Renesas Synergy™ Platform

Migrating Synergy Software Projects from SSP v1.0.0 to SSP v1.1.0

R20AN0412EU0101
Rev.1.01
Nov 18, 2016

Introduction

This application note describes how to migrate your Renesas Synergy projects from Renesas Synergy Software Package (SSP) v1.0.0 to SSP v1.1.0. More information on this topic is available in this Synergy Knowledge Base article.

Goals and Objectives

The objective of this document is to detail the process of migrating a project built in e^2 studio 4.2.0.012 with SSP v1.0.0 to a project compatible with e^2 studio 5.0.0.043 with SSP v1.1.0.

Prerequisites

The reader of this application note is assumed to have some experience with the Renesas e^2 studio ISDE and SSP and have a project you wish to migrate from SSP v1.0.0 to SSP v1.1.0.

Required Resources

To perform the procedures in this application note, you will need:

- A PC running Microsoft® Windows® 7 with the following Renesas software installed:
  - e^2 studio ISDE 4.2.0
  - e^2 studio ISDE 5.0.0
  - Synergy Software Package (SSP) 1.0.0
  - Synergy Software Package (SSP) 1.1.0
  - An existing Synergy project built in e^2 studio 4.2.0.012 (referred to as e^2 studio 4.2 in this document) with SSP v1.0.0

You can download the required Renesas software from the Renesas Synergy Gallery (https://synergygallery.renesas.com).

Time Required

You can perform the procedure in this application note in approximately one hour for a typical design. A very complex design (using the Messaging Framework and/or GUIX extensively) could take much longer; a less complex design (without the messaging Framework or GUIX) might take less than one hour. The high-level steps involved are:

1. Back up your existing SSP 1.0.0 project.
2. Open your existing e^2 studio 4.2 (SSP 1.0.0) project in e^2 studio 5.0.
3. Update the SSP license path.
4. Migrate the Synergy Configuration for your project.
5. If your SSP 1.0.0 project uses the Messaging Framework, migrate the Messaging Framework.
6. The GUIX version used by SSP v1.1.0 has been updated. If your SSP 1.0.0 project uses GUIX, migrate to GUIX 5.3.0.

The remainder of this application note describes the above steps in detail.

Contents

1. Create a Backup of the SSP v1.0.0 Project

IMPORTANT! Back up your project before beginning the migration procedure. This migration procedure involves multiple manual steps, and therefore you must back up the project before migration so that the known working state can be recovered if necessary. Do not skip this step!

To create a backup of your project, open your project built in e² studio 4.2 and export a known working backup:

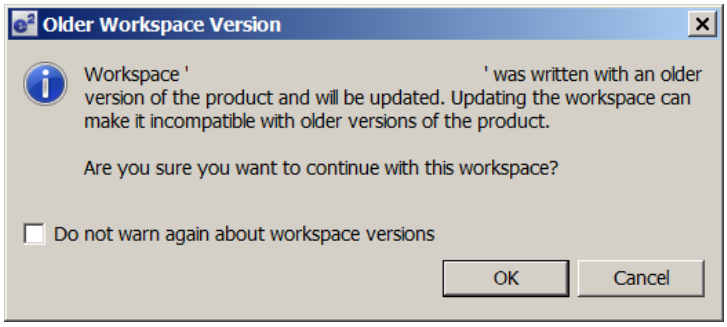
1. Open your project in e² studio 4.2 with SSP 1.0.0 installed.
2. Build the project and run it if desired to confirm it is in a known working state.
3. Go to **File > Export**.
4. Under **General**, select **Archive File**. Click **Next**.
5. Select the button **Select All**.
6. Browse to a location to store your backup. We recommend C:\Renesas\Backups\<project_name>_1.0.0.zip, where <project_name> is the name of your project. Any other location you will remember is also acceptable.
7. Click **Finish**.
8. Open configuration.xml (Synergy Configuration tool). Click the **Threads** tab.
9. Check all Threads to see if you are using the **Communications Framework on sf_el_nx_comms**. If you are using this module, write down which thread it is in and all of its properties from the Properties window. You will need to add this module to the migrated project manually.

2. Open your e² studio 4.2 Project in e² studio 5.0

IMPORTANT! Back up your project as described in Section 2 before beginning this section.

Open your project built in e² studio 4.2 in e² studio 5.0. Some module stacks may require a manual process to complete the project migration. The following example shows how to update the HMI Thermostat Application:

1. Open e² studio 5.0. You can either use the same workspace folder you used for e² studio 4.2 and migrate it or create a new workspace for e² studio 5.0. If you choose to migrate your e² studio 4.2 workspace, you will see the message below. Click **OK** to continue.



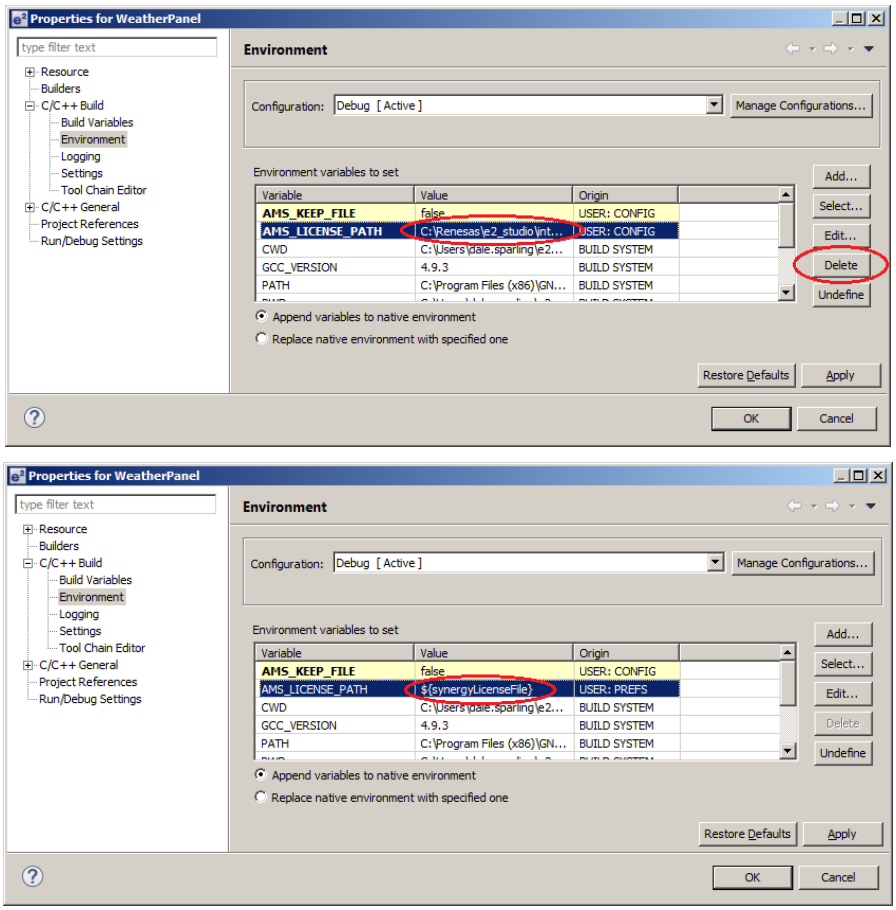
3. Update your SSP License Path

e² studio 4.2 stored the path to the SSP license in each project file in an environment variable called AMS_LICENSE_PATH. This variable was filled out on the first screen of the New Project wizard and could be modified using a screen found in the project settings at **Project > Properties > C/C++ General > Synergy License**.

With e² studio 5.0, a new default SSP license path is stored in the workspace. This path is set for the first time in the New Project wizard just like before, but the screen that modifies this path has been moved from the project settings to the workspace settings at **Window > Preferences > C/C++ > Renesas > Synergy License**.

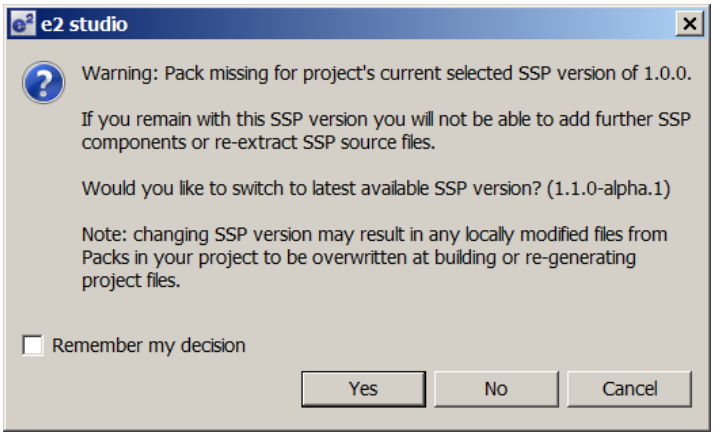
Projects in e² studio 5.0 still hold the AMS_LICENSE_PATH for those cases where a specific license path may be desired for the project. To use the default path defined in the workspace this environment variable should be set to the special macro - ${synergyLicenseFile}.

All projects migrated from e² studio 4.2 will have absolute license paths stored in the project file. To replace these absolute paths with the macro, delete the value that is there and the special macro will appear in its place.

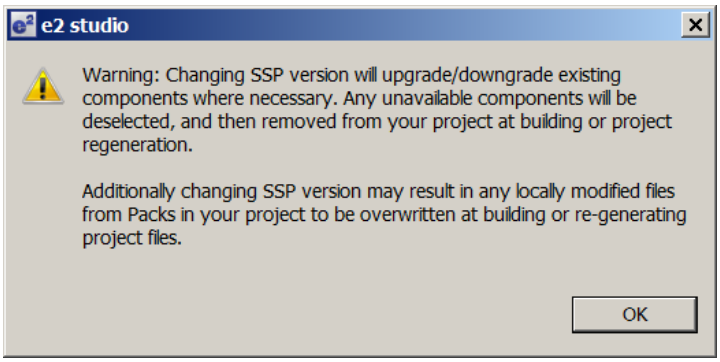


4. Migrate the Synergy Configuration

1. Open configuration.xml in the root of the project.
2. You will see the following message. Click **Yes** to update the project.



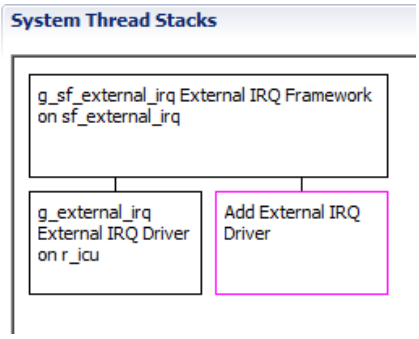
3. You will see the following warning message. Click **OK**.



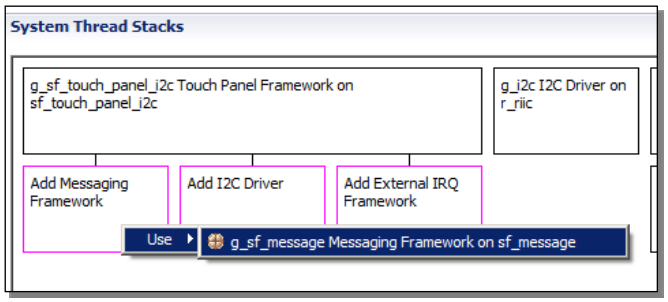
4. Select **File** > **Save** to save the updated configuration.xml (Synergy Configuration).
5. Close the Synergy Configuration tab. Reopen configuration.xml to make the Synergy Configuration tab visible.
6. Click on the **Threads** tab.

   Note: If you do not close and reopen the Synergy Configuration tool as described in the previous step, you may see prompts to add modules to a stack even though the stack is already complete.

   An example of this is shown in the image below. There is already an External IRQ Driver in the stack, and there is an extra prompt to Add External IRQ Driver. If you see extra prompts (like those in the example below), close and reopen the Synergy Configuration tool.



7. If you were using the **Communications Framework on sf_el_nx_comms** before (as identified in Section 1), add the module to the thread it was in before migration and configure it in the Properties window.
8. Highlight a thread. Boxes with a pink color outline may need to be updated. Click the box and select **Use** if available to complete the stack. If **Use** is not an option, check to see if the module is optional. Some boxes will say they are optional in the prompt. For example, if you are using a UART driver you may see “Add DTC Driver for Transmission [Recommended but optional]”. If the module is optional and you were not using it in your previous project, do not add it yet. If the module is not explicitly marked as optional in the pink box, and **Use** is not available (this is uncommon during project migration), select **Add** to add the required module. Configure any newly added modules in the properties window.



9. Complete the process described in the previous step on all threads in the project.
10. Review all threads in the project and confirm that the only pink boxes remaining are optional modules that are not used in your project.

    If your project does not use the Messaging Framework or GUIX, you have completed the project migration. You

can build and run the project. If your project uses the Messaging Framework, follow the steps in Section 4. If your project uses GUIX, follow the steps in Section 5.

5. Migrate the Messaging Framework

This section explains how to migrate the Messaging Framework. If your project does not use the Messaging Framework **(Framework Services > All > sf_message** is not checked on the Components tab), skip this section.

The process below explains how to migrate your project built in e^2 studio 4.2 (with SSP v1.0.0), supported by the Messaging Framework utility tool named sf_message_configurator, to e^2 studio 5.0, supported by Messaging Configurator in the Synergy Configuration tool.

5.1 Configure Event Classes

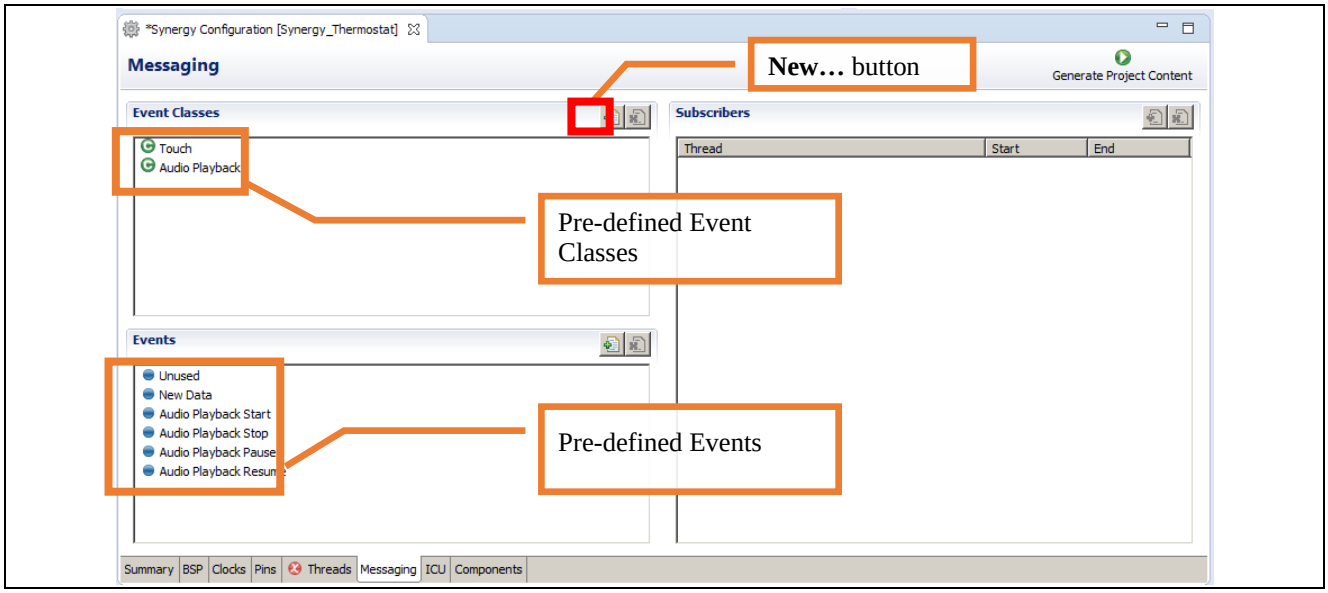
Configure Event Classes using the Synergy Project Editor in e^2 studio 5.0:

1. Look for the sf_message_port.h file in your project built on e^2 studio 4.2. Typically, the file is located in the /src/ directory. In this directory, you can find Event Class code definition, which will be displayed in the **Event Classes** pane in the **Messaging** tab of the Synergy Project Editor.

```
/** Messaging framework event class code definition */
typedef enum
{
    SF_MESSAGE_EVENT_CLASS_SYSTEM,
    SF_MESSAGE_EVENT_CLASS_ERROR_LOG,
    SF_MESSAGE_EVENT_CLASS_AUDIO,
    SF_MESSAGE_EVENT_CLASS_TOUCH,
    SF_MESSAGE_EVENT_CLASS_TIME,
    SF_MESSAGE_EVENT_CLASS_TEMPERATURE,
    SF_MESSAGE_EVENT_CLASS_DISPLAY,
} sf_message_event_class_t;
```

2. Open the **Messaging** tab in the Synergy Project Editor. You can see predefined Event Classes and Events in the **Event Classes** and **Event** panes.

   Use the predefined Events and Event Classes for the Touch Panel Framework and the Audio Playback Framework modules, which use the Messaging Framework for passing events.

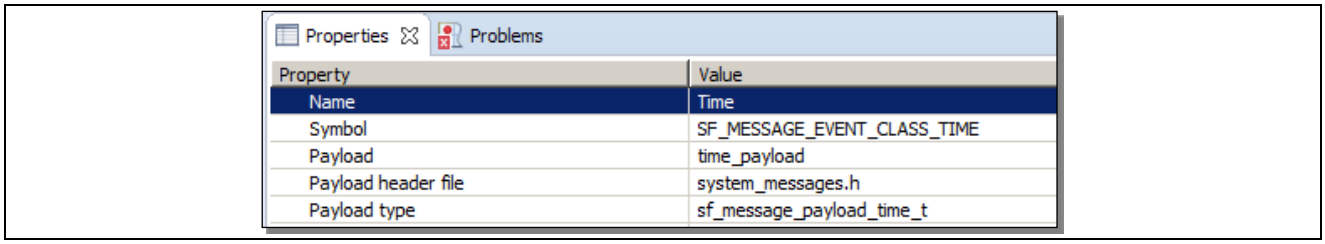


3. Click the **New…** button and add user-defined Event Classes on the Event Classes pane. Add the proper information for the payload of your Event Class as shown below:

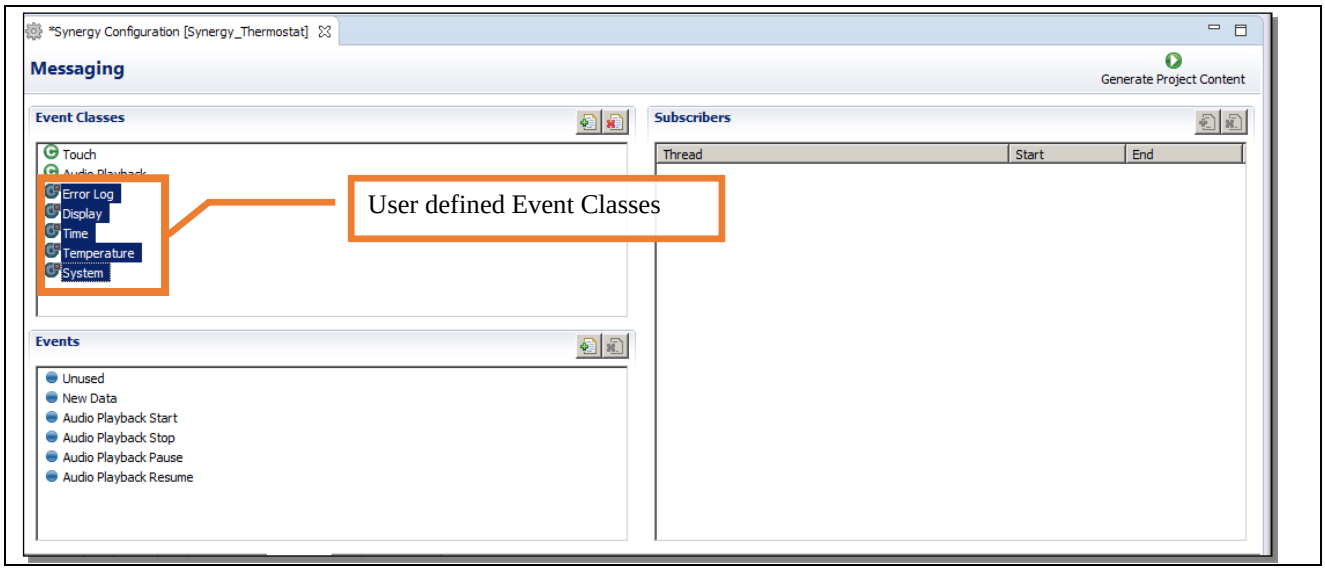
Name	*: To be one of the names of the members in sf_message_payload_t.*
Payload	*: Event code*
Payload header file	*: Specify the header file which defines the payload structure.*
Payload type	*: Type of the payload structure.*

Note that any Event Classes needs an associated Event Class payload. For example, for the following payload defined in system_message.h, the Event Class properties must be configured as follows:

```
typedef struct st_sf_message_payload_time
{
        sf_message_header_t   header;
        rtc_time_t time;
} sf_message_payload_time_t;
```



4. Complete the Event Classes addition.



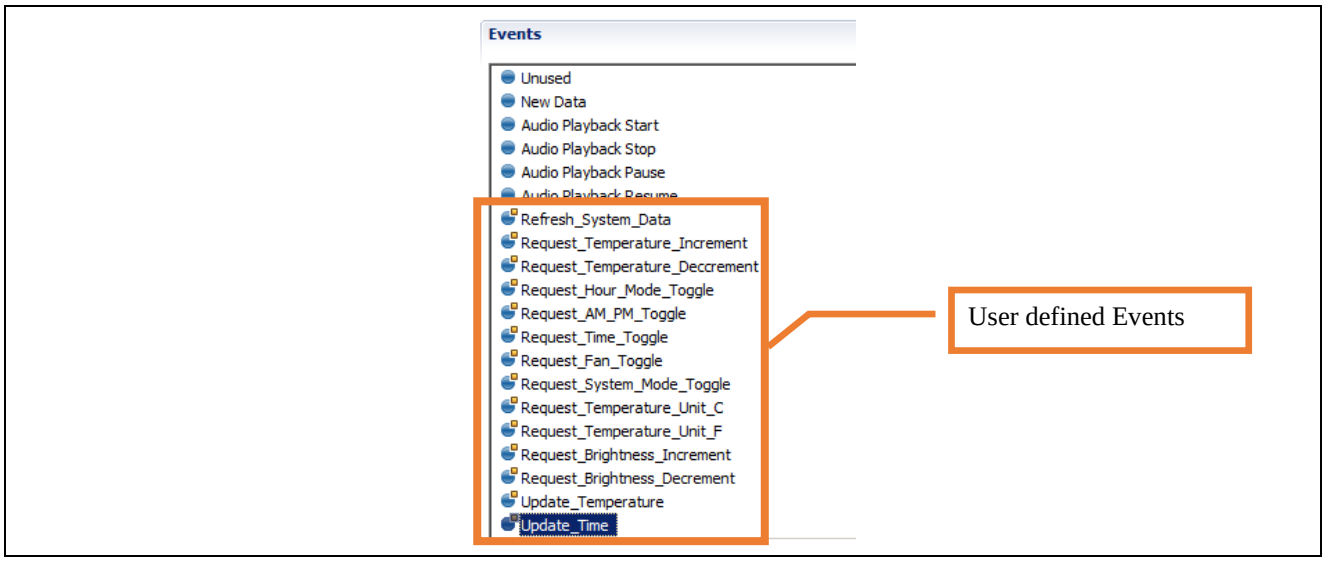
5.2 Configure Events

Configure Events with the Synergy Project Editor in e^2 studio 5.0:

1. Look for the sf_message_port.h file in your existing project again. You can find the Event code definition there. The Event code is displayed in the **Events** pane in the **Messaging** tab in the Synergy Project Editor.

```
/** Messaging framework event code definition --- */
typedef enum
{
    /** Events used by SSP audio framework. */
    SF_MESSAGE_EVENT_AUDIO_START,
    SF_MESSAGE_EVENT_AUDIO_STOP,
    SF_MESSAGE_EVENT_AUDIO_PAUSE,
    SF_MESSAGE_EVENT_AUDIO_RESUME,

    /** Some classes do not use the event enumeration. */
    SF_MESSAGE_EVENT_UNUSED,

    /** New data is available. */
    SF_MESSAGE_EVENT_NEW_DATA,

    /** System data refresh. */
    SF_MESSAGE_EVENT_REFRESH_SYSTEM_DATA,

    /** Temperature set point adjustment. */
    SF_MESSAGE_EVENT_REQUEST_TEMPERATURE_INCREMENT,
    SF_MESSAGE_EVENT_REQUEST_TEMPERATURE_DECREMENT,

    /** Time setting. */
    SF_MESSAGE_EVENT_REQUEST_HOUR_MODE_TOGGLE,
    SF_MESSAGE_EVENT_REQUEST_AM_PM_TOGGLE,
    SF_MESSAGE_EVENT_REQUEST_TIME_UPDATE,

    /** System settings. */
    SF_MESSAGE_EVENT_REQUEST_FAN_TOGGLE,
    SF_MESSAGE_EVENT_REQUEST_SYSTEM_MODE_TOGGLE,

    /** Temperature unit settings. */
    SF_MESSAGE_EVENT_REQUEST_TEMPERATURE_UNIT_C,
    SF_MESSAGE_EVENT_REQUEST_TEMPERATURE_UNIT_F,

    /** Display Settings. */
    SF_MESSAGE_EVENT_REQUEST_BRIGHTNESS_INCREMENT,
    SF_MESSAGE_EVENT_REQUEST_BRIGHTNESS_DECREMENT,

    /** Volume Settings. */
    SF_MESSAGE_EVENT_AUDIO_PLAY,
    SF_MESSAGE_EVENT_REQUEST_VOLUME_INCREMENT,
    SF_MESSAGE_EVENT_REQUEST_VOLUME_DECREMENT,

    /** Temperature data. */
    SF_MESSAGE_EVENT_UPDATE_TEMPERATURE,

    /** Time data. */
    SF_MESSAGE_EVENT_UPDATE_TIME,
} sf_message_event_t;
```

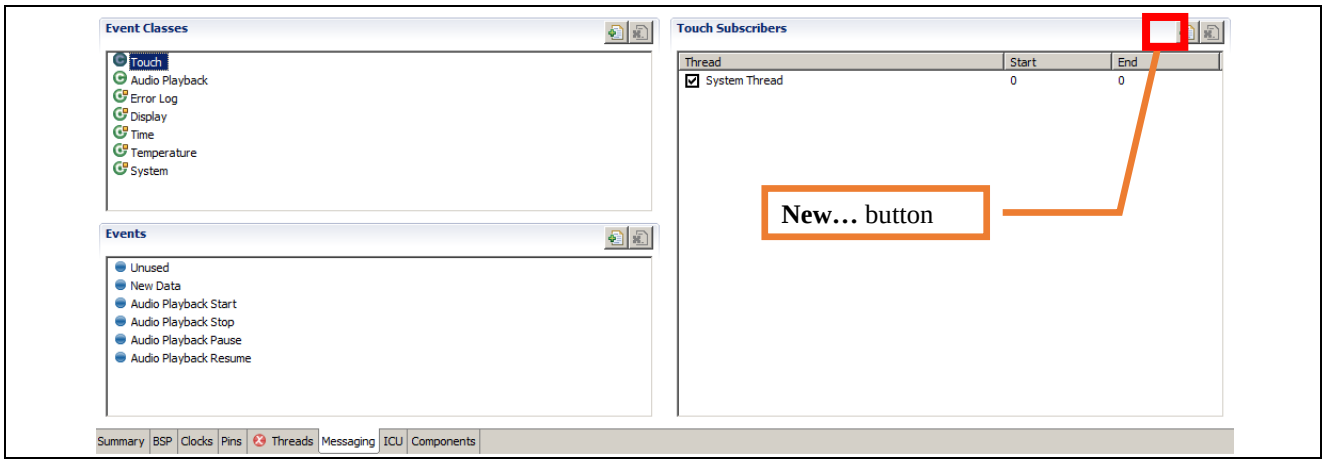
2. Open the **Messaging** tab in the Synergy Project Editor.
3. Click the **New…** button to add user-defined Event codes on the **Events** pane in the **Messaging** tab.
4. The list of Events in the file sf_message_port.h file shown in 1 are displayed in the **Events** pane in **Messaging** tab.



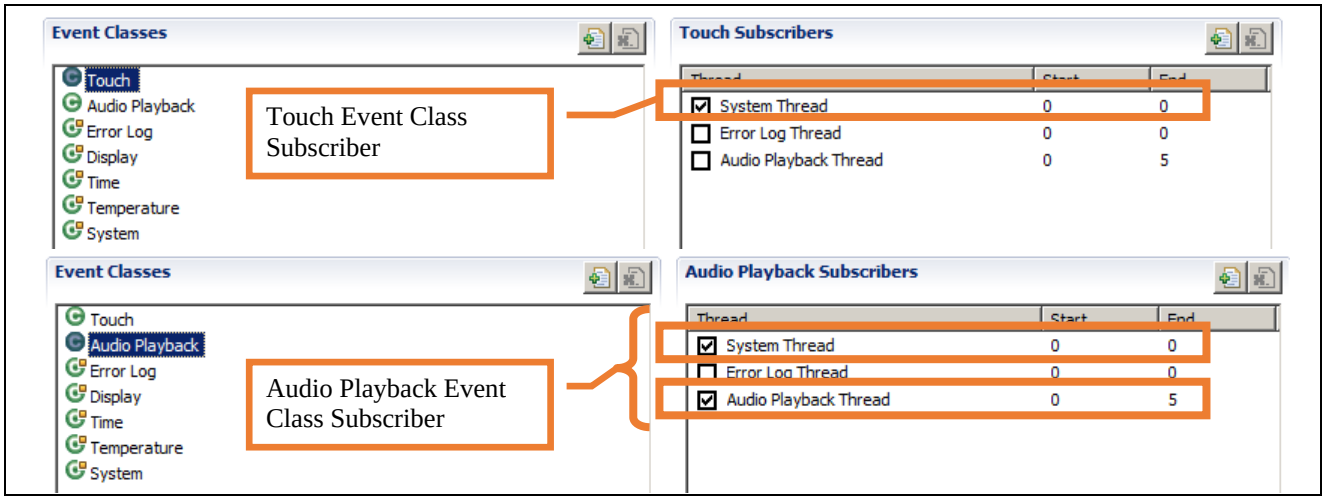
5.3 Configure Subscribers

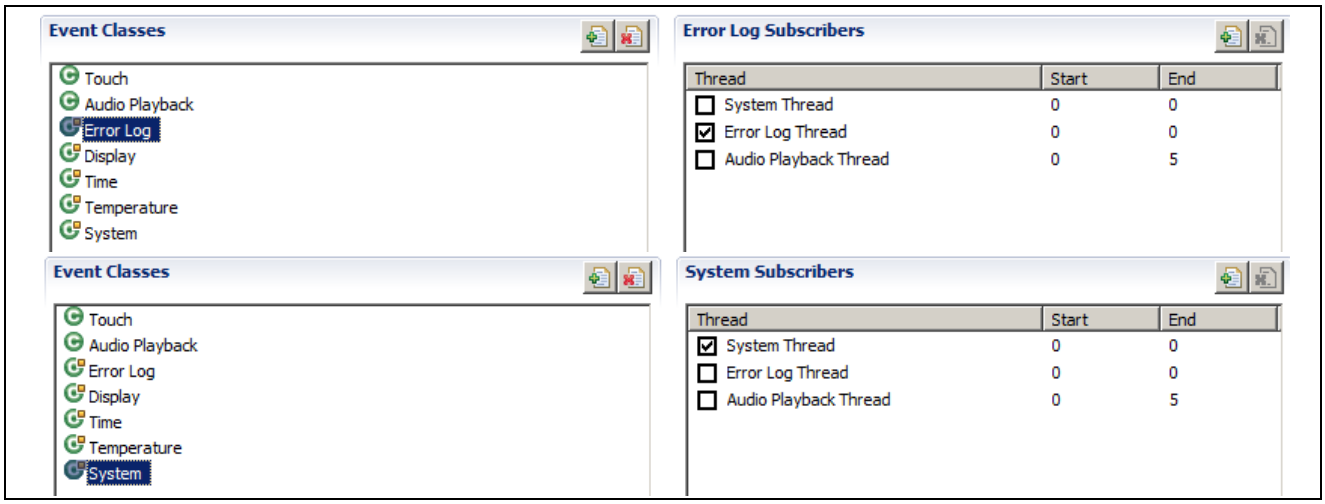
Configure Event Class subscribers in the Synergy Project Editor in e^2 studio 5.0:

1. Click the **New…** button to add subscriber threads for Event Classes on the **<Event Class name> Subscribers** pane. Then, click and highlight an Event Class on the **Event Classes** pane and click on the checkbox just left to the thread name on **<Event Class name> Subscribers** pane to register the thread as the Event Class subscriber.



2. Shown below are examples of registered subscribers for each Event Classes. For instance, the Touch Event Class message is subscribed by the System Thread only in this example. The Audio Playback Event Class is subscribed by two threads, System Thread and Audio Playback Thread. (See section 4.4 for details about Audio Playback.)





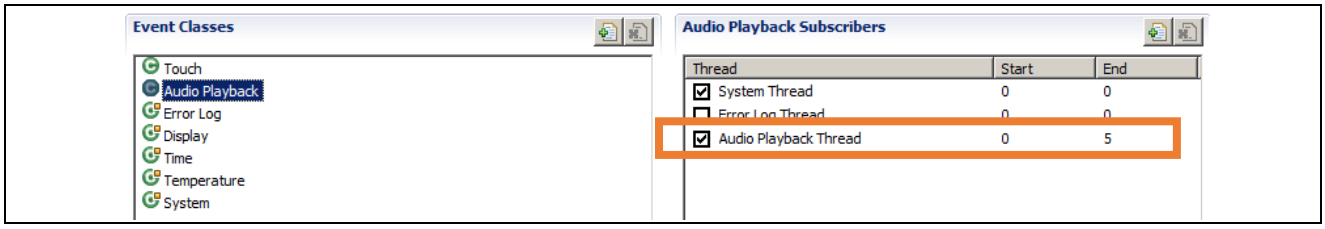
The Synergy Project Editor automatically generates Message Queues, which are required for the Messaging Framework, whereas in e^2 studio 4.2, you had to add Message Queues manually. The Message Queues are generated automatically in the file /src/synergy_gen/message_data.c.

5.4 Migrate Audio Playback Events

This section explains how to migrate the Audio Playback Event passing. You can configure the Event Class subscribers in the Synergy Project Editor for thread instances listed in the Threads pane on Threads tab. Only thread instances that are listed in the Threads tab are allowed to be an Event Class subscriber. Since the Audio Playback Framework module creates an internal thread at run-time and Audio Playback Event Class messages are pended on by this thread, you cannot configure the module internal thread as the subscriber on the Audio Playback Subscribers pane. This limitation may be removed in future e^2 studio versions.

To migrate Audio Playback Events:

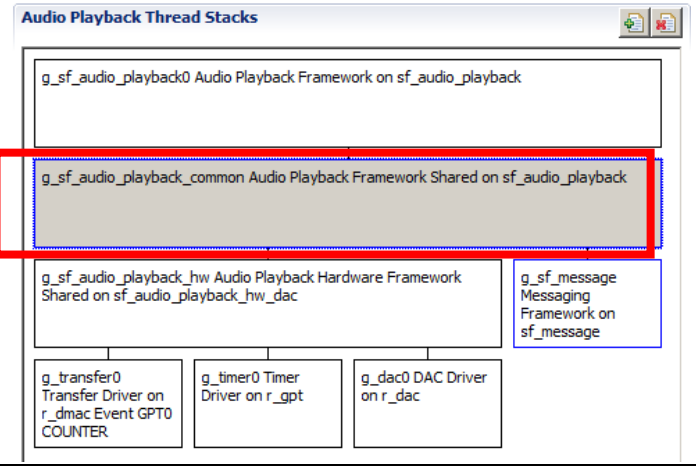
1. Create a thread named **Audio Playback Thread**. The name is arbitrary, but for this example use the name in this document.
2. Configure the thread as an **Audio Playback Event Class** subscriber.



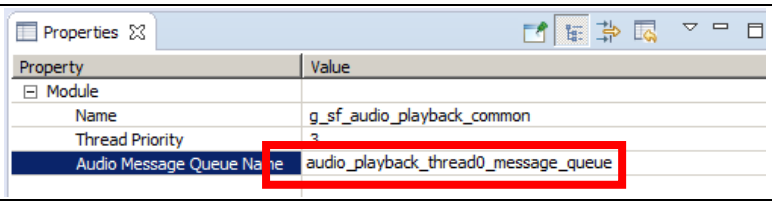
3. Click the **Generate Project Content** button to generate a Message Queue for the Audio Playback Thread. You can find the name of the Message queue in the file src/synergy_gen/message_data.c. The name of Message Queue in this example is audio_playback_thread0_message_queue.

```
TX_QUEUE audio_playback_thread0_message_queue;
```

4. Open the **Threads** tab in the Synergy Project Configurator and select the **Audio Playback Framework Shared on sf_audio_playback** module in **Audio Playback Thread Stacks**.



5. Change the property **Audio Message Queue Name** to the name of the queue identified previously: audio_playback_thread0_message_queue. This allows Audio Playback Framework module to be an Audio Playback Event Class subscriber.



5.5 Generate and Delete Code

Generate the code that sets up the Messaging Framework:

1. Click the **Generate Project Content** button to generate the files sf_message_port.h, sf_message_payloads.h and message_data.c. The two header files are created in the directory /synergy_cfg/ssp_cfg/framework/. The file message_data.c is created in the directory /src/synergy_gen/. Below are the examples for the auto-generated code:

    - sf_message_port.h

```
typedef enum e_sf_message_event_class
{
    SF_MESSAGE_EVENT_CLASS_TOUCH, /* Touch */
    SF_MESSAGE_EVENT_CLASS_AUDIO, /* Audio Playback */
    SF_MESSAGE_EVENT_CLASS_ERROR_LOG, /* Error Log */
    SF_MESSAGE_EVENT_CLASS_DISPLAY, /* Display */
    SF_MESSAGE_EVENT_CLASS_TIME, /* Time */
    SF_MESSAGE_EVENT_CLASS_TEMPERATURE, /* Temperature */
    SF_MESSAGE_EVENT_CLASS_SYSTEM, /* System */
} sf_message_event_class_t;
typedef enum e_sf_message_event
{
    SF_MESSAGE_EVENT_UNUSED, /* Unused */
    SF_MESSAGE_EVENT_NEW_DATA, /* New Data */
    SF_MESSAGE_EVENT_AUDIO_START, /* Audio Playback Start */
    SF_MESSAGE_EVENT_AUDIO_STOP, /* Audio Playback Stop */
    SF_MESSAGE_EVENT_AUDIO_PAUSE, /* Audio Playback Pause */
    SF_MESSAGE_EVENT_AUDIO_RESUME, /* Audio Playback Resume */
    SF_MESSAGE_EVENT_REFRESH_SYSTEM_DATA, /* Refresh_System_Data */
    SF_MESSAGE_EVENT_REQUEST_TEMPERATURE_INCREMENT, /* Request_Temperature_Increment */
    SF_MESSAGE_EVENT_REQUEST_TEMPERATURE_DECCREMENT, /* Request_Temperature_Deccrement */
    SF_MESSAGE_EVENT_REQUEST_HOUR_MODE_TOGGLE, /* Request_Hour_Mode_Toggle */
    SF_MESSAGE_EVENT_REQUEST_AM_PM_TOGGLE, /* Request_AM_PM_Toggle */
    SF_MESSAGE_EVENT_REQUEST_TIME_TOGGLE, /* Request_Time_Toggle */
    SF_MESSAGE_EVENT_REQUEST_FAN_TOGGLE, /* Request_Fan_Toggle */
    SF_MESSAGE_EVENT_REQUEST_SYSTEM_MODE_TOGGLE, /* Request_System_Mode_Toggle */
    SF_MESSAGE_EVENT_REQUEST_TEMPERATURE_UNIT_C, /* Request_Temperature_Unit_C */
    SF_MESSAGE_EVENT_REQUEST_TEMPERATURE_UNIT_F, /* Request_Temperature_Unit_F */
    SF_MESSAGE_EVENT_REQUEST_BRIGHTNESS_INCREMENT, /* Request_Brightness_Increment */
    SF_MESSAGE_EVENT_REQUEST_BRIGHTNESS_DECREMENT, /* Request_Brightness_Decrement */
```

```
    SF_MESSAGE_EVENT_UPDATE_TEMPERATURE, /* Update_Temperature */
    SF_MESSAGE_EVENT_UPDATE_TIME, /* Update_Time */
} sf_message_event_t;
extern TX_QUEUE system_thread_message_queue;
extern TX_QUEUE audio_playback_thread0_message_queue;
extern TX_QUEUE error_log_thread_message_queue;
#endif /* SF_MESSAGE_PORT_H_ */
```

- sf_message_payload.h

```
/* generated messaging header file - do not edit */
#ifndef SF_MESSAGE_PAYLOADS_H_
#define SF_MESSAGE_PAYLOADS_H_
#include "sf_touch_panel_api.h"
#include "sf_audio_playback_api.h"
#include "system_messages.h"
#include "system_messages.h"
#include "system_messages.h"
#include "system_messages.h"
typedef union u_sf_message_payload
{
    sf_touch_panel_event_t sf_touch_panel_event;
    sf_audio_playback_data_t sf_audio_playback_data;
    sf_message_payload_error_t error_log_payload;
    sf_message_payload_time_t time_payload;
    sf_message_payload_temperature_t temperature_payload;
    gx_system_update_message_t system_payload;
} sf_message_payload_t;
#endif /* SF_MESSAGE_PAYLOADS_H_ */
```

- message_data.c

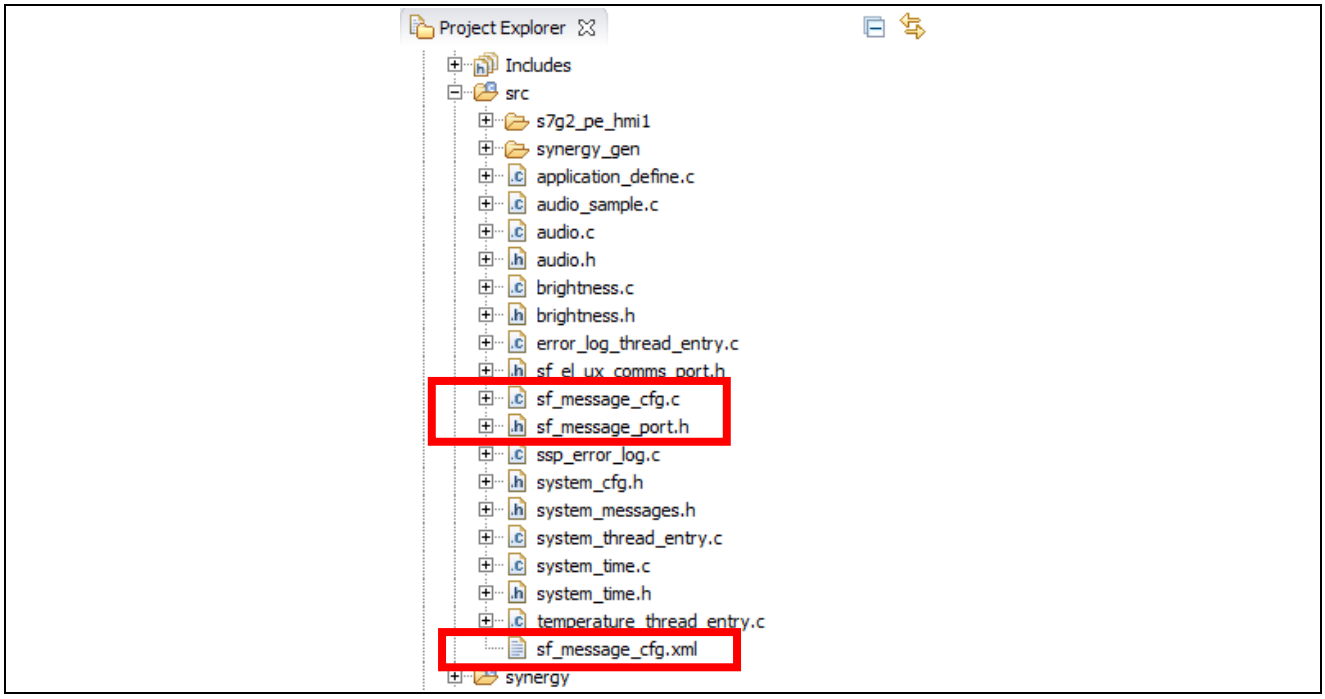
```
/* generated messaging source file - do not edit */
#include "sf_message.h"
#ifndef SF_MESSAGE_CFG_QUEUE_SIZE
#define SF_MESSAGE_CFG_QUEUE_SIZE (16)
#endif
TX_QUEUE system_thread_message_queue;
static uint8_t queue_memory_system_thread_message_queue[SF_MESSAGE_CFG_QUEUE_SIZE];
TX_QUEUE audio_playback_thread_message_queue;
static uint8_t queue_memory_audio_playback_thread_message_queue[SF_MESSAGE_CFG_QUEUE_SIZE];
TX_QUEUE error_log_thread_message_queue;
static uint8_t queue_memory_error_log_thread_message_queue[SF_MESSAGE_CFG_QUEUE_SIZE];
static sf_message_subscriber_t system_thread_message_queue_0_0              =
{ .        p_queue = &system_thread_message_queue, .instance_range          = { .start = 0, .end = 0 } };
static sf_message_subscriber_t audio_playback_thread_message_queue_0_5      =
{ .p_queue = &audio_playback_thread_message_queue, .instance_range          = { .start = 0, .end = 5 } };
static sf_message_subscriber_t error_log_thread_message_queue_0_0           =
{ .        p_queue = &error_log_thread_message_queue, .instance_range       = { .start = 0, .end = 0 } };
static sf_message_subscriber_t * gp_group_SF_MESSAGE_EVENT_CLASS_TOUCH[]    = { &system_thread_message_queue_0_0, };

static sf_message_subscriber_list_t g_list_SF_MESSAGE_EVENT_CLASS_TOUCH     =
{ .event_class = (uint16_t) SF_MESSAGE_EVENT_CLASS_TOUCH, .number_of_nodes = 1,
  .pp_subscriber_group = gp_group_SF_MESSAGE_EVENT_CLASS_TOUCH };
static sf_message_subscriber_t * gp_group_SF_MESSAGE_EVENT_CLASS_AUDIO[]    =
{ &audio_playback_thread_message_queue_0_5, &system_thread_message_queue_0_0, };
static sf_message_subscriber_list_t g_list_SF_MESSAGE_EVENT_CLASS_AUDIO     =
{ .event_class = (uint16_t) SF_MESSAGE_EVENT_CLASS_AUDIO, .number_of_nodes = 2,
  .pp_subscriber_group = gp_group_SF_MESSAGE_EVENT_CLASS_AUDIO };
static sf_message_subscriber_t * gp_group_SF_MESSAGE_EVENT_CLASS_TEMPERATURE[] = { &system_thread_message_queue_0_0, };
static sf_message_subscriber_list_t g_list_SF_MESSAGE_EVENT_CLASS_TEMPERATURE =
{ .event_class = (uint16_t) SF_MESSAGE_EVENT_CLASS_TEMPERATURE, .number_of_nodes = 1,
  .pp_subscriber_group = gp_group_SF_MESSAGE_EVENT_CLASS_TEMPERATURE };
static sf_message_subscriber_t * gp_group_SF_MESSAGE_EVENT_CLASS_ERROR_LOG[] = { &error_log_thread_message_queue_0_0, };
static sf_message_subscriber_list_t g_list_SF_MESSAGE_EVENT_CLASS_ERROR_LOG =
{ .event_class = (uint16_t) SF_MESSAGE_EVENT_CLASS_ERROR_LOG, .number_of_nodes = 1,
  .pp_subscriber_group = gp_group_SF_MESSAGE_EVENT_CLASS_ERROR_LOG };
static sf_message_subscriber_t * gp_group_SF_MESSAGE_EVENT_CLASS_TIME[]     = { &system_thread_message_queue_0_0, };
static sf_message_subscriber_list_t g_list_SF_MESSAGE_EVENT_CLASS_TIME      =
{ .event_class = (uint16_t) SF_MESSAGE_EVENT_CLASS_TIME, .number_of_nodes = 1,
  .pp_subscriber_group = gp_group_SF_MESSAGE_EVENT_CLASS_TIME };
static sf_message_subscriber_t * gp_group_SF_MESSAGE_EVENT_CLASS_SYSTEM[]   = { &system_thread_message_queue_0_0, };
static sf_message_subscriber_list_t g_list_SF_MESSAGE_EVENT_CLASS_SYSTEM    =
{ .event_class = (uint16_t) SF_MESSAGE_EVENT_CLASS_SYSTEM, .number_of_nodes = 1,
  .pp_subscriber_group = gp_group_SF_MESSAGE_EVENT_CLASS_SYSTEM };

sf_message_subscriber_list_t * p_subscriber_lists[] =
{ &g_list_SF_MESSAGE_EVENT_CLASS_TOUCH, &g_list_SF_MESSAGE_EVENT_CLASS_AUDIO,
  &g_list_SF_MESSAGE_EVENT_CLASS_TEMPERATURE, &g_list_SF_MESSAGE_EVENT_CLASS_ERROR_LOG,
  &g_list_SF_MESSAGE_EVENT_CLASS_TIME, &g_list_SF_MESSAGE_EVENT_CLASS_SYSTEM, NULL };
void g_message_init(void);
void g_message_init(void)
{
    tx_queue_create (&system_thread_message_queue, (CHAR *) "System Thread Message Queue", 1,
                     &queue_memory_system_thread_message_queue, sizeof(queue_memory_system_thread_message_queue));
    tx_queue_create (&audio_playback_thread_message_queue, (CHAR *) "Audio Playback Thread Message Queue", 1,
                     &queue_memory_audio_playback_thread_message_queue,sizeof(queue_memory_audio_playback_thread_message_queue));
    tx_queue_create (&error_log_thread_message_queue, (CHAR *) "Error Log Thread Message Queue", 1,
                     &queue_memory_error_log_thread_message_queue, sizeof(queue_memory_error_log_thread_message_queue));
}
```

Message Queues

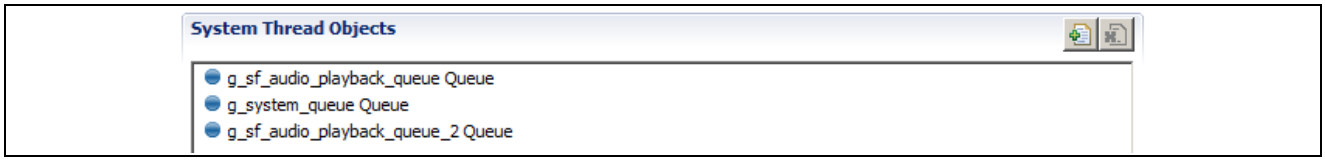
Subscriber List

Message Queue initializer

2. Delete the following three files, which were required for the project in e² studio 4.2. These files are no longer valid in e² studio 5.0:



3. In the **<Thread name> Objects** pane on **Threads** tab, delete the Message Queue objects that were used for the Messaging Framework in e² studio 4.2. Any queues for the Messaging Framework are generated by the Synergy Project Editor in /src/synergy_gen/message_data.c. You no longer need to add queues to the **<Thread name> Objects** pane.



5.6 Change the Application

To update your application code, modify the name of the queue used for the 2nd argument in the Messaging Framework pend API in every single Event Class subscriber thread. Use the name of the queue generated in message_data.c by the Synergy Project Editor. Here is an example:

```
…
/** Event loop. */
while (1)
{
    /** Wait for an error message. */
    sf_message_payload_error_t * p_message;
    err = g_sf_message.p_api->pend(g_sf_message.p_ctrl,
                                   &error_log_thread_message_queue,
                                   (sf_message_header_t **) &p_message,
                                   TX_WAIT_FOREVER);
    /** Print error message */
    if (SSP_SUCCESS == err)
    {
        sprintf(p_str, "%s %d %d\r\n", p_message->module, p_message->err,
                                                          (int) p_message->line);
        g_sf_comms.p_api->write(g_sf_comms.p_ctrl, p_str, strlen(p_str), 10);

        /** Release buffer. */
        g_sf_message.p_api->bufferRelease(g_sf_message.p_ctrl,
                                          (sf_message_header_t *)p_message,
                                          SF_MESSAGE_RELEASE_OPTION_NONE);
    }
}
```

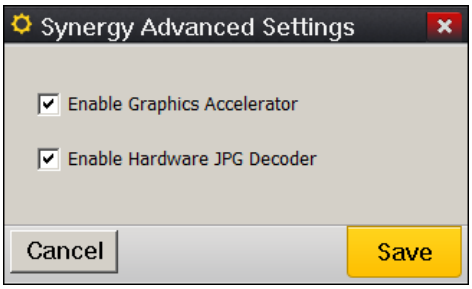
6. Migrate to GUIX 5.3.0

In SSP v1.1.0, GUIX is updated to version 5.3.0. This procedure describes how to update graphics files for use with the updated GUIX library in the SSP.

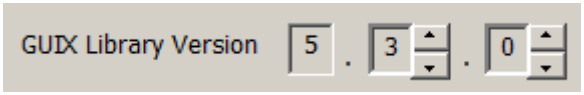
6.1 Update GUIX Resources

To update the GUIX resources files:

1. Download GUIX Studio version 5.3.0.1 from the Renesas Synergy Gallery and install it.
2. Open GUIX Studio version 5.3.0.1
3. Select **Project > Open Project**. Browse to the .gxp project file used to generate resources for your application.
4. Select **Configure > Project/Displays**.
5. Next to Target CPU, click **Advanced Settings**. Confirm that Graphics Accelerator and Hardware JPG Decoder are enabled.



6. Set the **GUIX Library Version** to **5.3.0**.



7. Set the **GUIX Library Version** to **5.3.0**.
8. Click **Save**.
9. Select **Project** > **Generate All Output Files**.
10. If your project uses GUIX Port module (sf_el_gx), be aware that the following two module configuration properties were moved from the sf_el_gx module to the GUIX on gx module:
    - 2DG Rendering Support
    - JPEG Rendering Support

    Open the **Threads** tab in the Synergy Project Editor, select **GUIX on gx** module, and change the following two values to yes if your GUIX project uses the Synergy graphics hardware accelerations:
    - Enable Synergy 2D Drawing Engine Support
    - Enable Synergy JPEG Support

The GUIX resources are now updated to be compatible with the updated GUIX v5.3.0 library in the SSP.

Website and Support

Support: https://synergygallery.renesas.com/support

Technical Contact Details:

- America: https://renesas.zendesk.com/anonymous_requests/new
- Europe: https://www.renesas.com/en-eu/support/contact.html
- Japan: https://www.renesas.com/ja-jp/support/contact.html

All trademarks and registered trademarks are the property of their respective owners.

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	May 9, 2016	-	Initial Version
1.01	Nov 18, 2016	-	Minor format changes

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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2016 Renesas Electronics Corporation. All rights reserved.

Colophon 5.0