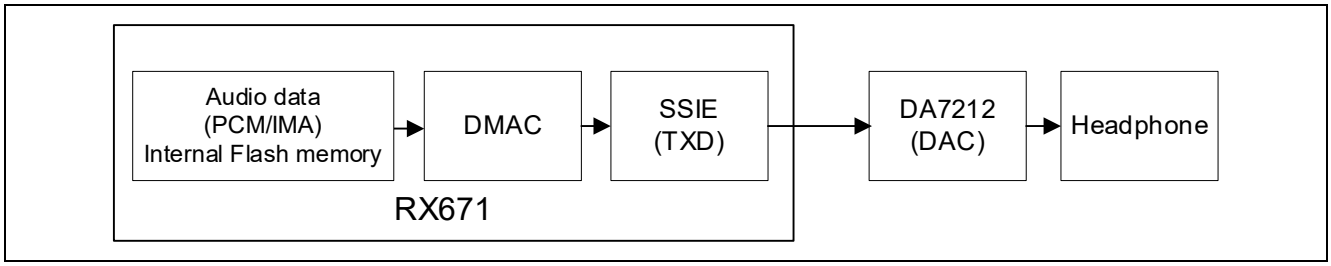


**Figure 1.4 Audio Player block diagram**

Supported file types

— PCM 8/16 bit or IMA ADPCM*1 WAV file Maximum size 2,023,424 bytes.
Data stored in internal flash memory Address 0xFFE10000 Size (maximum) 0x1EE000
Supported files
- WAV file PCM 8 bit monaural
- WAV file PCM 8 bit stereo
- WAV file PCM 16 bit monaural
- WAV file PCM 16 bit stereo
- WAV file IMA ADPCM monaural *1
- WAV file IMA ADPCM stereo *1

Note: 1. IMA ADPCM is an operation sample and does not guarantee operation.

PCM data transfer format

— PCM data transfer format: I2S
— Sampling rate: RX671 master 96/48/32/24/16/12/8 kHz
RX671 slave 96/48/44.1/32/24/22.05/16/12/11.025/8 kHz

Application examples: Audio player, audio synthesis system

# 2. Operating conditions

## 2.1 Operation mode setting

In this program, the following three modes can be selected by SW selection at startup.

**Table 2.1 Operation mode selection**

| Item | Operating mode | Startup operation | LED |
|---|---|---|---|
| 1 | Sine wave output mode | Normal startup (No SW operation) | LED1 lights up. |
| 2 | Microphone audio I/O mode | Press and hold the SW1 to start | LED2 lights up. |
| 3 | Audio Player mode | Press and hold the SW2 to start | LED3 lights up. |

## 2.2 Configuration parameters

**Table 2.2 List of configuration parameters**

| Configuration item | | Configuration item label | Setting method |
|---|---|---|---|
| RX671 SSIE Master/Slave | | SSIE_CH0_CLK_MODE | SSI module code generation<br>• Ch0 Clock Supply Mode |
| Sampling rate *1 | | SSIE_MCLK | SSI module code generation<br>• Master Clock |
| Number of quantization bits | | SSIE_CH0_DATA_WIDTH | SSI module code generation<br>• Ch0 PCM data width |
| Sine wave frequency *2 | Left channel | WAVE_FREQ_HZ | r_ssi_dma_rx_tx_main.h<br>• Macro definition change |
| | Right channel | WAVE_FREQ2_HZ | |
| Volume setting | Headphones | DAC_L_GAIN_OUTSIN_DB | r_ssi_dma_rx_tx_main.h<br>• Macro definition change |
| | | DAC_R_GAIN_OUTSIN_DB | |
| | | DAC_L_GAIN_THROUGH_DB | |
| | | DAC_R_GAIN_THROUGH_DB | |
| | | DAC_L_GAIN_AUDIO_PLAYER_DB | |
| | | DAC_R_GAIN_AUDIO_PLAYER_DB | |
| | | HP_L_GAIN_OUTSIN_DB | |
| | | HP_R_GAIN_OUTSIN_DB | |
| | | HP_L_GAIN_THROUGH_DB | |
| | | HP_R_GAIN_THROUGH_DB | |
| | | HP_L_GAIN_AUDIO_PLAYER_DB | |
| | | HP_R_GAIN_AUDIO_PLAYER_DB | |

Notes: 1. Valid when RX671 is master
2. Sine wave frequency setting is valid only in sine wave output mode
3. SSI module code generation:
rx671_example_ssi_dma_rx_tx.scfg --> components --> r_ssi_api_rx --> properties

### 2.2.1 RX671 SSIE master and slave

Switches the RX671 SSIE communication operation mode between master and slave.

**Table 2.3 Master/Slave settings**

| SSIE_CH0_CLK_MODE | Operating mode | Status |
|---|---|---|
| 0 | Master | RX671 has MCLK output<br>Sampling rate setting (SSIE_MCLK) is valid |
| 1 (Initial value) | Slave | RX671 has MCLK input<br>Sampling rate setting (SSIE_MCLK) is invalid |

### 2.2.2 Sampling rate

If the RX671 is the master, the sampling rate is set by the value of SSIE_MCLK.

**Table 2.4 Sampling rate setting**

| SSIE_MCLK *1 | Sampling rate |
|---|---|
| 256u | 96 kHz |
| 512u (Initial value) | 48 kHz |
| 768u | 32 kHz |
| 1024u | 24 kHz |
| 1536u | 16 kHz |
| 2048u | 12 kHz |
| 3072u | 8 kHz |

Note: 1. If the RX671 is a slave, the value of SSIE_MCLK is ignored.

Sampling rate calculation formula

Sampling rate = AUDIO_CLK / SSIE_MCLK

*In the case of this evaluation board: AUDIO_CLK＝24576000 [Hz]

### 2.2.3 Number of quantization bits

Sets the number of quantization bits for audio data.

**Table 2.5 Number of quantization bits**

| Configuration item label | Configuration value |
|---|---|
| SSIE_CH0_DATA_WIDTH | 32 bits / 24 bits / 22 bits / 20 bits / 18 bits / 16 bits (initial value) / 8 bits |

### 2.2.4 Sine wave frequency

Sets the sine wave frequency in sine wave output mode.

**Table 2.6 Sine wave frequency**

| Configuration channel | Configuration item label | Configuration value [Hz] |
|---|---|---|
| Monaural / L-ch | WAVEFREQ_HZ *1 | 1000 (Initial value) |
| R-ch | WAVEFREQ2_HZ *1 | (500) *2 |

Notes: 1. The configuration value is set to a multiple of 25 Hz (due to the specifications of sine wave generation software).
2. Make WAVEFREQ2 undefined when it is monaural.

### 2.2.5 Volume setting

You can set the DAC and headphone amplifier gain for L and R, respectively, by using the following definitions.

**Table 2.7 Volume setting**

| Gain setting | Setting range [dB] | Description |
|---|---|---|
| DAC_L_GAIN_OUTSIN_DB | -77.25 ~ 12 (0.75step) | In sine wave output mode<br>DAC L-ch digital gain |
| DAC_R_GAIN_OUTSIN_DB | -77.25 ~ 12 (0.75step) | In sine wave output mode<br>DAC R-ch digital gain |
| DAC_L_GAIN_THROUGH_DB | -77.25 ~ 12 (0.75step) | In on-board microphone I/O mode<br>DAC L-ch digital gain |
| DAC_R_GAIN_THROUGH_DB | -77.25 ~ 12 (0.75step) | In on-board microphone I/O mode<br>DAC R-ch digital gain |
| DAC_L_GAIN_AUDIO_PLAYER_DB | -77.25 ~ 12 (0.75step) | In Audio Player mode<br>DAC L-ch digital gain |
| DAC_R_GAIN_AUDIO_PLAYER_DB | -77.25 ~ 12 (0.75step) | In Audio Player mode<br>DAC R-ch digital gain |
| HP_L_GAIN_OUTSIN_DB | -57 ~ 6 (1step) | In sine wave output mode<br>HP L-ch amplifier gain |
| HP_R_GAIN_OUTSIN_DB | -57 ~ 6 (1step) | In sine wave output mode<br>HP R-ch amplifier gain |
| HP_L_GAIN_THROUGH_DB | -57 ~ 6 (1step) | In on-board microphone I/O mode<br>HP L-ch amplifier gain |
| HP_R_GAIN_THROUGH_DB | -57 ~ 6 (1step) | In on-board microphone I/O mode<br>HP R-ch amplifier gain |
| HP_L_GAIN_AUDIO_PLAYER_DB | -57 ~ 6 (1step) | In Audio Player mode<br>HP L-ch amplifier gain |
| HP_R_GAIN_AUDIO_PLAYER_DB | -57 ~ 6 (1step) | In Audio Player mode<br>HP R-ch amplifier gain |

Table 2.8 shows the volume set for each operation mode.

**Table 2.8 Volume settings for each operation mode**

| Operating mode | DAC L/R digital gain | HP_L/R amplifier gain |
|---|---|---|
| Sine wave output | 0 dB | -20 dB |
| On-board microphone I/O | 0 dB | +6 dB |
| Audio Player | 0 dB | -20 dB |

# 3. Program description

## 3.1 Processing flow

### 3.1.1 Overall operating flow

Figure 3.1 shows the main processing flow. Figure 3.2 shows the audio application processing flow.

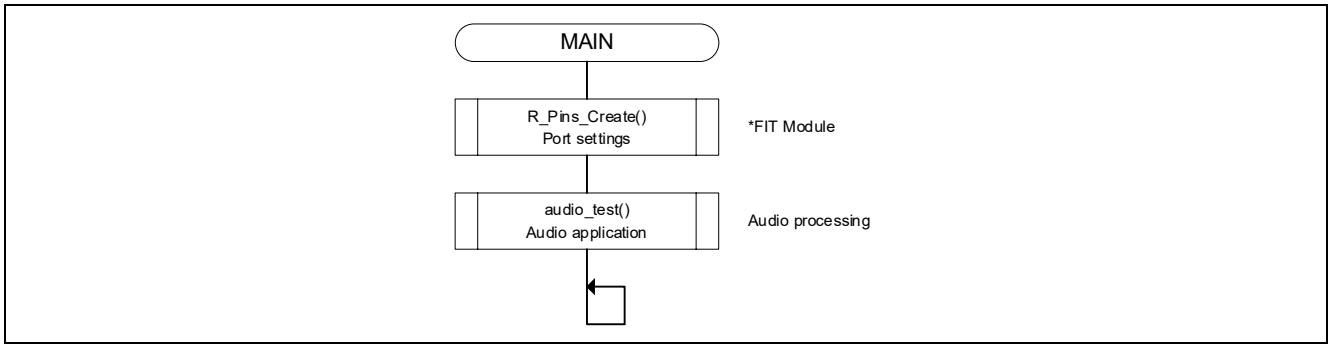


**Figure 3.1 Main processing flow**

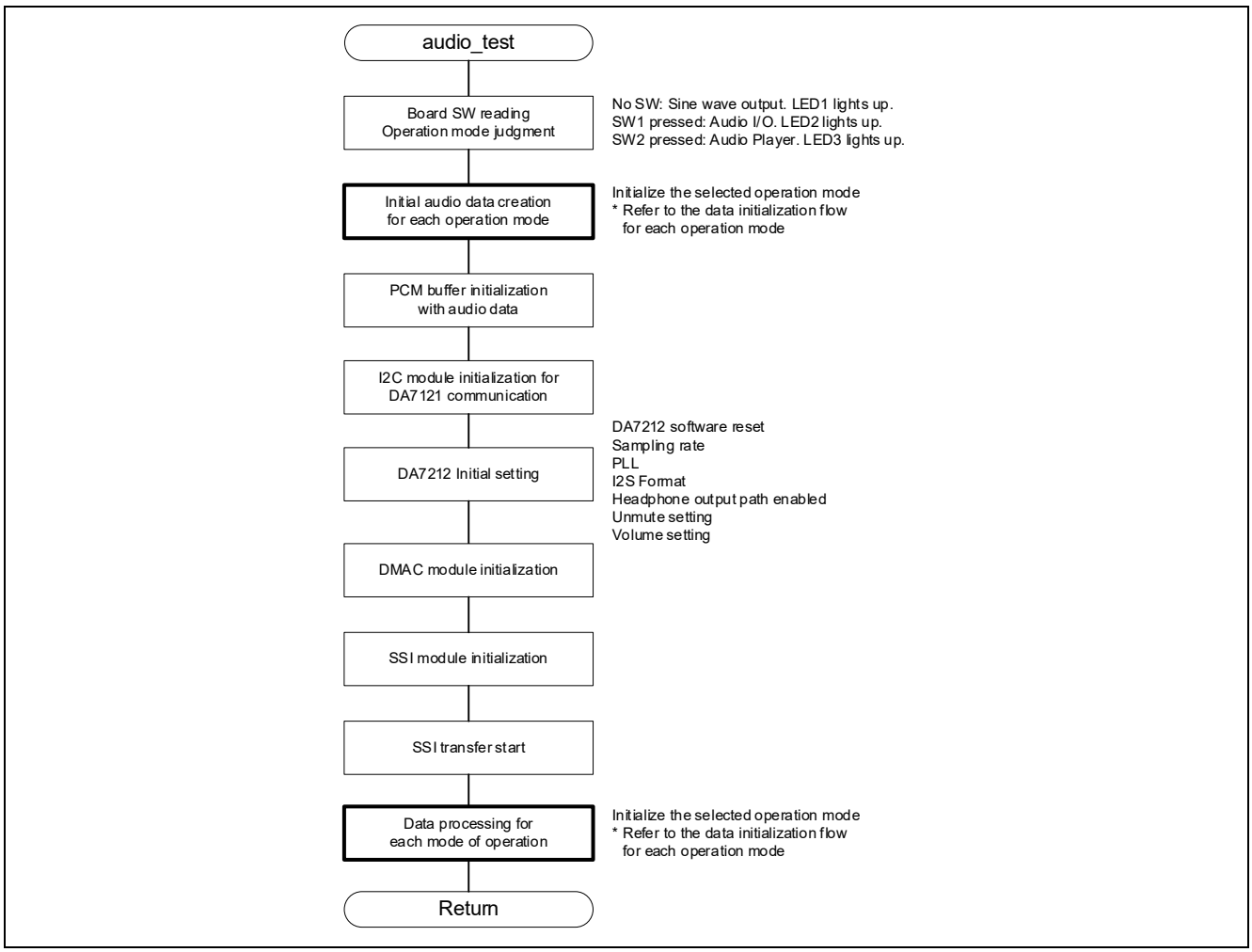


**Figure 3.2 Audio application processing flow**

### 3.1.2 Sine wave output mode processing flow

In this mode, a sine wave of the frequency set by the parameter (WAVE_FREQ) is generated by software, and transmitted to the Audio Codec device via SSIE communication through DMAC.

Audio data from the microphone is received via SSIE communication, but it is not used.

Figure 3.3 shows the data initialization flow in sine wave output mode.

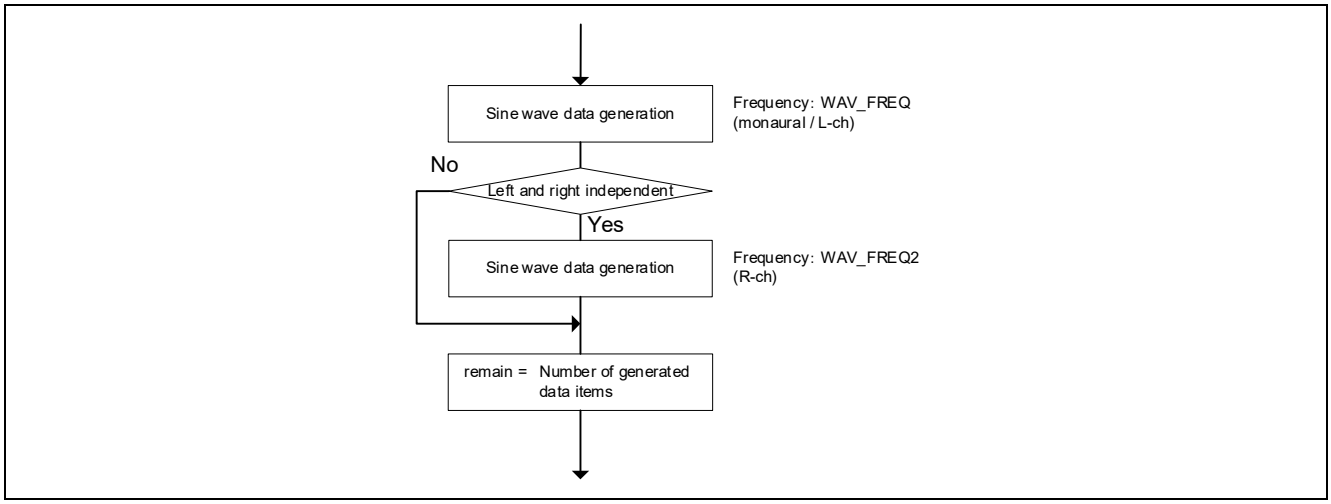


**Figure 3.3 Data initialization flow in sine wave output mode**

In the sine wave output mode, data processing is performed through DMAC.

### 3.1.3 Audio Input/Output mode processing flow

In this mode, audio data from the microphone (ICS-43434) on the evaluation board is received via SSIE communication and transferred to the receive buffer through DMAC. The data is then copied to the transmit buffer by the program, and transmitted to the Audio Codec device via SSIE communication through DMAC.

Figure 3.4 shows the Audio Input/Output mode data processing flow.

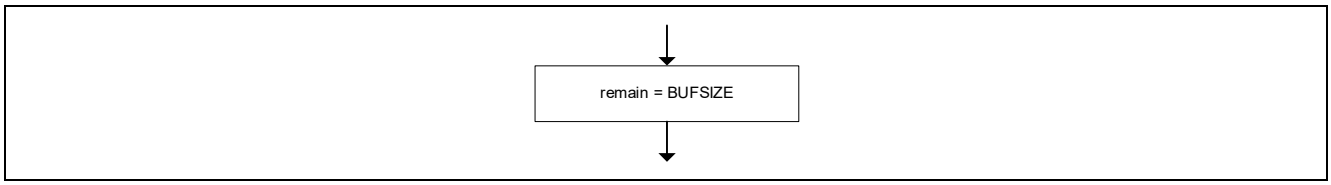


**Figure 3.4 Data initialization flow in audio Input/Output mode**

Note: In audio Input/Output mode, audio data received via SSIE is used, so the initialization processing of the data is not required.

Figure 3.5 shows the Audio Input/Output mode data processing flow.

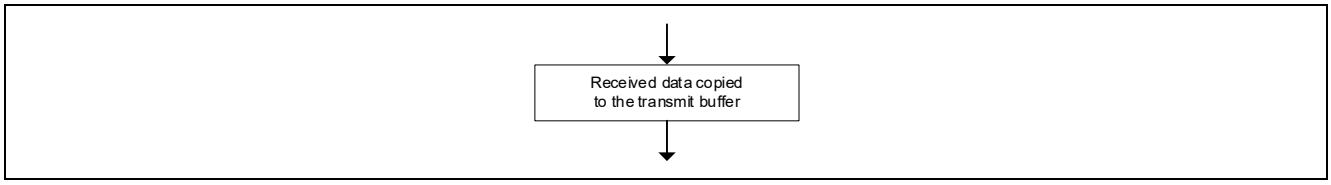


**Figure 3.5 Data processing flow in audio Input/Output mode**

Audio data received via SSIE from the microphone is copied to the transmit buffer and then transferred from the transmit buffer to SSIE through DMAC.

### 3.1.4 Audio Player mode processing flow

In this mode, data is read from the internal flash memory (in which audio data is recorded) and sent to the Audio Codec device via SSIE communication through DMAC.

Audio data from the microphone is received via SSIE communication, but it is not used.

Figure 3.6 shows the Audio Player mode data initialization flow.

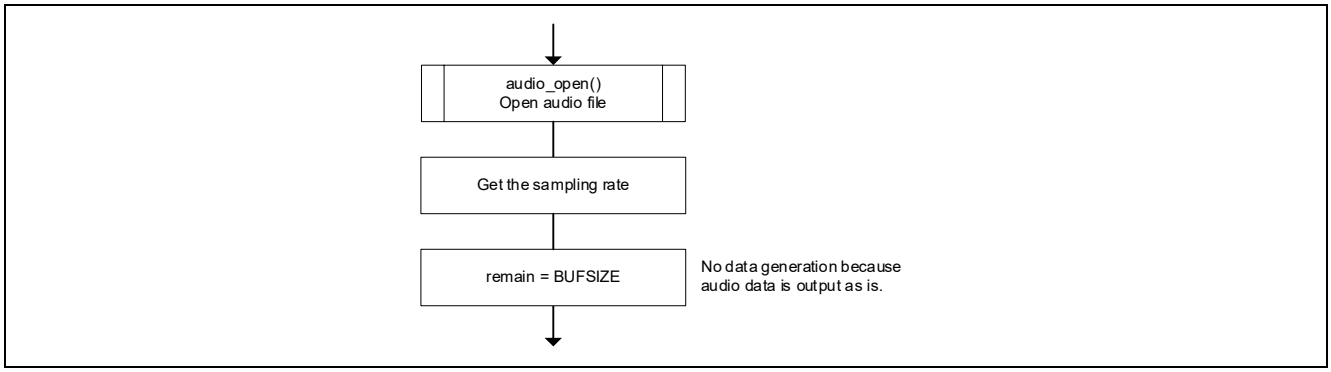


**Figure 3.6 Audio Player mode data initialization flow**

In the Audio Player mode, recorded audio data is read and audio information is set.

Figure 3.7 shows the Audio Player mode data processing flow.

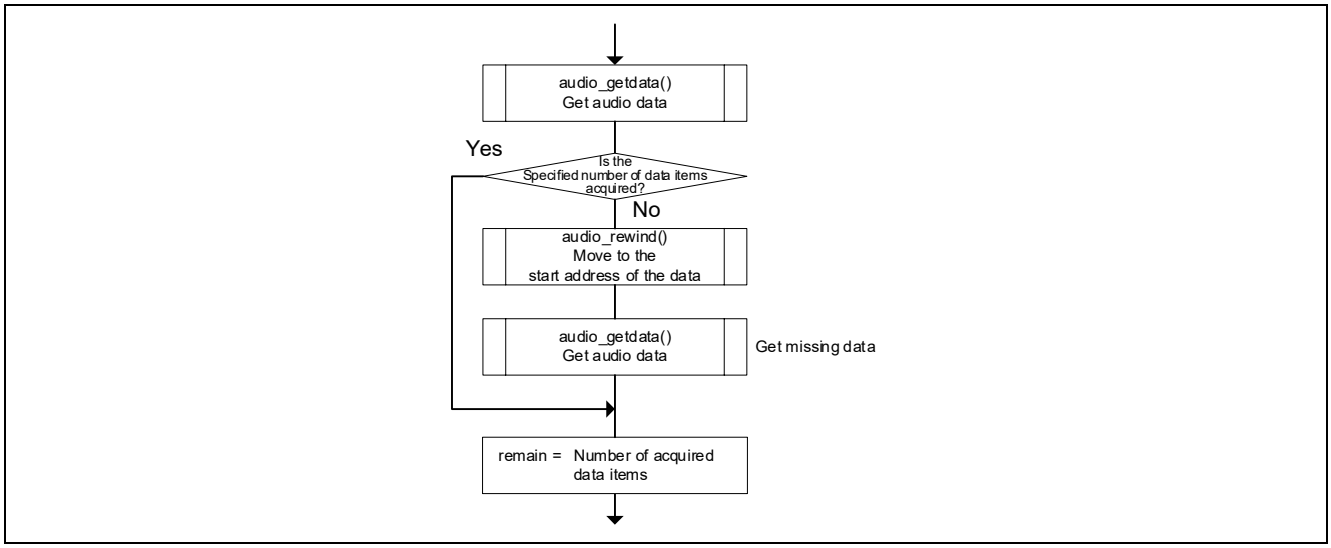


**Figure 3.7 Audio Player mode data processing flow**

## 3.2 File configuration

Figure 3.8 shows the Project file structure of the application note.

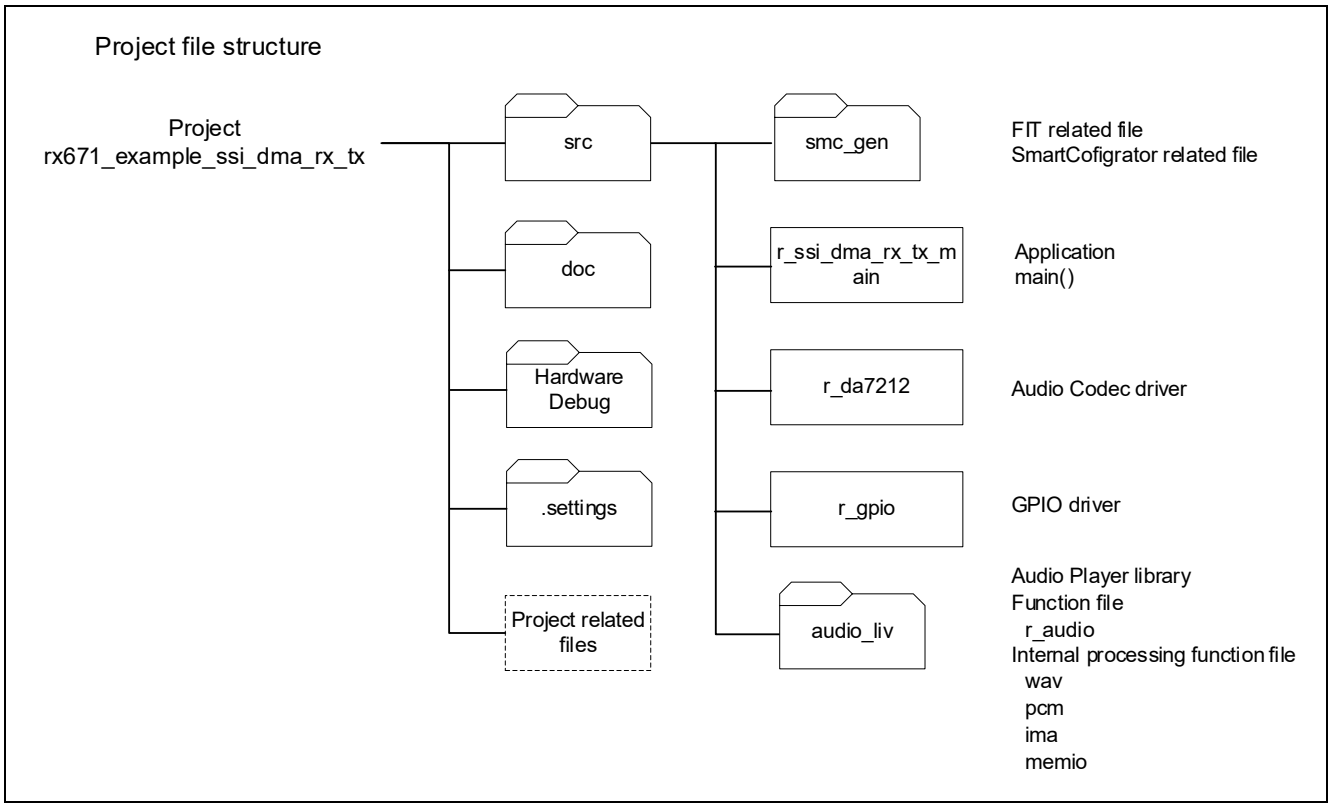


**Figure 3.8 Project file structure**

## 3.3 Function

### 3.3.1 Function configuration

Figure 3.9 shows the function configuration used in this program.

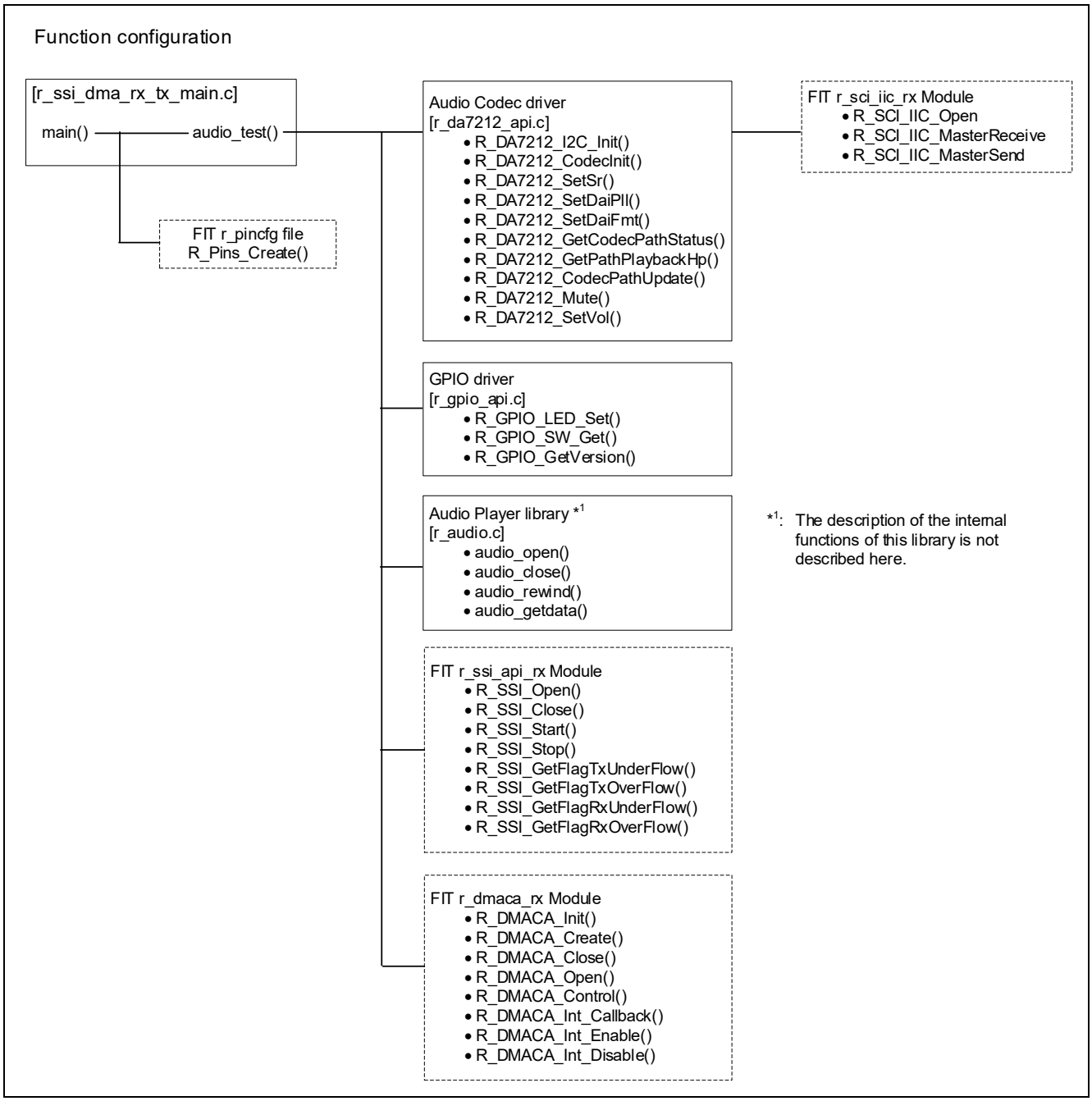


**Figure 3.9 Program function structure**

### 3.3.2 Application functions

Table 3.1 lists the application functions used in this application.

**Table 3.1 List of application functions**

| Function Name | Function | File |
|---|---|---|
| main | Entry function of this application | r_ssi_dma_rx_tx_main.c |
| audio_test | Audio processing body | |

| main | |
|---|---|
| **Function** | Entry function |
| **Format** | void main(void) |
| **Argument** | None |
| **Return value** | None |
| **Processing** | MCU pin definition: FIT Module R_Pins_Create()<br>Start audio processing body audio_test () |

| audio_test | |
|---|---|
| **Function** | Audio processing body |
| **Format** | int32_t audio_test(void) |
| **Argument** | None |
| **Return value** | -1: When an error occurs (During normal operation, the internal operation is repeated, so the function does not end.) |
| **Processing** | For details about the processing, see Figure 3.2 Audio application processing flow. |

### 3.3.3 Audio Codec driver functions

Table 3.2 lists the Audio Codec driver functions used in this application.

**Table 3.2 List of Audio Codec driver functions**

| Function Name | Function | File |
|---|---|---|
| R_DA7212_I2C_Init | Initializing I2C for communication with DA7212 | r_da7212_api.c |
| R_DA7212_CodecInit | Initializing DA7212 | |
| R_DA7212_SetSr | Setting the sampling rate for Audio Codec | |
| R_DA7212_SetDaiPll | Setting PLL for Audio Codec | |
| R_DA7212_SetDaiFmt | Setting the I2S format for Audio Codec | |
| R_DA7212_GetCodecPathStatus | Getting the current control register setting value of Audio Codec | |
| R_DA7212_GetPathPlaybackHp | Getting the control register setting value of headphone output of Audio Codec | |
| R_DA7212_Mute | Muting/unmuting functions of Audio Codec | |
| R_DA7212_SetVol | Setting the volume of Audio Codec | |
| R_DA7212_CodecPathUpdate | Register setting function for Audio Codec | |

| R_DA7212_I2C_Init | |
|---|---|
| **Function** | Initialize the IIC to communicate with DA7212 |
| **Format** | int32_t R_DA7212_I2C_Init(void) |
| **Argument** | None |
| **Return value** | 0: success, -1: error |
| **Processing** | Executes the FIT module R_SCI_IIC_Open(), and returns whether the processing was successful or failed. |

| R_DA7212_CodecInit | |
|---|---|
| **Function** | Initialize DA7212 |
| **Format** | int32_t audio_test(void) |
| **Argument** | None |
| **Return value** | 0: success, -1: error |
| **Processing** | Resets DA7212 by software, and sets all the control register settings to OFF. |

| R_DA7212_SetSr | |
|---|---|
| **Function** | Setting the sampling rate for Audio Codec |
| **Format** | int32_t R_DA7212_SetSr(int32_t sampling_rate) |
| **Argument** | int32_t sampling_rate Sampling rate |
| **Return value** | 0: success, -1: error |
| **Processing** | Sets the value corresponding to the specified sampling rate in the register. |

| R_DA7212_SetDaiPll | |
|---|---|
| **Function** | Setting the PLL for Audio Codec |
| **Format** | int32_t R_DA7212_SetDaiPll(int32_t sampling_rate, uint32_t sys_mclk) |
| **Argument** | int32_t sampling_rate Sampling rate<br>uint32_t sys_mclk SYS_MCLK |
| **Return value** | 0: success, -1: error |
| **Processing** | Calculates the appropriate PLL setting (based on sampling_rate and sys_mclk) and sets it in the register. |

| R_DA7212_SetDaiFmt | |
|---|---|
| **Function** | Setting the I2S format for Audio Codec |
| **Format** | int32_t R_DA7212_SetDaiFmt(int32_t data_width) |
| **Argument** | int32_t data_width Word length |
| **Return value** | 0: success, -1: error |
| **Processing** | Calculates the communication format (based on data_width and DA7212_CODEC_SLAVE) and sets it in the register. |

| R_DA7212_GetCodecPathStatus | |
|---|---|
| **Function** | Getting the current control register setting of Audio Codec |
| **Format** | void R_DA7212_GetCodecPathStatus(uint8_t * amp_tbl) |
| **Argument** | uint8_t * amp_tbl Control table pointer |
| **Return value** | None |
| **Processing** | Gets the control register setting. |

| R_DA7212_GetPathPlaybackHp | |
|---|---|
| **Function** | Setting headphone output and getting the control register value |
| **Format** | void R_DA7212_GetPathPlaybackHp(uint8_t * amp_tbl) |
| **Argument** | uint8_t * amp_tbl Control table pointer |
| **Return value** | None |
| **Processing** | Sets headphone output and gets the control register setting. |

| R_DA7212_Mute | |
|---|---|
| **Function** | Mute control for Audio Codec |
| **Format** | int32_t R_DA7212_Mute(e_hal_controls_t controls, int32_t mute) |
| **Argument** | e_hal_controls_t controls Control target<br>int32_t mute 0: unmute, 1: mute |
| **Return value** | 0: success, -1: error |
| **Processing** | Mutes or unmutes the specified target. |

| R_DA7212_SetVol | |
|---|---|
| **Function** | Setting the volume of Audio Codec |
| **Format** | int32_t R_DA7212_SetVol(e_hal_controls_t amp, double vol) |
| **Argument** | e_hal_controls_t controls Control target<br>doble vol Volume |
| **Return value** | 0: success, -1: error |
| **Processing** | Controls the volume of the specified module. |

| R_DA7212_CodecPathUpdate | | |
|---|---|---|
| **Function** | Register setting function for Audio Codec | |
| **Format** | int32_t R_DA7212_CodecPathUpdate(uint8_t * amp_tbl) | |
| **Argument** | uint8_t * amp_tbl | Control table pointer |
| **Return value** | 0: success, -1: error | |
| **Processing** | Sets the control table value in the register. | |

### 3.3.4 GPIO driver functions

Table 3.3 lists the GPIO driver functions used in this application.

**Table 3.3 List of GPIO driver functions**

| Function Name | Function | File |
|---|---|---|
| R_GPIO_LED_Set | Controlling LED lighting | r_gpio_api.c |
| R_GPIO_SW_Get | Getting the port status of the switch of the evaluation board | |
| R_GPIO_GetVersion | Getting the driver version | |

| R_GPIO_LED_Set | | |
|---|---|---|
| **Function** | Controlling LED lighting | |
| **Format** | int32_t R_GPIO_LED_Set(int32_t no, int32_t offon) | |
| **Argument** | int32_t no | LED number 1: LED1, 2: LED2, 3: LED3 |
| | int32_t offon | 0: LED OFF, 1: LED ON |
| **Return value** | 0: success, -1: error | |
| **Processing** | Performs H/L control for the specified port, and sets the LED to ON or OFF | |

| R_GPIO_SW_Get | | |
|---|---|---|
| **Function** | Getting the port status of the switch of the evaluation board | |
| **Format** | int32_t R_GPIO_SW_Get(int32_t no) | |
| **Argument** | int32_t no | SW number 1: SW1, 2: SW2 |
| **Return value** | 0/1: pin status, -1: error | |
| **Processing** | Gets the port status of the specified switch. | |

| R_GPIO_GetVersion | |
|---|---|
| **Function** | Getting the driver version |
| **Format** | uint32_t R_GPIO_GetVersion(void) |
| **Argument** | None |
| **Return value** | GPIO driver version<br>(Top 2 bytes are the major version number and the bottom 2 bytes are the minor version number.) |
| **Processing** | Gets the GPIO driver version. |

### 3.3.5 Audio Player library functions

Table 3.4 lists the dedicated functions used when executing this application in the Audio Player mode.

**Table 3.4 List of functions used when executing Audio Player mode**

| Function Name | Function | File |
|---|---|---|
| audio_open | Opening audio files | r_audio.c |
| audio_close | Closing audio files | |
| audio_rewind | Returning the playback position of the audio file to the top | |
| audio_getdata | Reading data from audio files | |

| audio_open | | |
|---|---|---|
| **Function** | Opening audio files | |
| **Format** | int32_t audio_open(const void *data, uint32_t size, int32_t ch, int32_t bits, audio_t *audio) | |
| **Argument** | const void *data | Audio file storage address |
| | uint32_t size | Audio file size |
| | int32_t ch | Number of output data channels (1: monaural, 2: stereo) |
| | int32_t bits | Number of quantization bits of output data |
| | audio_t *audio | Pointer to store the Audio handle |
| **Return value** | 0 | Normal |
| | -2 | The file could not be opened (AUDIO_E_FILE) |
| | -4 | Unsupported data format (AUDIO_E_NOSPT) |
| **Processing** | Opens an audio file, and stores its Audio handle on the audio structure. audio_rewind(), audio_getdata(), audio_close() are used with this handle. | |

| audio_close | | |
|---|---|---|
| **Function** | Closing audio files | |
| **Format** | int32_t audio_close(audio_t *audio) | |
| **Argument** | audio_t *audio | Audio handle |
| **Return value** | 0 | Normal |
| | -5 | Unknown error (AUDIO_E_UNKNOWN) |
| **Processing** | Closes the specified audio file. | |

| audio_rewind | | |
|---|---|---|
| **Function** | Returning the playback position of the audio file to the top | |
| **Format** | int32_t audio_rewind(audio_t *audio) | |
| **Argument** | audio_t *audio | Audio handle |
| **Return value** | 0 | Normal |
| | -5 | Unknown error (AUDIO_E_UNKNOWN) |
| **Processing** | Returns the playback position of the audio file to the top. | |

| audio_getdata | | |
|---|---|---|
| **Function** | Reading data from audio files | |
| **Format** | int32_t audio_getdata(audio_t *audio, void *p, uint32_t samples) | |
| **Argument** | audio_t *audio | Audio handle |
| | void *p | The storage position of the read-out audio data (Pointer) |
| | long samples | Maximum number of samples of data to read |
| **Return value** | A value greater than or equal to 0 | Number of samples of read data (No more data if 0) |
| | -2 | File open failure (AUDIO_E_FILE) |
| | -4 | Unsupported data format (AUDIO_E_NOSPT) |
| | -5 | Unknown error (AUDIO_E_UNKNOWN) |
| **Processing** | Reads the specified number of audio data items from the specified read position. | |

## 4. FIT module

Table 4.1 lists the API functions used in this project.

**Table 4.1 List of API functions to use**

| FIT module file name | FIT version | Contents |
|---|---|---|
| r_bsp | 7.42 | Board support package module (AN1685) |
| r_dmaca_rx | 3.20 | DMAC module (AN2063) |
| r_ssi_api_rx | 2.03 | SSI API module (AN2150) |
| r_sci_iic_rx | 2.71 | SCI simple I2C module (AN1691) |

For the API functions and their functional description, refer to relevant FIT application notes.

# 5. Importing a Project

After importing the sample code, make sure to confirm build and debugger setting.

## 5.1 Importing a Project into e² studio

Follow the steps below to import your project into e² studio. Pictures may be different depending on the version of e² studio to be used.

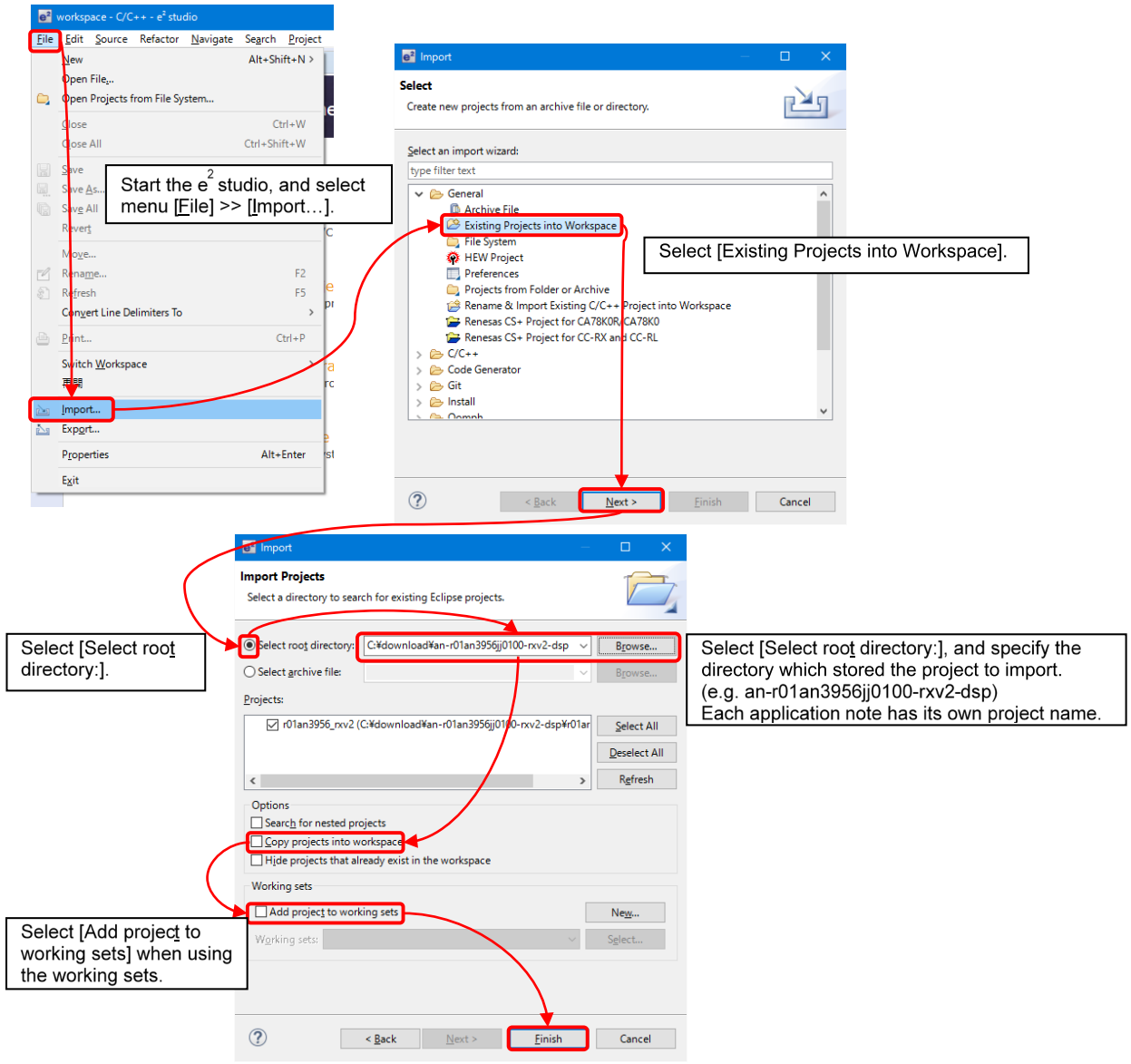


**Figure 5.1 Importing a Project into e² studio**

## 5.2 Importing a Project into CS+

Follow the steps below to import your project into CS+. Pictures may be different depending on the version of CS+ to be used.

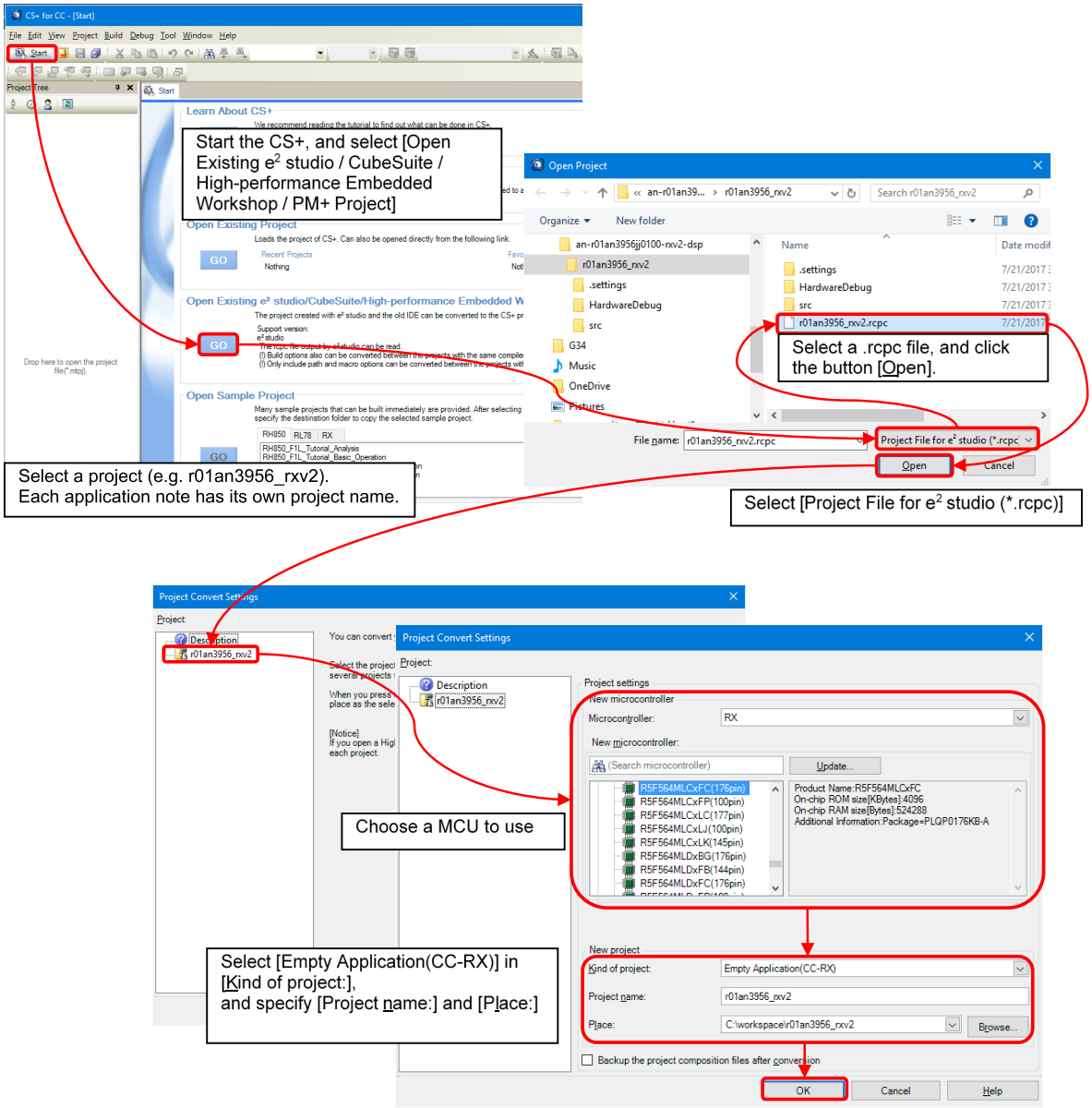


Figure 5.2 Importing a Project into CS+

## 6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

## 7. Reference Documents

Firmware Integration Technology Application notes

Firmware Integration Technology User's Manual (R01AN1833)
RX Family Board Support Package Module Firmware Integration Technology (R01AN1685)
RX Family SSI Module Firmware Integration Technology (R01AN2150)
RX Family Simple I2C Module Firmware Integration Technology (R01AN1691)
RX Family DMAC Module Firmware Integration Technology (R01AN2063)
PCM data transfer sample program Firmware Integration Technology using the RX Family SSI Module (R01AN2825)
(The latest versions can be downloaded from the Renesas Electronics website.)

Manuals, data sheets

RX671 Group User's Manual Hardware Edition (R01UH0899)
EK-RX671 v1 User's Manual (R20UT5234)
(The latest versions can be downloaded from the Renesas Electronics website.)

Technical updates, tool news

(The latest versions can be downloaded from the Renesas Electronics website.)

## Revision History

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Jan. 10, 2023 | - | First edition issued. |
| 1.01 | Feb. 13, 2024 | 14 | Fixed Figure 3.9. |
| | | 17 | Fixed format for R_DA7212_SetVol function. |
| | | 18 | Added R_GPIO_GetVersion function. |
| | | Program | Fixed define for I/O port corresponding to SW2.<br>Fixed file name for Audio Codec driver and GPIO driver. |

# General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

# 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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

## 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/.