Renesas Synergy™ Platform

JPEG Decode 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 JPEG Decode HAL module is a generic API for JPEG decode processing implemented on `r_jpeg`. The JPEG Decode HAL module supports the Synergy™ JPEG Codec peripheral. The JPEG Decode Framework Module is a ThreadX®-aware high-level API for JPEG Framework module applications and is implemented on `sf_jpeg_decode`; it provides thread-safe access to the Synergy JPEG hardware on a Synergy MCU Group. A user-defined callback can be created to detect hardware supported events.

Contents

1. JPEG Decode Framework Module Features

The JPEG Decode Framework module has the following features:

- Provides thread-safe access to the Synergy JPEG hardware.
- Supports JPEG decompression using the JPEG Decode HAL module.
- Supports a polling mode that allows an application to wait for the JPEG Decoder to complete.
- Supports an interrupt mode with user-supplied callback functions.
- Configures parameters such as horizontal and vertical subsample values, horizontal stride, decoded pixel format, input and output data format and color space.
- Obtains the size of the image prior to decoding it.
- Supports putting coded data in an input buffer and an output buffer to store the decoded image frame.
- Supports streaming coded data into the JPEG Decoder module. This feature allows an application to read a coded JPEG image from a file or from a network without buffering the entire image.
- Configures the number of image lines to decode. This feature enables the application to process the decoded image on the fly without buffering the entire frame.
- Supports the input decoded formats YCbCr444, YCbCr422, YCbCr420, and YCbCr411.
- Supports the output formats ARGB8888 and RGB565.
- Returns an error when the JPEG image's size, height, and width do not meet the requirements.
- Supports the wait API function to suspend/resume the thread for synchronizing with the JPEG hardware supported events.

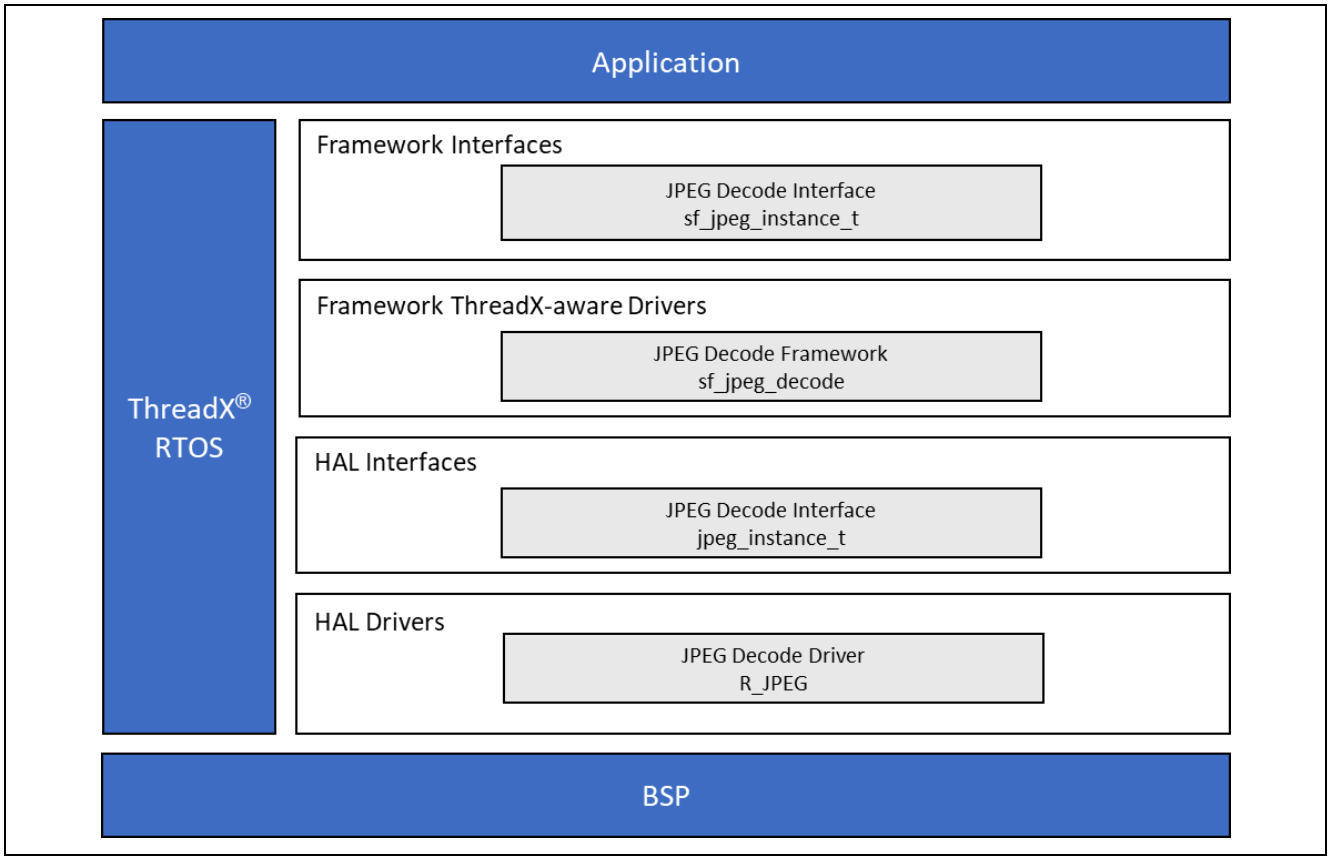


Figure 1. JPEG Decode Framework Module Block Diagram

2. JPEG Decode Framework Module APIs Overview

The JPEG Decode Framework module defines APIs for opening, closing, setting alarms, and starting and stopping RTC operations. 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. JPEG Decode Framework Module API Summary

Function Name	Example API Call and Description
.open	`g_sf_jpeg_decode0.p_api->open(g_sf_jpeg_decode0.p_ctrl, g_sf_jpeg_decode0.p_cfg);` Open the JPEG Decode Framework.
.inputBufferSet	`g_sf_jpeg_decode0.p_api->inputBufferSet(g_sf_jpeg_decode0.p_ctrl, p_buffer, size);` Assign input buffer to JPEG codec for storing input data
.outputBufferSet	`g_sf_jpeg_decode0.p_api->outputBufferSet(g_sf_jpeg_decode0.p_ctrl, p_buffer, buffer_size);` Assign output buffer to JPEG codec for storing output data
.linesDecodedGet	`g_sf_jpeg_decode0.p_api->linesDecodedGet(g_sf_decode0.p_ctrl, p_lines);` Return the number of lines decoded into the output buffer
.horizontalStrideSet	`g_sf_jpeg_decode0.p_api->horizontalStrideSet(g_sf_jpeg_decode0.p_ctrl, stride);` Configure the horizontal stride value
.imageSubsampleSet	`g_sf_jpeg_decode0.p_api->imageSubsampleSet(g_sf_jpeg_decode0.p_ctrl, horizontal, vertical);` Configure the horizontal and vertical subsample settings
.wait	`g_sf_jpeg_decode0.p_api->wait(g_sf_jpeg_decode0.p_ctrl, p_status, timeout);` Wait for the current JPEG codec operation to finish with a timeout value given in ThreadX ticks
.statusGet	`g_sf_jpeg_decode0.p_api->statusGet(g_sf_jpeg_decode0.p_ctrl, p_status);` Retrieve current status of the JPEG codec module
.imageSizeGet	`g_sf_jpeg_decode0.p_api->imageSizeGet(g_sf_jpeg_decode0.p_ctrl, p_horizontal, p_vertical);` Retrieve image size during decoding operation
.pixelFormatGet	`g_sf_jpeg_decode0.p_api->pixelFormatGet(g_sf_jpeg_decode0.p_ctrl, p_color_space);` Get the input pixel format
.close	`g_sf_jpeg_decode0.p_api->close(g_sf_jpeg_decode0.p_ctrl);` Cancel an outstanding operation
.versionGet	`g_sf_jpeg_decode0.p_api->versionGet(&version);` Get version and store it in provided pointer p_version

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 Manual,* API References for the associated module.

Table 2. Status Return Values

Name	Description
SSP_SUCCESS	JPEG Decode driver is successfully opened.
SSP_ERR_ASSERTION	Assertion error.
SSP_ERR_IN_USE	Module already in use.
SSP_ERR_TIMEOUT	The wait operation times out, the underlying driver did not respond in time.
SSP_ERR_WAIT_ABORTED	System internal error occurred.

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. JPEG Decode Framework Module Operational Overview

The JPEG Decode Framework module implements the standard JPEG decode operation. It takes the data in an input buffer and applies the defined JPEG decode algorithm to the buffer; the output is then delivered to the defined output buffer location. A `wait` API function can be used to suspend/resume the thread for synchronization with JPEG hardware supported events.

3.1 JPEG Decode Framework Module Important Operational Notes and Limitations

3.1.1 JPEG Decode Framework Module Operational Notes

- You can start decoding JPEG-encoded data by calling the `open` API. To open the module, use the JPEG Decode Framework module instance that includes the API function pointer, the pointer to the control block and static configuration that is generated through the Synergy Project configurator in the e^2 studio for ISDE.
- Stop the JPEG Decode Framework module by calling the `close` API.
- An input buffer-streaming mode is available when an input-centric function is needed.
- An output buffer-streaming mode is available when an output-centric function is needed.
- Supports RGB565 and ARGBB888 output data-color formats.
- The JPEG Decode Framework module has a status flag in the control block; you can get the current status of the module through the `statusGet` API. The status is also reported through a user-callback function when specific events occur in the module.
- The JPEG Decode Framework module supports buffer-streaming mode for the input buffer in cases when the input buffer is smaller than the source image size. Set the next input frame as an input buffer every time there is a hardware-generated `INPUT_PAUSE` interrupt.
- The JPEG Decode Framework module supports buffer-streaming mode for the output buffer in cases when the resulting image is larger than the output buffer-size. Read and store data from the output buffer to make space for upcoming data every time there is a hardware generated `OUTPUT_PAUSE` interrupt.
- The input and output buffers should be 8-bytes aligned for the JPEG Decode Framework module to be successful. Otherwise, the APIs return error codes that indicate unsuccessful execution.

3.1.2 JPEG Decode Framework Module Limitations

The JPEG Framework module does not support JPEG-encode processing. Refer to the latest *SSP Release Notes* for any additional operational limitations for this module.

4. Including the JPEG Decode Framework Module in an Application

This section describes how to include the JPEG Decode 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 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 JPEG Decode 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 JPEG Decode Framework module is `g_sf_jpeg_decode0`. This name can be changed in the associated Properties window.)

Table 3. RTC Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
`g_sf_jpeg_decode0` JPEG Framework	Threads	New Stack> Framework> Graphics> JPEG Decode Framework on `sf_jpeg_decode`

When the JPEG Decode Framework module on `sf_jpeg_decode` is added to the thread stack as shown in the following figure, the configurator automatically adds the needed lower-level drivers. Any drivers that need additional configuration information will be highlighted in Red. The specific settings required can be viewed by hovering the cursor over the highlighted text or as suggested in the highlighted stack frame. For this stack, the reported requirements for the JPEG HAL module are to enable the decompression-interrupt priority and the data-transfer interrupt-priority interrupts.

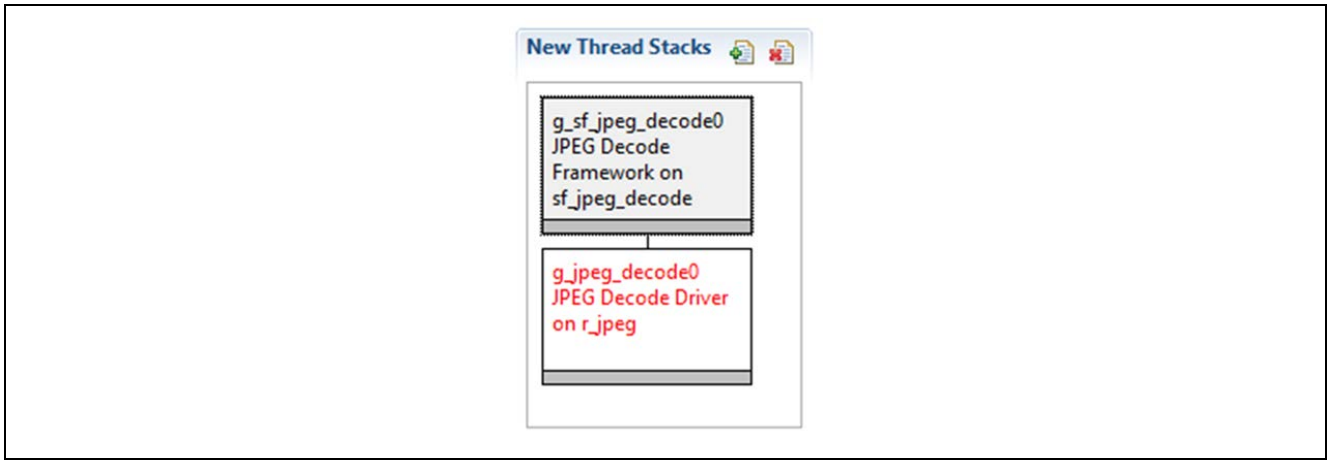


Figure 2. JPEG Decode Framework Module Stack

Decompression Process Interrupt (JEDI)

The JPEG decompression-process interrupt occurs when:

- The current decompression process is successfully completed.
- An error happens in the decompression process.
- Image size and pixel format are successfully read out.

Data Transfer Interrupt (JDTI)

The JPEG data-transfer interrupt occurs when:

- All the JPEG-coded data has successfully completed.
- The number of output image-data lines specified by `linesDecodedGet` has been transferred.
- The number of input image-data lines specified by `inputBufferSet` has been transferred.

5. Configuring the JPEG Decode Framework Module

The JPEG Decode Framework module must be configured by the user 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, that must be configured for lower-level modules for successful operation. Furthermore, 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 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 shown in the following tables for easy reference.

Note: You may want to open your ISDE, create the JPEG Decode Framework module and explore the property settings in parallel with looking over the configuration table settings in the following table. 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 for the JPEG HAL Module on r_jpeg

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enable or disable the parameter error checking.
Name	`g_jpeg_decode0`	The name to be used for a JPEG Decode module instance.
Byte Order for Input Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word-Byte Swap (7)(8)(5)(6)(3)(4)(2)(1) (Default: Normal Byte order)	Specify the byte order for input data. The order is swapped as specified in every 8-byte.
Byte Order for Output Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word-Byte Swap (7)(8)(5)(6)(3)(4)(2)(1) (Default: Normal Byte order)	Specify the byte order for output data. The order is swapped as specified in every 8-byte.
Output Data Color Format	Pixel Data RGB565 format, Pixel Data ARGBB888 format (Default: Pixel Data RGB565 format)	Specify the output data format.
Alpha value to be applied to decoded pixel data (only valid for ARGB8888 format)	255	Specify the alpha value for the output data format (only valid for ARGB8888 format).
Name of user callback function	NULL	Specify the name of user callback function.
Decompression Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	JPEG JEDI Interrupt selection
Data Transfer Interrupt priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	JPEG JDTI Interrupt selection

5.1 JPEG Decode Framework Module Clock Configuration

The JPEG Framework module uses the peripheral module `clock A (PCLKA)` to run the internal logic.

5.2 JPEG Decode Framework Module Interrupt Configuration

To enable interrupts, set the priority of the decompression interrupt and the data-transfer interrupt in the Properties window of the JPEG Decode Framework module in the ISDE.

5.3 JPEG Decode Framework Module Pin Configuration

The JPEG Decode Framework module does not use any pins.

6. Using the JPEG Decode Framework Module in an Application

The typical steps in using the JPEG Decode Framework module in an application are:

1. Initialize the JPEG Decode peripheral using the `open` API.
2. Set image subsample using the `imageSubSampleSet` API.
3. Set horizontal stride using the `horizontalStrideSet` API.
4. Set output buffer using the `outputBufferSet` API.
5. Set input buffer using the `inputBufferSet` API.
6. Wait for decode to complete with the `wait` API.
7. Check decode status with the `statusGet` API.
8. Close the instance with the `close` API (if needed).

The following figure illustrates these steps in a typical operational flow.

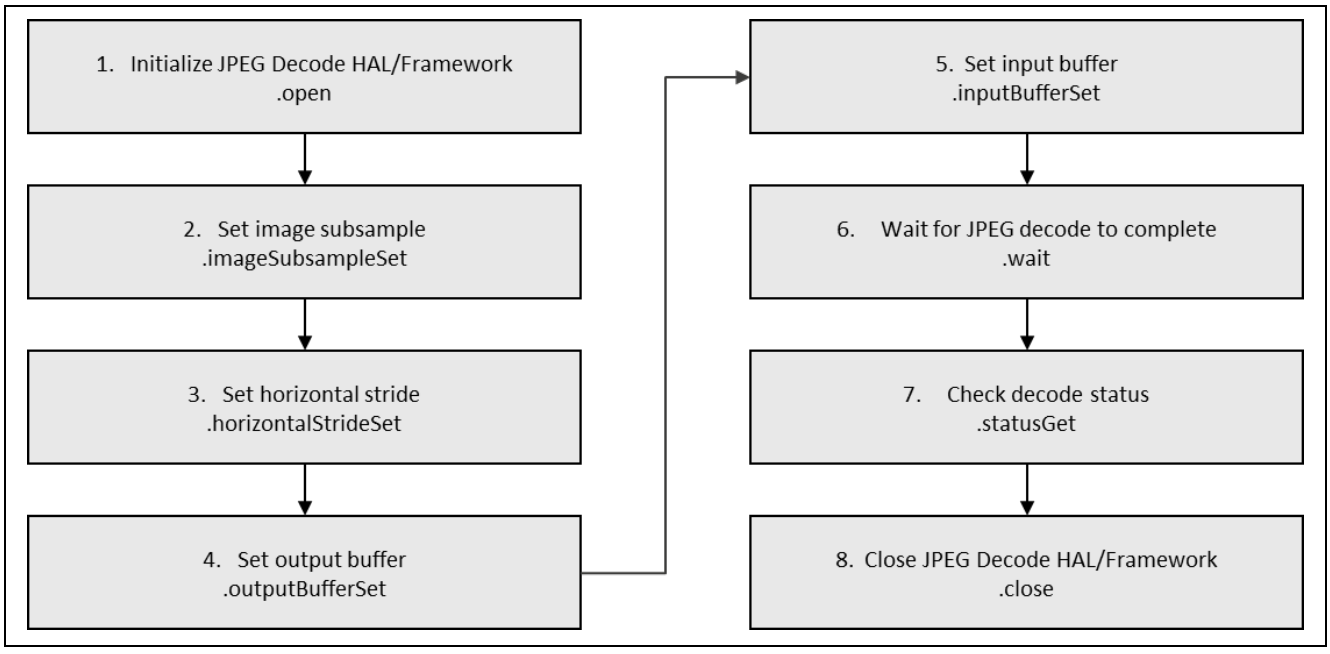


Figure 3. Flow Diagram of a Typical JPEG HAL/Framework Module Application

7. The JPEG Decode Framework Module Application Project

The application project associated with this module guide demonstrates the illustrated steps in a full design. The project can be found using the link provided in the References section at the end of this document. You may want to import and open the application project in the ISDE and view the configuration settings for the JPEG Decode Framework module. You can also read over the code in `jpeg_decodeimage_mg.c/.h`, `jpeg_decode_hal_api_mg.h`, `jpeg_decode_framework_api_mg.h` and `jpeg_decode_framework_thread_entry.c`. These files are used to illustrate the JPEG Decode Framework APIs in a complete design.

The application project demonstrates the typical use of the JPEG Decode Framework APIs. The application project main-thread entry initializes the JPEG Decode HAL and Framework modules in respective entry functions and decodes two different images stored in `sample_image1.c` and `sample_image2.c`; the output can be viewed in the ISDE's memory tab during debug operation. A user-callback function is entered when an event occurs during JPEG decoding. The user-specified callback function sets flags for executing programs to determine the course of actions. The following table identifies the target versions for the associated software and hardware used by the application project:

Table 5. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e² studio	6.2.1 or later	Integrated Solution Development Environment
SSP	1.5.0 or later	Synergy Software Platform
IAR EW for Synergy	8.23.1 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	6.2.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

A simple flow diagram of the application project is given in the following figures.

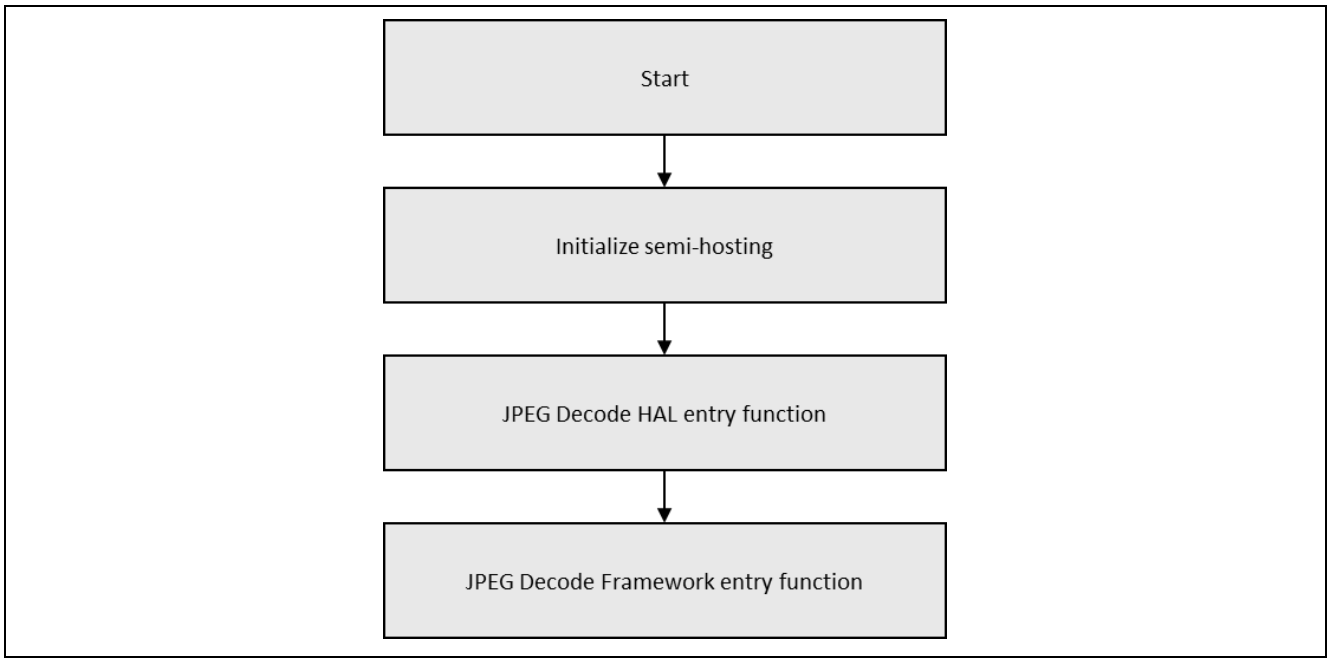


Figure 4. JPEG Decode Framework Module Application Project Flow Diagram

The complete application project can be found using the link provided in the References section at the end of this document. The `jpeg_decodeimage_mg.c/.h, jpeg_decode_hal_api_mg.h, jpeg_decode_framework_api_mg.h` and `jpeg_decode_framework_thread_entry.c` files are located in the project once it has been imported into the ISDE. You can open these files within the ISDE and follow along with the description provided to help identify key uses of APIs. Descriptions of what each of the files contain is given as well.

The `jpeg_decode_hal_api_mg.h` file is a header file that has inline functions that use HAL APIs for JPEG decoding. Similarly, `jpeg_decode_framework_api_mg.h` is a header file that has inline functions that use framework APIs provided in the first table of this module guide.

The `sample_image1.c and sample_image2.c` image files are stored in hexadecimal array format. The extended dump and code utility software tool is used to convert a JPEG image to hexadecimal C array (link is provided as described in the References section.) The `sample_image1.c` data is decompressed by the HAL APIs and the `sample_image2.c` data is decompressed by the framework decode application in this application project.

The `jpeg_decode_image_mg.h` file contains a function prototype, macros, and a declaration of flags that are used by the callback function.

The `jpeg_decode_image_mg.c` file contains the application code which uses HAL and Framework APIs to decode JPEG images. This file also contains the callback function to set flags when the JPEG decode events occur. This file includes code for JPEG decompression using the HAL and framework modules. The `jpeg_Decode_HAL_run()` function is an entry point to JPEG HAL decoding. The following figure explains the flow of this function.

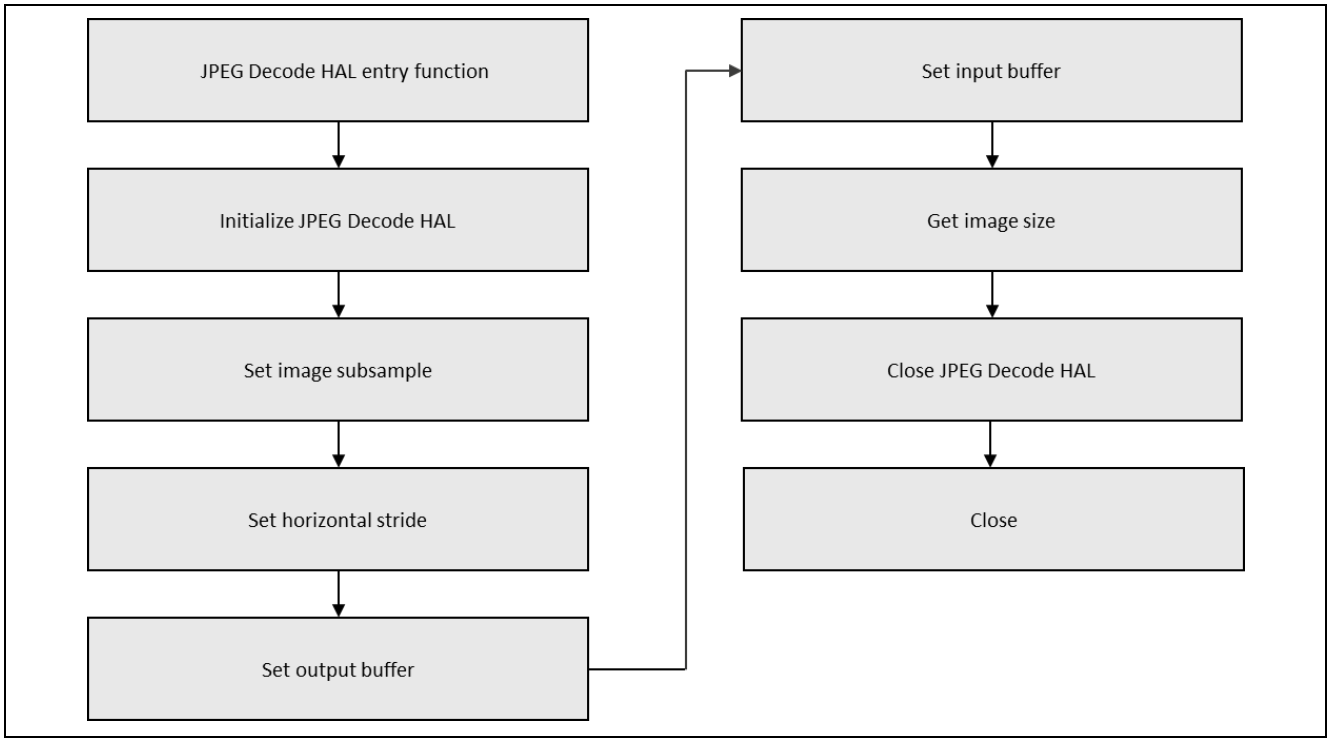


Figure 5. JPEG Decode Framework Module Application Project Flow Diagram

The `jpeg_Decode_Framework_run()` function is an entry point to JPEG HAL decoding. The following figure explains the flow of this function.

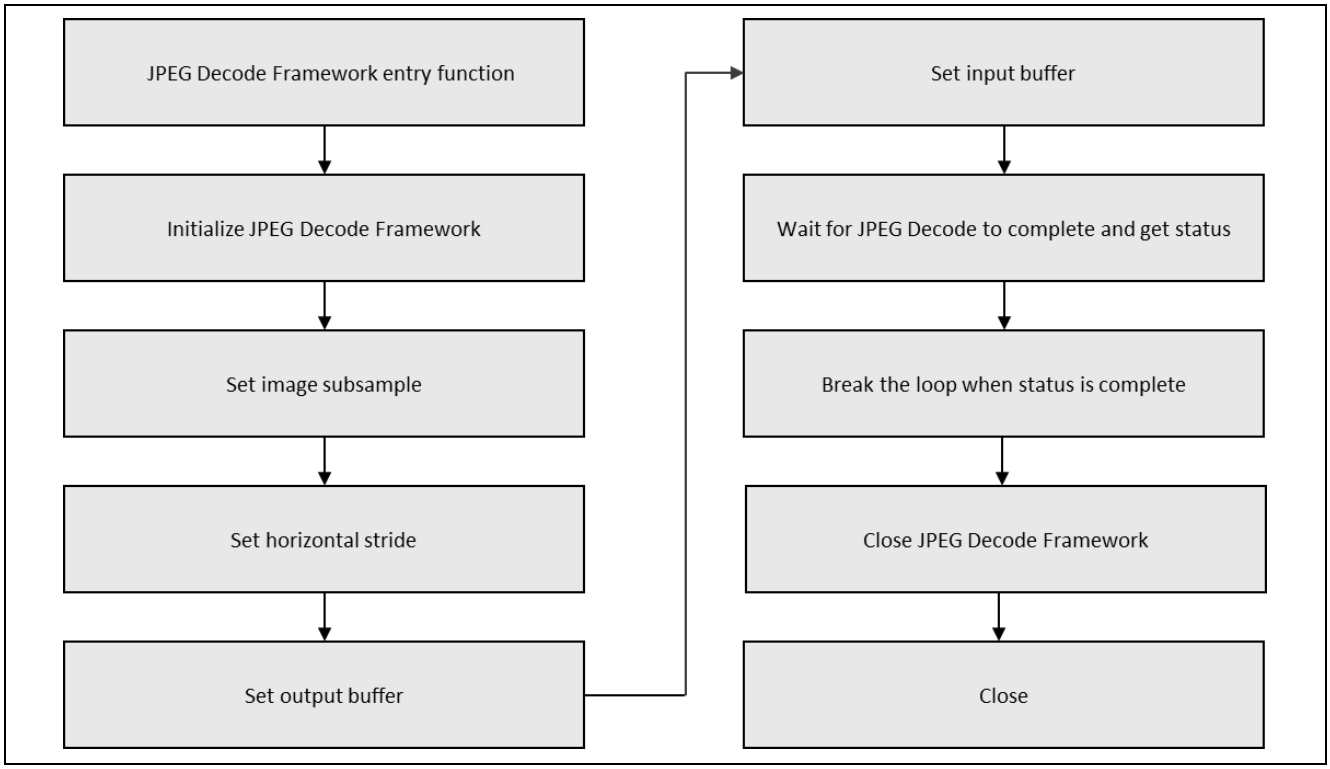


Figure 6. JPEG Decode Framework Module Application Project Flow Diagram

Note: This description assumes that you are familiar with using `printf()` with the Debug Console in the Synergy Software Package. If you are unfamiliar with this, refer to the How do I Use Printf() with the Debug Console in the Synergy Software Package Knowledge Base article, available as described in the References section at the end of this document. Alternatively, the user can see results via the watch variables in the debug mode.

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

Table 6. JPEG Decode Framework Module Configuration Settings for the Application Project

Resource	ISDE Property	Value Set
g_sf_jpeg_decode0 JPEG Decode Framework on r_jpeg	Parameter Checking	Enable
g_jpeg_decode0 JPEG Decode Driver on r_jpeg	Parameter Checking	Enabled
	Name	g_jpeg_decode0
	Byte Order for Input Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8)
	Byte Order for Output Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8)
	Output Data Color Format	Pixel Data RGB565 format
	Alpha value to be applied to decoded pixel data (only valid for ARGB8888 format)	255
	Name of user callback function	NULL (Locked by upper framework)
	Decompression Interrupt Priority	Priority 3
	Data Transfer Interrupt priority	Priority 3

Table 7. JPEG Decode HAL Module Configuration Settings for the Application Project

Resource	ISDE Property	Value Set
g_jpeg_decode1 JPEG Decode Driver on r_jpeg	Parameter Checking	Enabled
	Name	g_jpeg_decode1
	Byte Order for Input Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8)
	Byte Order for Output Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8)
	Output Data Color Format	Pixel Data RGB565 format
	Alpha value to be applied to decoded pixel data (only valid for ARGB8888 format)	255
	Name of user callback function	Jpeg_decode_callback
	Decompression Interrupt Priority	Priority 3
	Data Transfer Interrupt priority	Priority 3

8. Customizing the JPEG Decode Framework Module for a Target Application

Some configuration settings are normally changed by the developer from those shown in the application project. For example, the user can easily change the byte order for the input/output buffer, output-data format or the alpha value to be applied to the decoded pixel data. The user can also change the horizontal stride value, image subsample value, and input/output image size through the APIs.

9. Running the JPEG Decode Framework Module Application Project

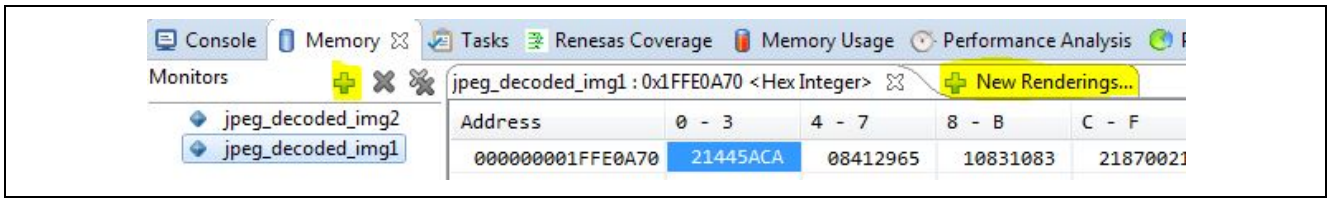
To run the JPEG Framework module application project and to see it executed on a target kit, you can simply import it into your ISDE, compile and run debug. See *Importing a Renesas Synergy Project.* (r11an0023eu0121-synergy-ssp-import-guide.pdf, included in this package) for instructions on importing the project into ISDE or IAR EW for Synergy and building/running the application.

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 JPEG Decode Framework module application project, simply follow these steps:

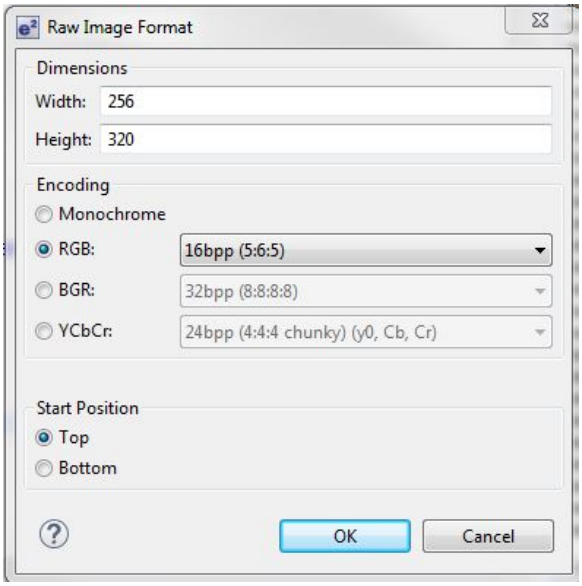
1. Import the attached example project `JPEG_Decode_MG_AP` to the ISDE or IAR EW for Synergy. For steps to import the example project, refer to the *Importing a Renesas Synergy Project*.
2. Compile the application without errors and warnings.
3. Connect to the host PC via a micro USB cable to J19 on SK-S7G2 Synergy MCU Group.
4. Start to debug the application.
5. LED1-3 toggle periodically when communication is ongoing. To view the decoded image, follow these steps:

   A. While the application example is running in debug mode, go to the Memory tab shown in the following figure and click the + sign to add the address locations jpeg_decoded_img1 and jpeg_decoded_img2. (Output buffer starting addresses).



   B. Click on tab **New Renderings** for each memory address and double click on the **Raw Image** option and click on **Raw Image Format** button.

   C. Fill in the information displayed in the following figure. The pop-up window opens after step 2 and press Ok.



   D. Observe the following outputs (output 1 and output 2 respectively).

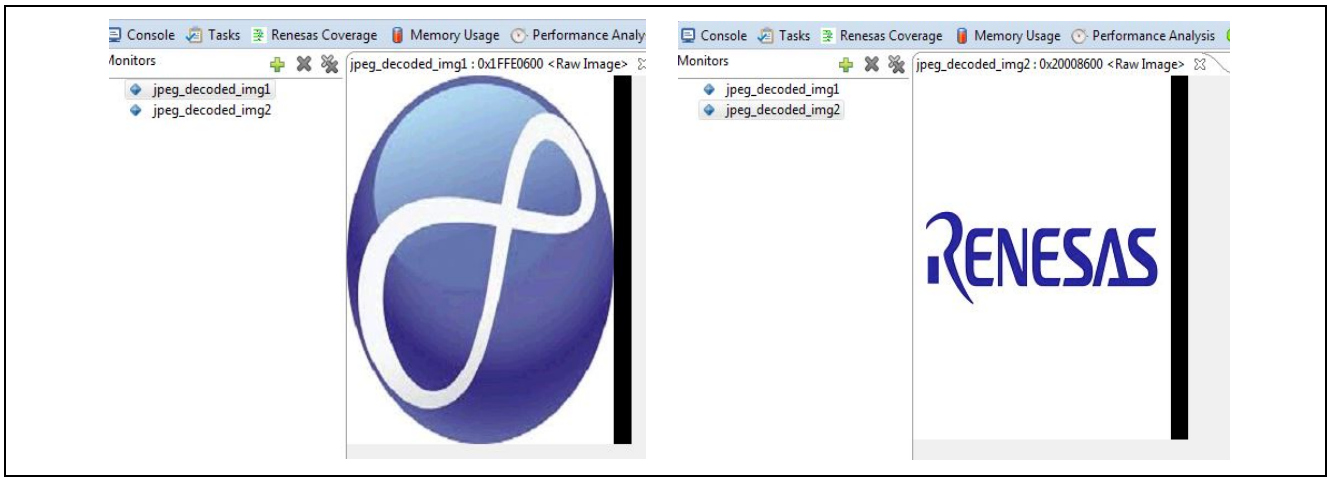


Figure 7. Example Output from JPEG Decode Framework Module Application Project

10. JPEG Decode Framework Module Conclusion

This module guide has provided all the background information needed to select, add, configure and use the JPEG Framework 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, such as conflicting configuration settings or the incorrect selection of lower-level drivers. The use of high-level APIs (as demonstrated in the 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, or in some cases, create lower-level drivers.

11. JPEG Decode Framework Module Next Steps

After you have mastered a simple JPEG Decode project, you may want to use these APIs in your example that decodes JPEG files and uses an LCD to display the decompressed image. Also, try using this example with streaming input and output buffers, high decompression settings, or images with different color formats.

12. JPEG Decode Framework Module Reference Information

SSP User Manual: Available in html format in the SSP distribution package and as a pdf from the Synergy Gallery (www.renesas.com/synergy/software).

Links to all the most up-to-date sf_jpeg_decode module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977548.

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	Aug.01.17	-	Initial Release
1.01	Jan.04.19	-	Migration for SSP 1.5.0

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.