Renesas Synergy™ Platform

GUIX "Hello World" for PE-HMI1

Introduction

This application note guides you through the process of creating a simple two screen GUI using GUIX™ Studio for the PE-HMI1. Its application demonstrates how easily a user can create and configure a new application using the Renesas Synergy™ Software Package (SSP).

The Synergy Software Package includes Express Logic's ThreadX® real-time operating system (RTOS), the X-Ware® suite of stacks (NetX™, USBX™, GUIX™, and FileX®), and a set of hardware drivers unified under a single robust framework. This powerful suite of tools provides a comprehensive integrated framework for rapid development of complex embedded applications.

The **Hello World** application was developed with e^2 studio using the Application Framework.

Target Device

PE-HMI1 board version 2.0

Minimum PC Recommendation

- Microsoft® Windows® 7 or later
- Intel® Core™ family processor running at 2.0 GHz or higher
- 8 GB memory
- 250 GB hard disk or SSD
- USB 2.0
- Connection to the Internet

Installed Software

- Synergy™ e^2 studio Integrated Solution Development Environment (ISDE) Version 2021 (21.7.0) or later
- Synergy™ Software Package (SSP) v2.2.0 or later
- GUIX Studio v6.1.8 or later

Note: If you do not have one of these software applications, you should install them before continuing. You can download the required Renesas software from the Renesas Synergy™ Gallery at: https://synergygallery.renesas.com

Software Files Provided

- `guiapp_event_handlers.c`
- `main_thread_entry.c`
- `R7FS7G27H2A01CBD.pincfg`

Purpose

This document seeks to guide you through the setup of a GUIX touch screen interface **Hello World** application in e^2 studio ISDE, where you configure hardware functions (LCD, timers, and I^{2}C interface), threads and message passing, interrupts, the LCD driver, and the touchscreen. It covers initial project setup in e^2 studio, along with basic debugging operations. It also instructs you in creating a simple GUI interface using the GUIX Studio editor. Once the application is running, it responds to touchscreen actions using Framework "Touch Panel V2 Framework on sf_touch_panel_v2", presenting a basic graphical user interface (GUI).

Intended Audience

The intended audience are developers designing GUI applications.

Contents

1. Overview 3
2. Importing the project into e² studio 3
3. Creating the project in e² studio 3
4. Configuring the project in e² studio 7
5. Creating the GUIX interface using GUIX Studio 20
6. Adding code for custom interface controls and building the project 33
7. Running the application 34
8. Appendix 36
Revision History 38

1. Overview

This application note shows how to setup a project and develop a simple GUI-based application using GUIX Studio.

2. Importing the project into e² studio

Note: This step is included to give the user the ability to skip the development steps and just jump to the point of verifying a working project on the PE-HMI1.

Most users SKIP THIS STEP and proceed to step 3 to create a project in e² studio. If you do import the project, skip to section 7 Running the application.

To skip the development walkthrough in this document and open a completed project in e² studio, see the *Renesas Synergy™ Project Import Guide* (`REN_r11an0023eu0121-synergy-ssp-import-guide_APN_20181022.pdf`) in this package. It contains instructions on importing the project into e² studio and building the project. The included `GUIX_Hello_World_PE-HMI1.zip` file contains the completed project.

3. Creating the project in e² studio

Start by creating a new project in e² studio.

1. Open e² studio by clicking on the **e² studio** icon in the **Windows Start Menu > All Programs > Renesas Electronics e²studio** folder.
2. If the **Workspace Launcher** dialog box appears, click **OK** to use the default workspace.

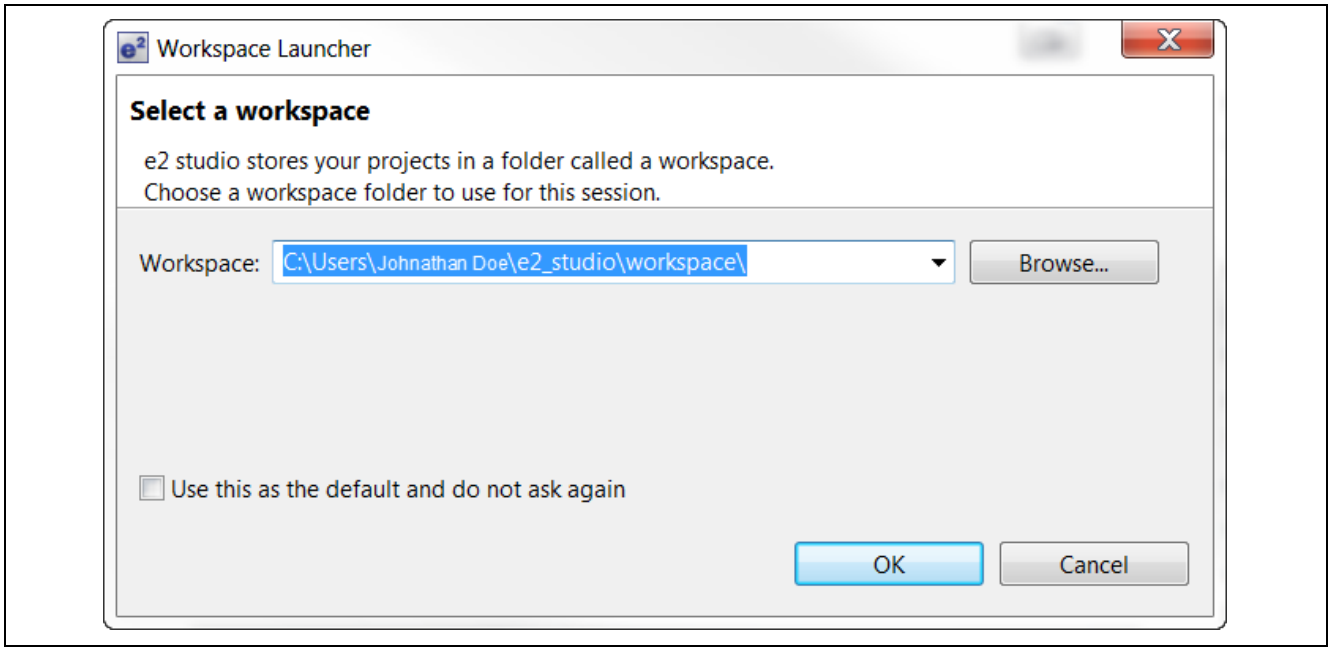


Figure 1. Workspace Launcher Dialog

3. Create a new workspace:
   From the **File** drop-down menu, select **Switch Workspace > Other…**
4. Append a workspace name:
   In the **Workspace Launcher** window, add text to the end of the workspace name to make it unique, such as **GUI_APP**. If you installed to the default location, the new workspace name will be **C:\Users\[user name]\e2_studio\workspace\GUI_APP**.
5. Click **OK** to create the new workspace.
6. Proceed past the **Welcome** screen by clicking in the **Workbench** area.

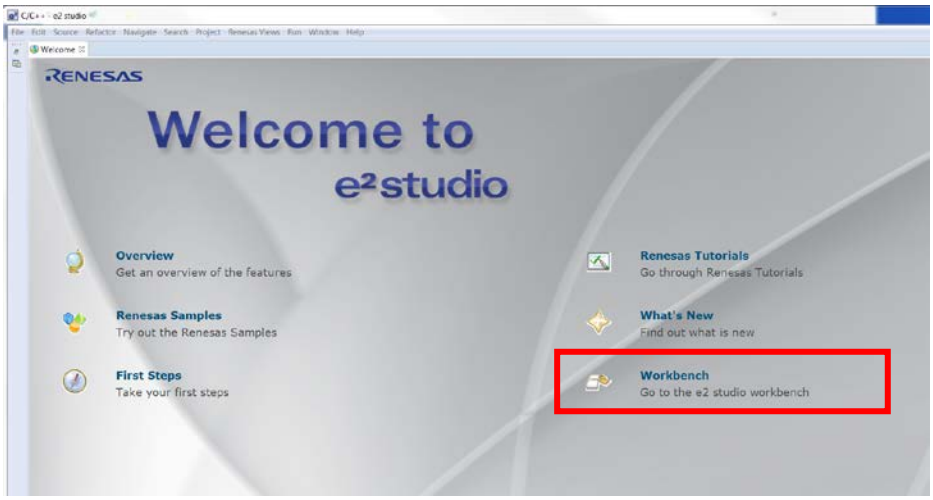


Figure 2. Close the Welcome Window by clicking in the Workbench Area

7. Start a new project by clicking the drop-down menu next to the **New** icon in the Tool Bar.

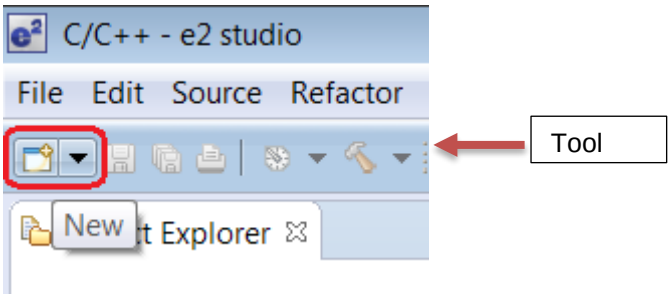


Figure 3. Start a New Project

8. Select **Synergy C/C++ Project** from the menu.

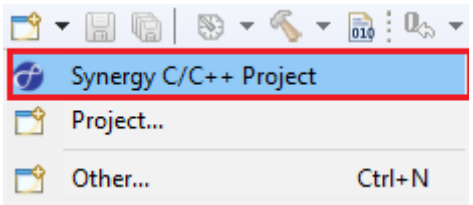


Figure 4. Select Synergy C/C++ Project in the drop-down menu

9. Select Renesas Synergy C Executable project.

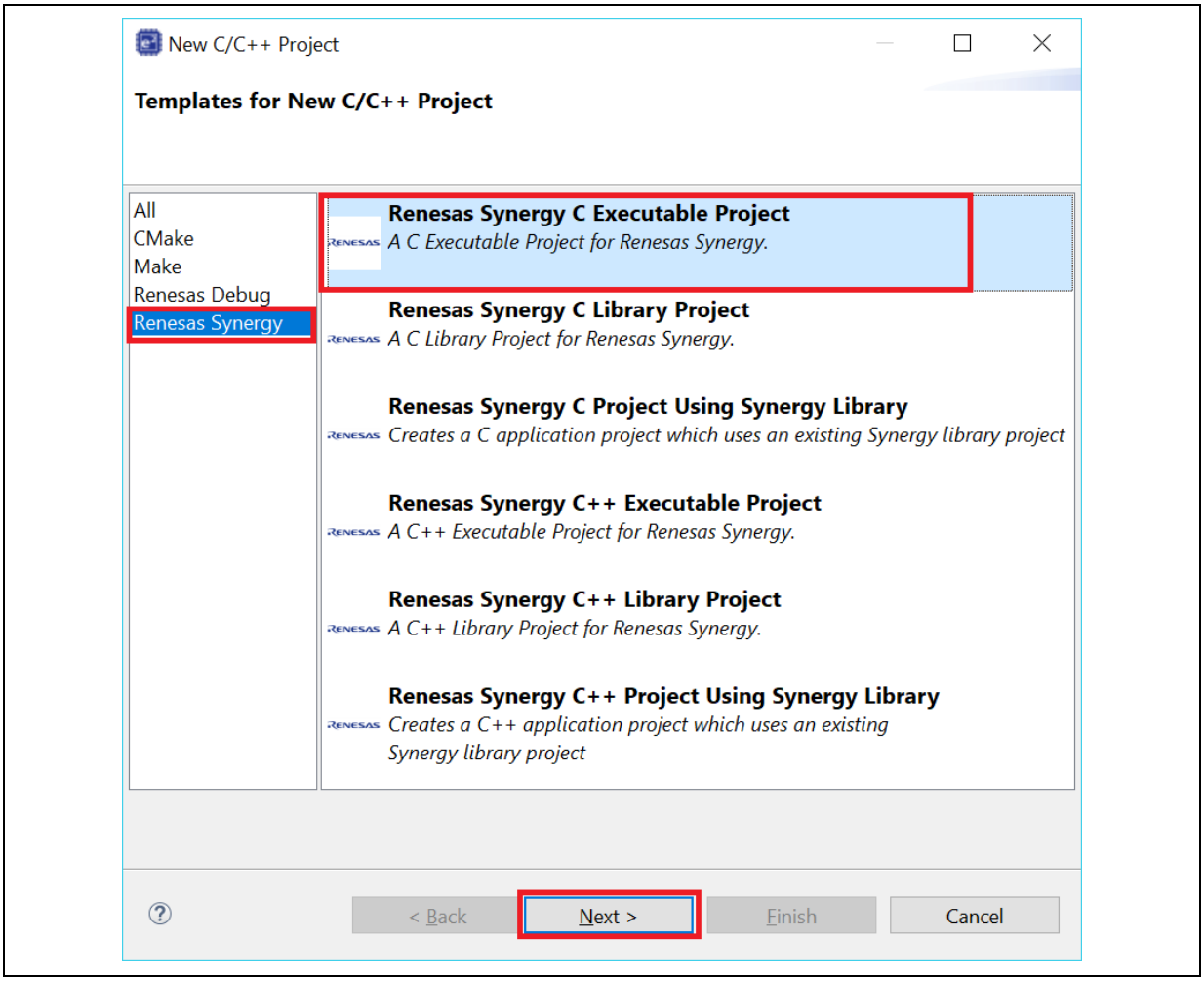


Figure 5. Project type selection

10. Enter a name for the project in the Project name text field. For example, **GUIApp**.

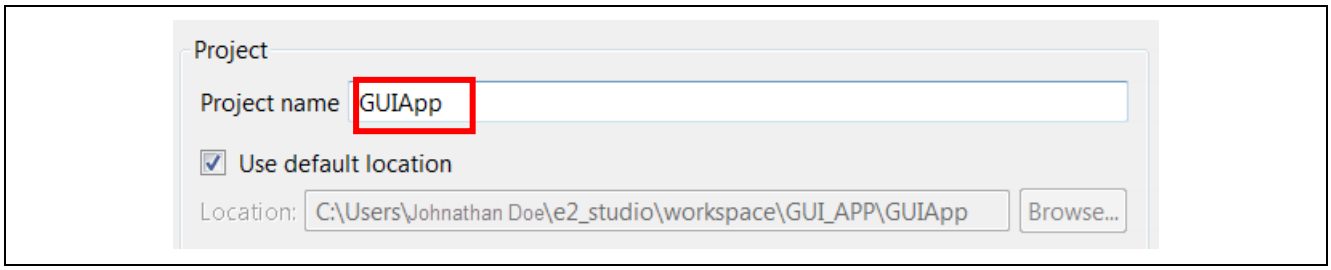


Figure 6. Enter a Project Name

11. On the top right of this page, verify that the **Toolchains** option is set to **GCC ARM Embedded**.

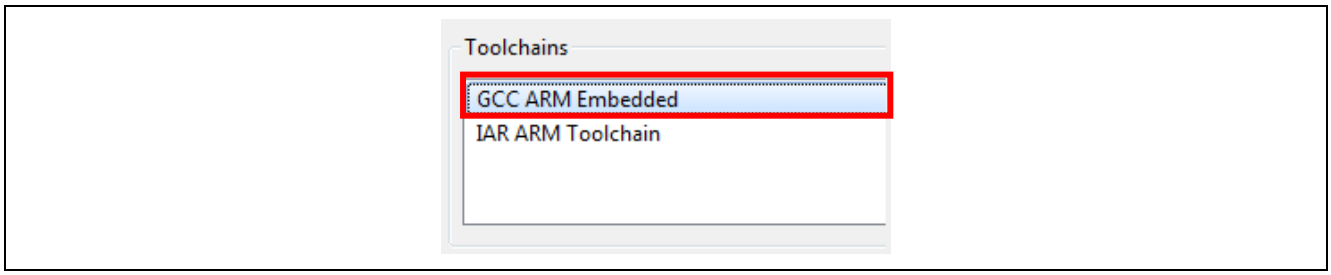


Figure 7. Verify GCC ARM Embedded Toolchain

12. Click the **Next** button to continue.
13. Under **Device Selection** (top left), select the **SSP version** 2.1.0 (or later).
14. For the **Board** field, select **S7G2 PE-HMI1**. The **Device** field updates automatically.

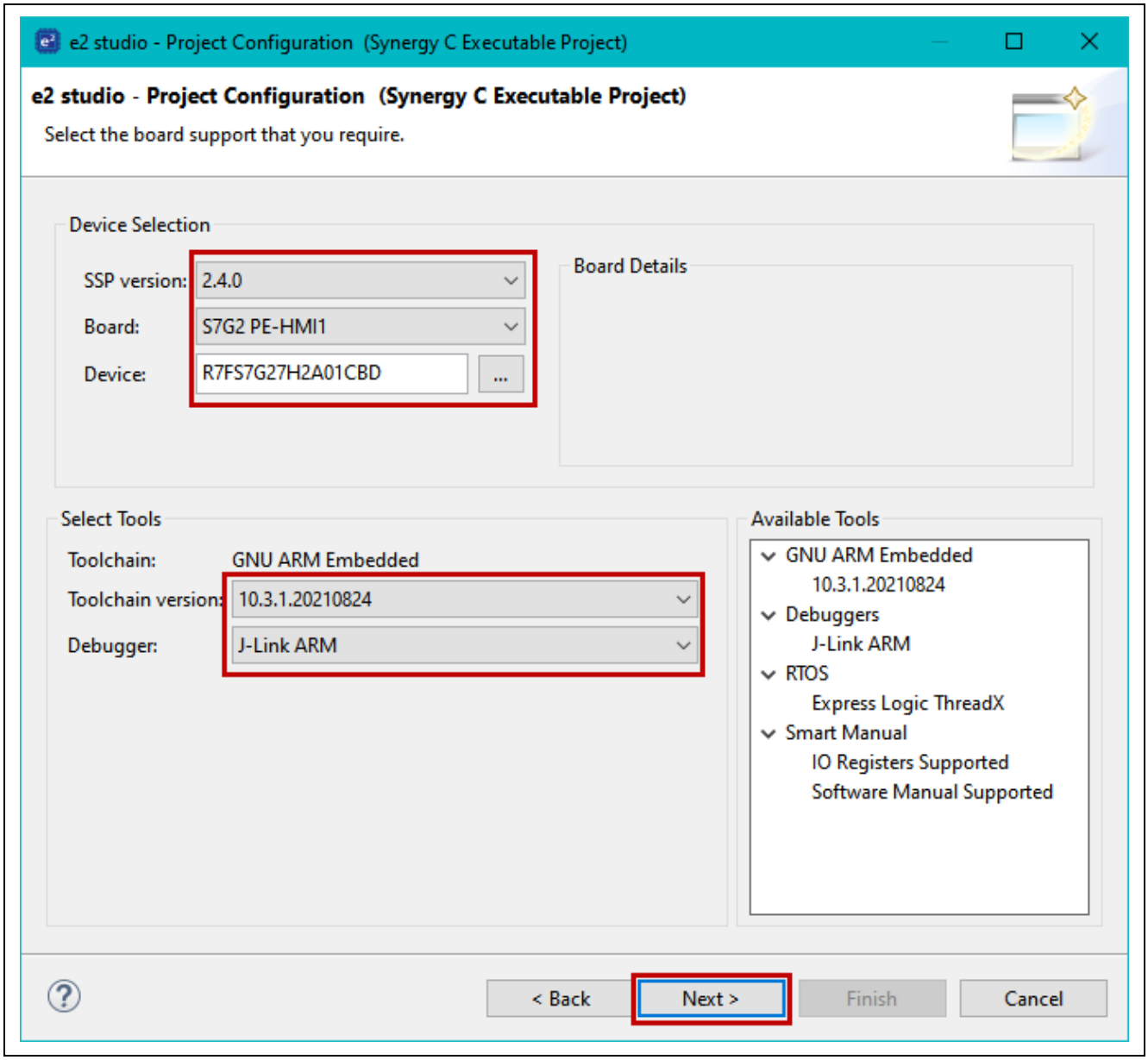


Figure 8. Device selection

15. Click the **Next** button to continue.
16. In the **Project Configuration Dialog**, select the option **BSP**.

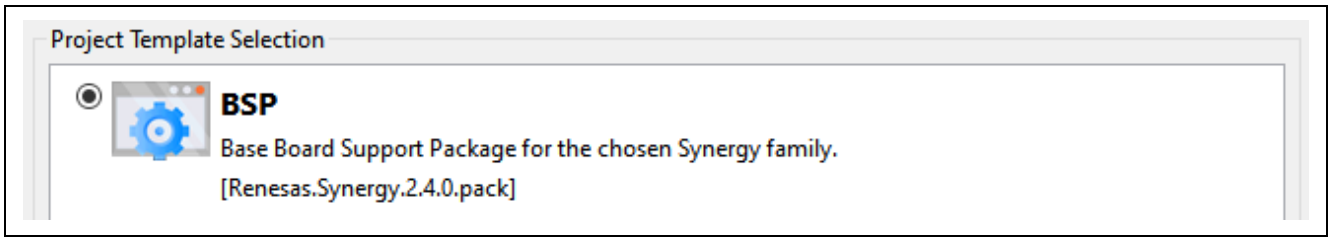


Figure 9. Select the BSP

17. Click the **Finish** button.

18. If you have not directed e² studio to remember your perspectives, e² studio will display the **Open Associated Perspective** dialog box. If opened, click **Yes** to acknowledge and close.

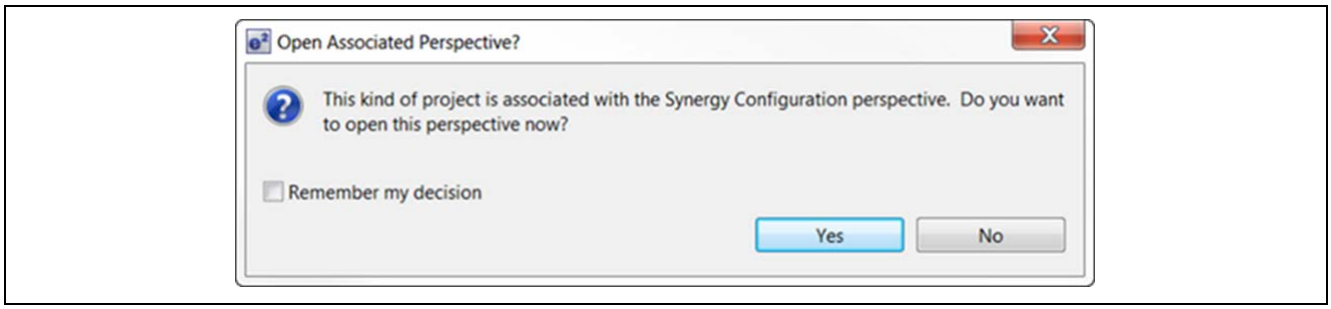


Figure 10. Open Perspective dialog box

When the project is created, screen could look like this image below.

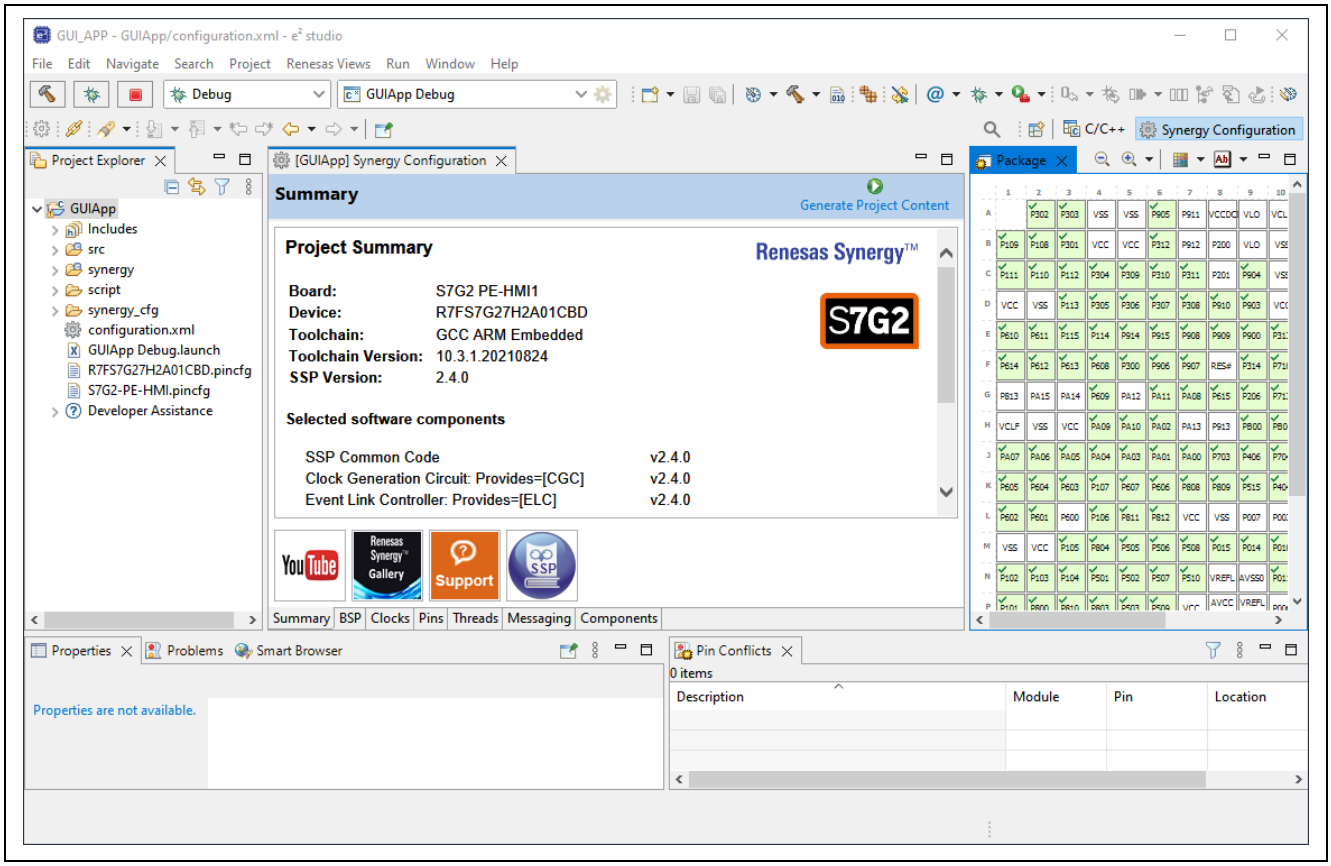


Figure 11. GUIApp Project

4. Configuring the project in e² studio

Once the project is successfully created in e2 studio ISDE, copy a new file of pin configuration and start to configure for GUI application.

1. Open **Windows Explorer** and navigate to where you put the files included with this application note. Locate the file `Source Files\ R7FS7G27H2A01CBD.pincfg`. Now drag the file from the Windows Explorer Window into the GUIApp e^2 studio **Project Explorer** window.
2. Open the **Synergy Configuration**, if it is not already open, by double clicking the **configuration.xml** file in the **Project Explorer Window**.

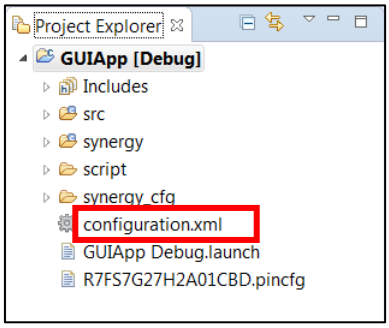


Figure 12. Selecting the configuration.xml file in Project Explorer

3. In [GUIApp] Synergy Configuration window. Select The **Pins** tab. Select **R7FS7G27H2A01CBD.pincfg** from the **Select pin configuration** drop list. like the image below.

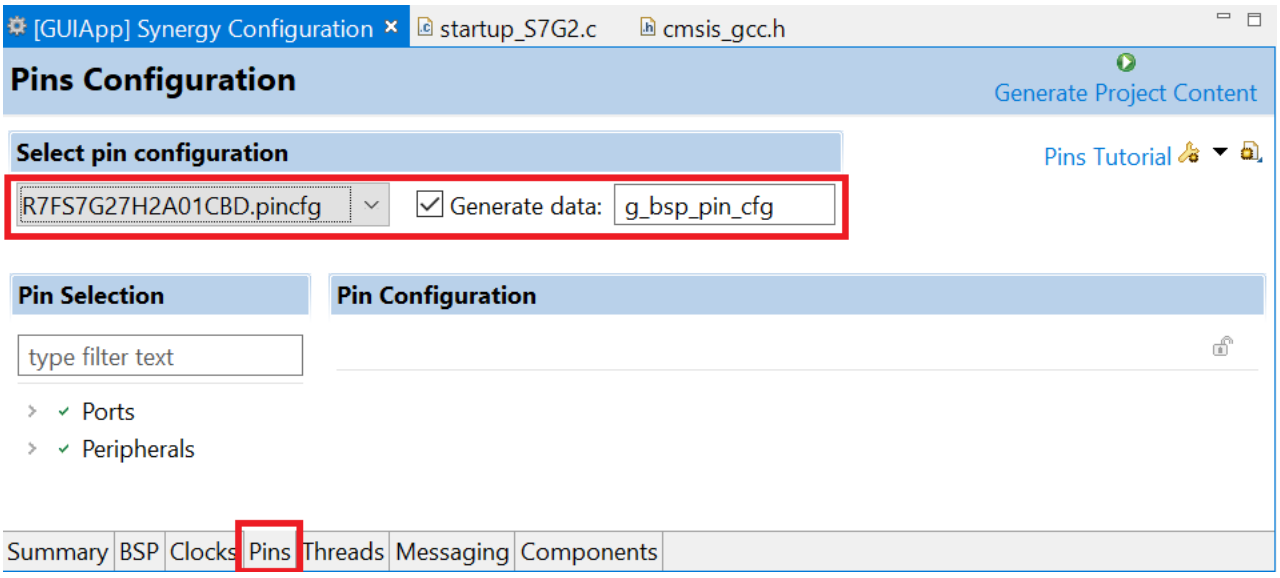


Figure 13. Selecting pin configuration and file replacement

4. In the **Synergy Configuration** window, click the **Threads** tab.

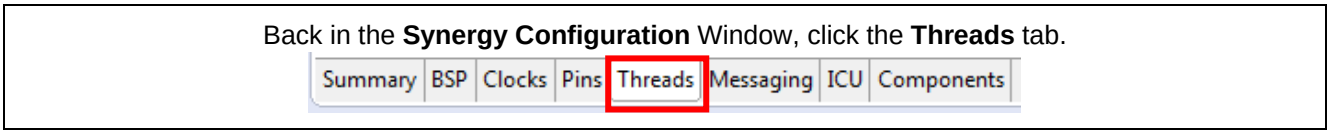


Figure 14. Synergy Configuration Threads Tab

5. Select the **HAL/Common thread** (on the top left).

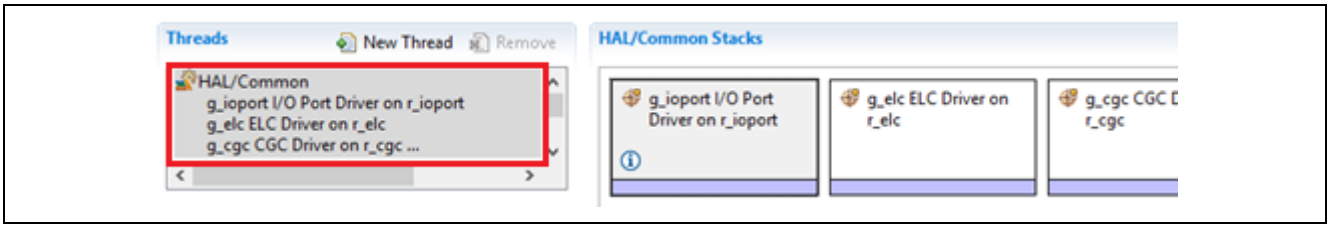


Figure 15. Threads

6. In the **HAL/Common Stacks** area, click the **New Stack** button.

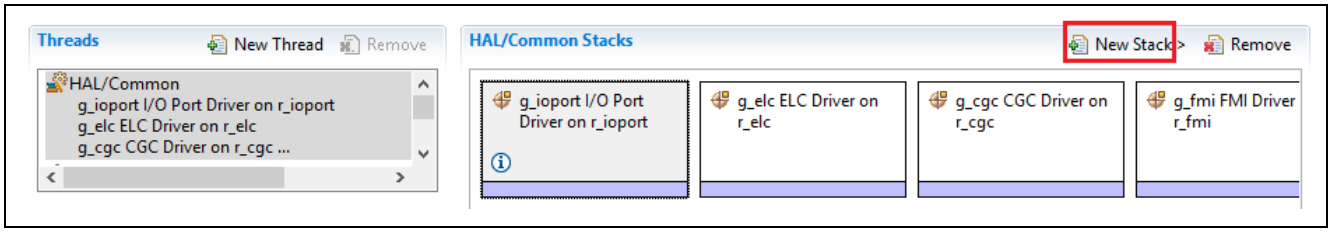


Figure 16. Add a Timer Driver Module to the HAL/Common Thread part 1

7. In the menu, select **Driver > Timers > Timer Driver on r_gpt**.

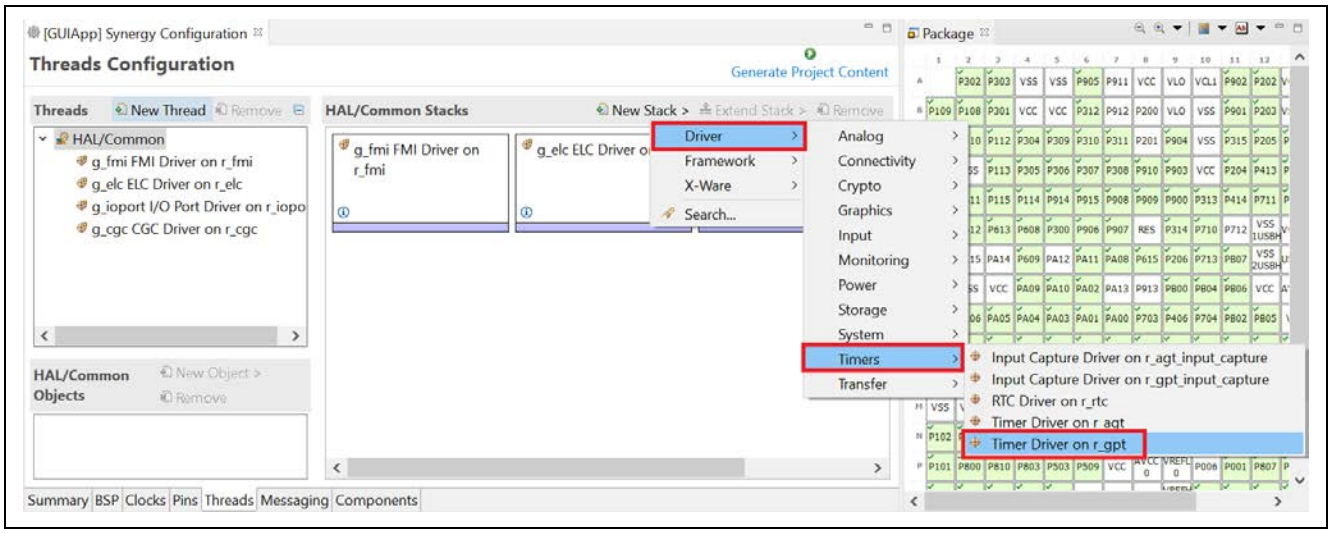


Figure 17. Add a Timer Driver Module to the HAL/Common Thread part 2

8. In the **HAL/Common Stacks** area, select the newly created module **g_timer Timer Driver on r_gpt** and configure the **Properties Window** of **Timer Driver on r_gpt**.

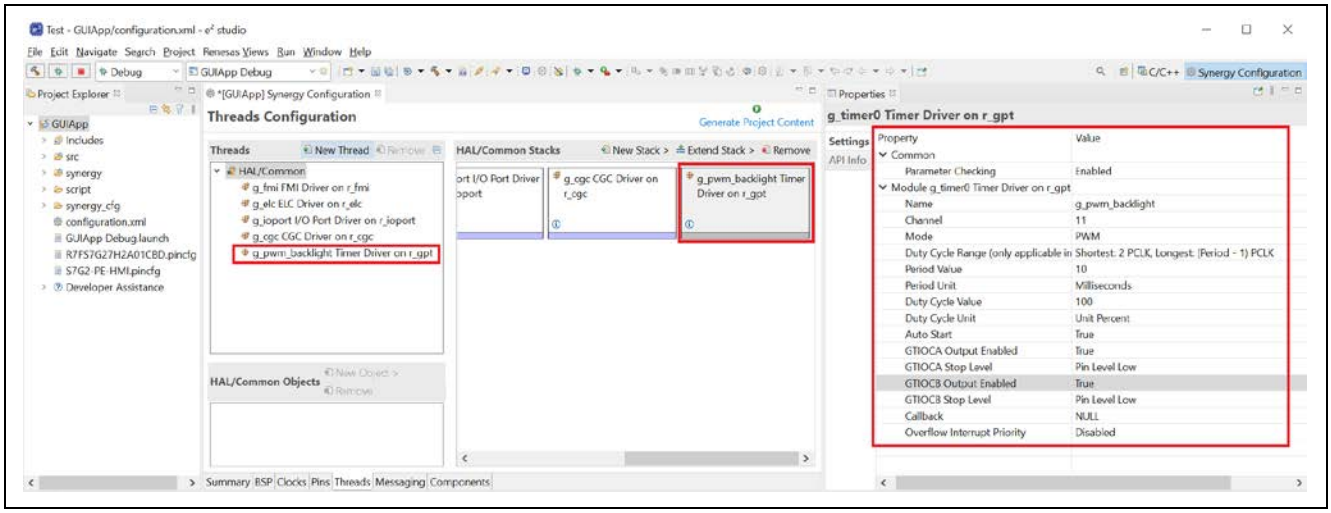


Figure 18. Select the Newly Created Timer Driver Module

The next steps add the required software to enable the touch screen and configure the LCD controller.

The touch screen requires several frameworks and drivers to be used. External interrupts determine when to read the data, an I²C driver handles the reads, and a framework translates the register data from the peripheral to touch coordinates the software can use.

9. Create a new thread by clicking **New Thread** in the **Threads** area.

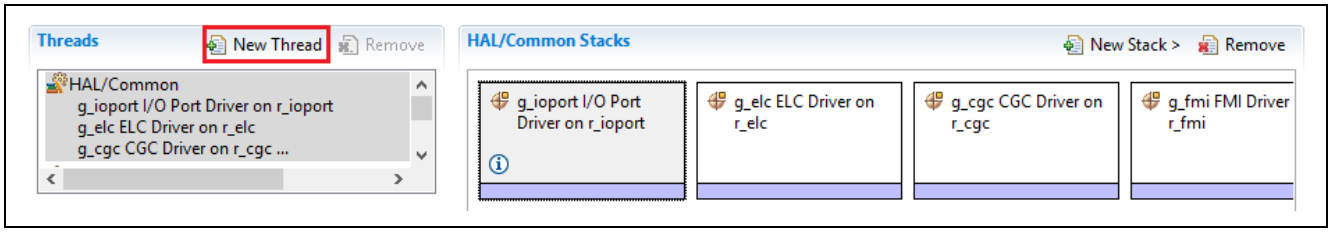


Figure 19. Create a New Thread

10. Click on **New Thread** to pull up the properties.
11. Edit the **Properties** to match the following:

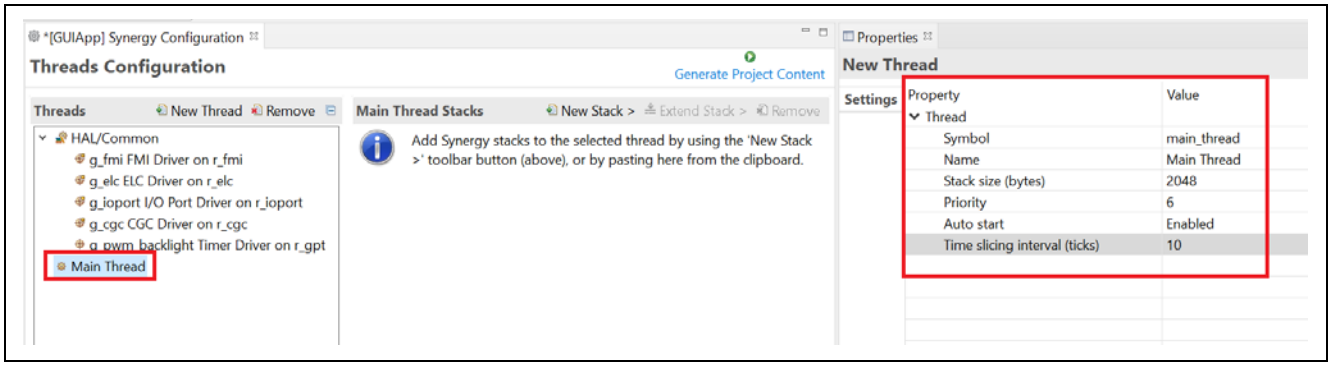


Figure 20. Configure Main Thread Properties

12. Back in the **Synergy Configuration Window**, **Threads tab, Main Thread Stacks** area and click on **New Stack**.

    Note: Be sure **Main Thread** is selected before adding new modules.

    In the **Synergy Configuration** window, **Threads** tab, **Main Thread Stacks** area, add a framework for the touch panel by selecting **New Stack**, then **Framework > Input > Touch Panel V2 Framework on sf_touch_panel_v2**.

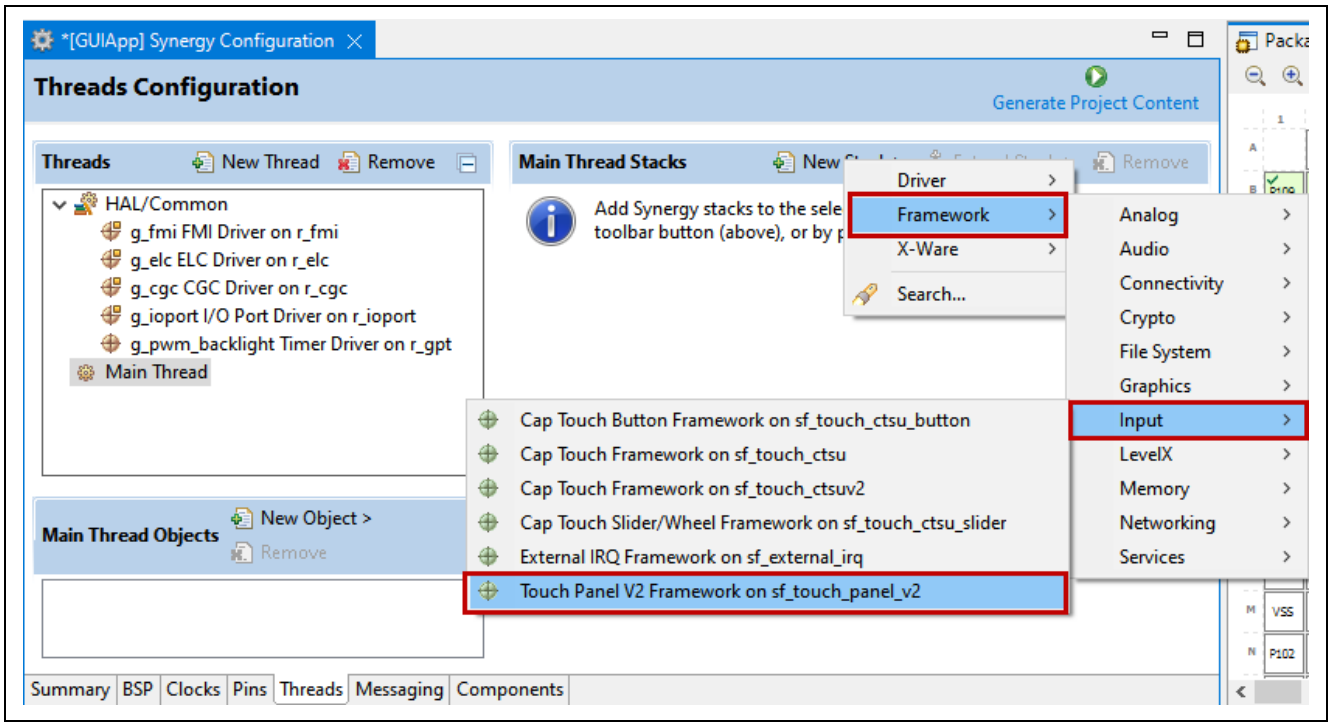


Figure 21. Adding Touch Panel Framework

13. In the **Synergy Configuration Window > Threads tab > Main Thread Stacks area**, click on **g_sf_touch_panel Touch Panel V2 Framework sf_touch_panel_v2**. Then configure the properties for **g_sf_touch_panel Touch Panel V2 Framework sf_touch_panel_v2**.

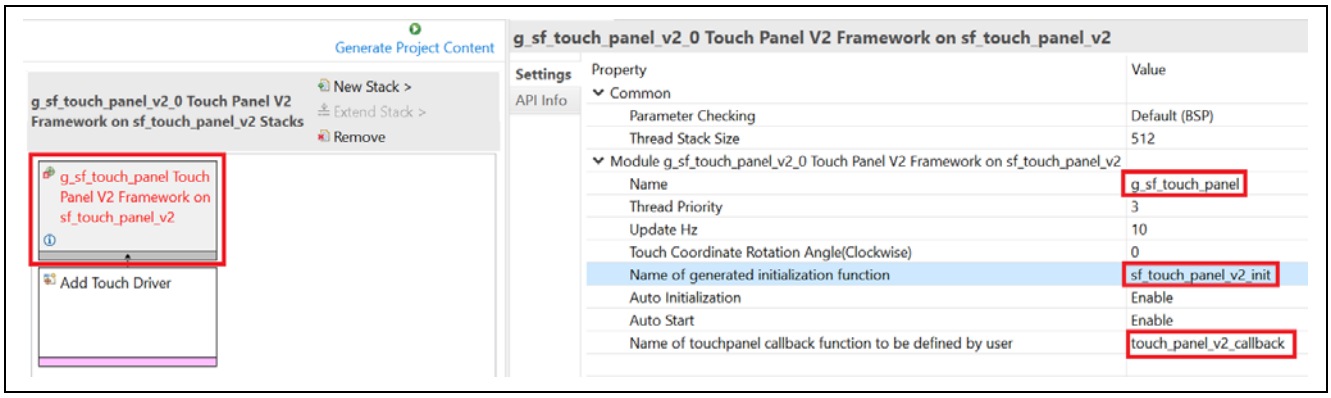


Figure 22. Configuring Touch Panel V2 Framework Properties

14. In the **Synergy Configuration Window** > **Threads tab** > **Main Thread Stacks** area, click on **Add Touch Driver > New > Touch_panel_chip_ft5x06**.

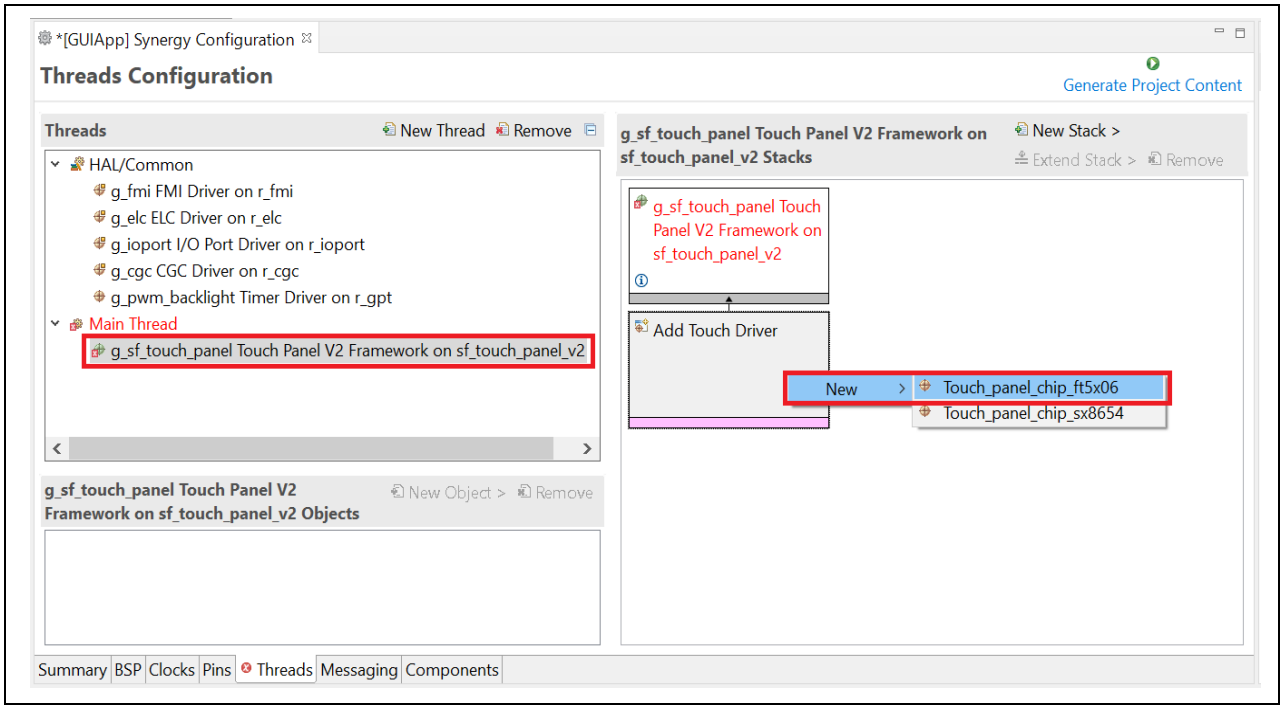


Figure 23. Add the Touch_panel_chip_ft5x06 Touch driver

15. Configure the **Touch_panel_chip_ftx06** properties as shown.

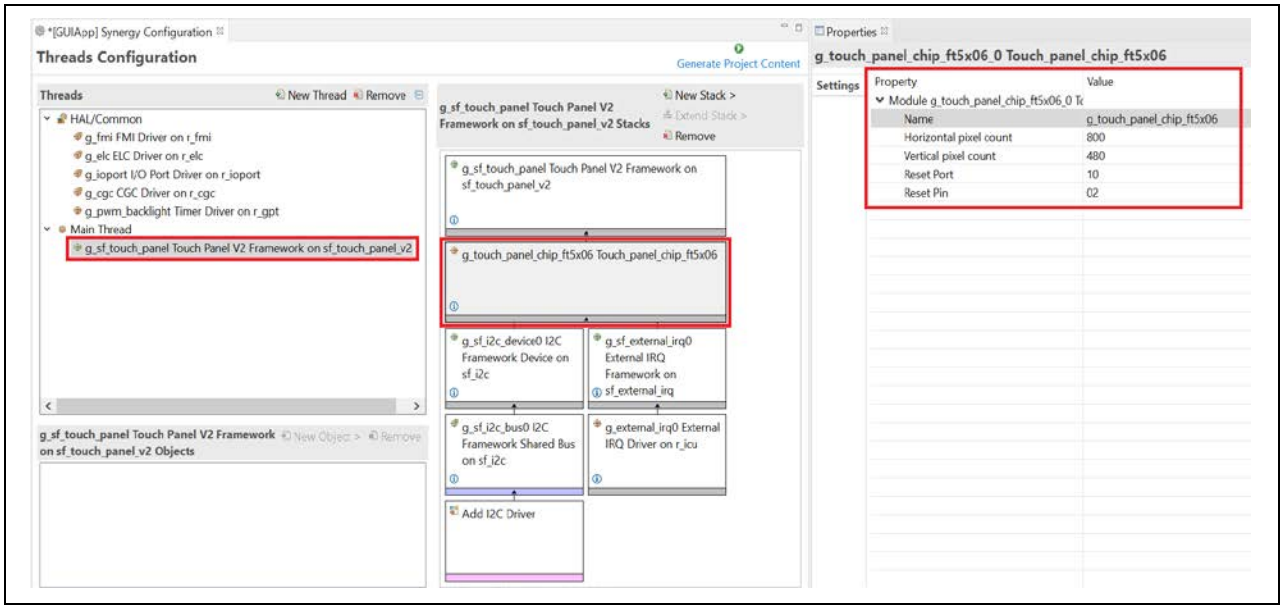


Figure 24. Configure Touch_panel_chip_ft5x06 Properties

Notice that the Synergy Configurator has now already created the external IRQ framework and has a placeholder for the external IRQ and I²C driver stacks (see Figure 25).

The Touch Panel V2 Framework module scans data from a touch controller and invokes the user registered touch panel callback when a touch event occurs. (If the user callback is not registered, the `sf_touch_panel_v2_api_t::touchDataGet` API function can be used to retrieve the data). The **SF External Interrupt** is a framework layer used by **the touch controller driver.**

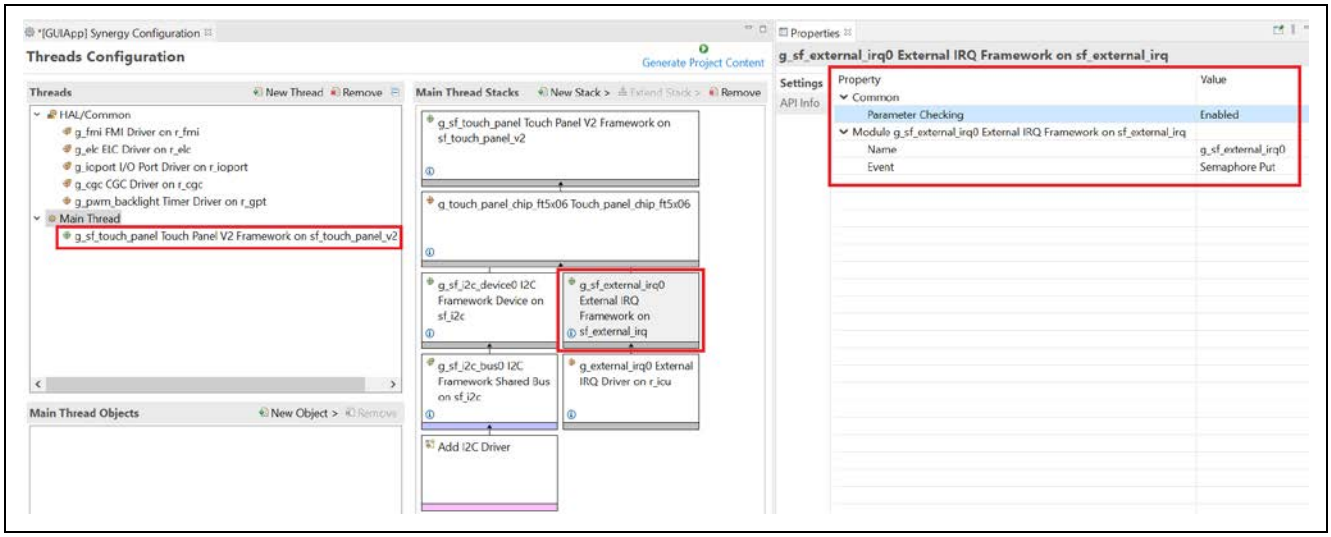


Figure 25. Configure the properties for External IRQ Framework Stack

16. Select the **External IRQ Driver on r_icu** and configure the following properties.

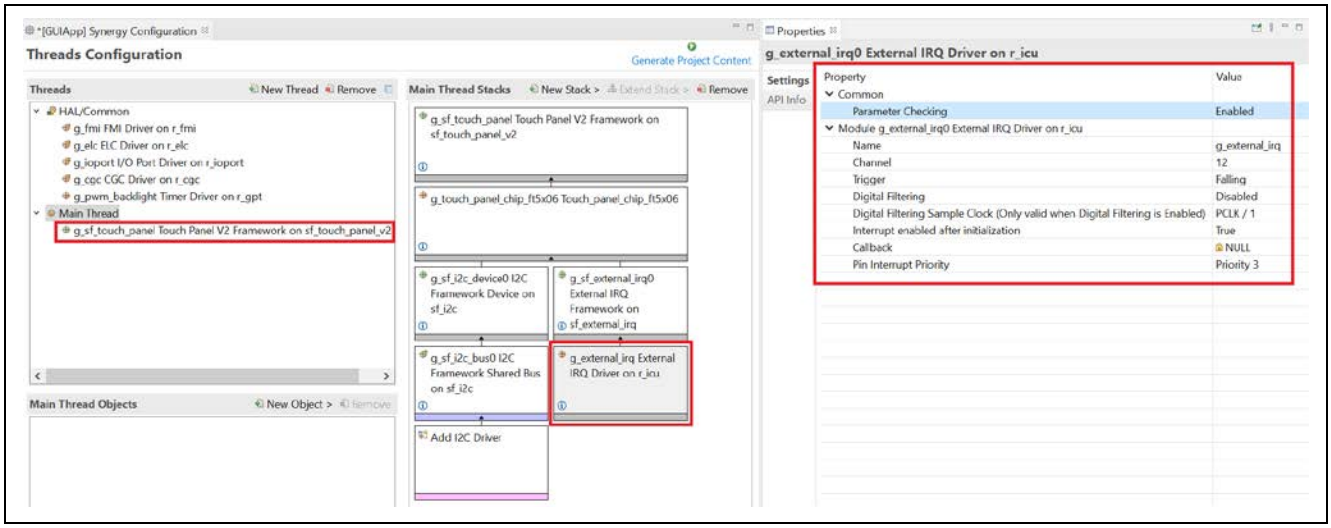


Figure 26. Configuring External IRQ Driver on r_icu Properties

17. In the **Synergy Configuration** window > **Threads** tab > **Main Thread Stacks** area, click on **g_sf_i2c_device0 I2C Framework Device on sf_i2c**. Then configure the properties for **g_sf_i2c_device0 I2C Framework Device on sf_i2c**.

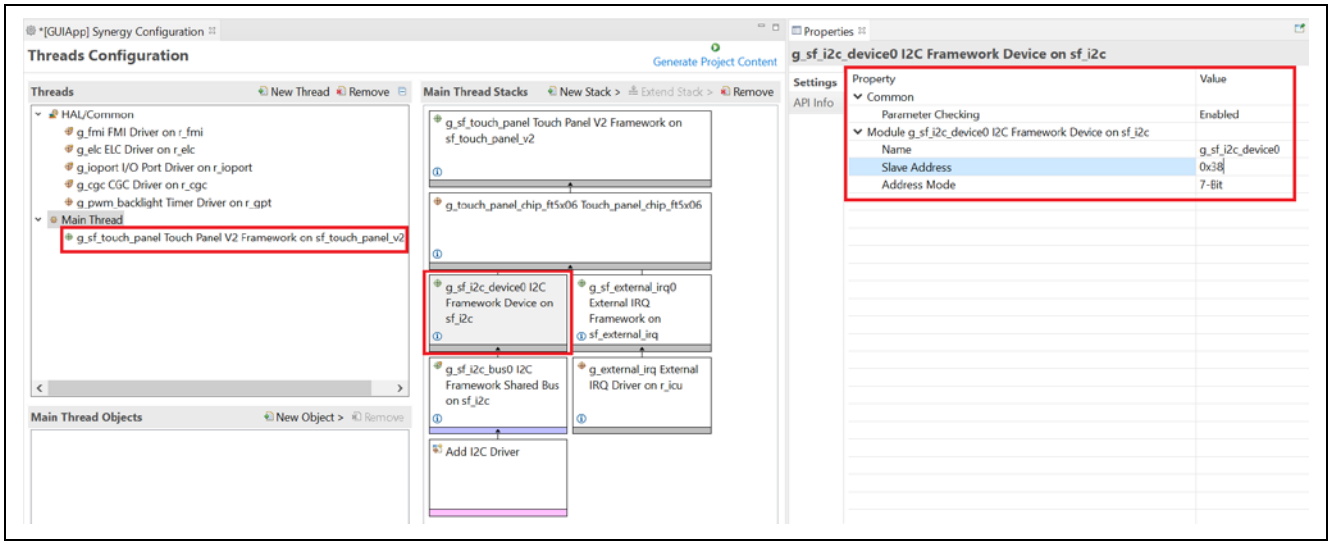


Figure 27. Configure the properties for g_sf_i2c_device0 I2C Framework Device on sf_i2c

18. In the **Synergy Configuration Window** > **Threads tab** > **Main Thread Stacks** area, click **g_sf_i2c_bus0 I2C Framework Shared Bus on sf_i2c >** Configure the properties for **g_sf_i2c_bus0 I2C Framework Shared Bus on sf_i2c**

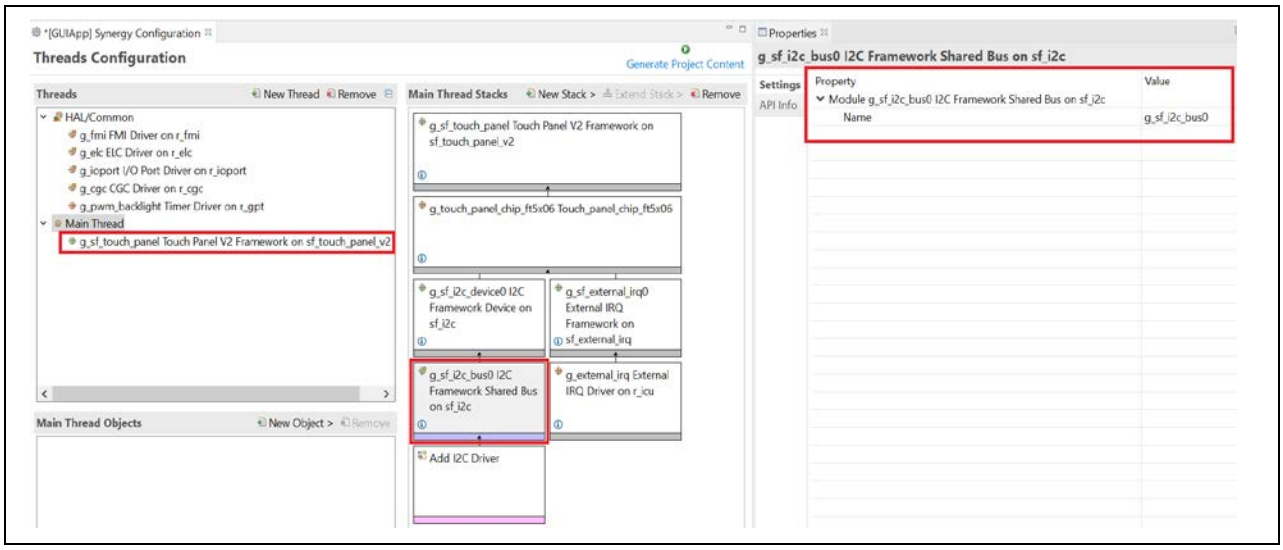


Figure 28. Configure g_sf_i2c_bus0 I2C Framework Shared Bus on sf_i2c Properties

19. In the **Synergy Configuration** window, **Threads** tab, **Main Thread Stacks** area, add a driver for the I²C bus by right-clicking **Add I2C Driver**, and then selecting **New > I2C Master Driver on r_iic**.

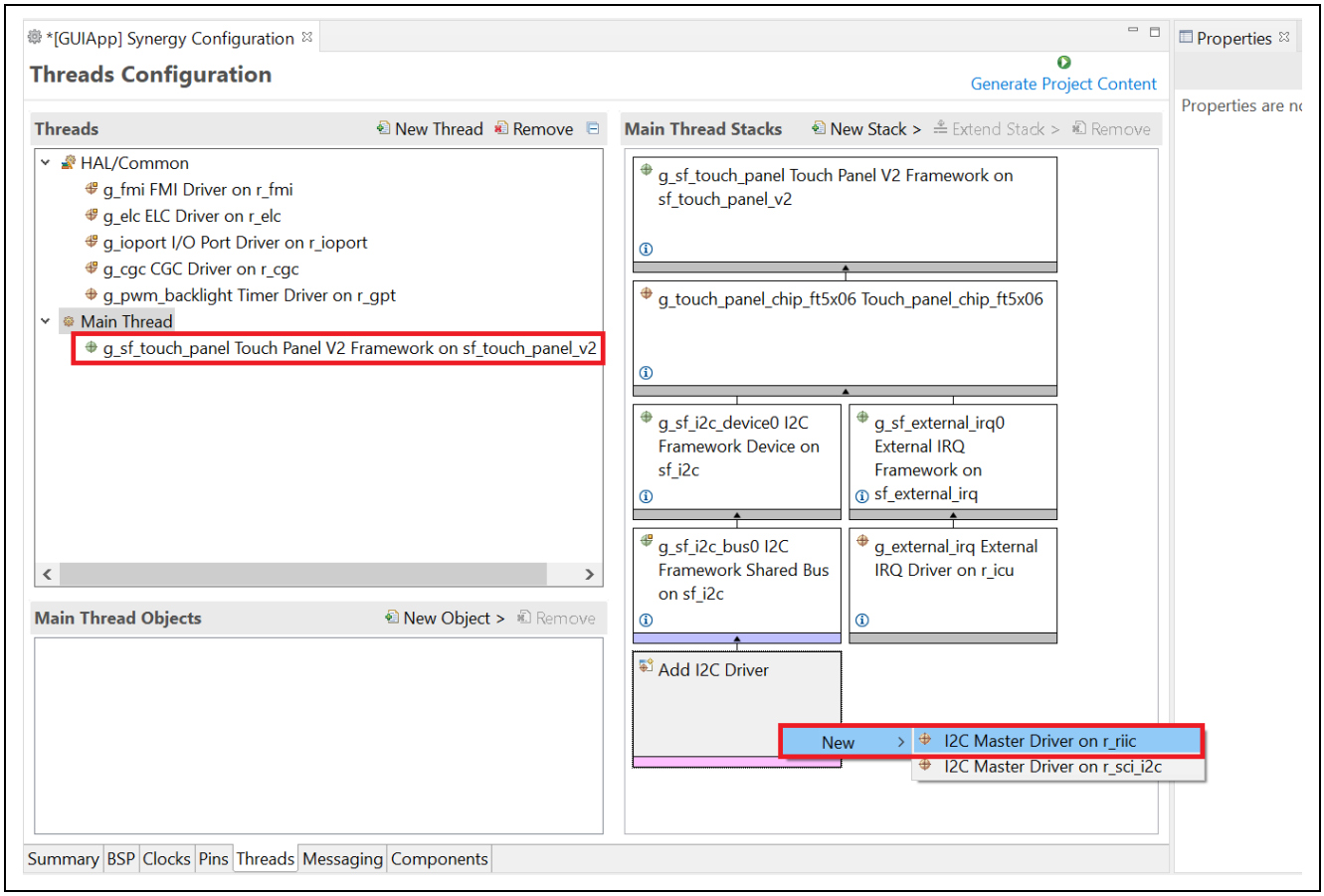


Figure 29. Adding I²C Driver I2C Master Driver on r_iic

20. In the **Synergy Configuration** window > **Threads** tab > **Main Thread Stacks** area, click on **I2C Master Driver on r_riic** and configure the Properties for **I2C Master Driver on r_riic**

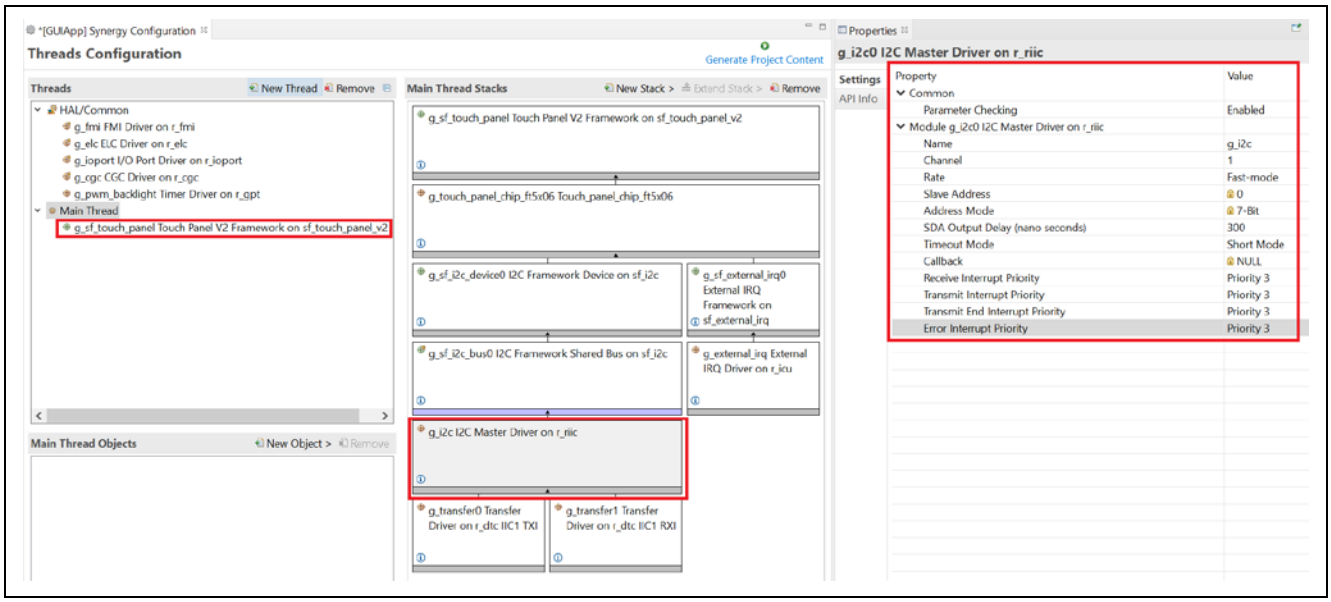


Figure 30. Configuring I²C Master Driver on r_riic

21. In the **Synergy Configuration** window > **Threads** tab > **Main Thread Stacks** area, click on **g_transfer0 Transfer Driver on r_dtc SCI7 TXI** and configure the properties for **g_transfer0 Transfer Driver on r_dtc SCI7 TXI**

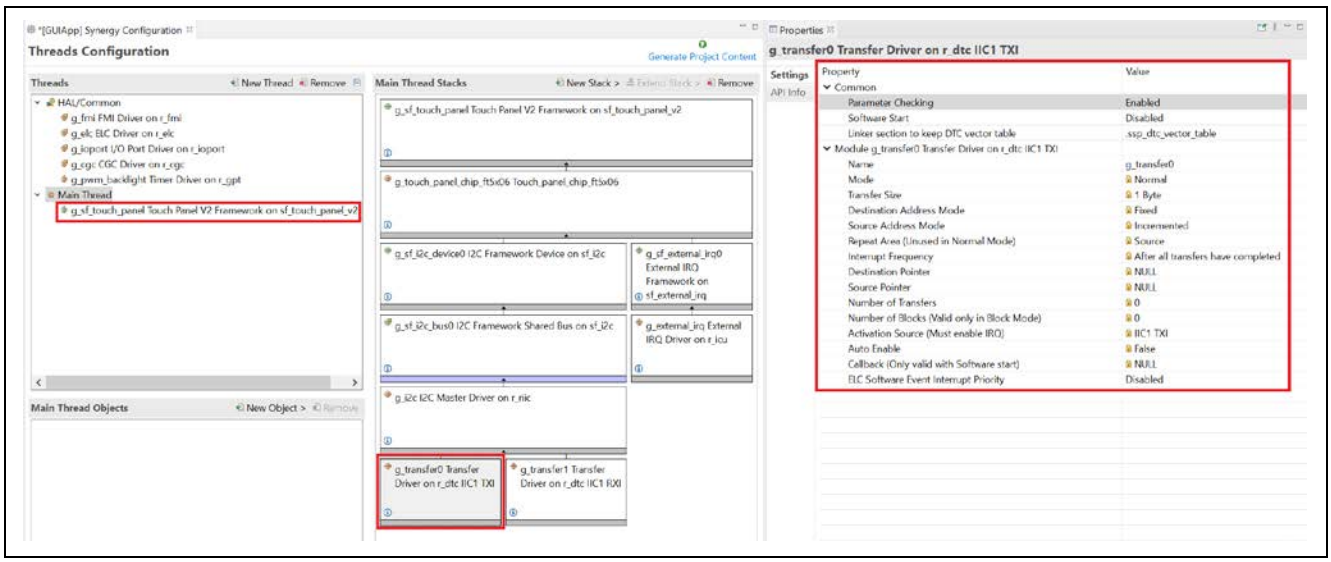


Figure 31. Configure the Properties of g_transfer0 Transfer Driver on r_dtc SCI7 TXI

22. In the **Synergy Configuration** window > **Threads** tab > **Main Thread Stacks** area, click on **g_transfer1 Transfer Driver on r_dtc SCI7 RXI** and configure the properties for **g_transfer1 Transfer Driver on r_dtc SCI7 RXI**.

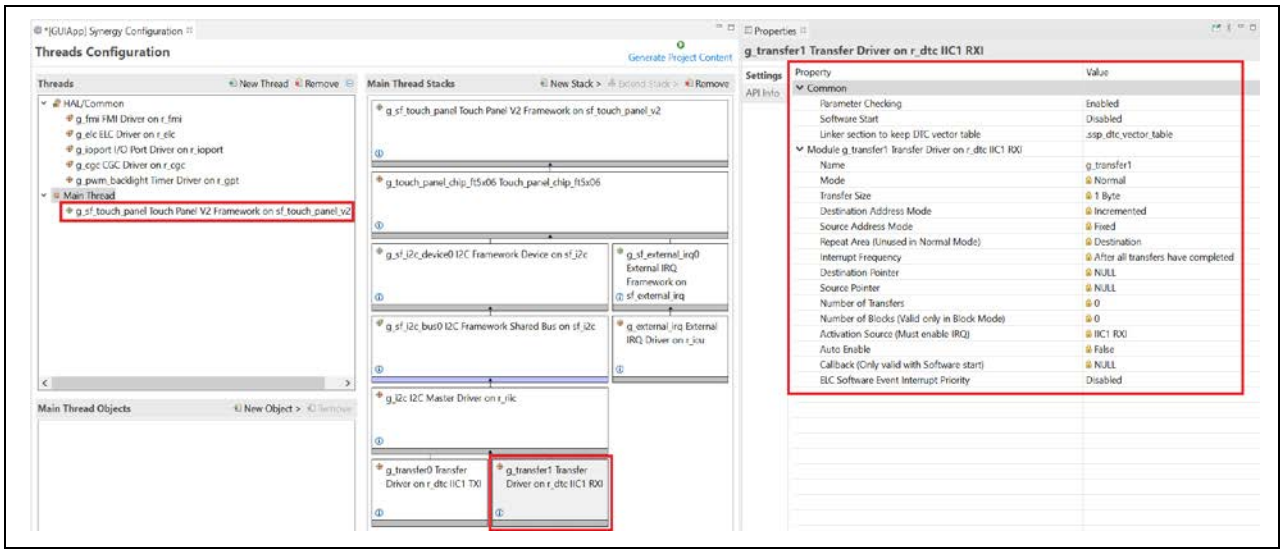


Figure 32. Configure the Properties of g_transfer1 Transfer Driver on r_dtc SCI7 RXI

23. Under **Main Thread Stacks**, select **New Stack**, then **X-Ware >GUIX >GUIX on gx**.

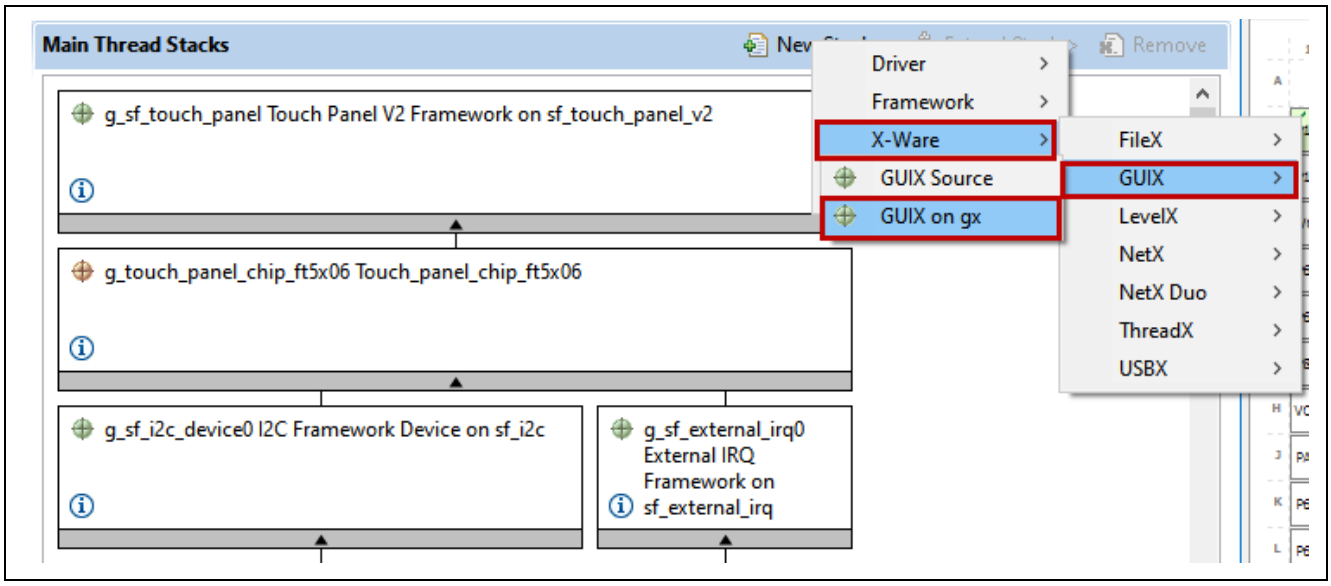


Figure 33. GUIX on gx

Notice that the Synergy Configurator has now already created the **GUIX Port on sf_el_gx framework**, **Display Driver** and has a placeholder for the JPEG decode and D/AVE hardware accelerator stacks.

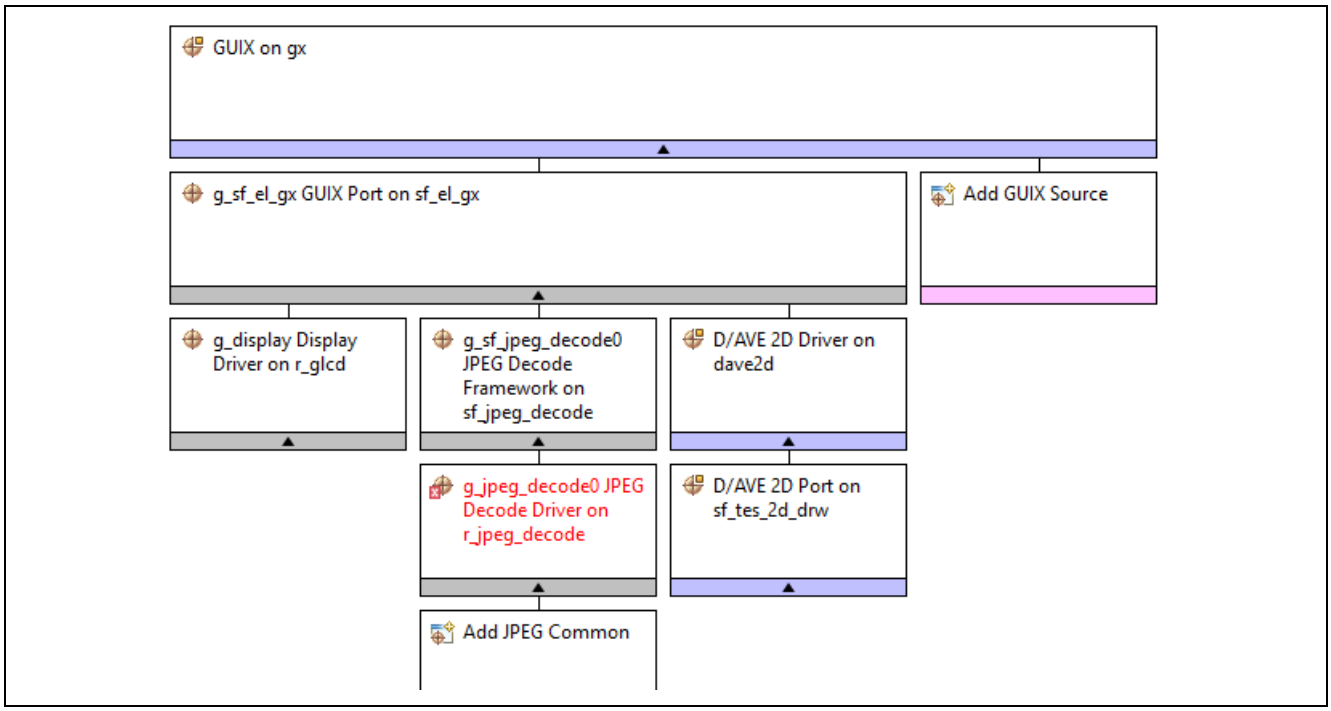


Figure 34. GUIX on gx

24. Select **GUIX on gx** and configure the following **Properties**.

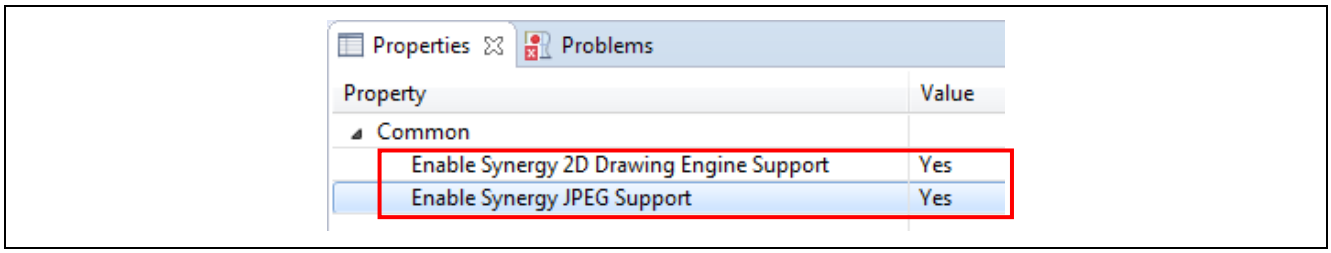


Figure 35. GUIX on gx Properties

25. Add **JPEG Common** to the Decode Driver on **r_jpeg_decode**.

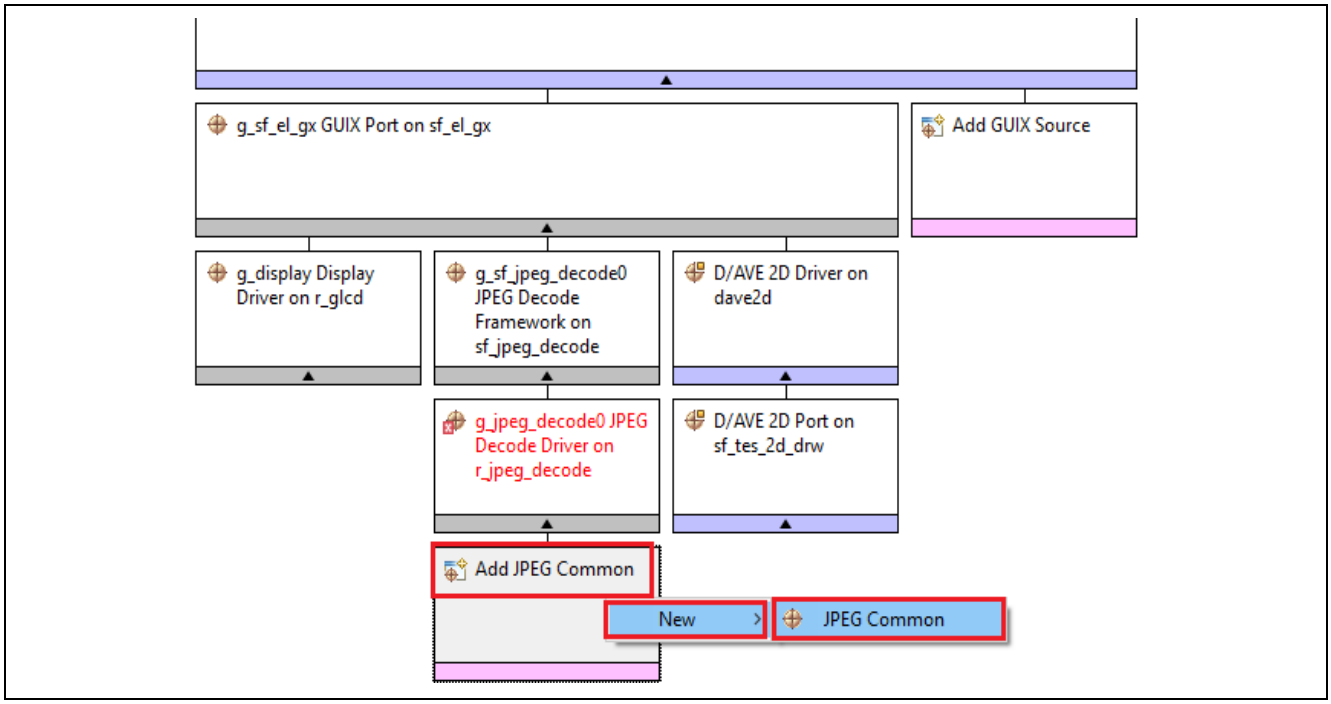


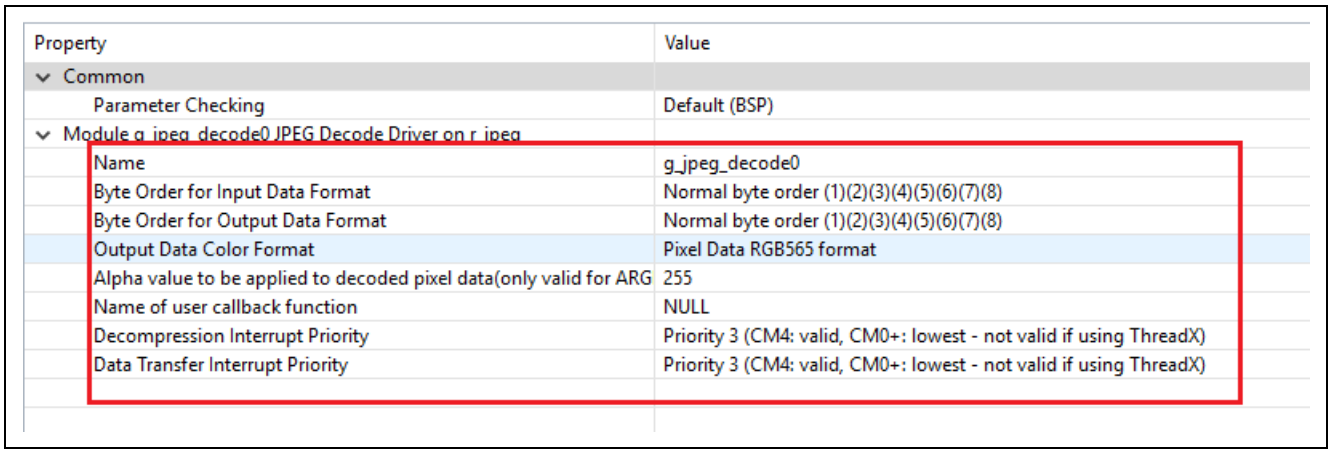
Figure 36. JPEG Common module

26. Select **GUIX Port on sf_el_gx** and configure the following under **Property**.

Property	Value
◢ Common	
Parameter Checking	Enabled
◢ Module g_sf_el_gx GUIX Port on sf_el_gx	
Name	g_sf_el_gx
Display Driver Configuration Inheritance	Inherit Graphics Screen 1
Name of User Callback function	NULL
Screen Rotation Angle(Clockwise)	0
GUIX Canvas Buffer (required if rotation angle is not zero)	Not used
Size of JPEG Work Buffer (valid if JPEG hardware acceleration e	1000
Memory section for GUIX Canvas Buffer	sdram
Memory section for JPEG Work Buffer	sdram

Figure 37. GUIX Port on sf_el_gx Properties

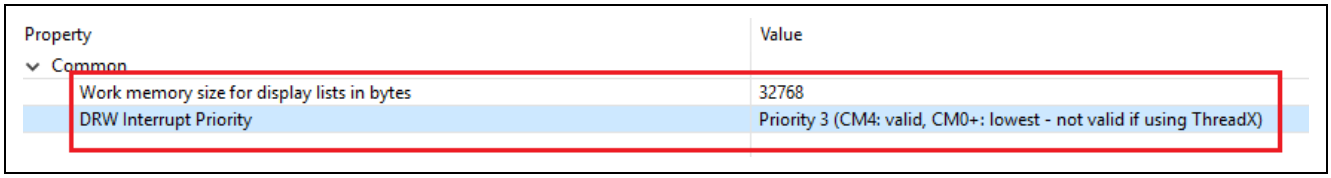
27. Select **JPEG Decode Driver on r_jpeg** and configure the following interrupt properties. Note that Priority 3 is just an arbitrary number.



Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_jpeg_decode0 JPEG Decode Driver on r_jpeg	
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 ARG	255
Name of user callback function	NULL
Decompression Interrupt Priority	Priority 3 (CM4: valid, CM0+: lowest - not valid if using ThreadX)
Data Transfer Interrupt Priority	Priority 3 (CM4: valid, CM0+: lowest - not valid if using ThreadX)

Figure 38. JPEG Decode Driver on r_jpeg Properties

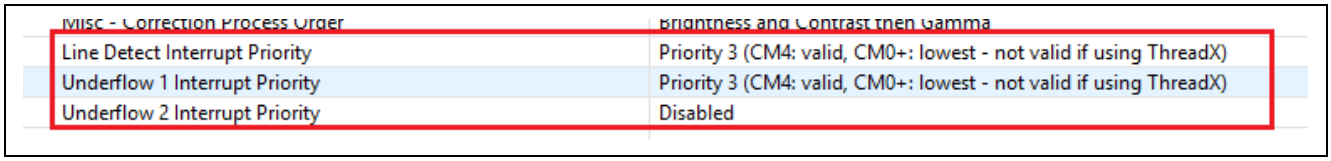
28. Under **Main Thread Stacks**, select **D/AVE 2D Port on sf_tes_2d_drw** and configure the following properties.



Property	Value
Common	
Work memory size for display lists in bytes	32768
DRW Interrupt Priority	Priority 3 (CM4: valid, CM0+: lowest - not valid if using ThreadX)

Figure 39. D/AVE 2D Port Properties

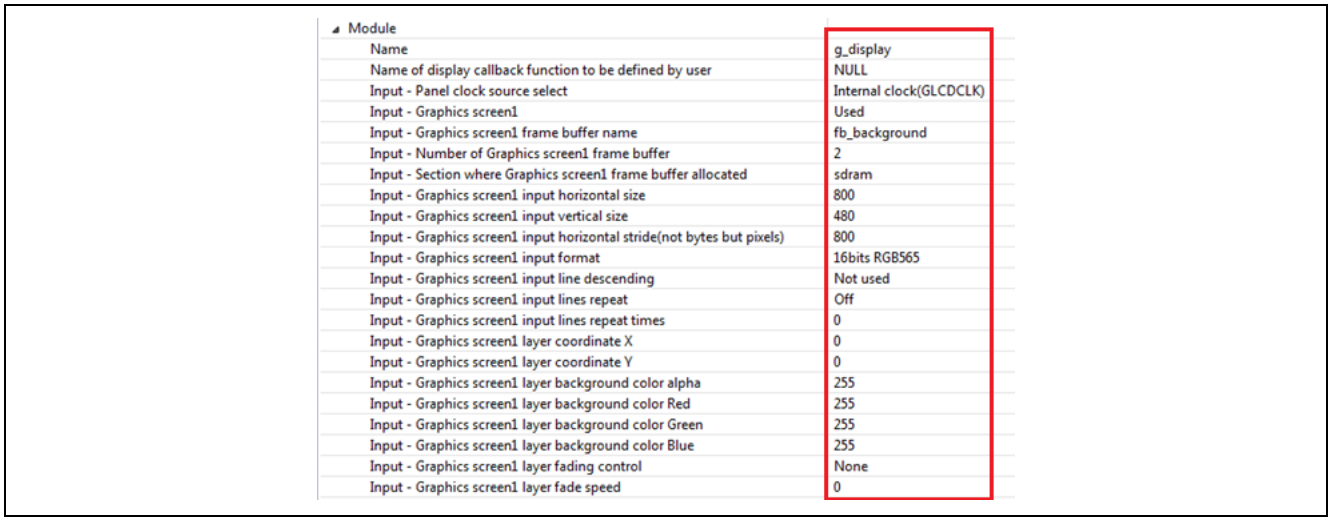
29. Under **Main Thread Stacks**, select **Display Driver on r_glcd** and configure the following interrupt properties.



Misc - Correction Process Order	Brightness and Contrast then Gamma
Line Detect Interrupt Priority	Priority 3 (CM4: valid, CM0+: lowest - not valid if using ThreadX)
Underflow 1 Interrupt Priority	Priority 3 (CM4: valid, CM0+: lowest - not valid if using ThreadX)
Underflow 2 Interrupt Priority	Disabled

Figure 40. Interrupt Properties

30. Scroll down to show the following Graphics Screen 1 properties.



Module	
Name	g_display
Name of display callback function to be defined by user	NULL
Input - Panel clock source select	Internal clock(GLCDCLK)
Input - Graphics screen1	Used
Input - Graphics screen1 frame buffer name	fb_background
Input - Number of Graphics screen1 frame buffer	2
Input - Section where Graphics screen1 frame buffer allocated	sdram
Input - Graphics screen1 input horizontal size	800
Input - Graphics screen1 input vertical size	480
Input - Graphics screen1 input horizontal stride(not bytes but pixels)	800
Input - Graphics screen1 input format	16bits RGB565
Input - Graphics screen1 input line descending	Not used
Input - Graphics screen1 input lines repeat	Off
Input - Graphics screen1 input lines repeat times	0
Input - Graphics screen1 layer coordinate X	0
Input - Graphics screen1 layer coordinate Y	0
Input - Graphics screen1 layer background color alpha	255
Input - Graphics screen1 layer background color Red	255
Input - Graphics screen1 layer background color Green	255
Input - Graphics screen1 layer background color Blue	255
Input - Graphics screen1 layer fading control	None
Input - Graphics screen1 layer fade speed	0

Figure 41. Graphics Screen 1 Properties

31. Configure the following output properties.

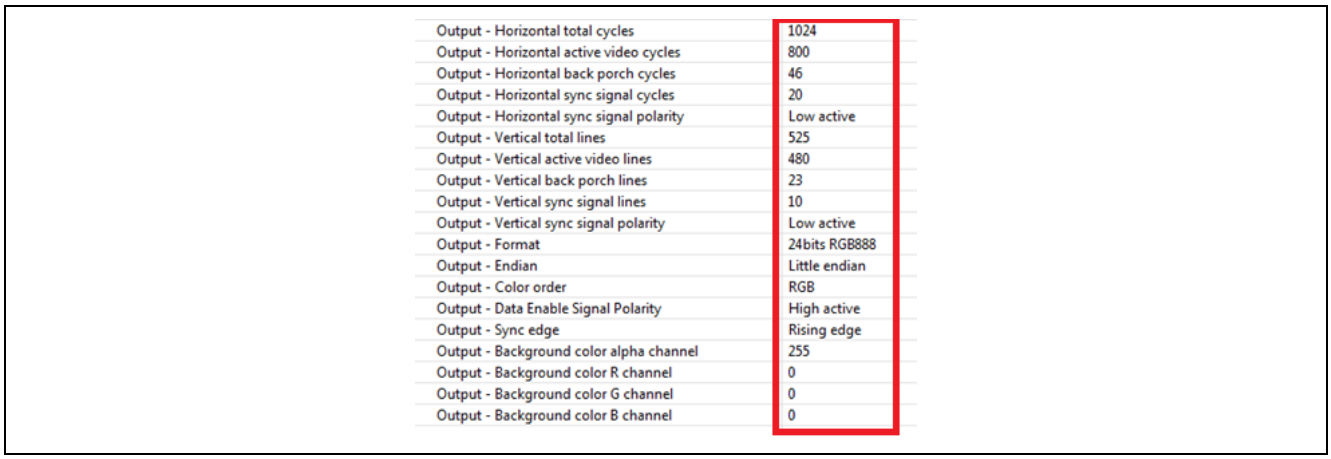


Figure 42. Output Screen 2 Properties

32. Change the following TCON settings to match.

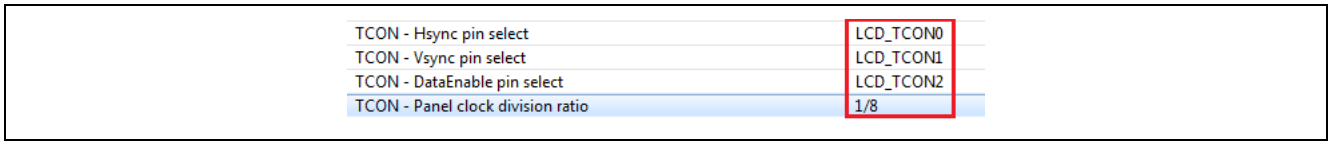


Figure 43. TCON Settings

33. Save the project by pressing **Ctrl + s** on the keyboard.
34. Click the **Generate Project Content** button to update the project files.

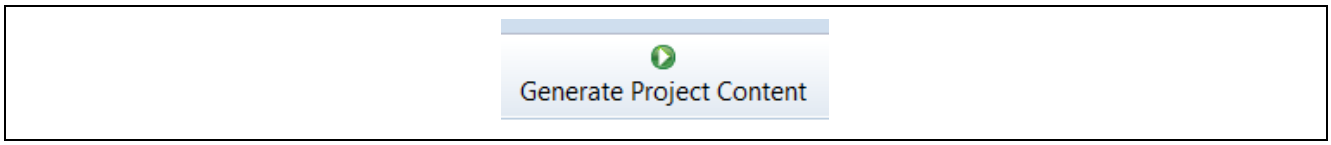


Figure 44. Generate Project Content

35. Open **Windows Explorer** and locate the files included with this application note. Locate the file `Source Files\main_thread_entry.c`. Drag the file from the **Windows Explorer Window** into the **src** folder inside the e² studio **Project Explorer** window.
    A. When asked how to import the selected files, click **OK** to copy the files.
    B. When asked if you want to overwrite, click **Yes**.

Note: This file contains the Main Thread event handling code. It reads low level touchscreen events from the queue and transforms them to graphical user interface actions.

5. Creating the GUIX interface using GUIX Studio

Now that the base project is set up, you can start adding the GUIX components.

10. Create a new folder named **gui** inside the **src** by right clicking on the **src** folder and selecting **New > Folder**.

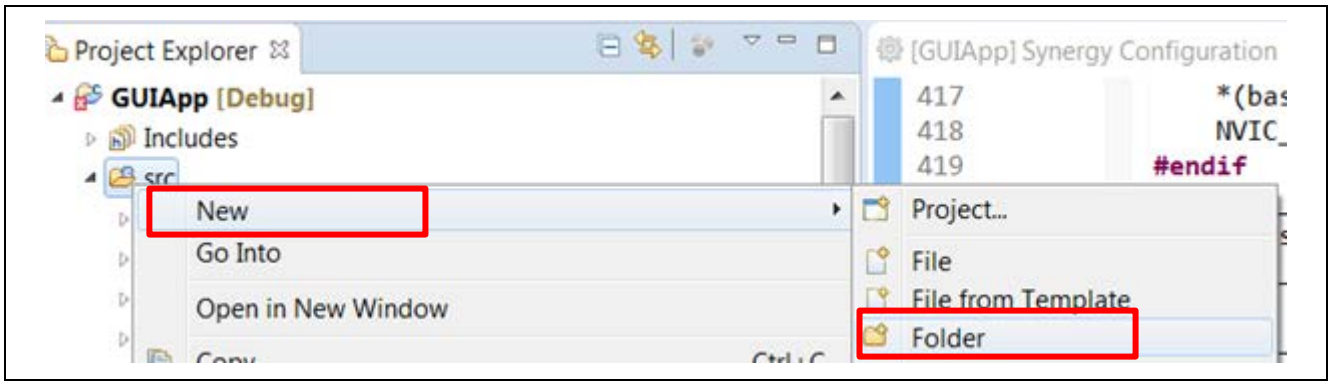


Figure 45. Creating a New Folder

11. Create another new folder named **guix_studio** in the root folder of the project by right-clicking **GUIApp** and selecting **New** > **Folder**. The final folder layout should now look like the following figure.

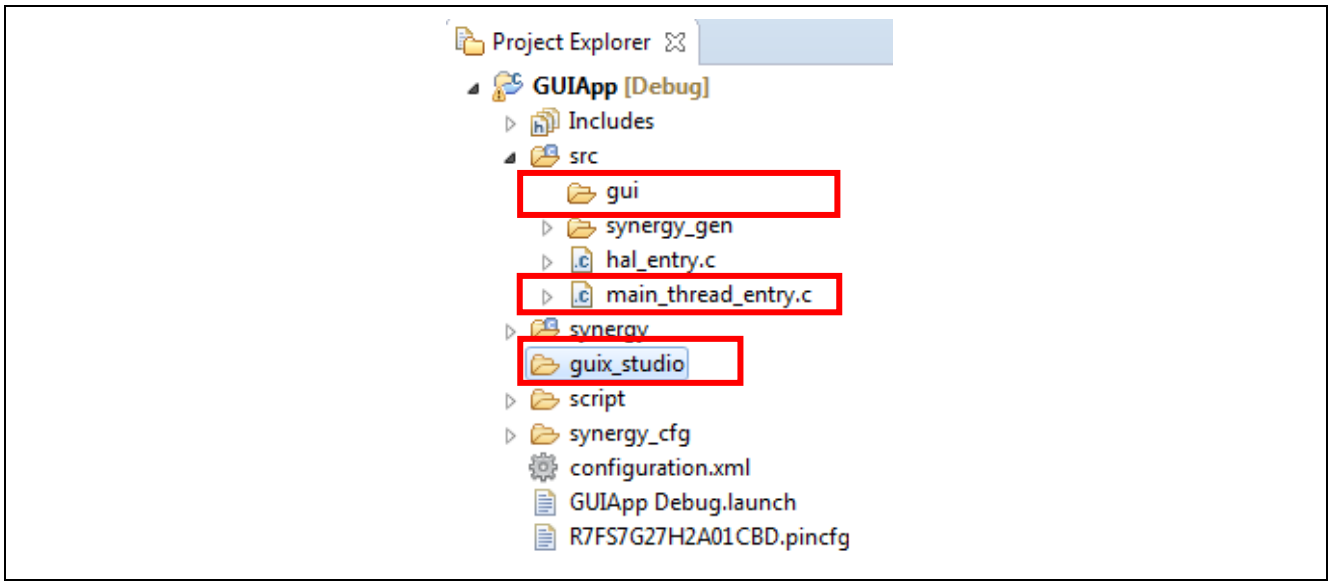


Figure 46. Final Folder List

12. Open GUIX Studio by clicking the desktop icon or by clicking the **GUIX Studio** icon in the **Windows Start** menu, **All Programs** > **Express Logic** > **GUIX Studio folder**.

Figure 47. Start GUIX Studio

13. In the **Recent Projects** dialog, click the button **Create New Project…**

Figure 48. Create New Project

14. Name the project **guiapp**.

Important: Filenames are generated by appending names to the project name. Be aware that the project name is case-sensitive. Later, files will be added to the project that you have named **guiapp**.

15. For the **Project Path**, browse to the location of the folder we created earlier called **guix_studio**.

Note: If you installed the tools into the default directories, the folder will be located at:
`C:\Users\[User]\e2_studio\workspace\GUI_APP\GUIApp\guix_studio.`

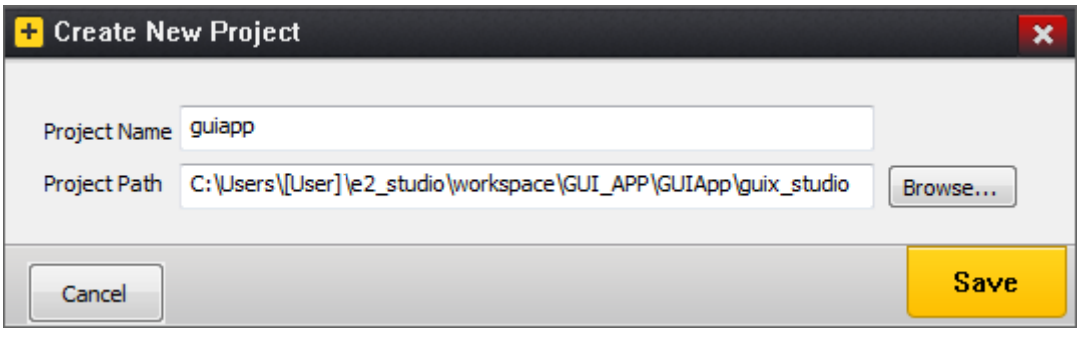


Figure 49. Create a New GUIX Project

16. Click **Save**.
17. Change the **Directories** for all three options to be:.**\src\gui**

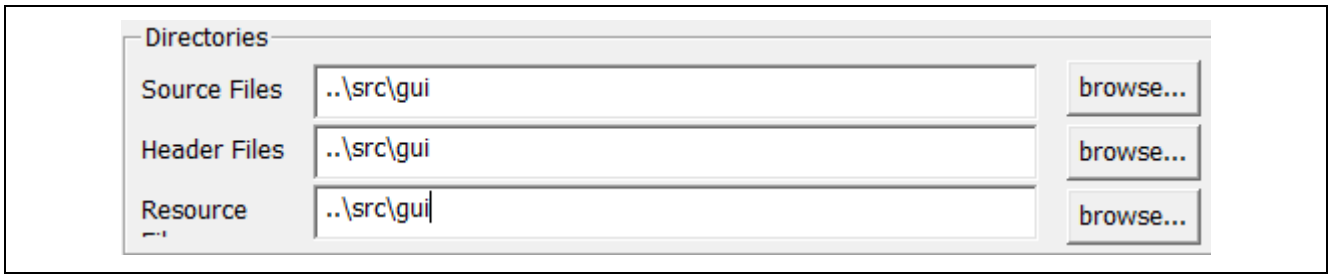


Figure 50. Correct the file Locations

CAUTION: Make sure you put in two dots ".." in the directories above.

18. Change the **Target CPU** setting to **Renesas Synergy**.
19. Change the **Toolchain** setting to **GNU**.

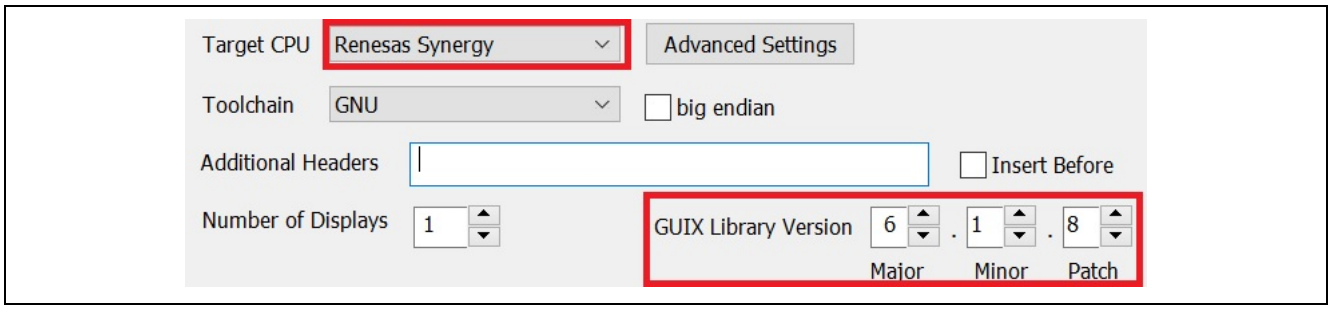


Figure 51. Target and GUIX version settings

20. Click the **Advanced Settings** button. A dialog appears.
21. Enable the **Enable 2D Drawing Engine** graphics accelerator and **Hardware JPEG Decoder** as shown in the following screen.

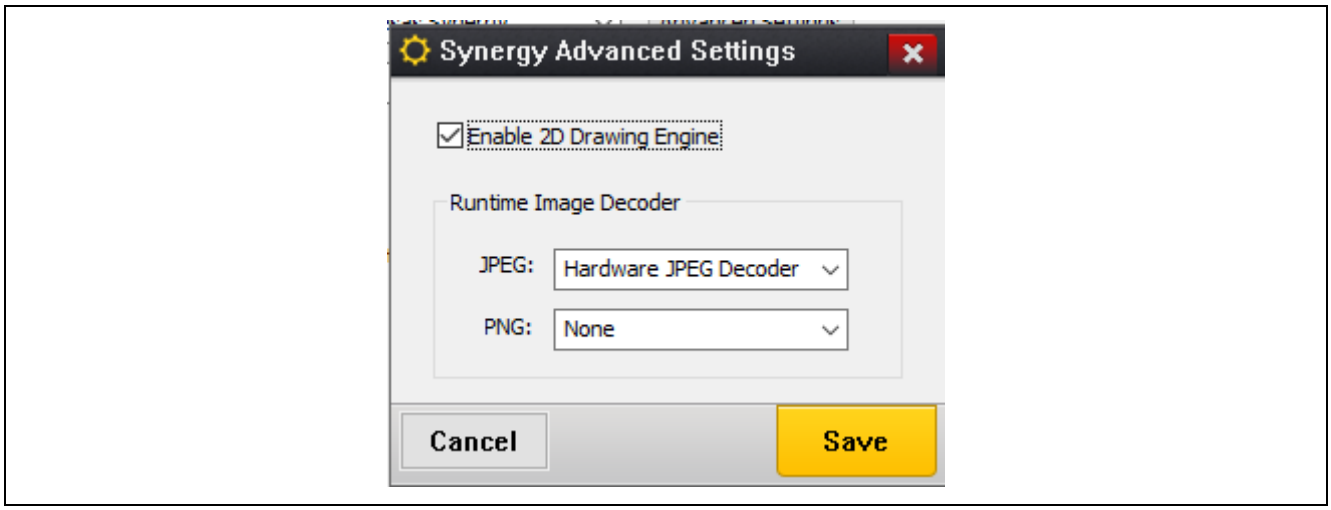


Figure 52. Synergy Advanced Settings

22. Click **Save**.
23. Set up the **Display Configuration** as shown in the following screen.

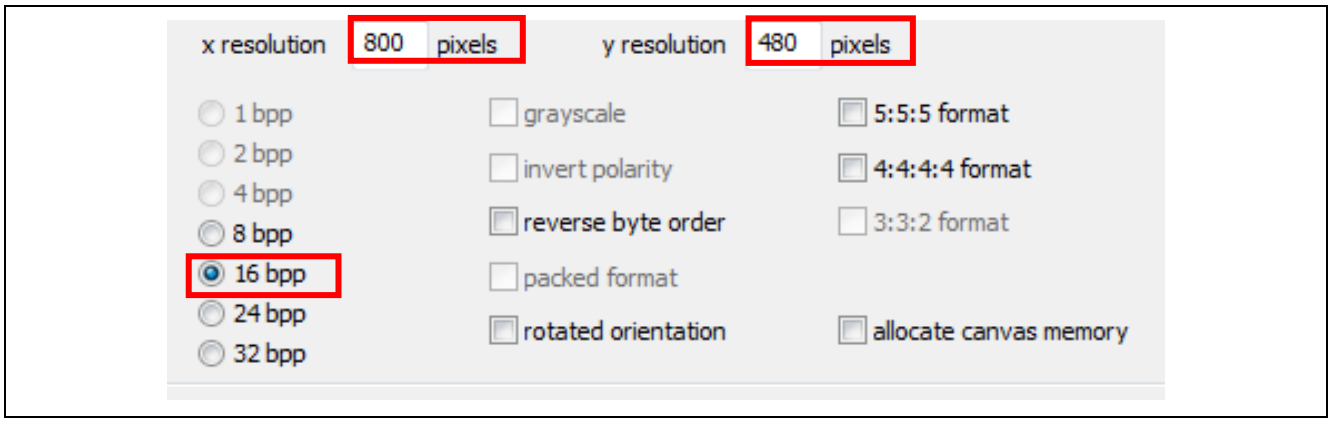


Figure 53. Configure the Display

24. Click **Save** to generate the project.
25. Right-click **display**_1 in the **Project Vie**w.
26. Select **Insert > Window > Window**.

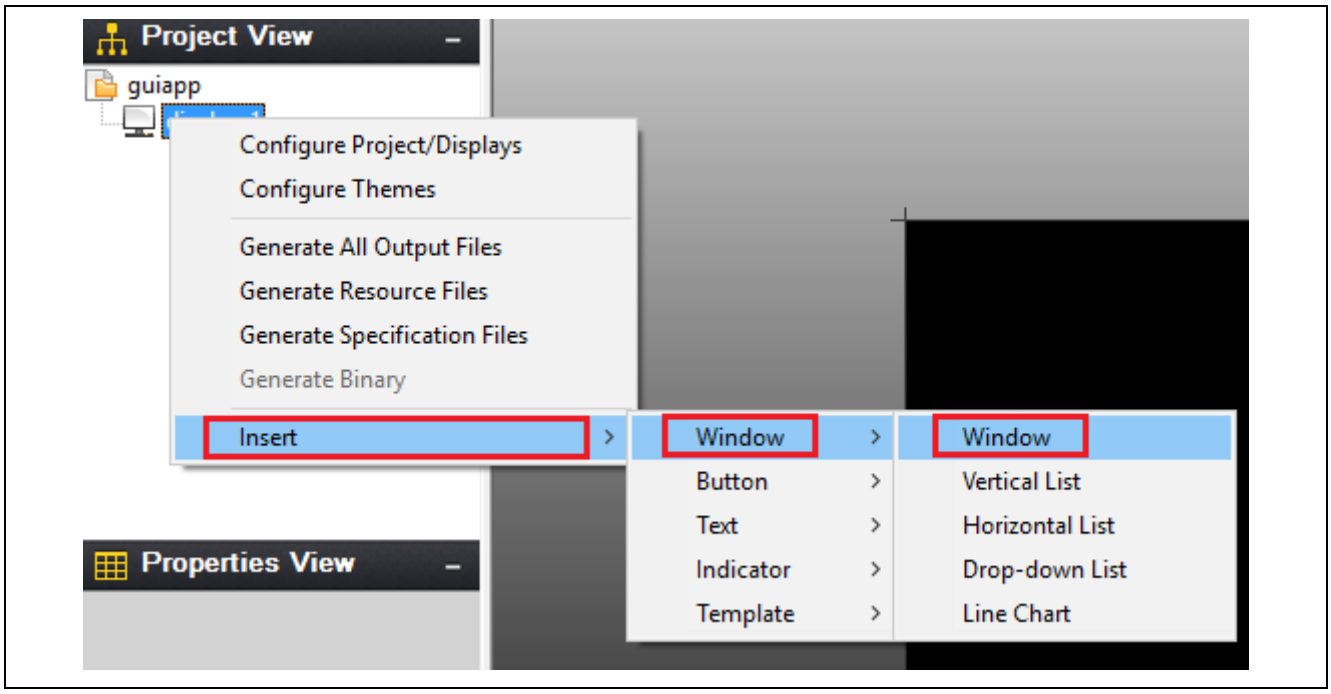


Figure 54. New Window

27. Modify the properties by selecting the new window and editing the **Properties View**. Update the current settings to match the following screen.

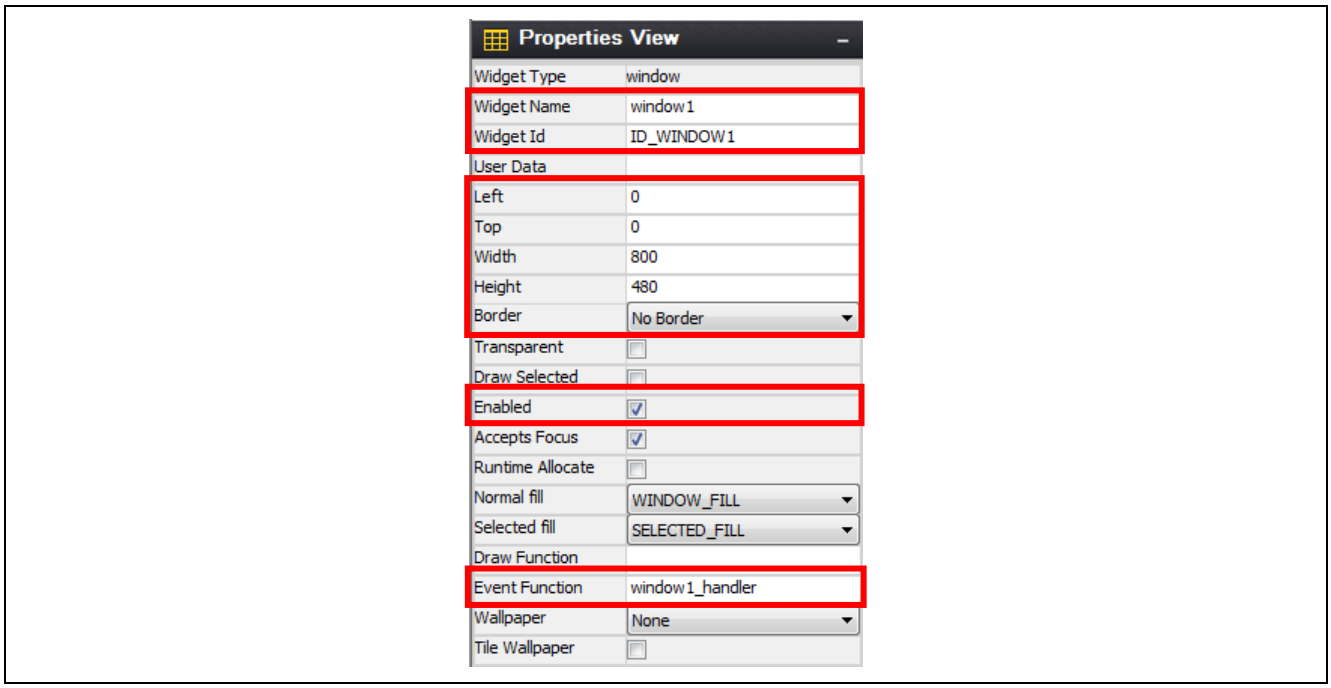


Figure 55. Configure Window1 Properties

28. In the **Project View** window, right click **display_1** and create another window by selecting **Insert** > **Window** > **Window**.
29. Modify the properties to match the following screen.

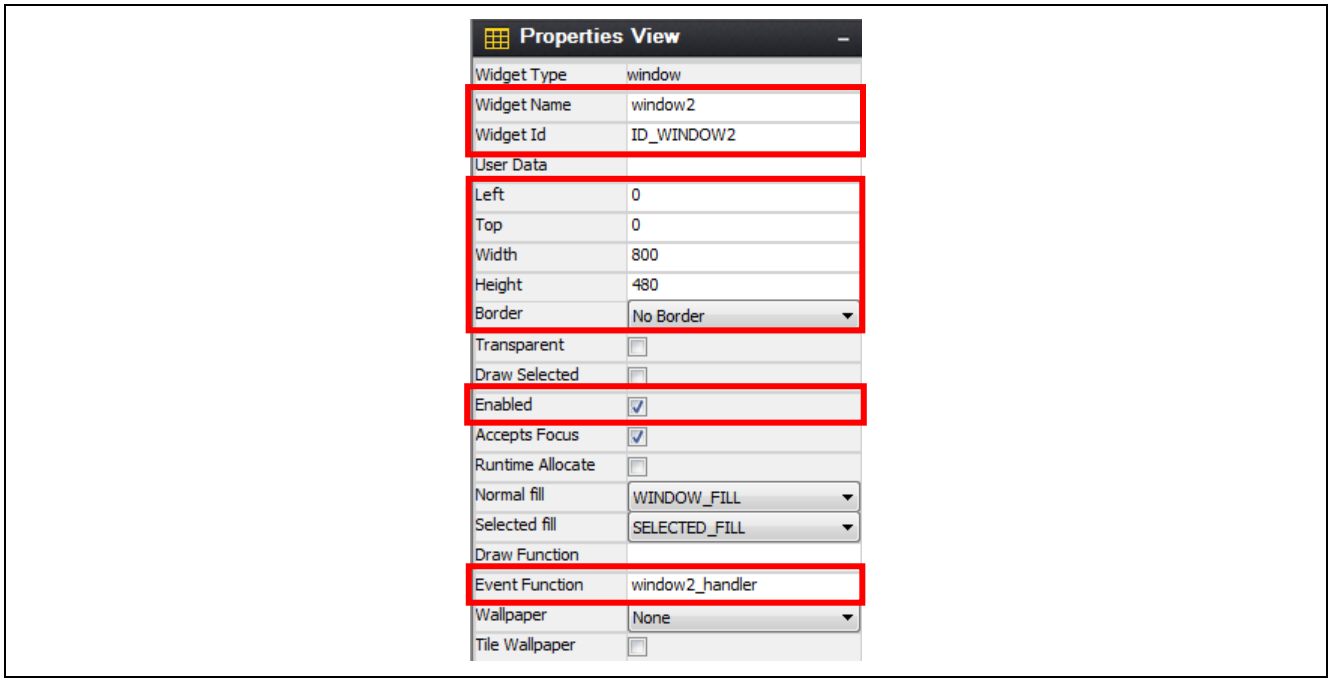


Figure 56. Configure Window2 Properties

30. In the **Project View**, right-click on **window1** and insert a **Button** (Text Button) by selecting **Insert** > **Button** > **Text Button**.

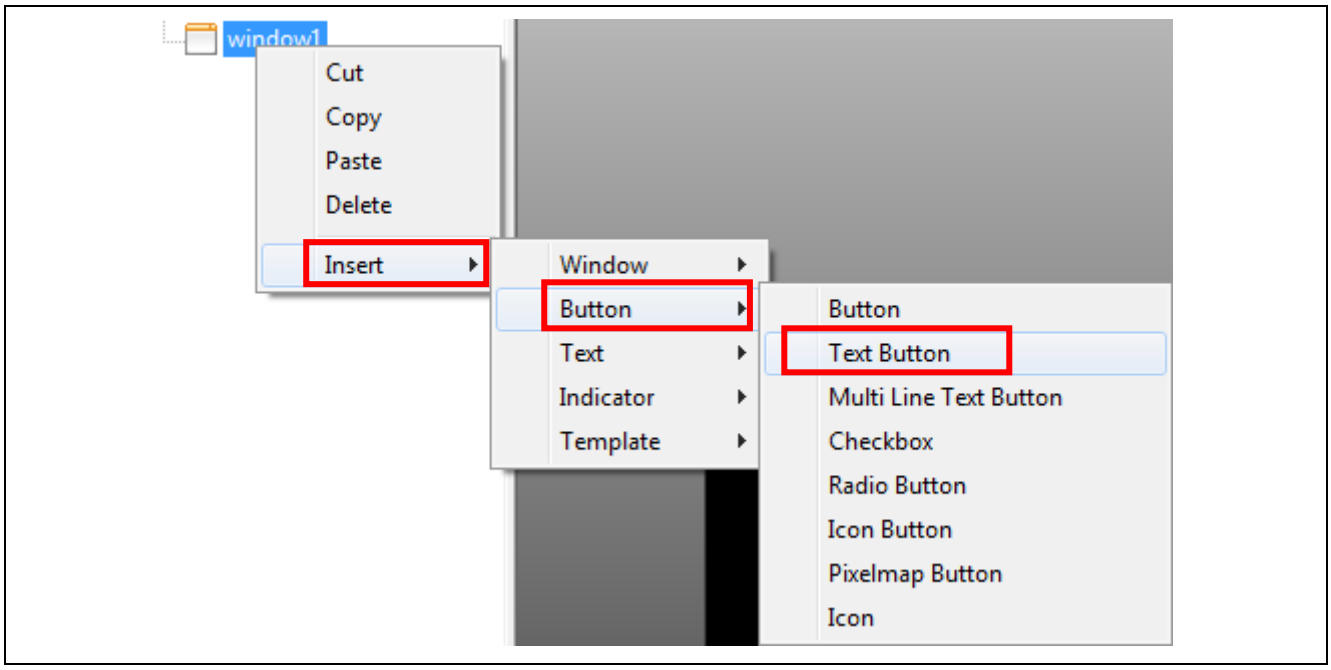


Figure 57. Add a New Text Button

31. In the **Project View**, right-click **window1** and insert a **Button**, **Checkbox** by selecting **Insert** > **Button** > **Checkbox**.

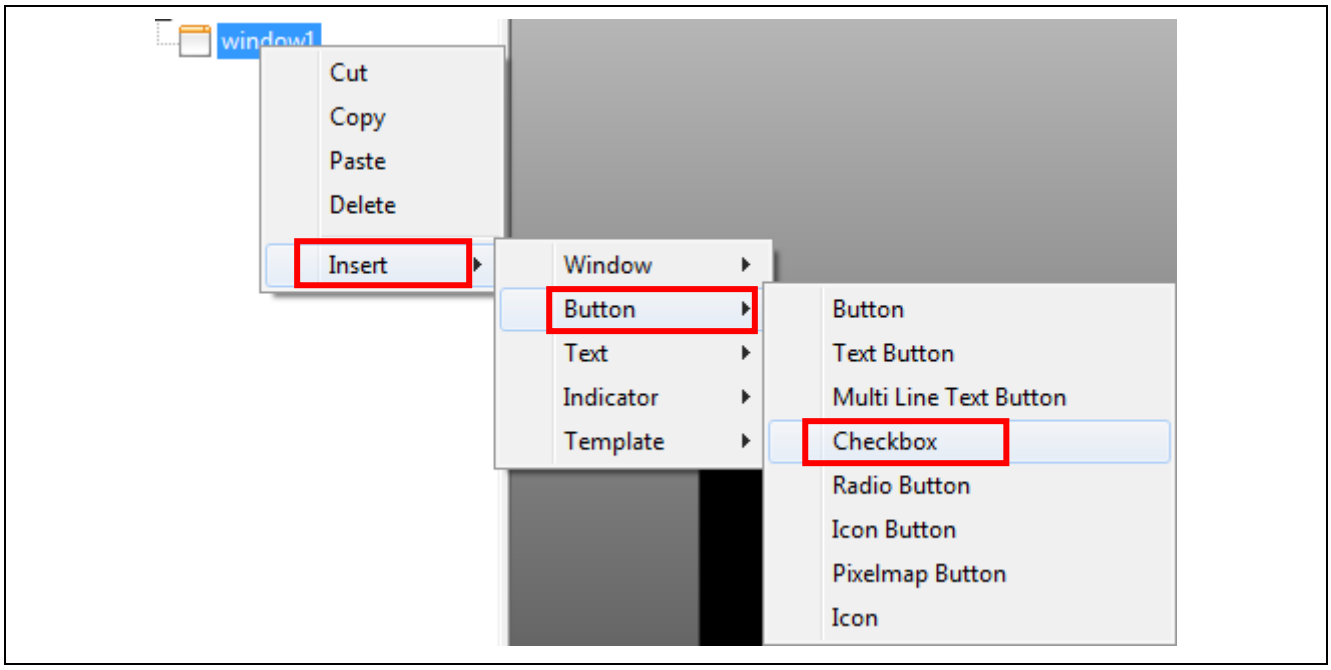


Figure 58. Add a New Checkbox

32. In the **Project View**, right-click **window1** and insert a **Text**, then **Prompt** by selecting **Insert** > **Text** > **Prompt**.

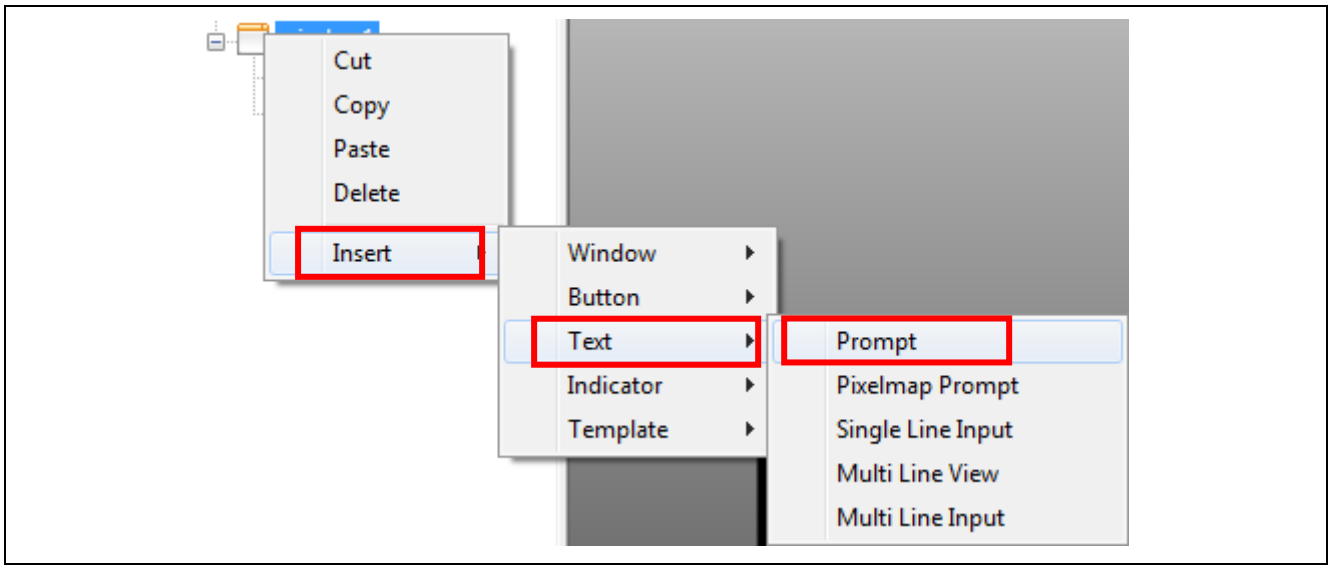


Figure 59. Adding New Prompt

33. In the **Project View**, right-click **window1** and **Insert** another **Text Prompt**.
34. In the Project View, right-click **window2** and **Insert** another **Text Prompt**.
35. In the Project View, right-click **window2** and **Insert** another **Text Prompt**.
36. If you have followed these directions correctly, your Project View should look like the following screen:

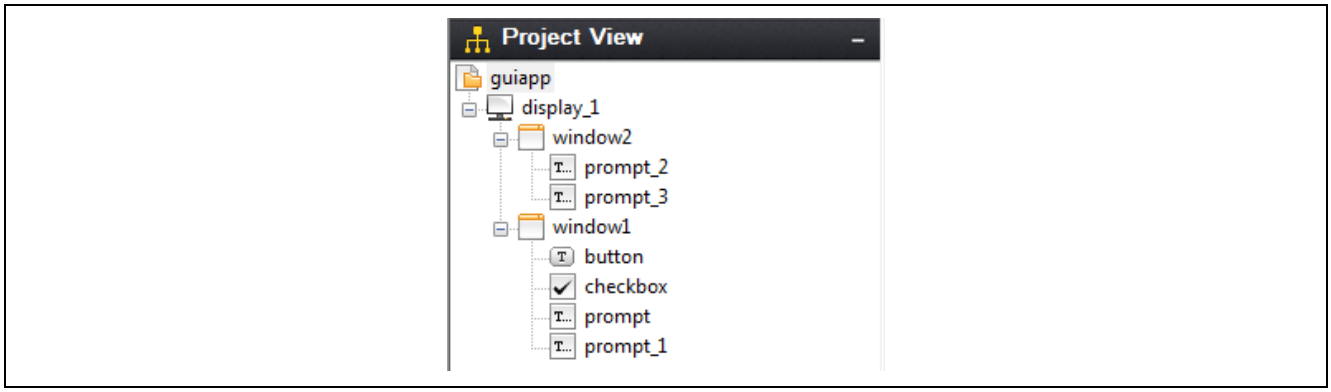


Figure 60. GUIX Project View

37. Press the + character on right of </> Strings to expand the Strings menu.

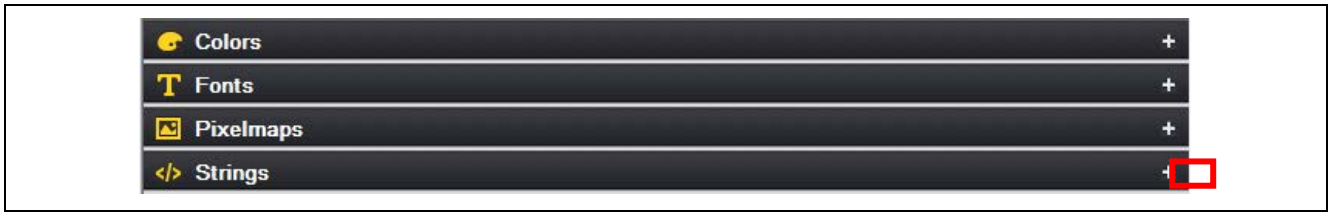


Figure 61. Strings Button

38. Double-click on any of the strings to open the **String Table Editor**.
39. Delete the existing strings by selecting them, then click the **Delete String** button in the **String Table Editor**.
40. Add the following **Strings** using the **Add String** button.

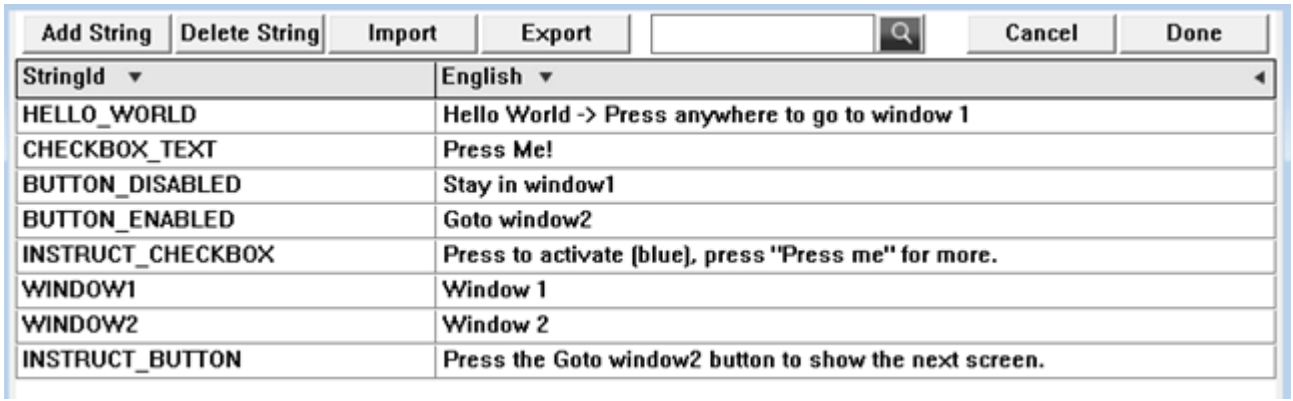


Figure 62. New Strings

41. When correct, click **Save**.
42. In the **Project View** under **window1**, click the button and then modify the properties in the Properties View to match the following.

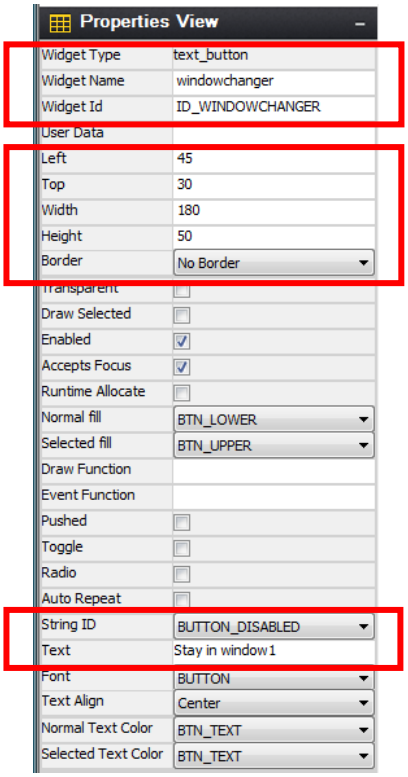


Figure 63. Configure the windowchanger Button properties

43. In the **Project View** under **window1**, click the checkbox and then modify the properties in the **Properties View** to match the following screen.

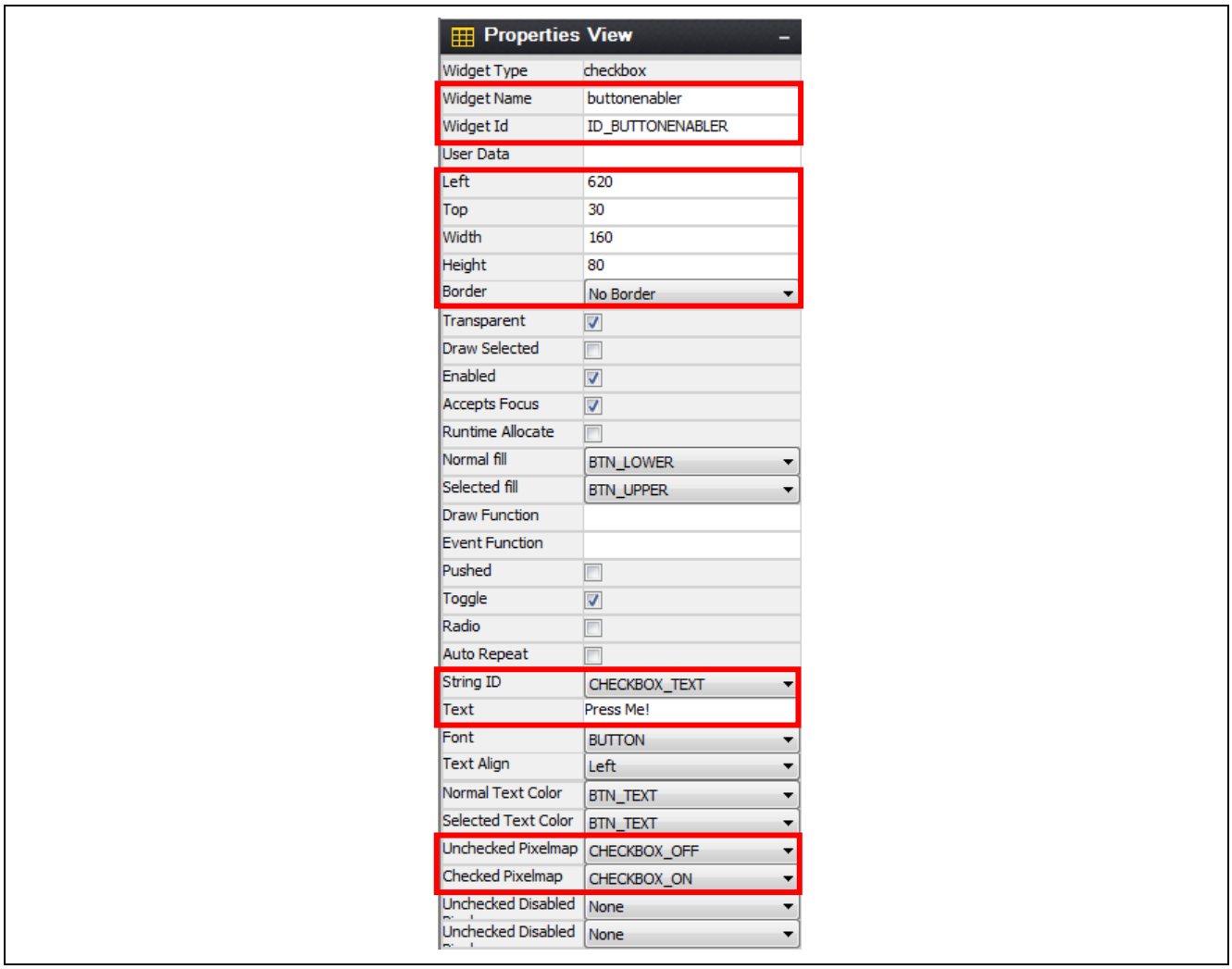


Figure 64. Configure Buttonenabler checkbox properties

44. In the **Project View** under **window1**, click **Prompt** and then modify the properties to match the following screen.

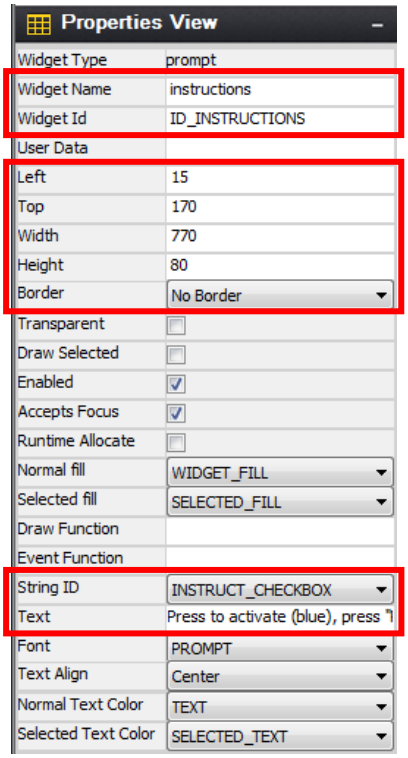


Figure 65. Configure Prompt properties

45. In the **Project View** under **window1**, click **prompt_1** then modify the properties to match the following screen.

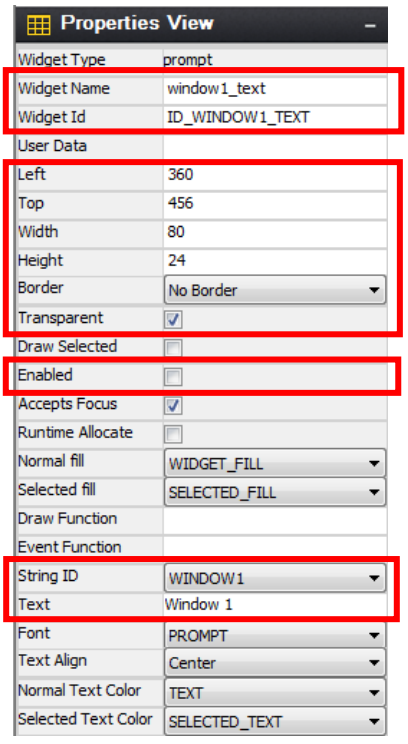


Figure 66. Configure Window Text properties

46. In the **Project View** under **window2**, click **prompt_2** and then modify the properties to match the following screen.

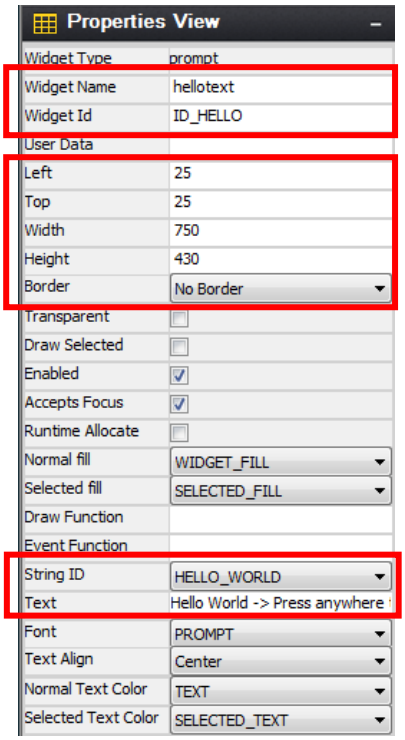


Figure 67. Configure Hello Text Prompt properties

47. In the **Project View** under **window2**, click **prompt_3** and then modify the properties to match the following screen.

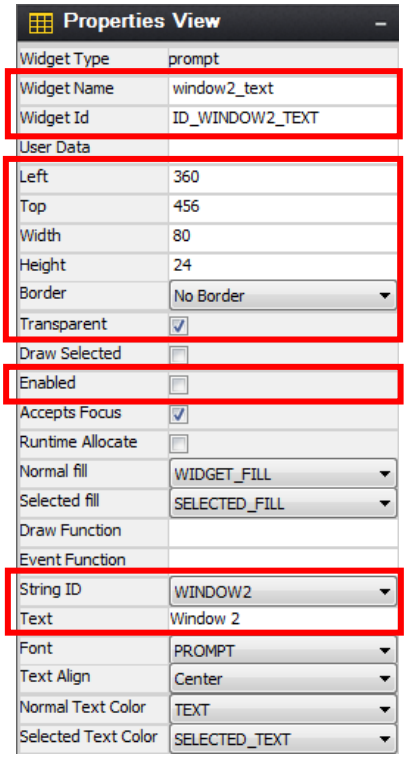


Figure 68. Configure Window Text properties

After these configuration steps, the two windows should now look similar to the following images:

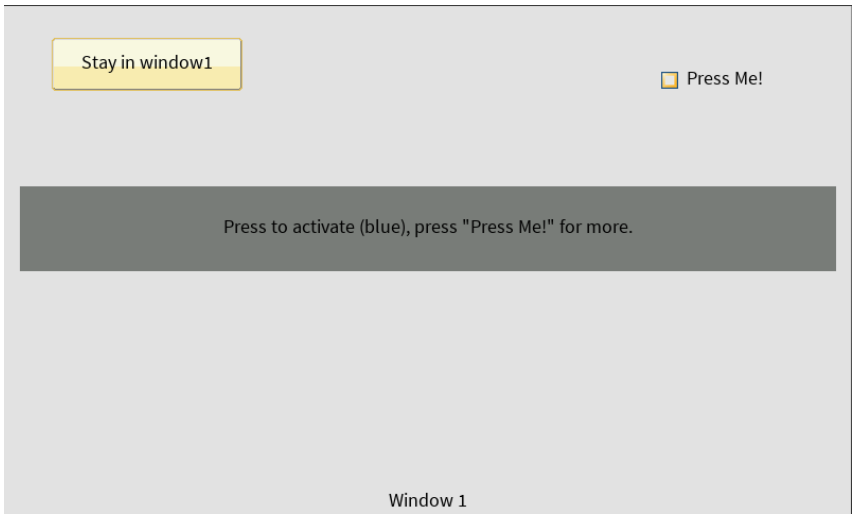


Figure 69. Configured Window1

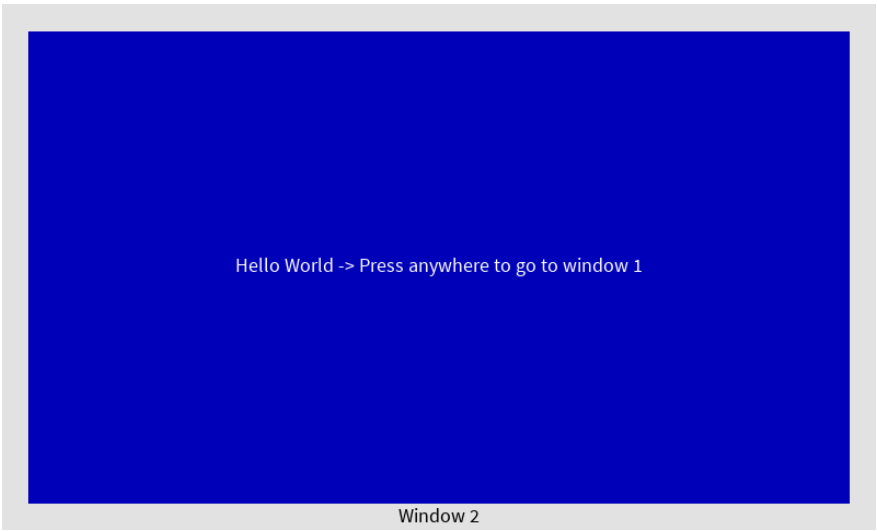


Figure 70. Configured Window2

48. Expand the Pixelmaps section on the right by clicking the **+**.
49. Click **System**.

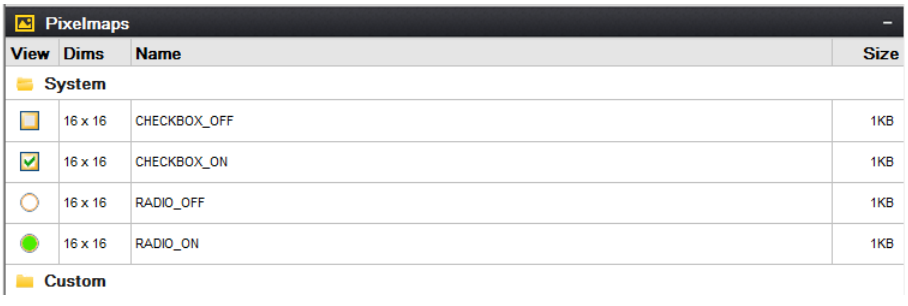


Figure 71. Configuration of Pixelmaps

50. Double-click **CHECKBOX_OFF** to edit the Pixelmap.
51. Deselect **Compress Output** and click **Save**.
52. Double-click **CHECKBOX_ON** to edit the Pixelmap.
53. Deselect **Compress Output** and click **Save**.

54. Save the project.

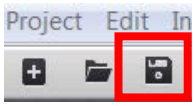


Figure 72. Save Project

55. From the pulldown menu, select **Project** > **Generate All Output Files**.

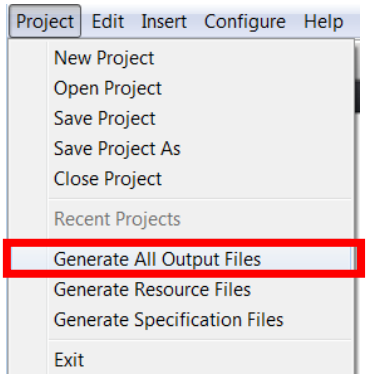


Figure 73. Generate All Output files

56. Click on **Generate**.

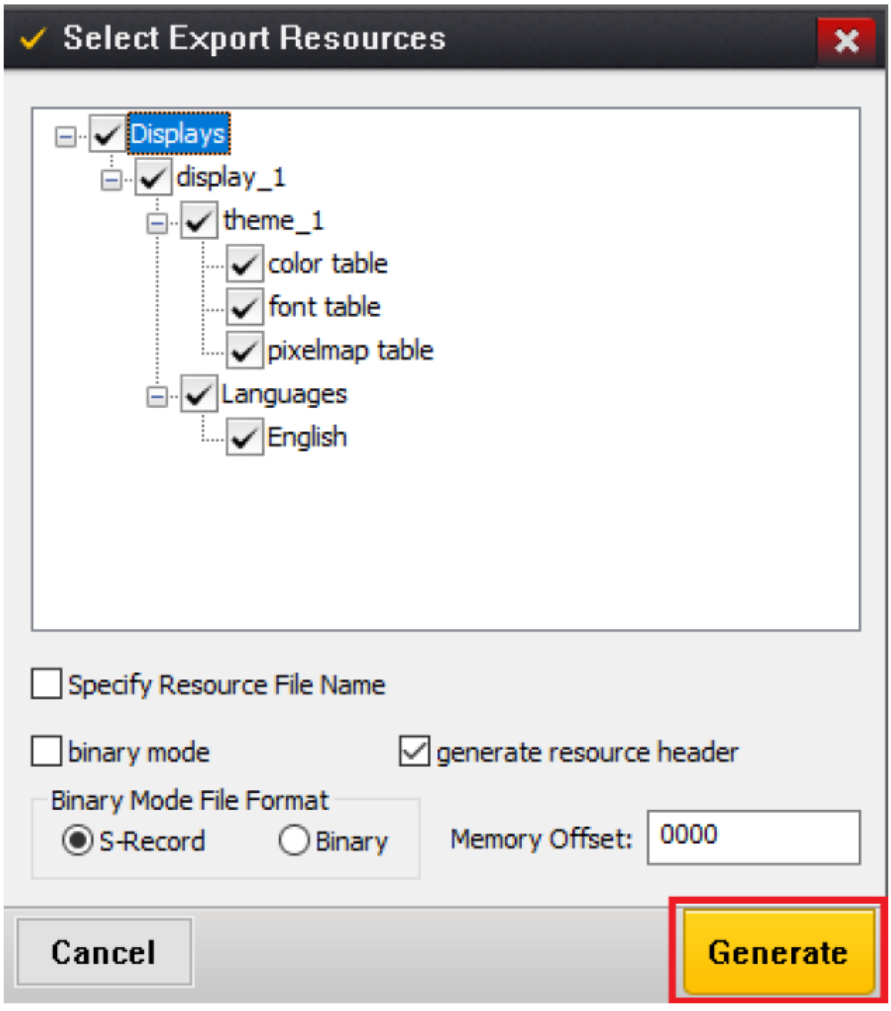


Figure 74. Select Export Resources

57. Return to e² studio.

6. Adding code for custom interface controls and building the project

1. Open **Windows Explorer** and navigate to where you put the files included with this application note. Locate the file `Source Files\guiapp_event_handlers.c`. Now drag the file from the Windows Explorer Window into the **src** folder inside the e² studio **Project Explorer** window.
2. When asked how to import the selected files, click **OK** to copy the files.

Note: This file contains the event management functions for the different graphical elements created in GUIX Studio (window1, window2).

GUIX handles the events that are required at a system level, but to handle custom commands like screen transitions and button actions, event handlers need to be defined. Shown below is the event handler for window1.

```
UINT window1_handler(GX_WINDOW *widget, GX_EVENT *event_ptr)
{
    UINT result = gx_window_event_process(widget, event_ptr);

    switch (event_ptr->gx_event_type)
    {
    case GX_SIGNAL(ID_BUTTONENABLER, GX_EVENT_TOGGLE_ON):
        button_enabled = true;
        update_text_id(widget->gx_widget_parent, ID_WINDOWCHANGER, GX_STRING_ID_BUTTON_ENABLED);
        update_text_id(widget->gx_widget_parent, ID_INSTRUCTIONS, GX_STRING_ID_INSTRUCT_BUTTON);
        break;
    case GX_SIGNAL(ID_BUTTONENABLER, GX_EVENT_TOGGLE_OFF):
        button_enabled = false;
        update_text_id(widget->gx_widget_parent, ID_WINDOWCHANGER, GX_STRING_ID_BUTTON_DISABLED);
        update_text_id(widget->gx_widget_parent, ID_INSTRUCTIONS, GX_STRING_ID_INSTRUCT_CHECKBOX);
        break;
    case GX_SIGNAL(ID_WINDOWCHANGER, GX_EVENT_CLICKED):
        if(button_enabled){
            show_window((GX_WINDOW*)&window2, (GX_WIDGET*)widget, true);
        }
        break;
    default:
        gx_window_event_process(widget, event_ptr);
        break;
    }

    return result;
}
```

Events can be routed based on the ID of the widget and the signal from GUIX. For example, the **checkbox ID_BUTTONENABLER** can have two states; **GX_EVENT_TOGGLE_ON** and **GX_EVENTS_TOGGLE_OFF**. When the box is unchecked and then pressed, the event **GX_EVENT_TOGGLE_ON** is sent to the handler, after the box will be checked.

3. Build the project by clicking the **Hammer** icon below the menu bar. If all steps were followed correctly, there should be no errors reported in the build output.

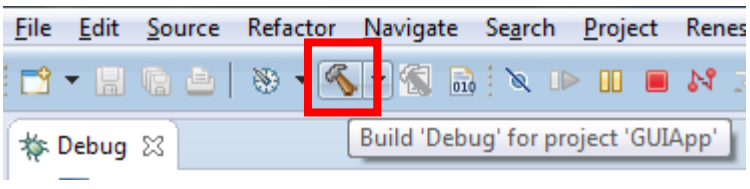


Figure 75. Build Button

7. Running the application

1. Power the PE-HMI1 and connect the J-Link Lite Cortex M debugger to the PC and PE-HMI1.
   Note: The application is not yet ready to be run on the target hardware. The following steps are necessary to run it.
2. Click the drop-down menu for the **debug** icon.

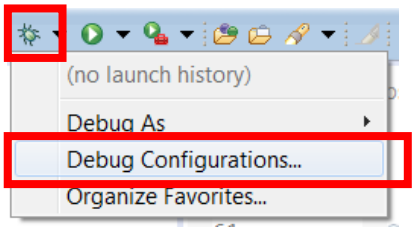


Figure 76. Debug Options

3. Select the **Debug Configurations…** option
4. Under the **Renesas GDB Hardware Debugging** section, select **GUIApp Debug**.
5. Click on the **Debug** button to start debugging.
   Note: If the **Debug** button is greyed out, then there is likely to be an issue with the build. Check all steps for mismatched options.

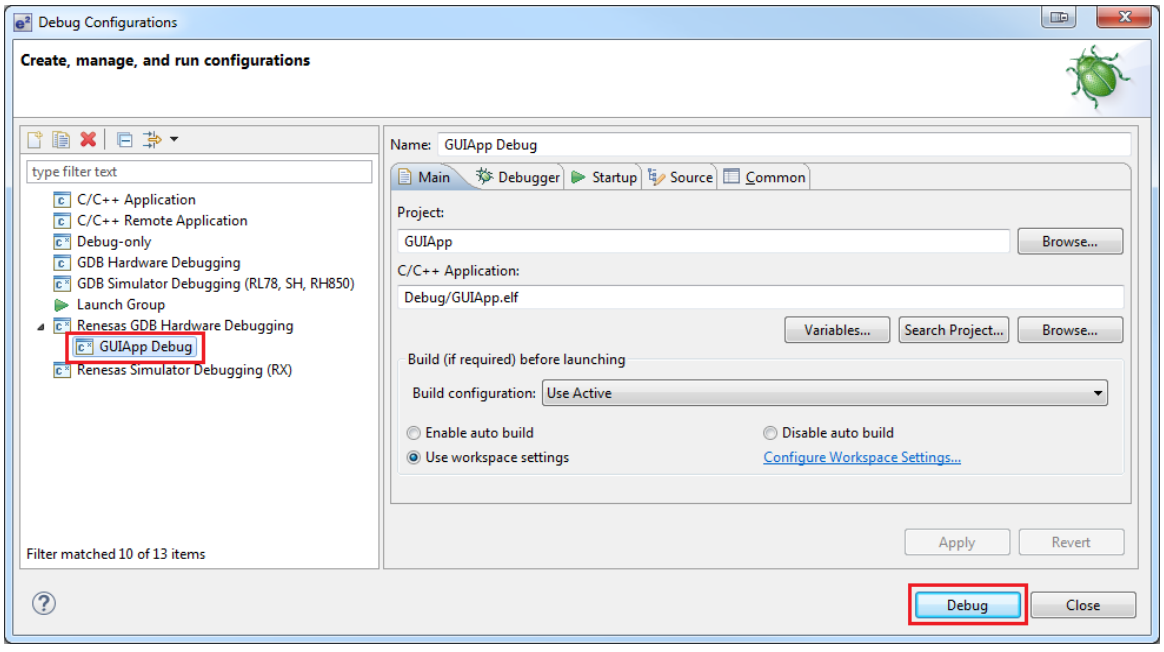


Figure 77. Debug Configurations

6. If asked to confirm a Perspective Switch, click **Yes**. (If you have previously instructed e² studio to remember your decision, this dialog box will not be displayed.)

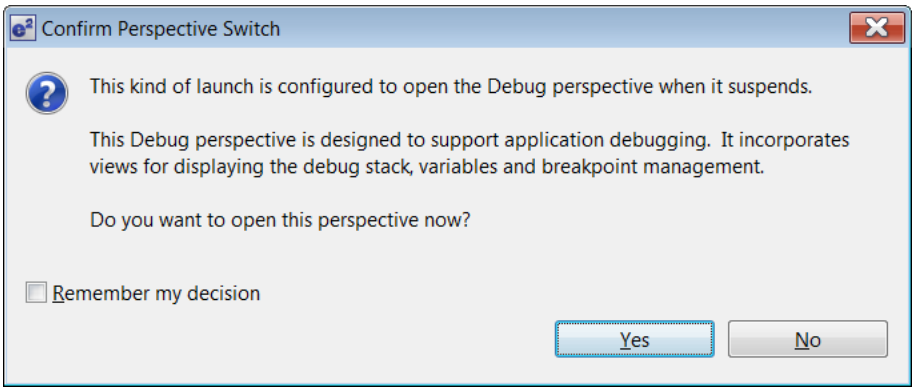


Figure 78. Perspective Switch Dialog

7. Press **F8** or the **Resume** button to start the application. It will stop at `main`.

Figure 79. Resume Button

8. Press **F8** or the **Resume** button to run the code.
   Note: The GUI created earlier should display on the screen.
9. Overview of the demo:

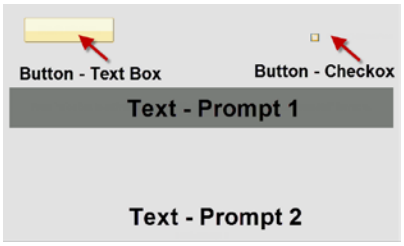


Figure 80. Window1

A. The figure above shows **Window1**. In this window are four elements:

- **Button – Checkbox**: Use this button to enable navigating to **Window2**. Text is set to **Press Me!** and it is unchecked. When the user presses within the **Checkbox** active area, the event **window1_handler** is activated. This event is picked up inside `guiapp_event_handlers.c` where the code toggles the checkbox then sets the text in **Text –Prompt 1** and **Button – Text Box** to the appropriate message.
- **Button – Text Box**: This box simply shows what window you will go to if you press outside the **Text –Prompt 1** area. (See **Button – Checkbox** to see how it is changed.) Press in this area to activate the **window1_handler** event which is picked up by `guiapp_event_handlers.c`, where the code changes the window to **window2**.
- **Text – Prompt 1**: This area instructs the user how to control the demo. (See **Button – Checkbox** for how it is changed.)
- **Text – Prompt 2:** This prompt is used to show the user which window they are in. It never changes (always shows **window1**).

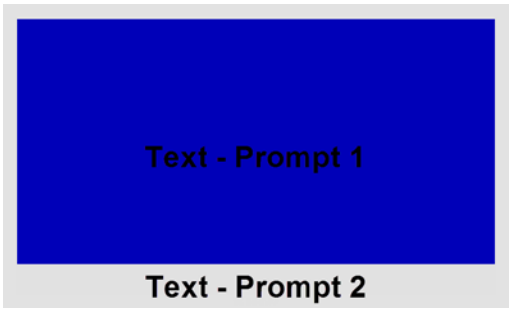


Figure 81. Window2

B. The preceding figure shows **window2**. In this window are two elements:

- **Text – Prompt 1**: This area presents **Hello World** and instructs the user how to return to **window1**. Pressing in this area initiates the **window2_handler** event which is picked up by `guiapp_event_handlers.c` and changes the active window to **window1**.
- **Text – Prompt 2:** This Prompt is used to show the user which window they are in. It never changes (always shows **window2**).

10. Press **Ctrl** + **F2** or the stop button to end the debug session.

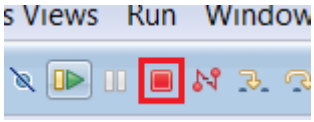


Figure 82. Stop Button

11. This concludes the GUIX **Hello World** for PE-HMI1.

8. Appendix

The GUIX image resources files are default stored in the internal code flash. The resource files can also be stored in the external flash such as QSPI. Refer the Knowledgebase link (https://en-support.renesas.com/knowledgeBase/18054800) to know more about using QSPI for storing the image resource files.

Website and Support

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

Synergy Platform MCUs	www.renesas.com/renesas-synergy-platform-mcus
Synergy Software Package	www.renesas.com/synergy/ssp
Software add-ons	www.renesas.com/synergy/addons
SSP Components	www.renesas.com/synergy/sspcomponents
MCU Components	www.renesas.com/synergy/components-synergy-mcus
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:	
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	Jan.22.16	All	Created Initial Document
1.10	Apr.13.16	All	Updated to SSP 1.1.0
1.11	Nov.18.16	Title	Minor formatting changes
1.12	Jan.09.17	All	Updated to SSP 1.2.0.b.1
1.13	Mar.03.17	All	Updated to SSP 1.2.0
1.14	Sep.13.17	All	Updated to SSP 1.3.0
1.15	Feb.28.18	All	Updated to SSP v1.4.0
1.16	Jun.18.18	—	Sample codes updated.
1.17	Sep.07.18	—	Updated to SSP v1.5.0
1.18	Mar.08.19	—	Updated to SSP v1.6.0
1.19	Apr.02.19	—	Updated package to include necessary files. There are no changes to the content of this document.
1.20	Aug.11.21	—	Updated for SSP v1.6.0 “Touch Panel V2 Framework”
1.21	Oct.21.21	—	Updated to SSP v2.1.0
1.22	Apr.21.23	—	Deleted SSP licensing section and messaging framework references

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/.