Renesas Synergy™ Platform

GUIX "Hello World" for DK-S7G2

Introduction

This application note guides you through the process of creating a simple two-screen GUI using GUIX Studio for the DK-S7G2 Synergy MCU kit. Its application project demonstrates how easily you 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 under e² studio using the Synergy Framework.

Target Device

DK-S7G2 board version 3.1 and 4.1

Minimum PC Recommendation

- Microsoft® Windows® 7/8/10
- Intel® Core™ family processor running at 2.0 GHz or higher (or equivalent processor)
- 8 GB memory
- 250 GB hard disk or SSD
- USB 2.0
- Connection to the Internet

Installed Software

- Synergy™ e² 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 it before continuing.

Provided Software Files

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

Purpose

To guide you through the setup of a GUIX touch screen interface for the **Hello World** application in e² studio, where you configure the drivers and framework included with the SSP. Project setup in e² studio includes setup of basic debugging operations. When you have the configuration ready, you can set up the LCD Controller, touch screen drivers, and messaging framework to communicate with application tasks. You can also create a simple GUI interface using the GUIX Studio editor. Once the application is running, it responds to touchscreen actions using the Touch Panel V2 Framework on sf_touch_panel_v2 Framework, presenting a basic graphical user interface (GUI).

Intended Audience

The intended audience are users who want to design GUI applications.

Contents

1. Overview

This application note shows how to set up 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 allow you to skip the development steps and go to the point of verifying a working project on the DK-S7G2. Most users SKIP THIS STEP and continue to section 3 to create a project in e² studio. If you do choose to import the project, go to section 7, Running the Application.

To skip the development walkthrough steps 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 the package. It contains instructions on importing the project into e² studio and building the project. The included `GUIX_Hello_World_DK-S7G2.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 e² studio** folder.
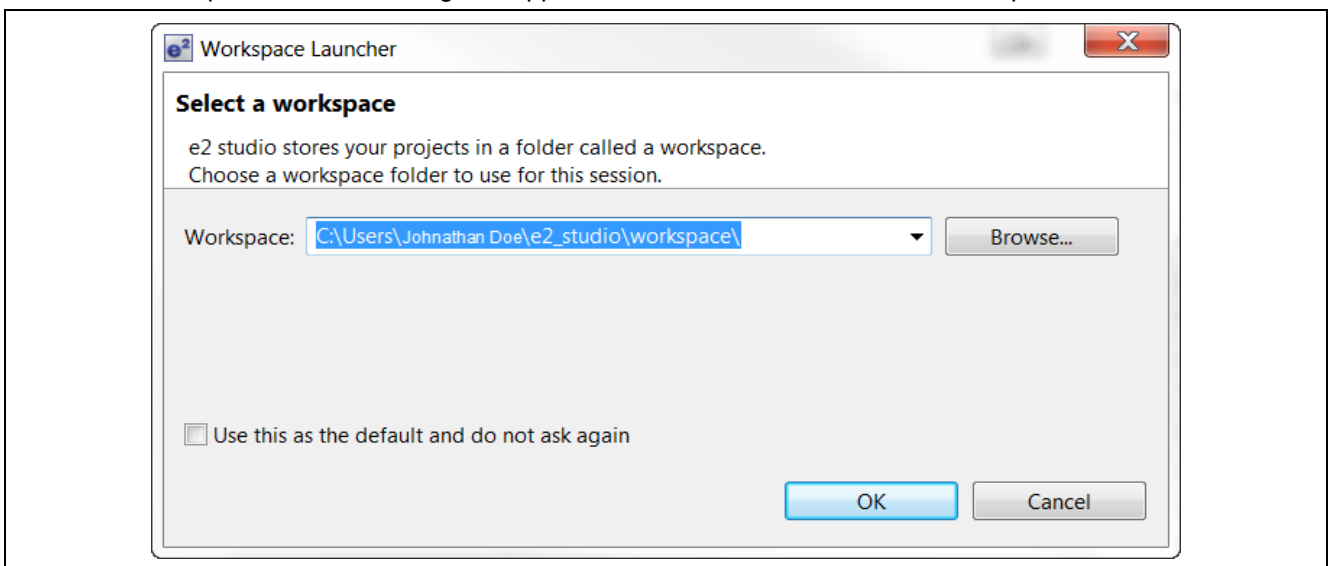2. If the workspace launcher dialog box appears, click **OK** to use the default workspace.


Figure 1. Workspace Launcher Dialog

3. Create a new workspace:
   From the **File** pulldown 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\[your name]\e2_studio\workspace\GUI_APP**.
5. Click **OK** to create the new workspace.

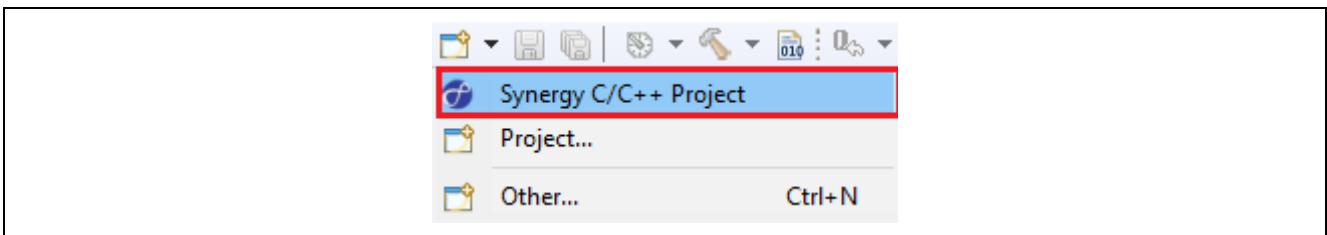6. Click in the Workbench area to proceed past the Welcome Screen.
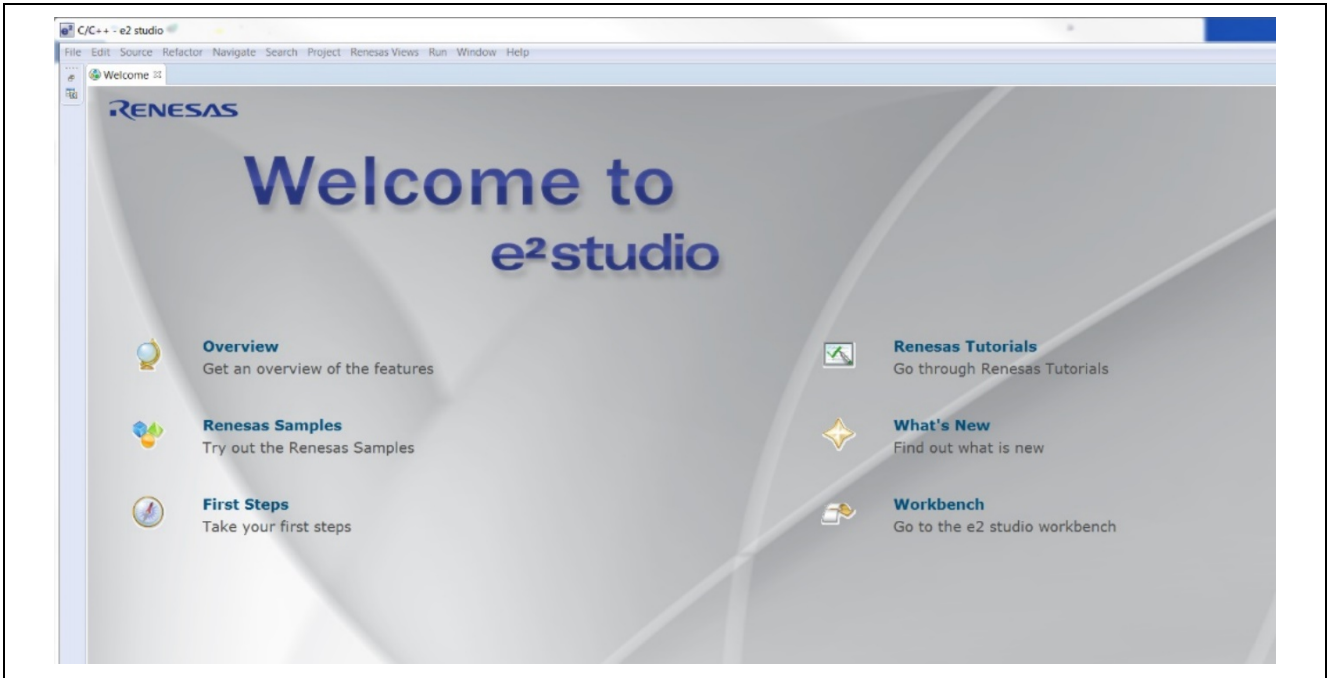
Figure 2. Close the Welcome Window by Clicking in the Workbench Area

7. Start a new project by clicking the dropdown menu ▼ next to the **New** icon in the Tool Bar.
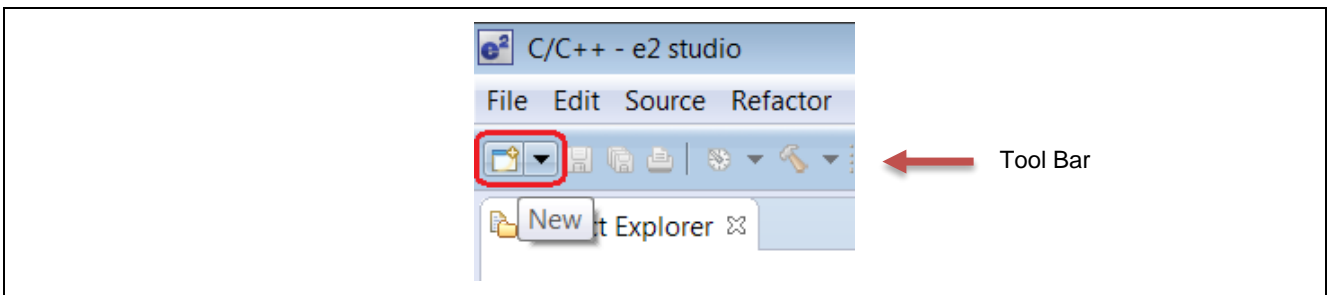
Figure 3. Start a New Project

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

Figure 4. Select Synergy C/C++ Project in the Dropdown Menu

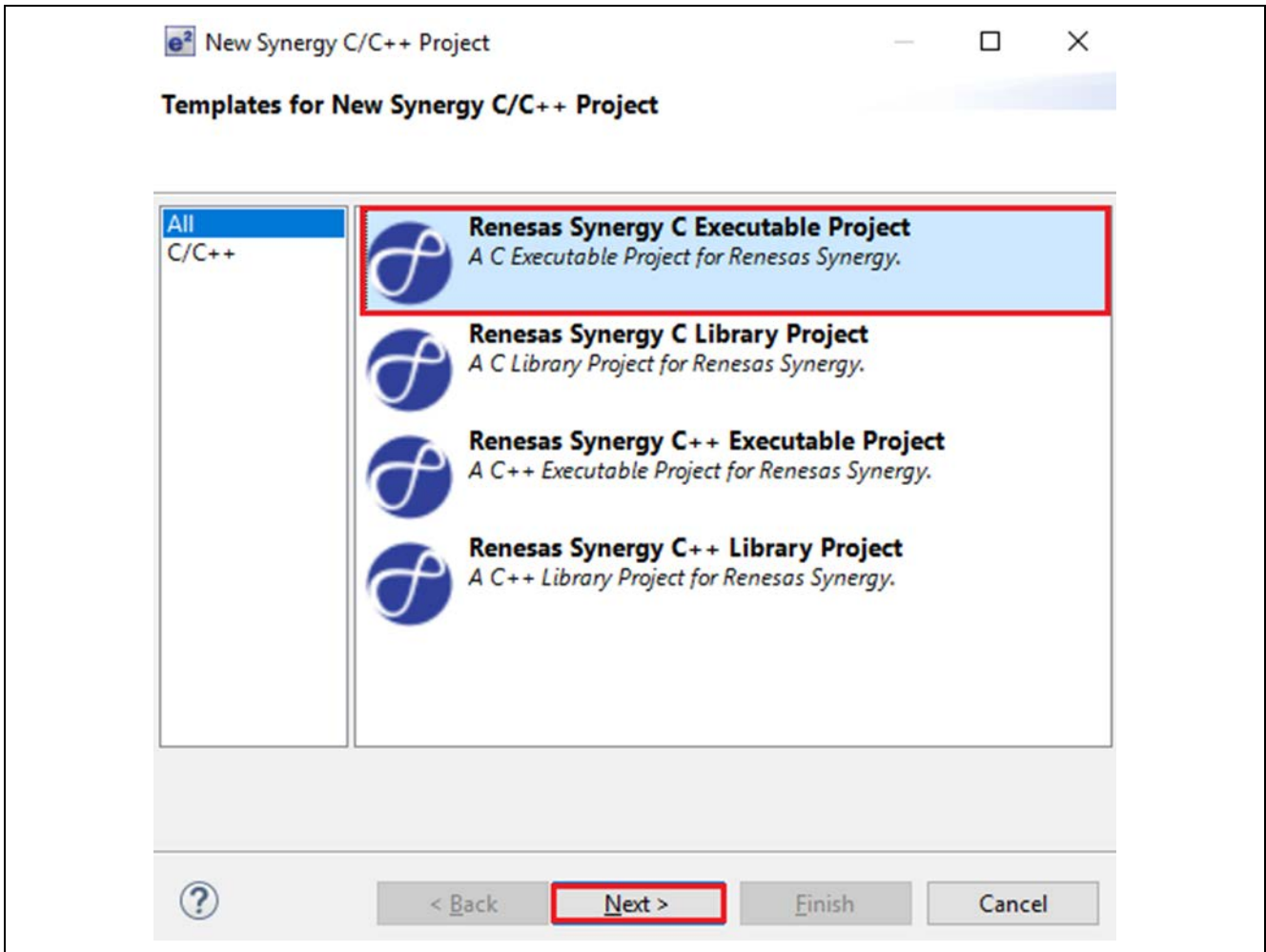9. Select **Renesas Synergy C Executable project**.

Figure 5.   Project Type Selection

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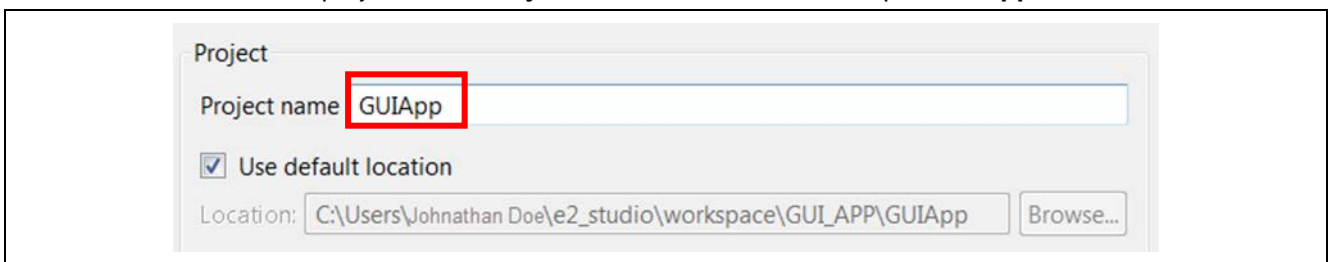

Figure 6.   Enter a Project Name

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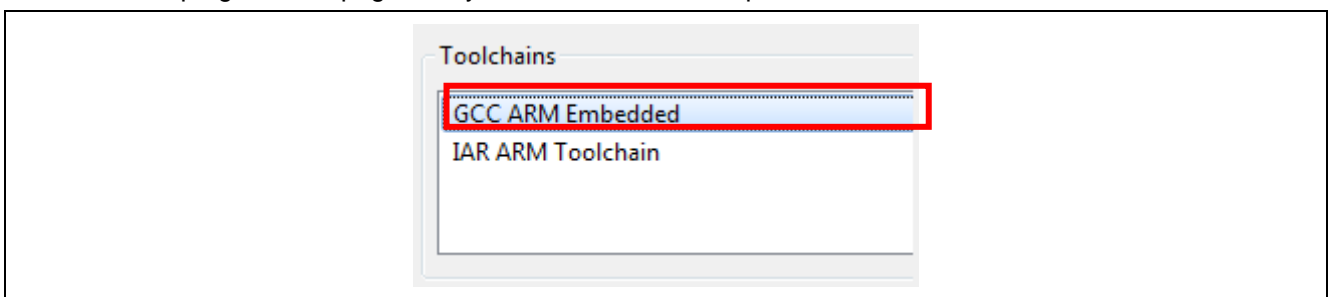

Figure 7.   Verify GCC ARM Embedded Toolchain

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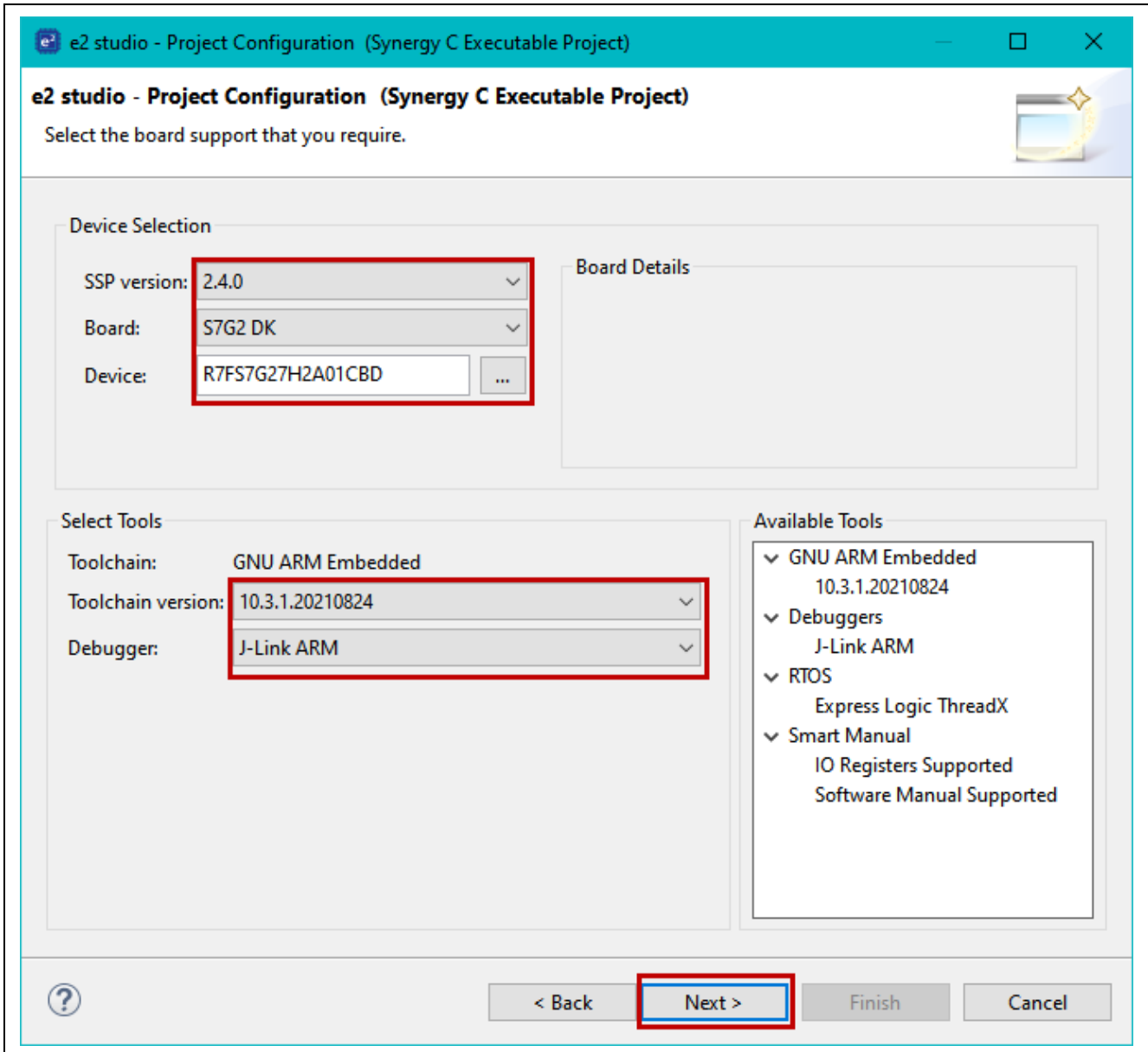

Figure 8.   Device Selection

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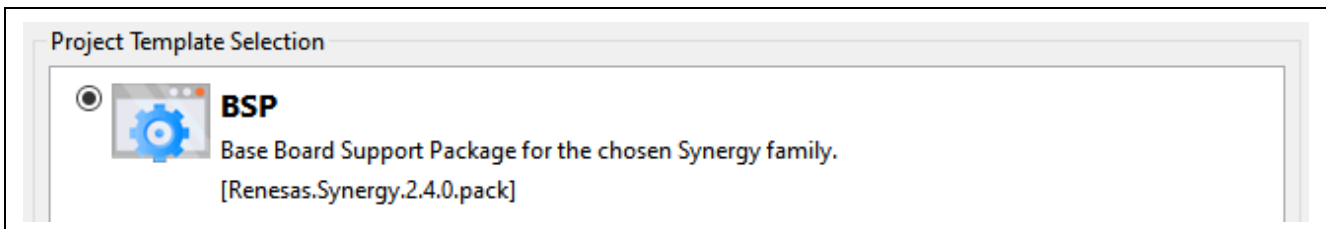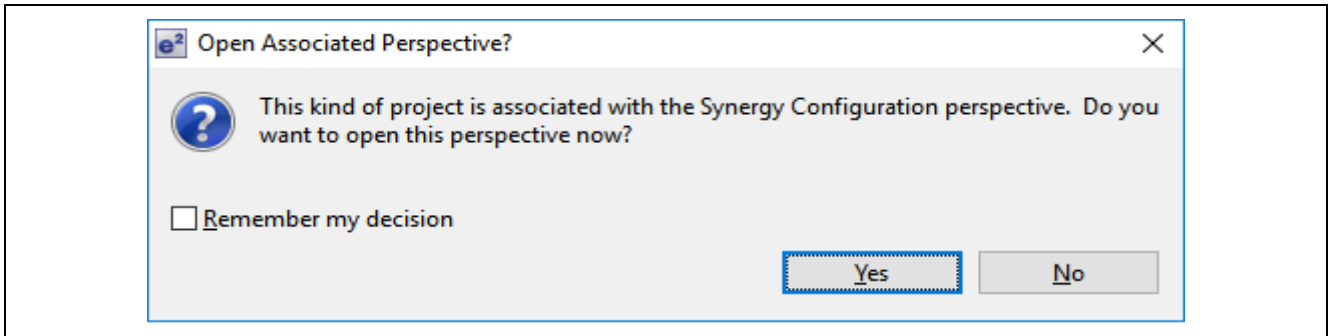16. In the **Project Template Selection**, select the option **BSP**.


Figure 9.   Select the BSP

17. Click the **Finish** button.

18. If you have not previously 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 e² studio has finished creating the project, it displays the following screen.
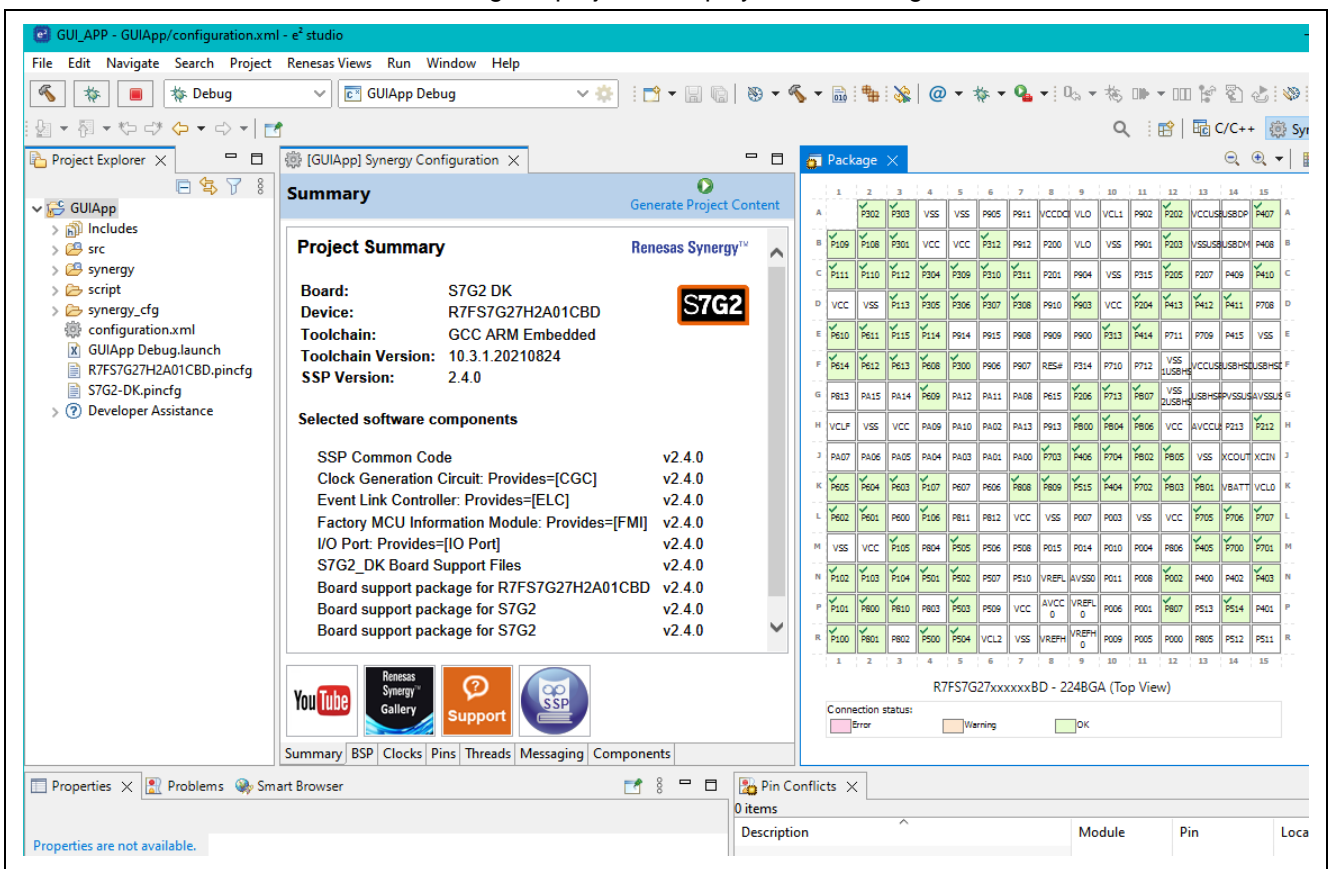
Figure 11.   GUIApp Project

4. Configuring the project in e² studio

Once successfully created in e² studio ISDE, the project can be configured for the GUI application.

1. Open the **Synergy Configuration**, if not already open, by double-clicking the **configuration.xml** file in the **Project Explorer Window**.
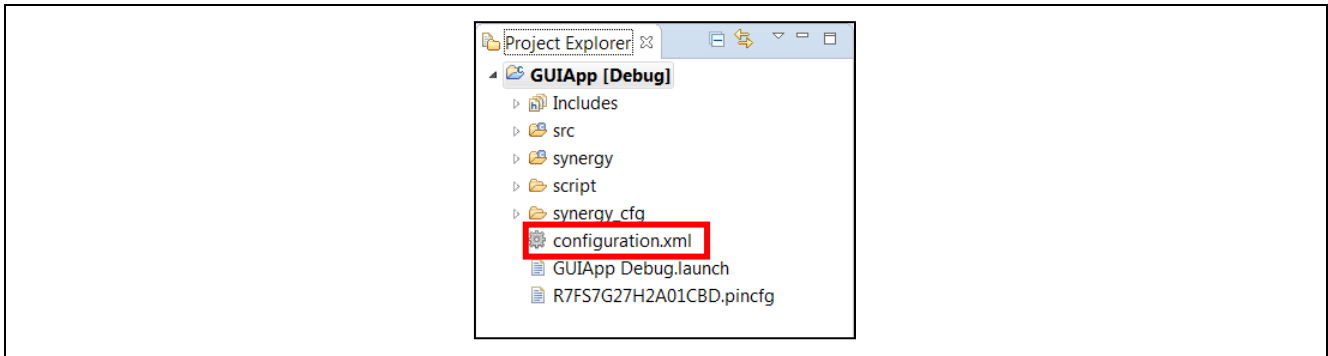

Figure 12. Selecting the Configuration.xml File in Project Explorer

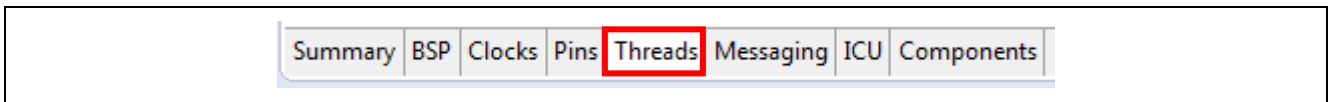2. In the **Synergy Configuration Window**, click the **Threads** tab.

Figure 13. Synergy Configuration Threads Tab
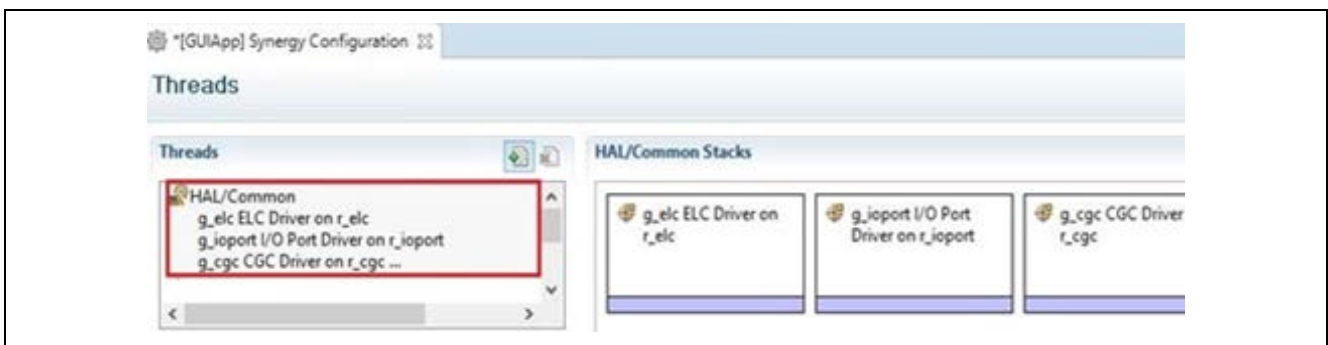
3. Select the **HAL/Common** thread.

Figure 14. Threads

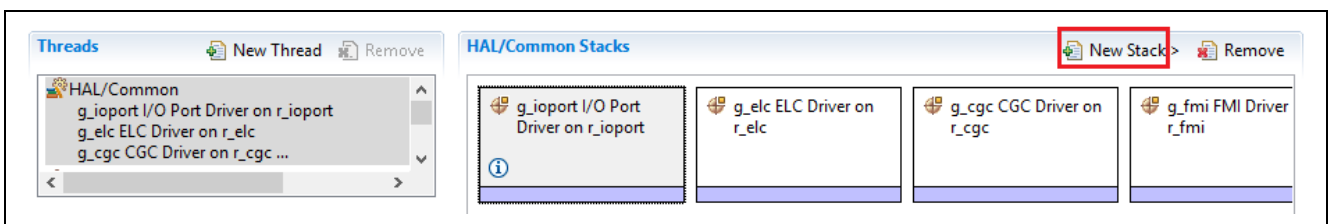4. In the **HAL/Common Stacks** area, click the **New Stack** button.

Figure 15. Add a Timer Driver Module to the HAL/Common Thread Part 1

5. In the **New Stack** menu, select **Driver** > **Timers** > **Timer Driver on r_gpt**.
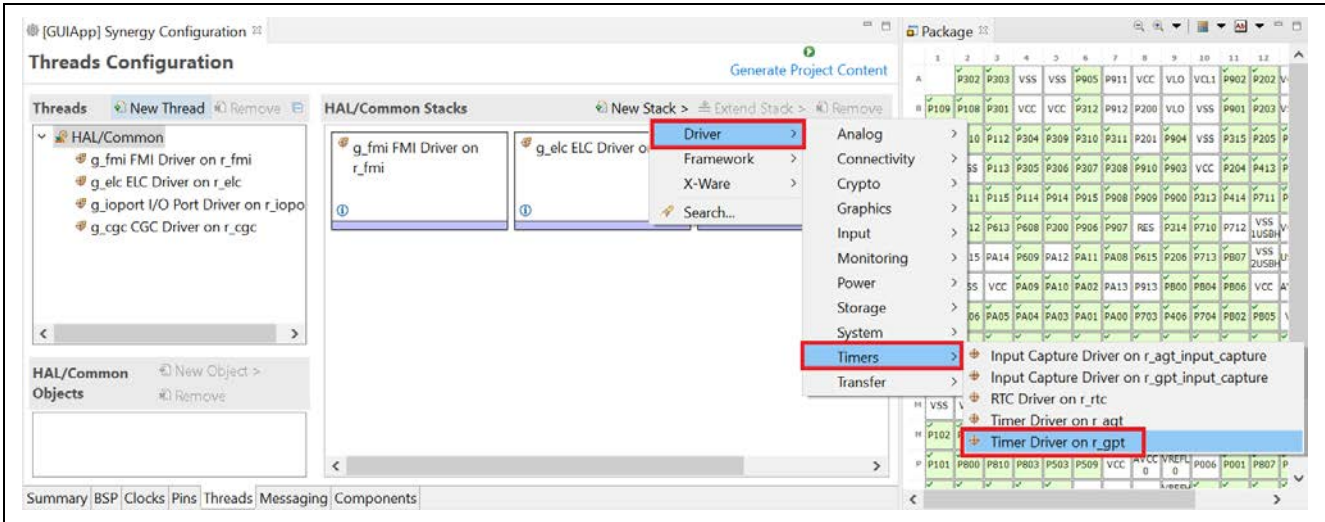

Figure 16.   Add a Timer Driver Module to the HAL/Common Thread Part 2

6. In the **HAL/Common Modules** area, select the newly created module **g_timer0 Timer Driver on r_gpt**.
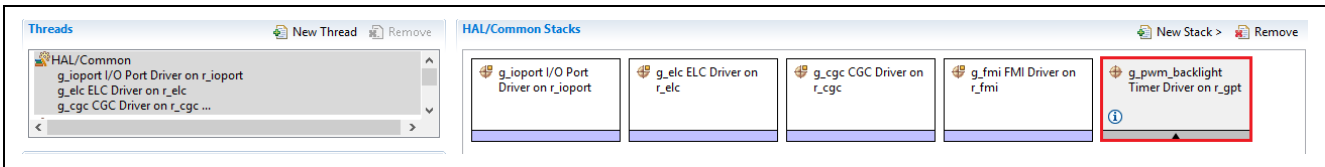

Figure 17.   Select the Newly Created Timer Driver Module

7. In the **Properties Window**, change the **Properties** to match those in Figure 18. Hint: Change the channel to 2!

Figure 18.   Configure the PWM Module

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

The touch screen requires several frameworks and drivers to be used. External interrupts are used to know when to read the data. An I^{2}C driver handles the reads. A framework translates the register data from the peripheral to touch coordinates the software can use.

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

Figure 19.   Creating a New Thread
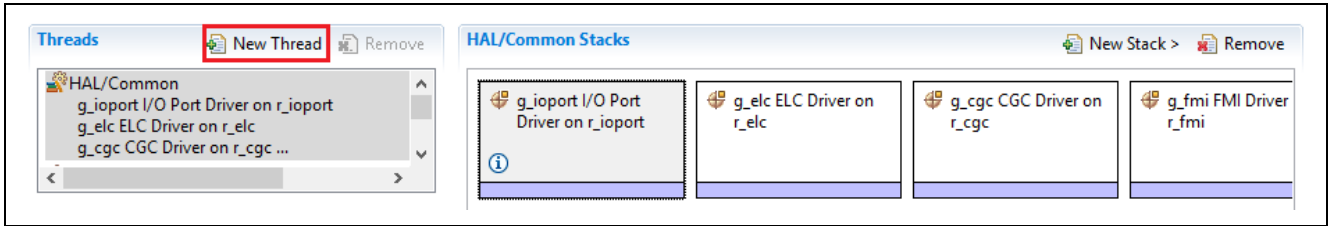
9. Click on **New Thread** to pull up the **Properties** tab.
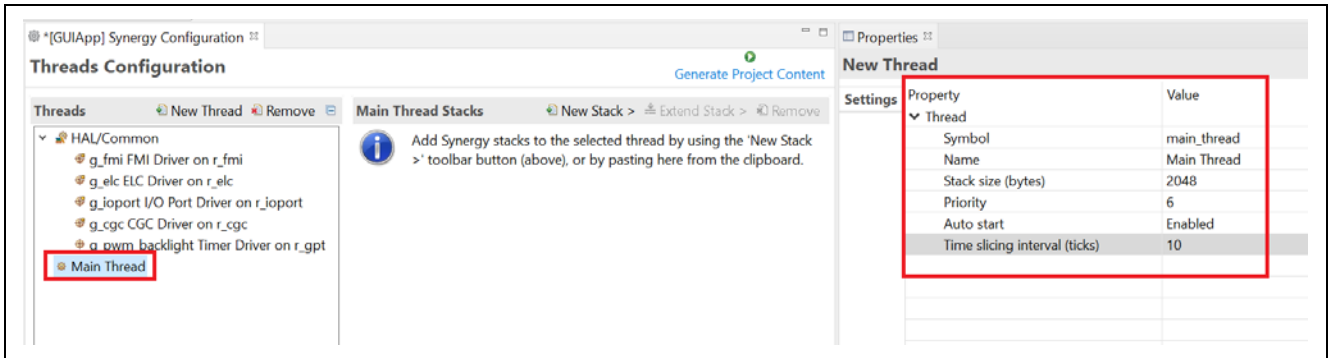10. Edit the **Properties** to match.

Figure 20.   Configuring the Main Thread Properties

11. Back in the S**ynergy Configuration** window > **Threads** tab > **Main Thread Stacks** area, click **New Stack**.
    Note: Be sure **Main Thread** is selected before adding new modules.
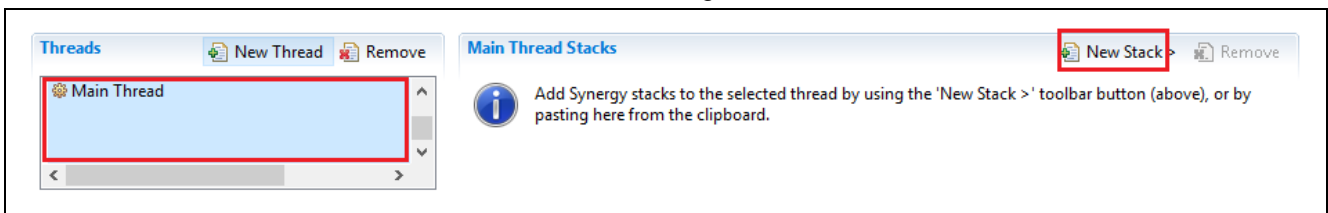
Figure 21   Main Thread Stacks

12. Add a framework for the touch panel by selecting **New Stack > Framework > Input > Touch Panel V2 Framework on sf_touch_panel_v2**.
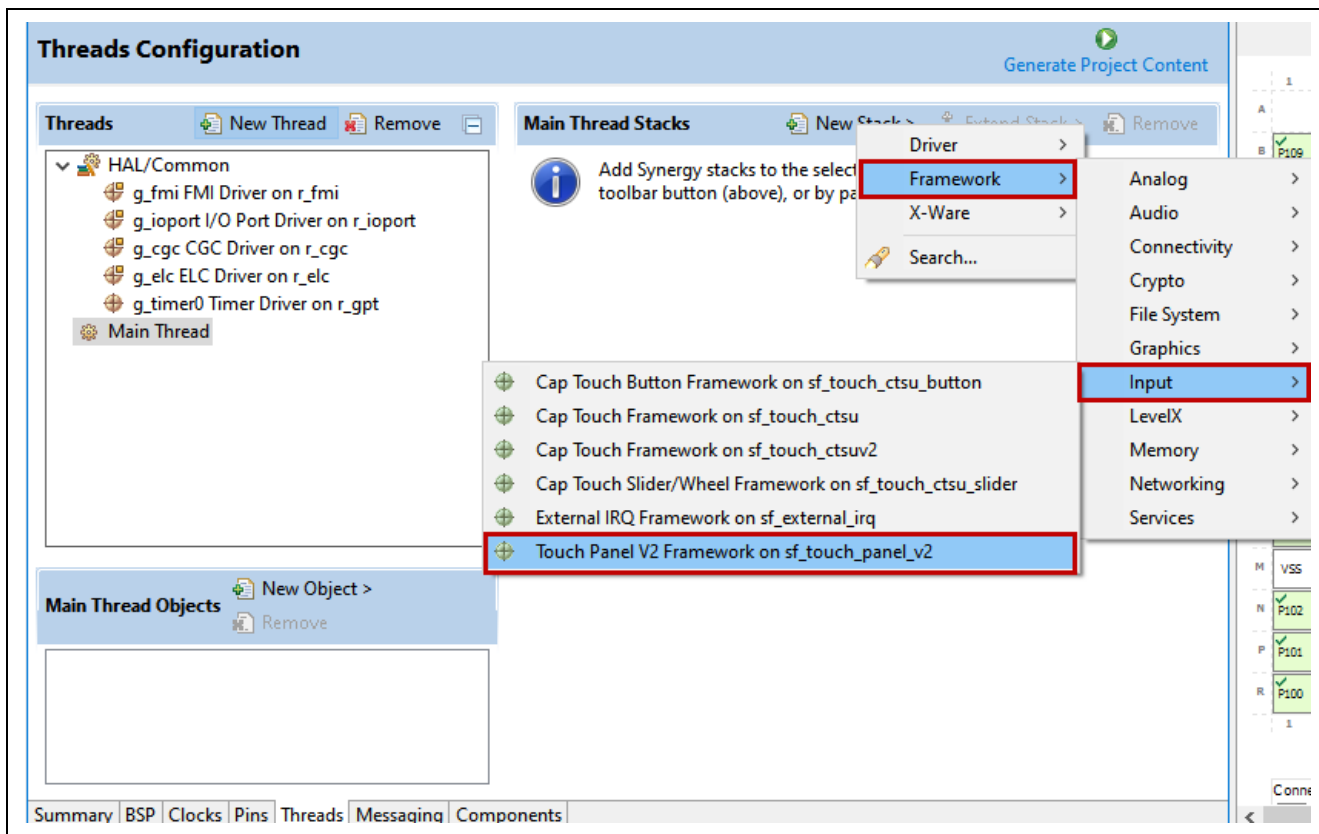
Figure 22.   Add Touch Panel V2 Framework on sf_touch_panel_v2.

13. In the S**ynergy Configuration** window **> Threads** tab **> Main Thread Stacks** area, click on **g_sf_touch_panel Touch Panel V2 Framework sf_touch_panel_v2**. Configure the properties for g_**sf_touch_panel Touch Panel V2 Framework sf_touch_panel_v2**.

Figure 23.   Configure the Touch Panel V2 Framework Properties

14. In the S**ynergy Configuration** window **> Threads** tab **> Main Thread Stacks** area, click on **Add Touch Driver > New > Touch_panel_chip_sx8654**.

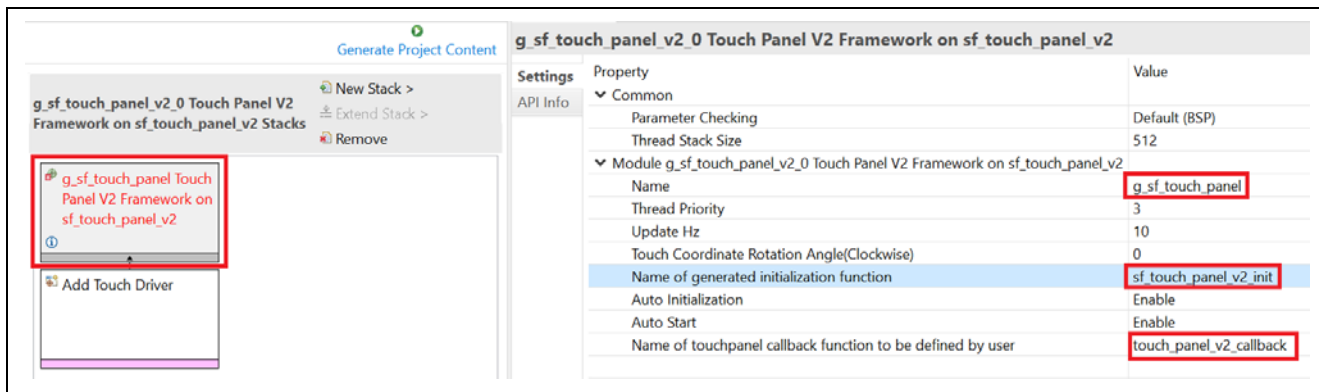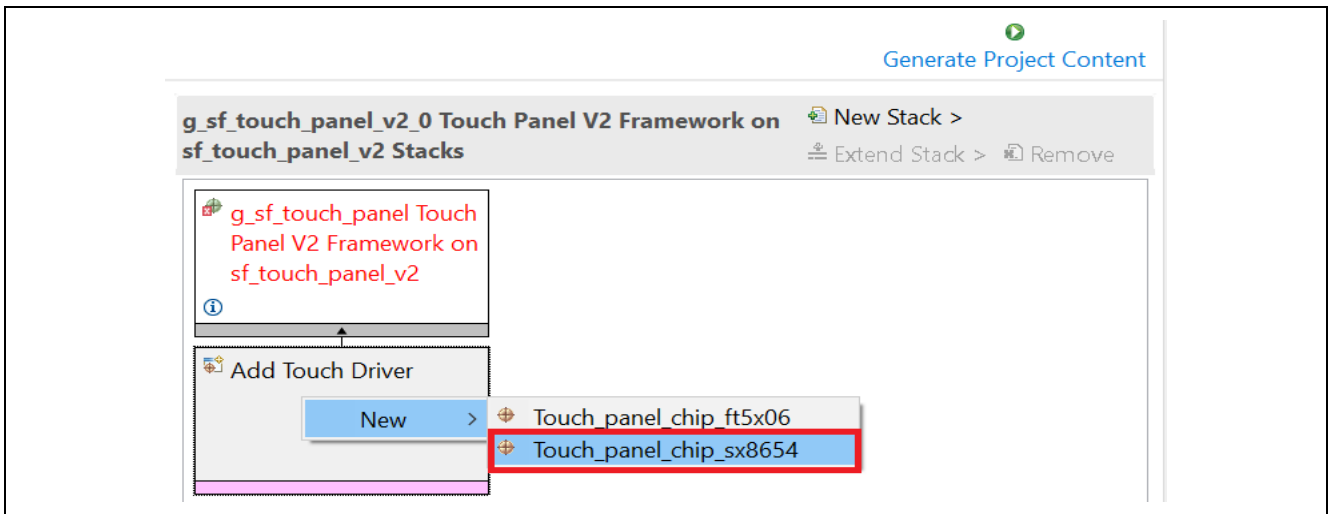**Figure 24.   Add the Touch_panel_chip_sx8654 Touch driver**

15. Configure the **Touch_panel_chip_sx8654** properties as shown.
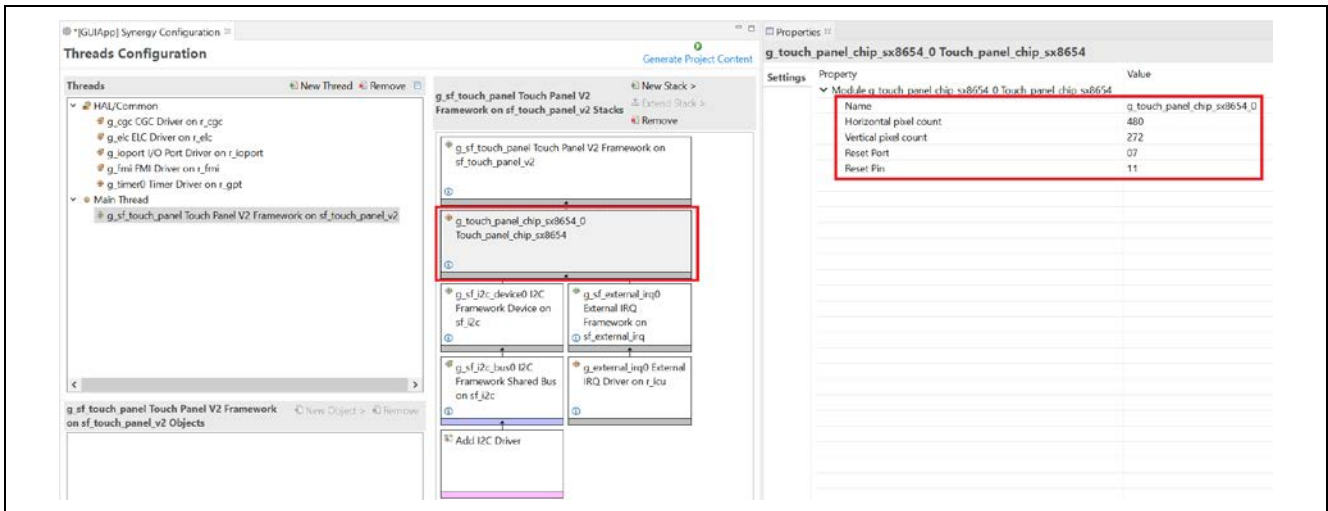
Figure 25.   Configure Touch_panel_chip_sx8654 Properties

The Synergy Configurator has created the external IRQ framework and driver and has a placeholder for the I²C driver and Touch driver.

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**.
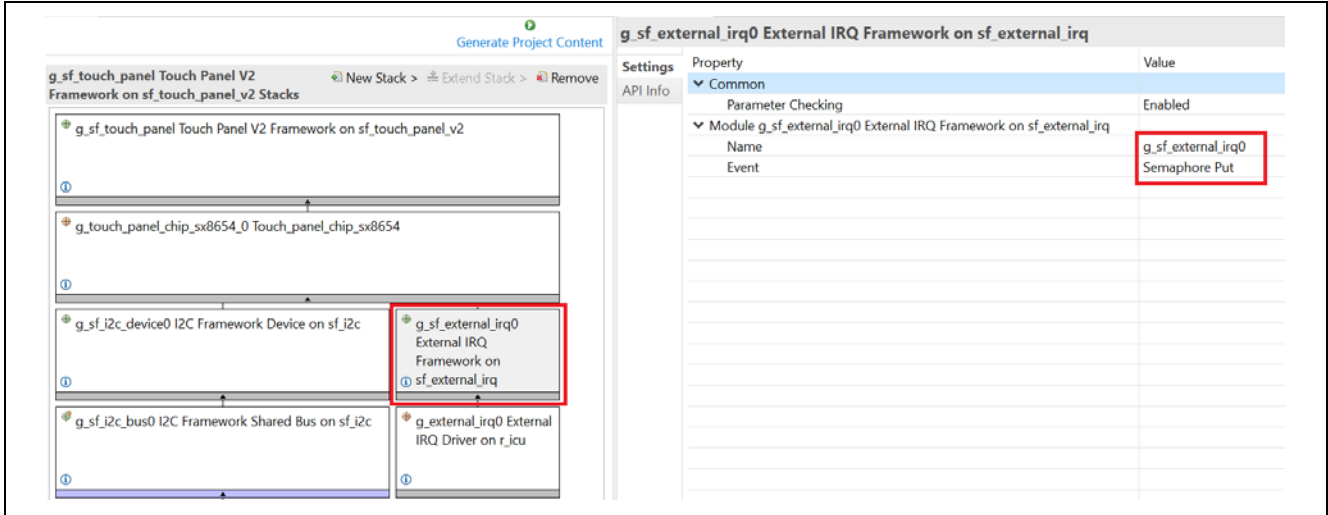
Figure 26.   Configure the properties for External IRQ Framework Stack

16. Select **External IRQ Driver on r_icu**. Configure the properties for the new module as shown.
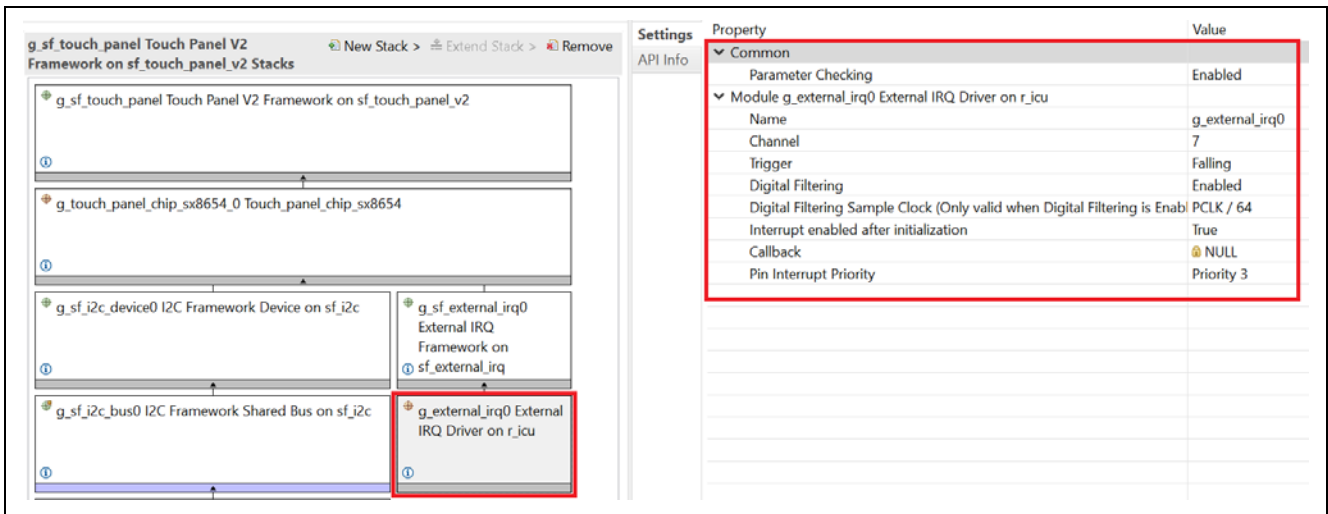
Figure 27.   Configure the Properties for IRQ Driver on r_icu

17. In the S**ynergy Configuration** window > **Threads** tab > **Main Thread Stacks** area, click on **g_sf_i2c_device0 I2C Framework Device on sf_i2c**. Configure the properties for **g_sf_i2c_device0 I2C Framework Device on sf_i2c**.
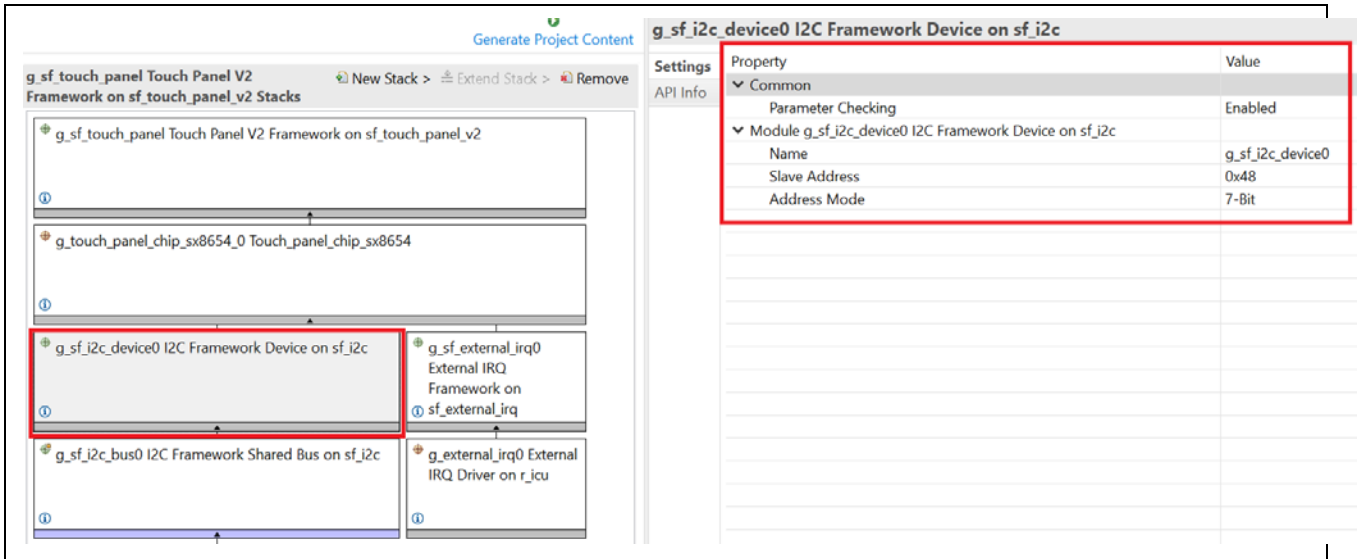
Figure 28.   Configure the Properties for g_sf_i2c_device0 I2C Framework Device on sf_i2c.

18. In the S**ynergy 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**
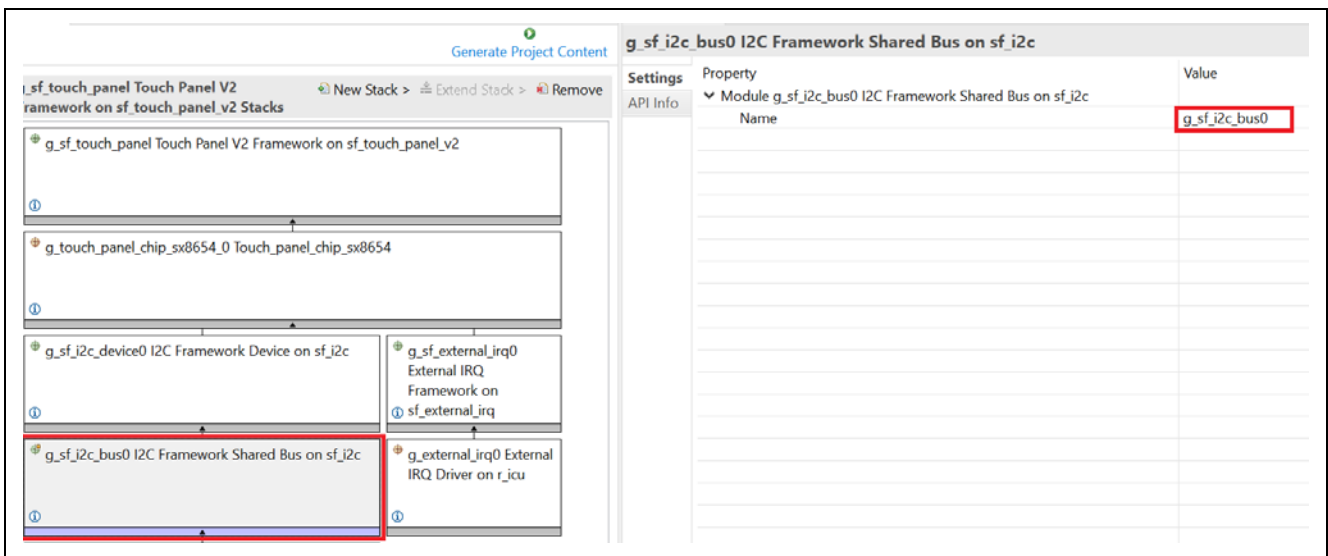
Figure 29.   Configure g_sf_i2c_bus0 I2C Framework Shared Bus on sf_i2c Properties

19. In the **Synergy Configuration** window > **Threads** tab > **Main Thread Stacks** area, click on **Add I2C Driver** > **New** > **I2C Master Driver on r_sci_i2c**
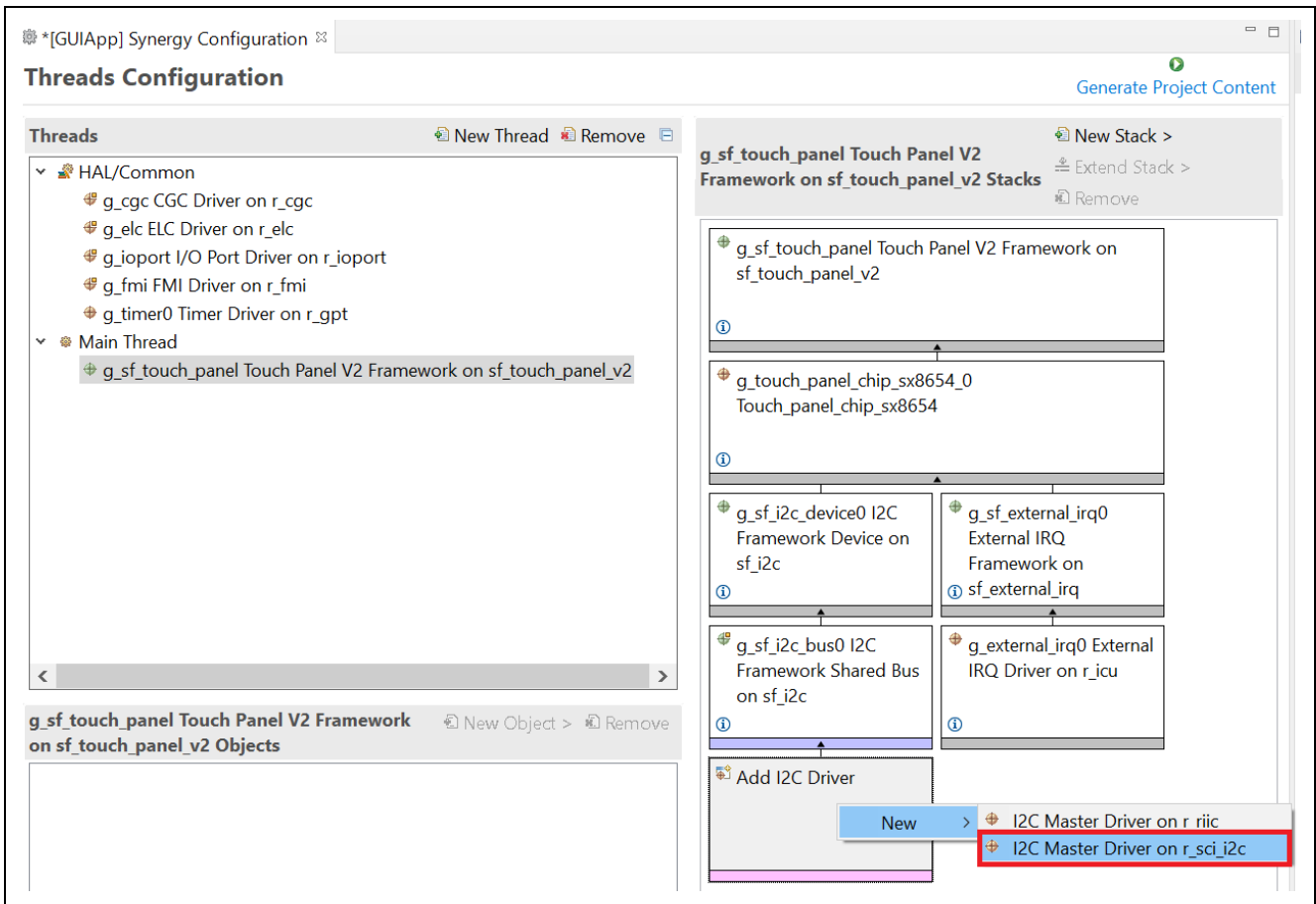
Figure 30. Add I2C Master Driver on r_sci_i2c

20. In the **S**ynergy **Configuration** window > **Threads** tab > **Main Thread Stacks** area, click on **I2C Master Driver on r_sci_i2c**. Configure the properties for **I2C Master Driver on r_sci_i2c**

Figure 31. Configure the Properties of Master Driver on r_sci_i2c

21. In the **S**ynergy **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 32. Configure the Properties of g_transfer0 Transfer Driver on r_dtc SCI7 TXI

22. In the S**ynergy 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**
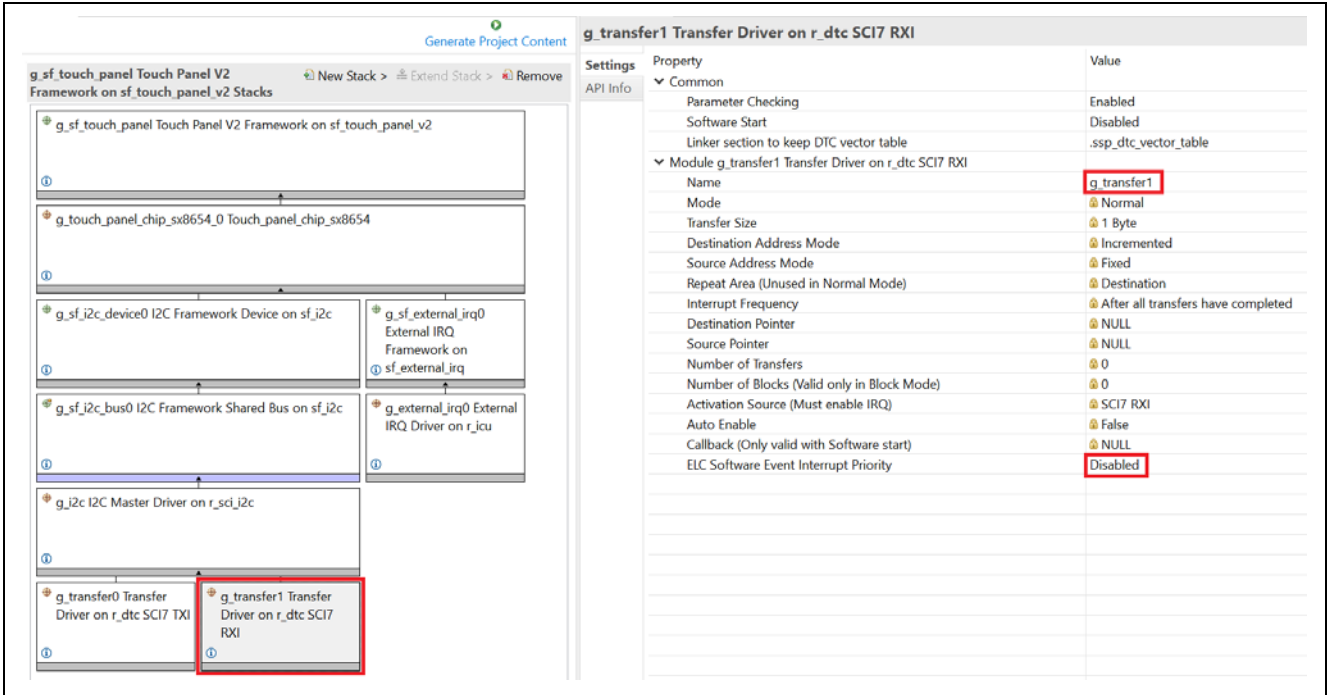
Figure 33.   Configure the Properties of g_transfer1 Transfer Driver on r_dtc SCI7 RXI

23. Under **Main Thread Stacks**, select **New Stack**, and then **X-Ware > GUIX > GUIX on gx**.
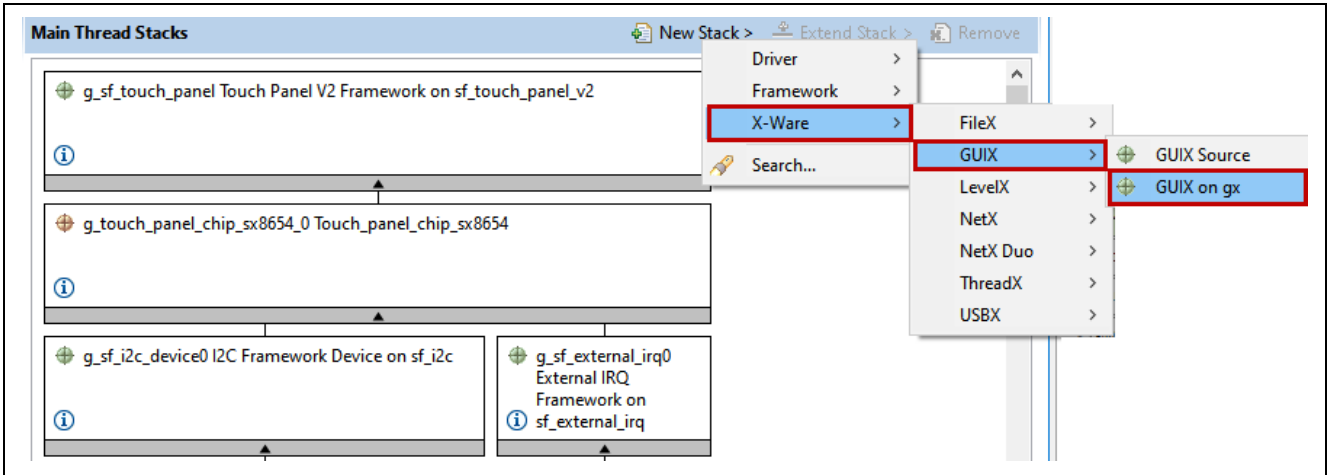
Figure 34.   GUIX on gx

Notice that the Synergy Configurator has now already created the **GUIX Port on sf_el_gx framework**, **Display Driver**, **JPEG decoder**, and **D/AVE hardware accelerator drivers** as shown in the following figure.
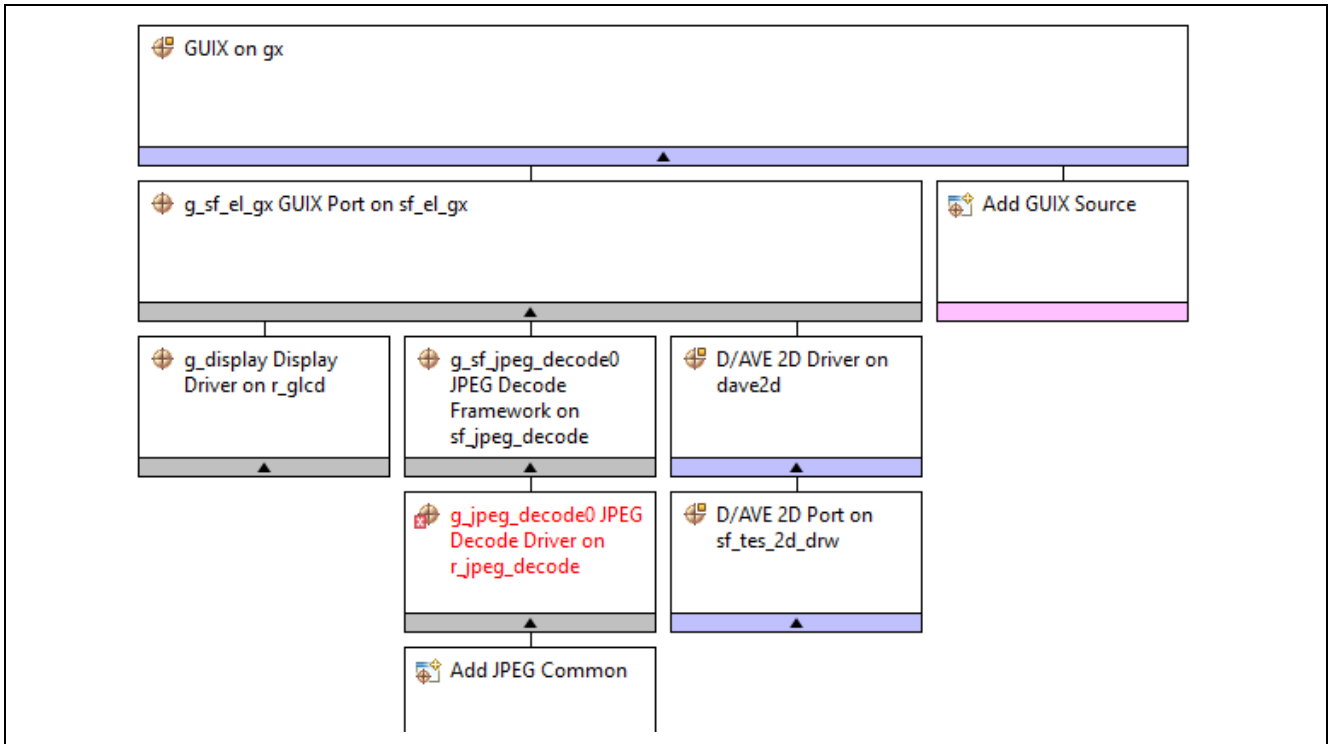
Figure 35.   GUIX on gx

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

Figure 36.   GUIX on gx Properties

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

Figure 37.   JPEG Common Module

26. Select **GUIX Port on sf_el_gx** and configure the properties as shown.

Figure 38.   GUIX Port on sf_el_gx Properties

27. Select the **JPEG Decode Driver on r_jpeg** and configure the interrupt properties as shown. 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 ARGB8888 format)	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 39.　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 properties as shown.

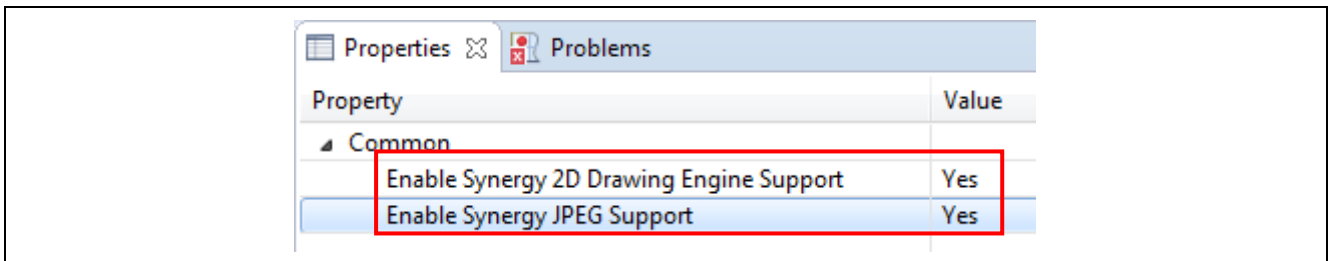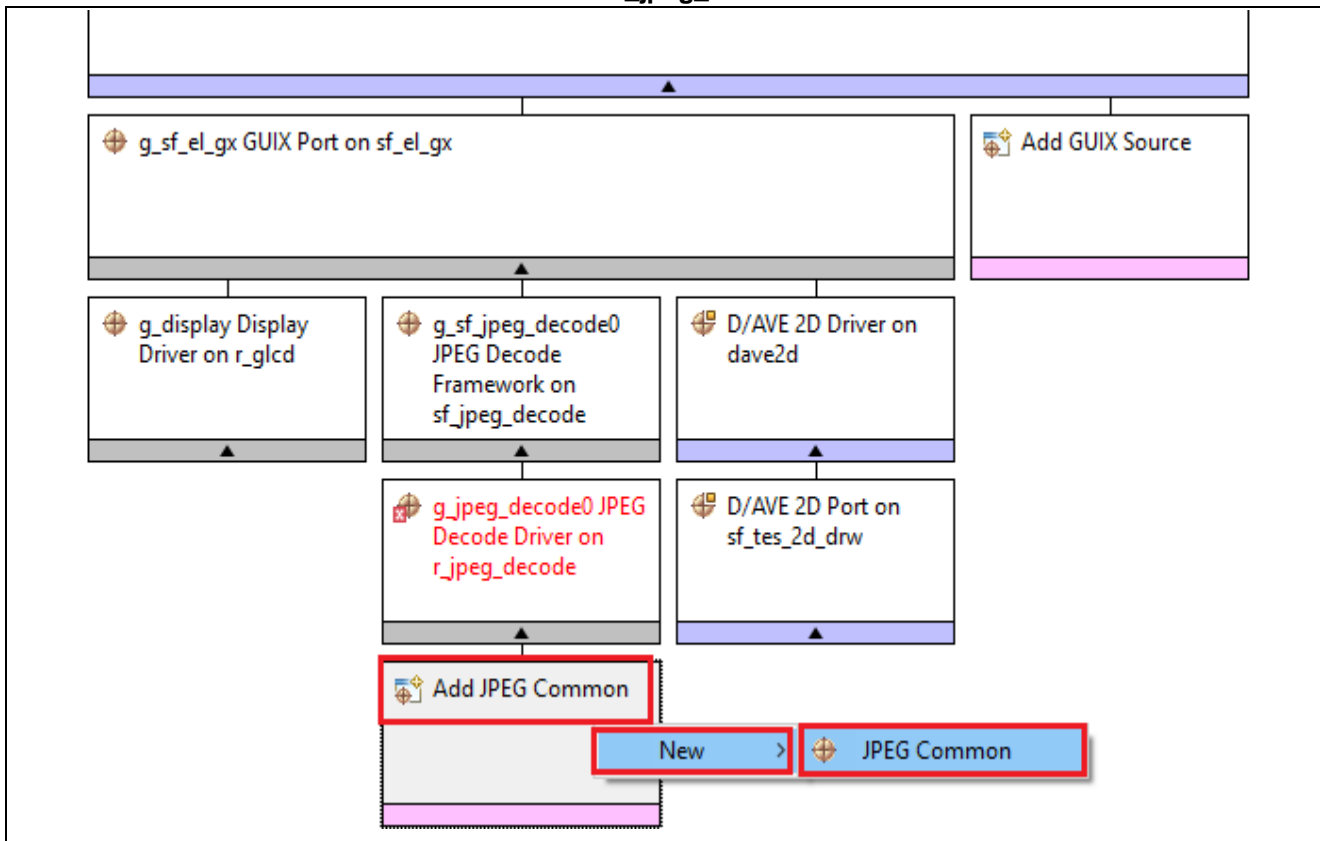| Property	Value
▲ Common	
Work memory size for display lists in	32768
DRW Interrupt Priority	Priority 3 (CM4: valid, CM0+: lowest - not valid if using ThreadX)

Figure 40.　D/AVE 2D Port Properties

29. Under **Main Thread Stacks**, select **Display Driver on r_glcd** and configure the Interrupt Properties as shown.

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 41.　Interrupt Properties

30. Configure the **Graphics Screen 1** properties as shown.

Property	Value
▲ Common	
Parameter Checking	Default (BSP)
▲ Module g_display0 Display Driver on r_glcd	
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	480
Input - Graphics screen1 input vertical size	272
Input - Graphics screen1 input horizontal stride(not bytes but pixels)	480
Input - Graphics screen1 input format	16bits RGB565

Figure 42.　Graphics Screen 1 Properties

31. Configure the **Output** properties as shown.

|	
Output - Horizontal total cycles	582
Output - Horizontal active video cycles	480
Output - Horizontal back porch cycles	43
Output - Horizontal sync signal cycles	41
Output - Horizontal sync signal polarity	Low active
Output - Vertical total lines	286
Output - Vertical active video lines	272
Output - Vertical back porch lines	12
Output - Vertical sync signal lines	10
Output - Vertical sync signal polarity	Low active
Output - Format	16bits RGB565

Figure 43.   Output Screen 2 Properties

32. Configure the TCON pins and clock as shown.

TCON - Hsync pin select	LCD_TCON1
TCON - Vsync pin select	LCD_TCON2
TCON - DataEnable pin select	LCD_TCON0
TCON - Panel clock division ratio	1/16

Figure 44.   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.
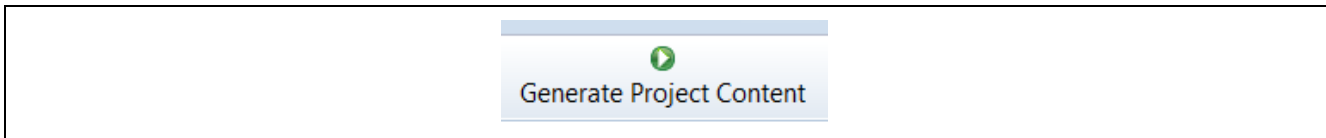
Figure 45.   Generate Project Content

35. Close the **Synergy Configuration** window.
36. Open Windows Explorer and go to the directory 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` root directory 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 pin configuration for the DK-S7G2 Synergy MCU.
37. In the Synergy Configuration window, under the Pins tab Select the Import the pin configuration, as shown.

Select pin configuration

S7G2-DK.pincfg   ☑ Generate data: g_bsp_pin_cfg
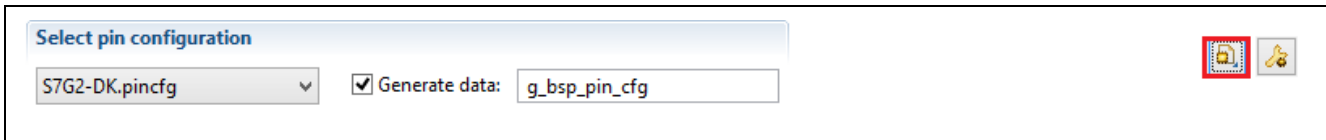
Figure 46.   Synergy Pin Configuration

38. Click **File System** to access the pin configuration given in the source file as shown.
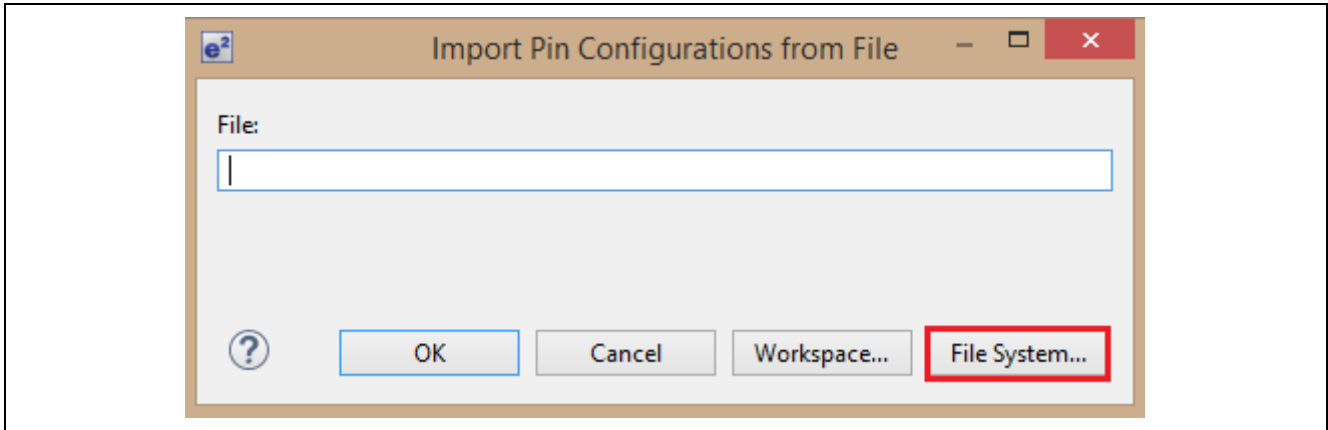
Figure 47.   Importing Pin Configuration

39. Select the pin configuration in the source file and press **Open** as shown. Click **OK** to import the pin configurations.
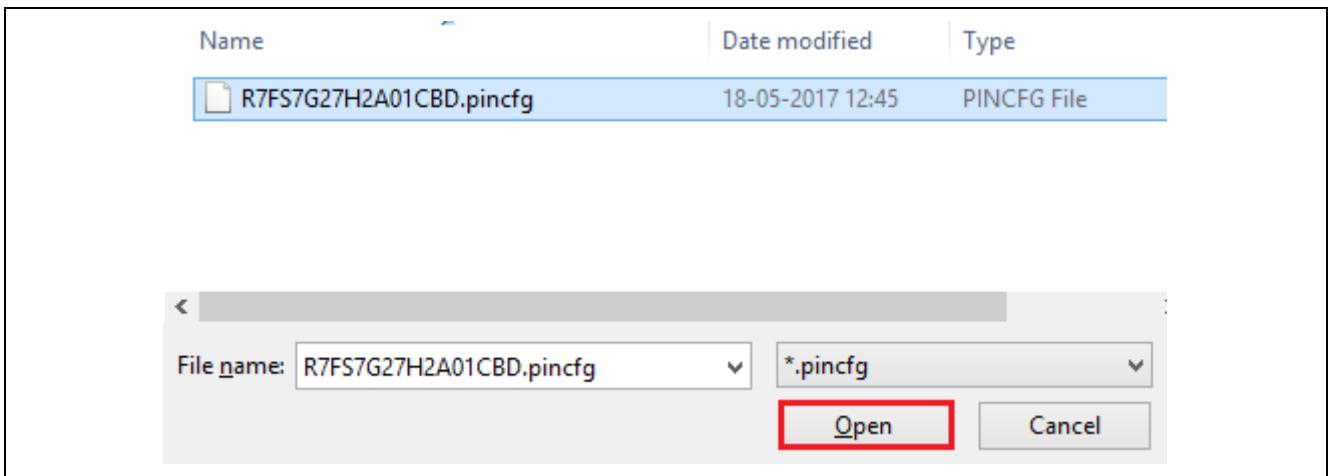
Figure 48.   Synergy Configuration

40. If you get the following error message when importing the pin configuration file, click **OK**.

Figure 49.   Conflict Found Error Box

41. Once you select the reference pin configuration, you will get a pin error message, **Pin Dangling**. This means two functions are using the same pin. As shown, the error is in the **Peripherals**.

Figure 50. Selecting Pin Configuration

42. Click **Peripherals**, then **Storage:SDHI**. Select **SDHI0** and go to **DAT7** to change it to **None** as shown.

Figure 51. Making P313 Usable

The next steps show how to configure pins of the S7G2 MCU to control the LCD panel and touch screen on DK-S7G2 MCU. Proceed to Step 43 to skip this optional informational section.

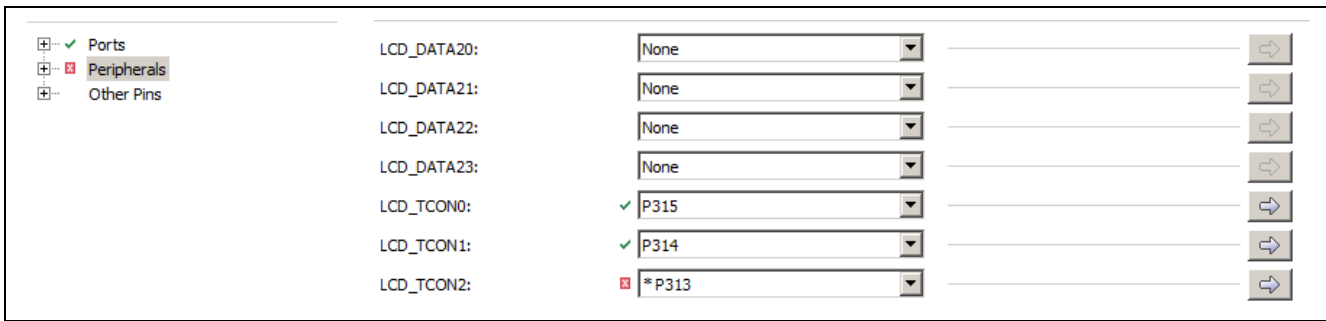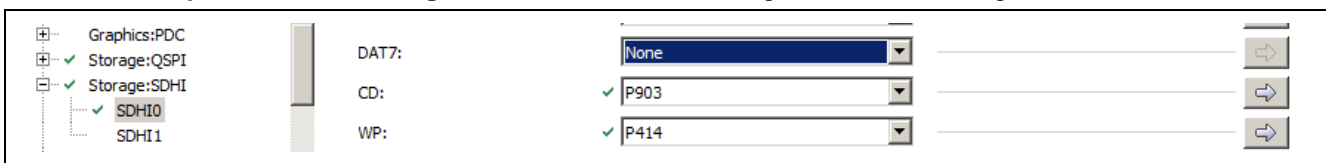The **Timer Driver on r_gpt** is used to configure the peripheral as a PWM to control the backlight level using a hardware pin on the S7G2 Synergy MCU. For the DK-S7G2 MCU, the pin that controls the backlight for the LCD is located on P7_12, as shown from the following snippet showing the DK-S7G2 MCU breakout board schematic.
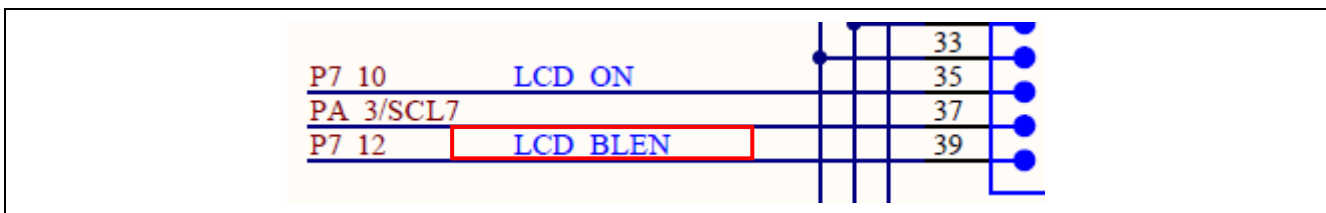
Figure 52. LCD Backlight Pin

Since an existing pin configuration is being used, it is not necessary to set this pin up using the pin configurator. If you are interested, follow the steps below to see how it was configured.

43. Select the **Pins** tab in the **Synergy Configuration Window**.
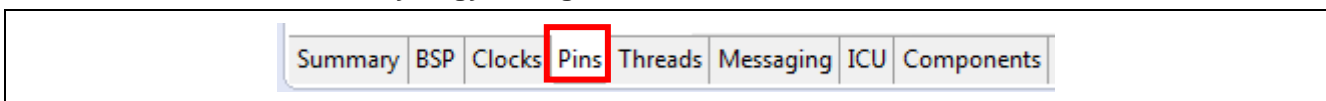
Figure 53. Configuration Pins

44. Expand **Ports** and **P7** to show the port 7 pins.
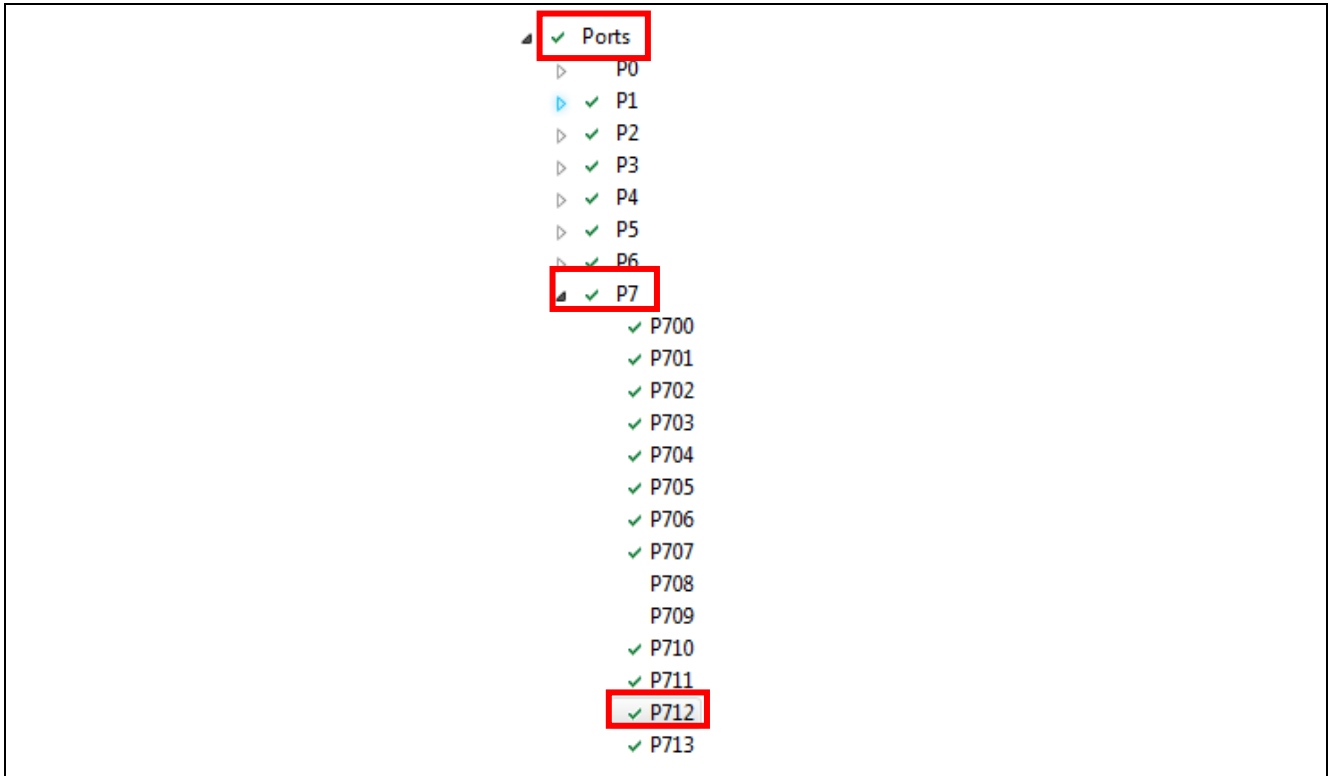
Figure 54. Port 7 Pins

45. Select **P712** to show the options for this pin.
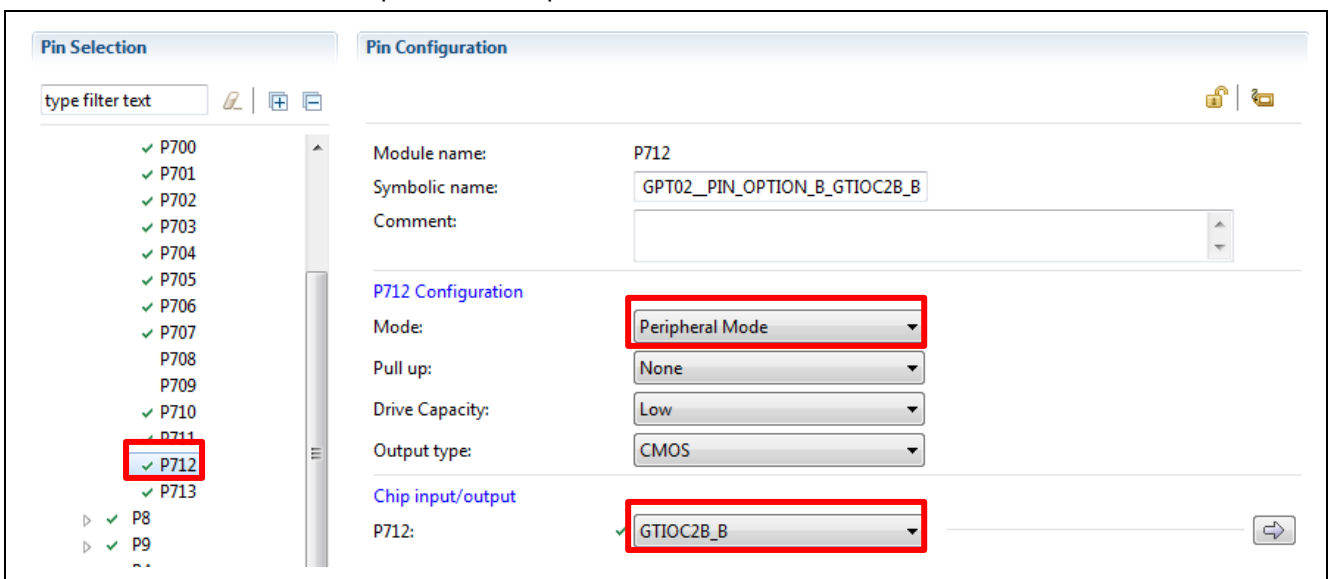
Figure 55. Pin Configuration for P7_12

— Module name: Selected **GPT2**.

— Pin Group Selection: Selected **Mixed**

— Operation Mode: Selected channel **GTIOCA or GTIOCB** output type: Only selected channel **GTIOCB** for the pin P712. This option changes based on the mode setting. In this case, GTIOC2B_B is selected to use as the **Timers:GPT 2 B** output.

— (Hint: Disable pin P712 first before select and setting **Timers:GPT ; GPT2**.

Pins can also be configured using the peripheral as a starting point.

This view shows the pins that are available for different functions.
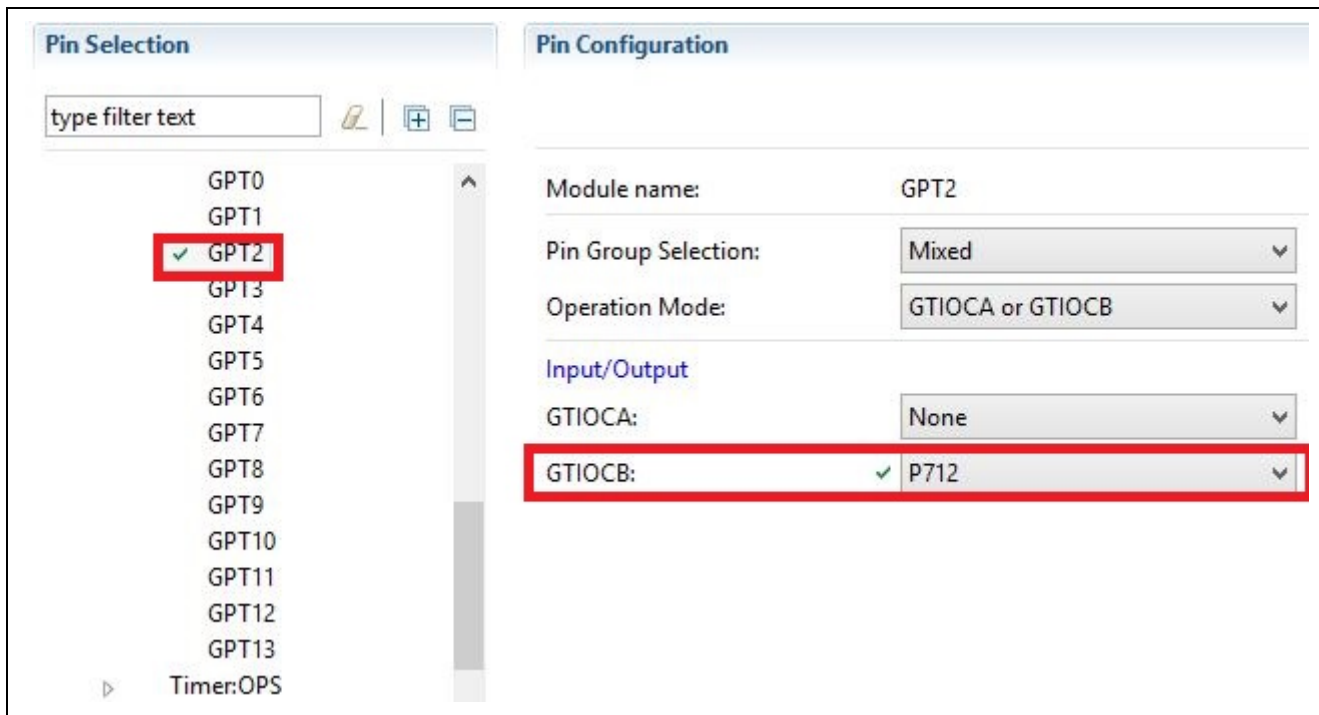
Figure 56.　GPT02 Pin Configuration

In most cases, after you enable and select a pin, it is automatically configured. This pin can be configured by pressing 🠖, which shows the following screen.

The interrupt for the touch controller is located on pin **P0_1** as seen in the breakout board schematic.

Figure 57.　Touch IRQ

The touch panel pin is configured as an IRQ in the **Pin Configuration** window.
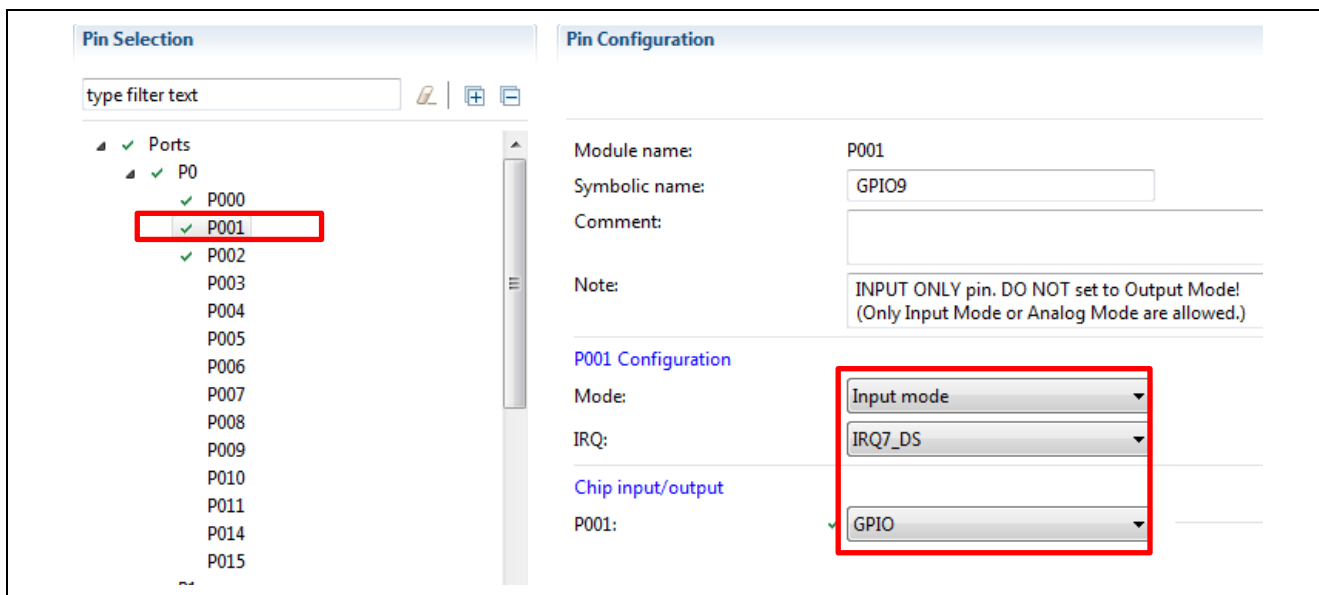
Figure 58.　P0_01 Pin Configuration

For the LCD board schematic, see the touch controller, which is the SX8656.
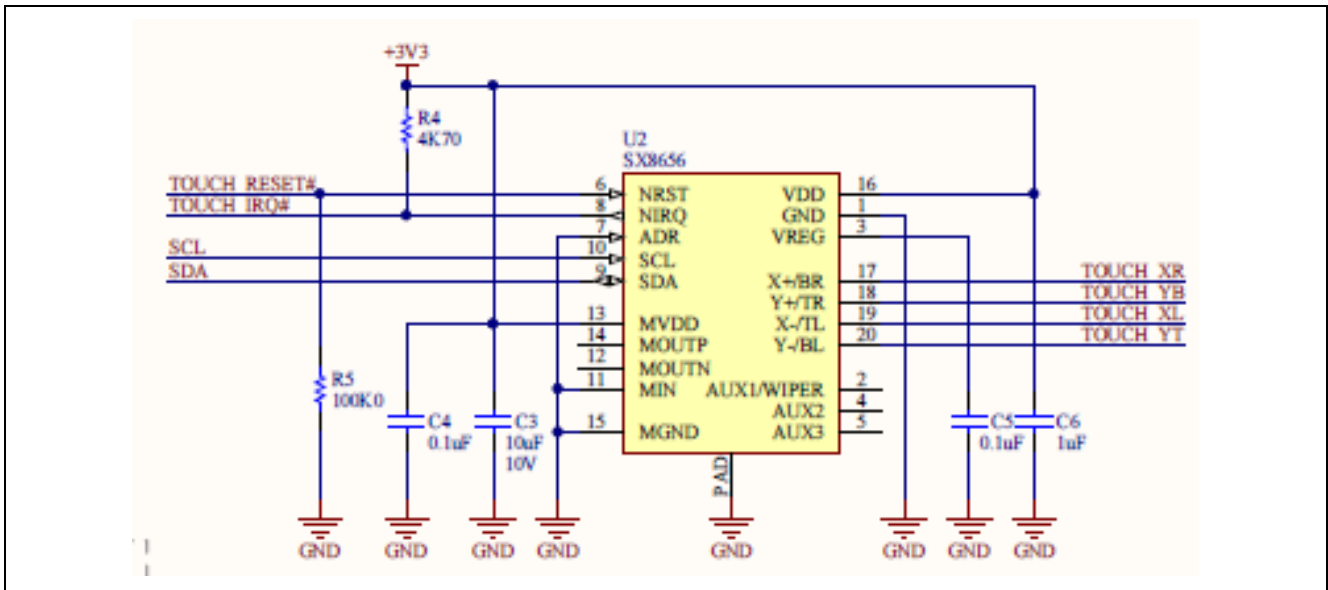
Figure 59.   Touch Controller

The following figure shows the touch controller's reset pin is located on P07_11. To make use of this function, set the pin up as a GPIO output.

Figure 60.   P7_11 Pin Configuration

The SCI driver can be configured for different serial communication protocols for the DK-S7G2 pin PA_3 and PA_2 are used to handle the I2C functionality.

Figure 61.   I2C Pins

The pins are configured in the Pin Configurator under the peripheral section.
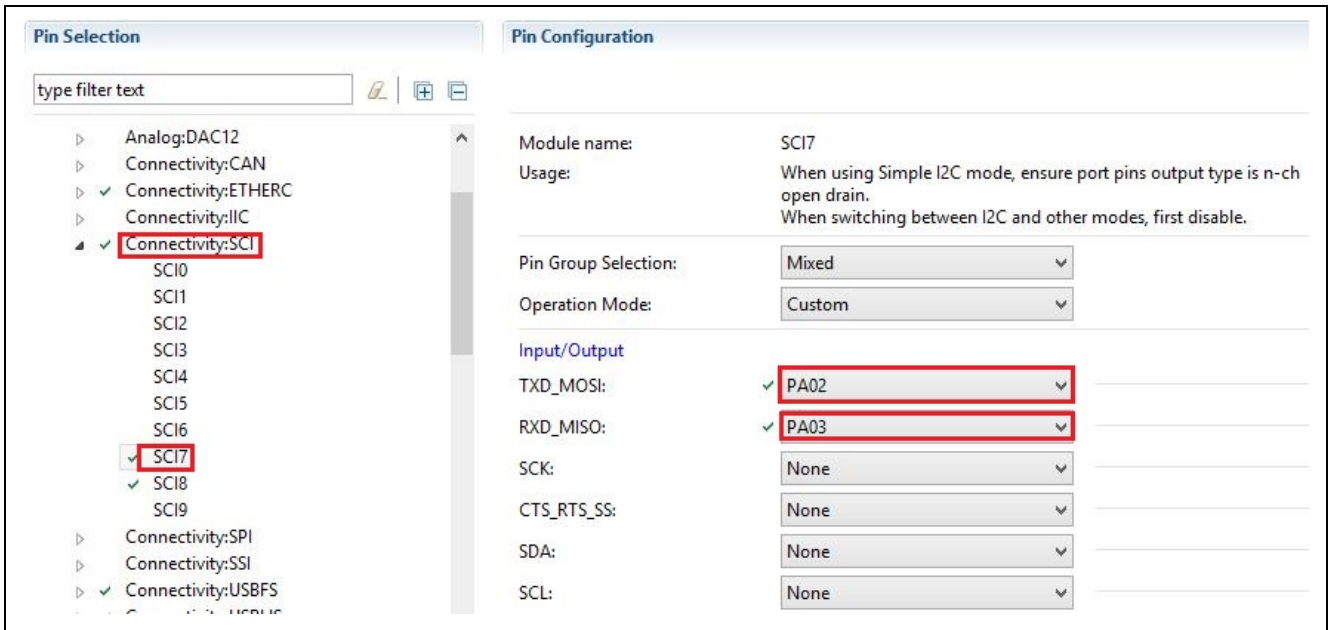
Figure 62. SCI7 Configuration

The LCD pin configuration is based on Option B for the **g_lcd** controller as seen in the pin configurator below.

Figure 63. GLCD Pin Option B

The breakout board schematic shows the full list of pins. The DK-D7G2 MCU uses a 16-bit LCD interface.
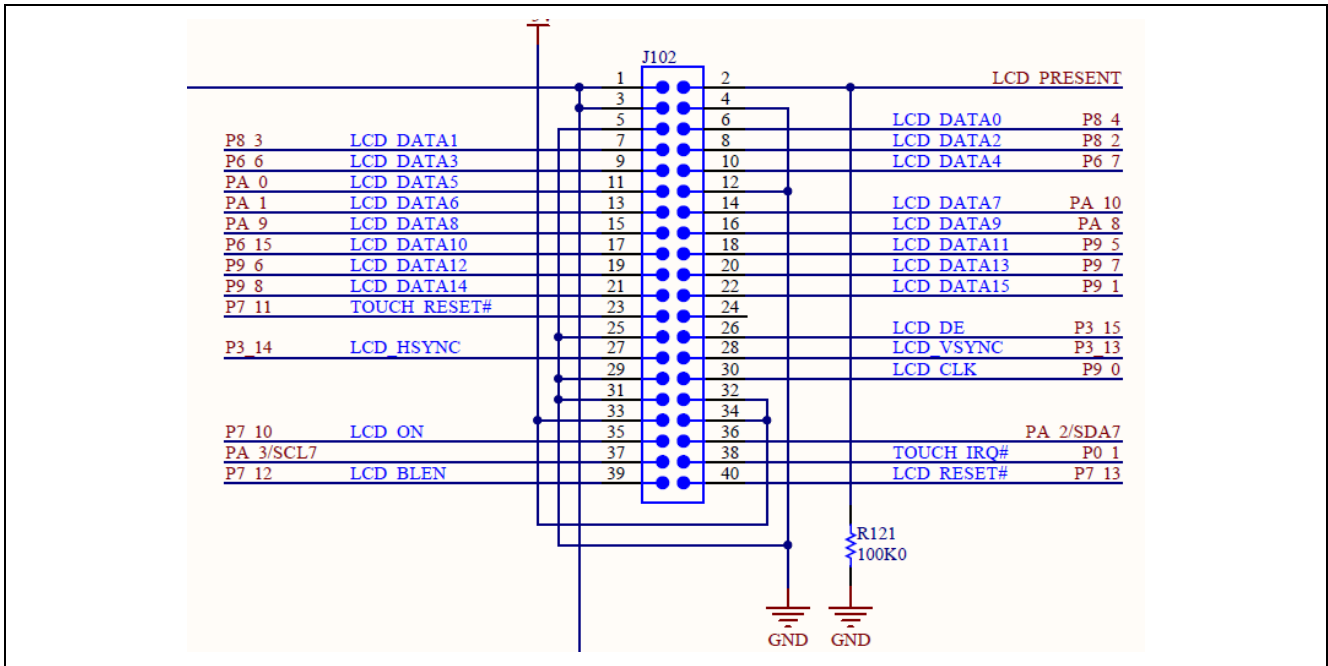
Figure 64.   LCD Connector Pin Out

46. Save the project by pressing **Ctrl + s** on the keyboard.
47. Click the **Generate Project Content** button to update the project files.
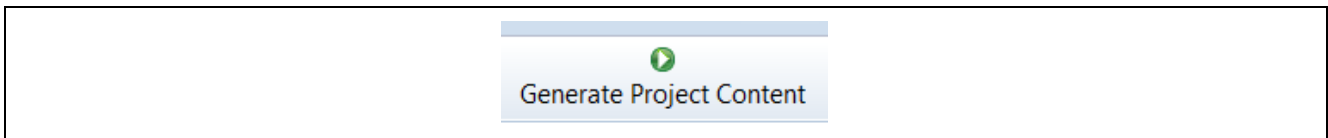
Figure 65.   Generate Project Content

48. Open **Windows Explorer** and find where you put the files included with this application note. Locate the file `Source Files\main_thread_entry.c`. Now 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 has been set up, GUIX components can be added.

1. Create a new folder named **gui** inside the **src** folder by right clicking on the **src** folder and selecting **New > Folder**.
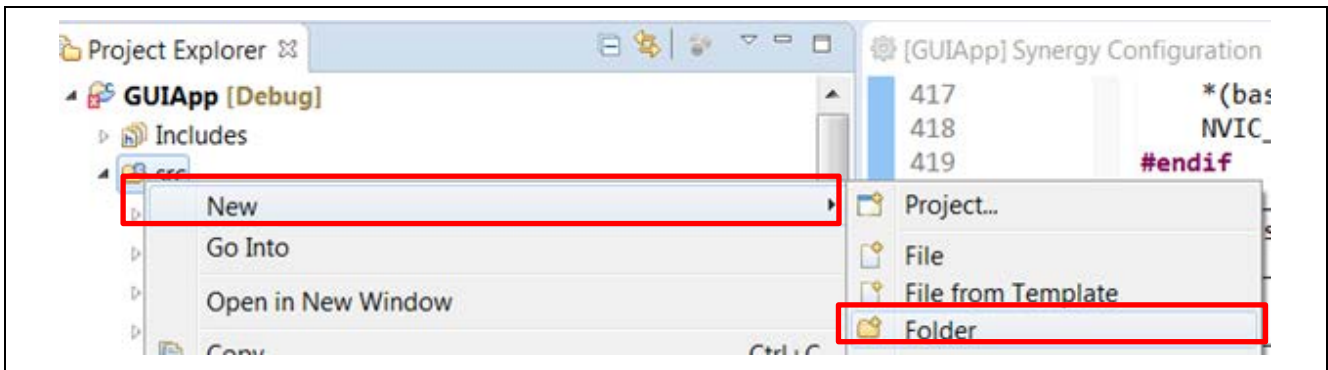
Figure 66.   Creating a New Folder

2. Create another new folder named **guix_studio** in the root folder of the project by right clicking on **GUIApp** and selecting **New > Folder**. The final folder layout should look like the following figure.
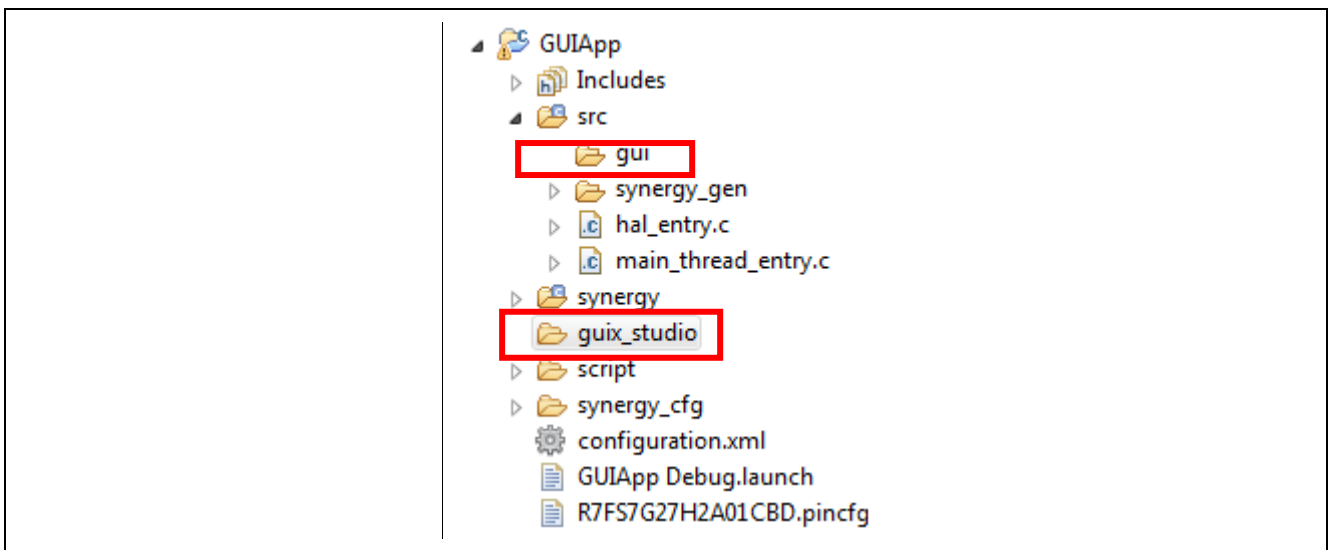
Figure 67.   Final Folder List

3. Open GUIX Studio by clicking the desktop icon. The version of GUIX Studio must be 6.1.8.0 or later.

Figure 68.   Start GUIX Studio

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

Figure 69.   Create New Project

5. Name the project **guiapp**.
   WARNING: Filenames will be generated by appending names to the project name. You must be careful to make names case sensitive when you define your project name. Later, when files are added to the project, it's assumed that you have called this GUIX project **guiapp**.
6. 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 70.   Create a New GUIX Project

7. Click **Save**.
8. Change the Directories for all three options to be **..\src\gui**.
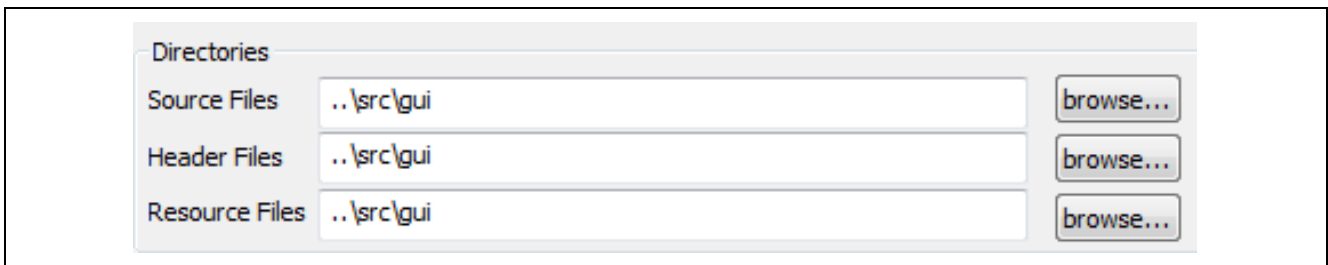
Figure 71.   Correct the file locations

   Caution: Make sure you put in two dots **..** in the directories above.
9. Change the **Target CPU** setting to **Renesas Synergy**.
10. Change the **Toolchain** setting to **GNU** and **GUIX Library version 6.1.8** to **.4.1**.
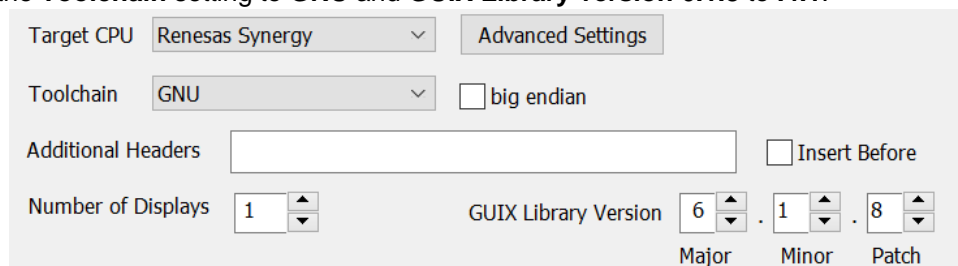
Figure 72.   Target and GUIX version settings

11. Click the **Advanced Settings** button. A dialog window appears.
12. Set **Enable 2D Drawing Engine** to enable the graphics accelerator and **Hardware JPEG Decoder** as shown.
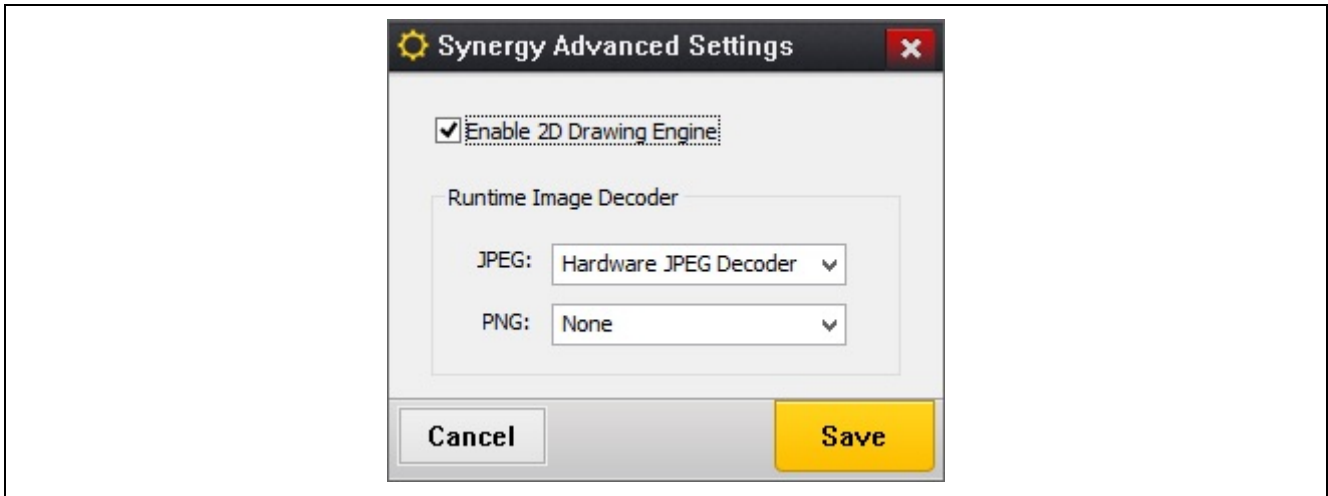
Figure 73.   Synergy Advanced Settings

13. Click **Save**.
14. Set up the **Display Configuration** as shown.
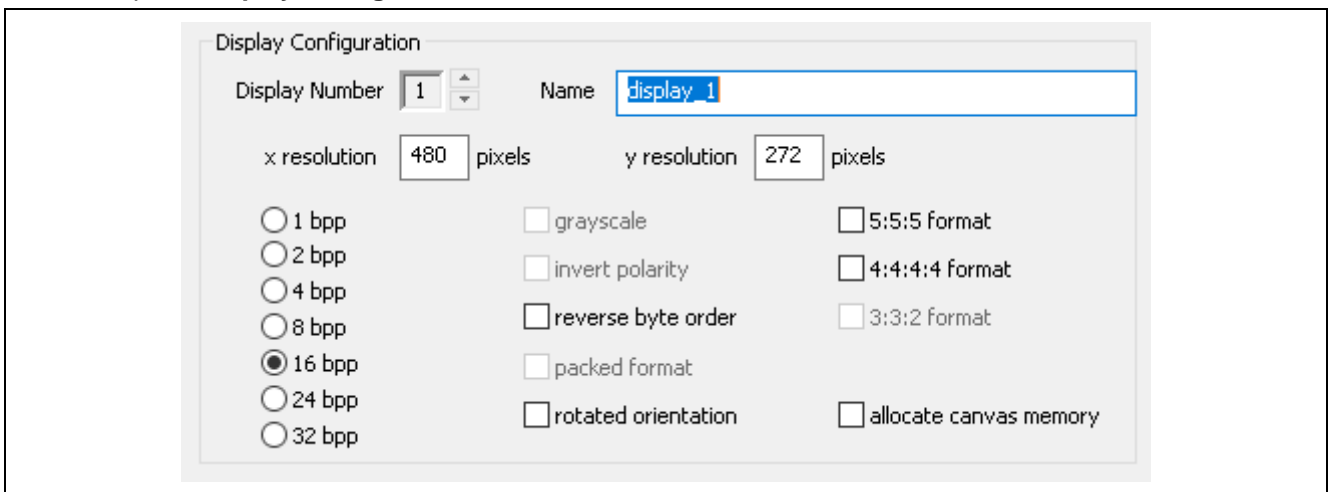
Figure 74.   Configure the Display

15. Click **Save** to generate the project.
16. Right-click on **display_1** in the project view.
17. Select **Insert > Window > Window**.
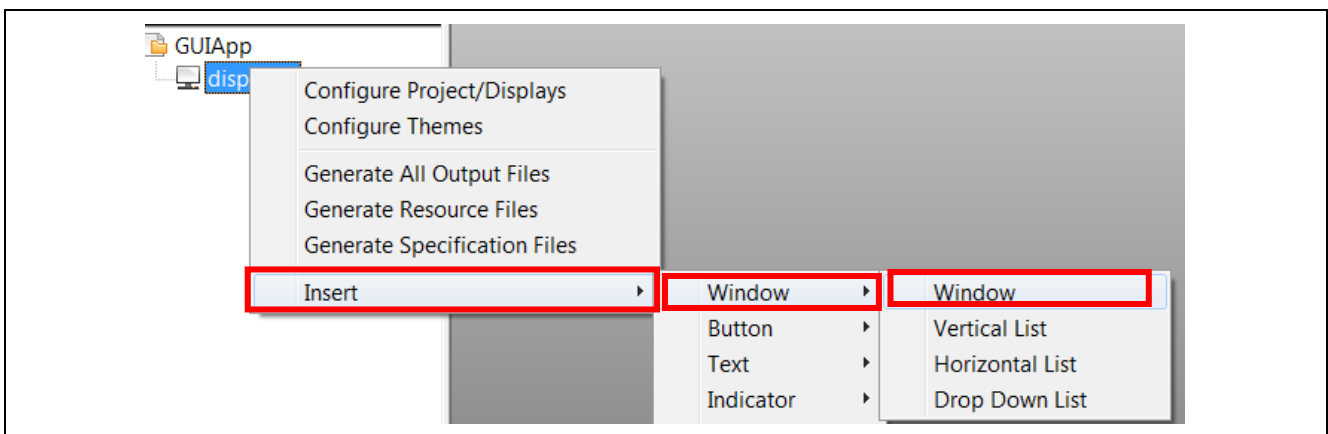
Figure 75.   New Window

18. Modify the properties by selecting the new window and editing the Properties View. Update the current settings to match those shown.
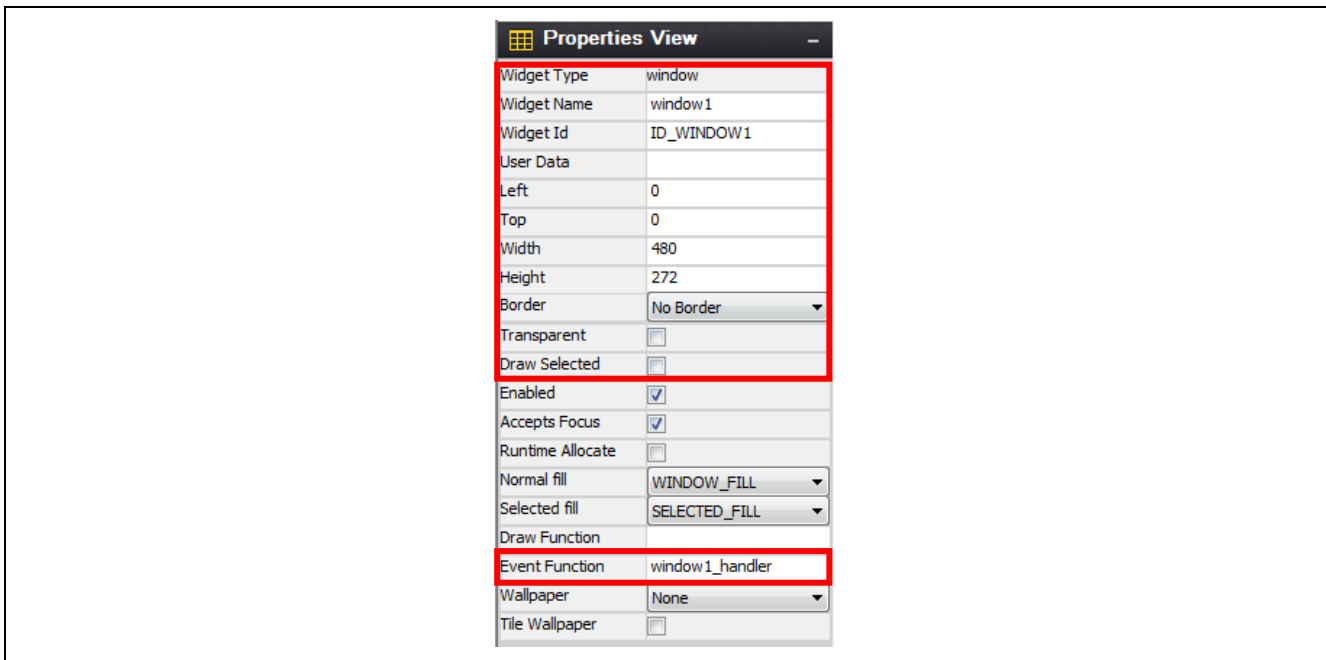
Figure 76.   Configure Window1 Properties

19. In the **Project View Window**, right click on **display_1** and create another window by selecting **Insert > Window > Window**.

20. Modify the properties to match the following figure.
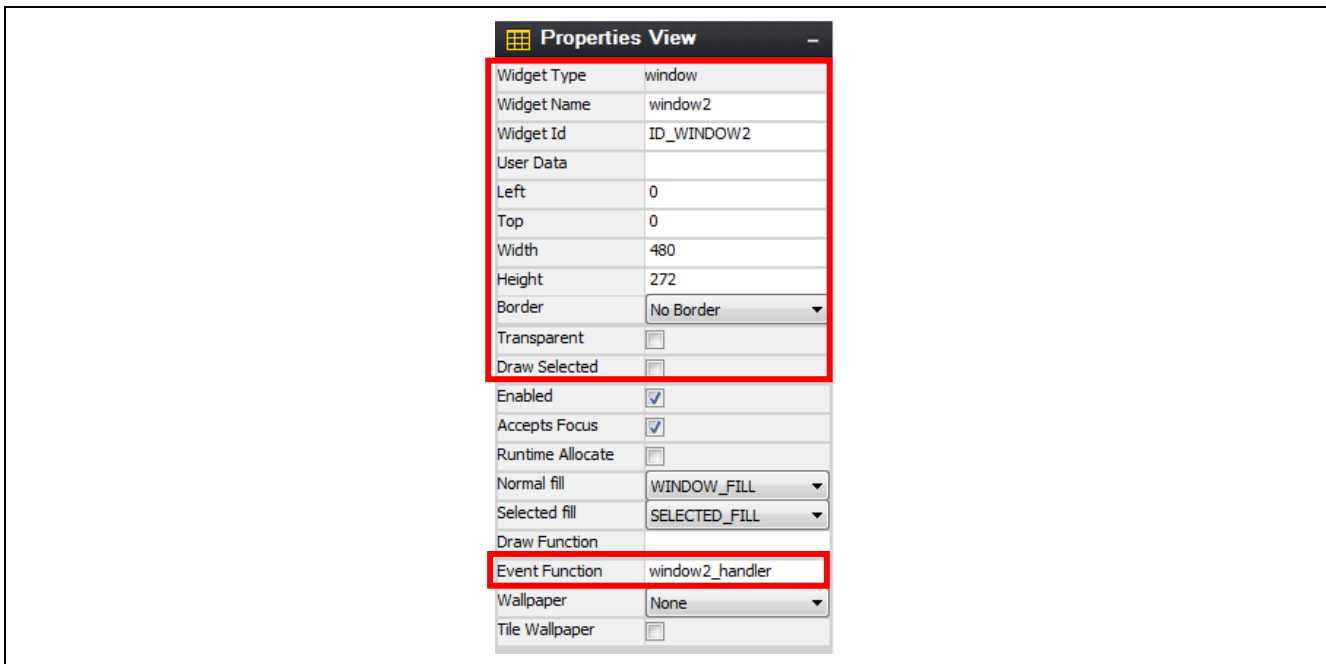
Figure 77.   Configure Window2 Properties

21. In the **Project View**, right-click on **window1** and insert a Button (Text Button) by selecting **Insert > Button > Text Button**.
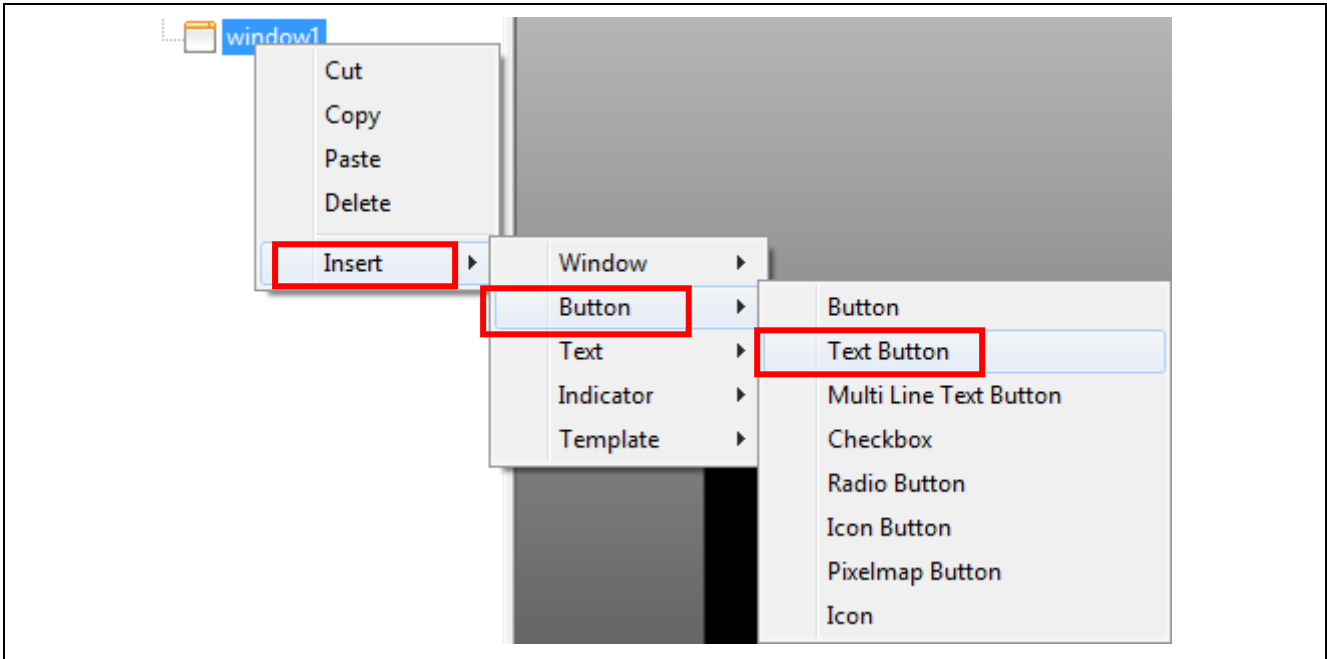
Figure 78.   Add a New Text Button

22. In the **Project View**, right-click on **window1** and insert a Button Checkbox by selecting **Insert > Button > Checkbox**.
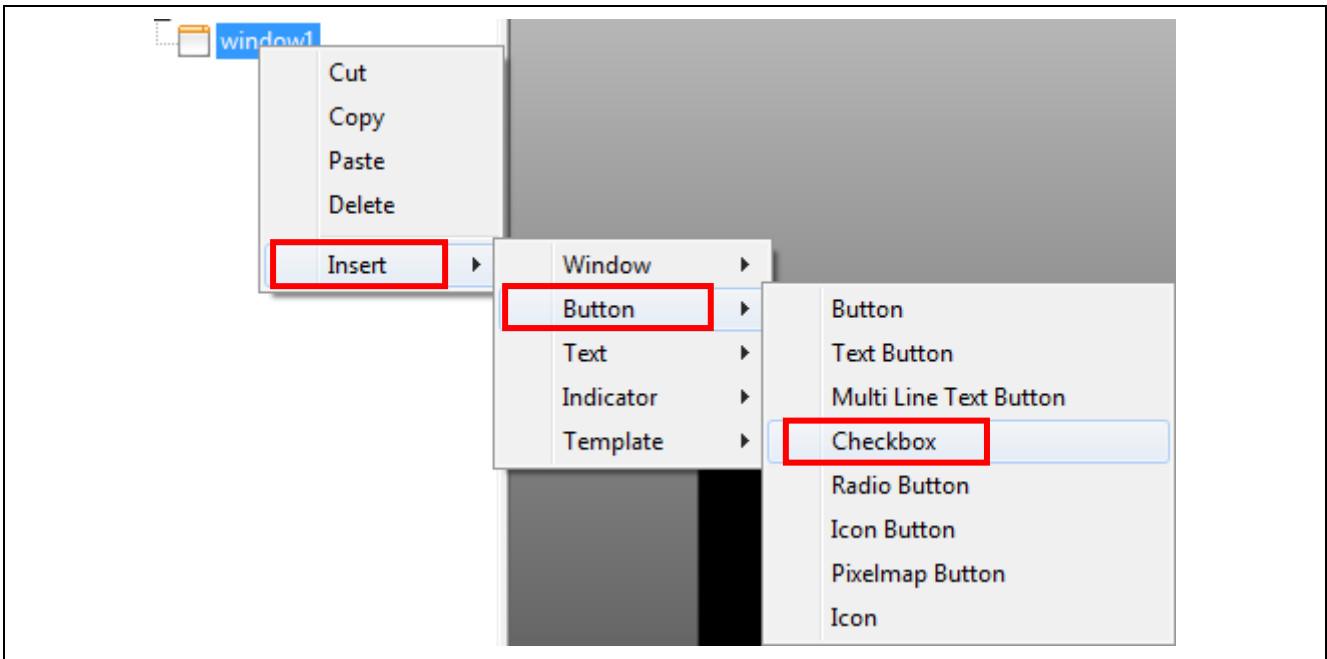
Figure 79.   Add a New Checkbox

23. In the **Project View**, right-click on **window1** and insert a Text Prompt by selecting **Insert > Text > Prompt**.
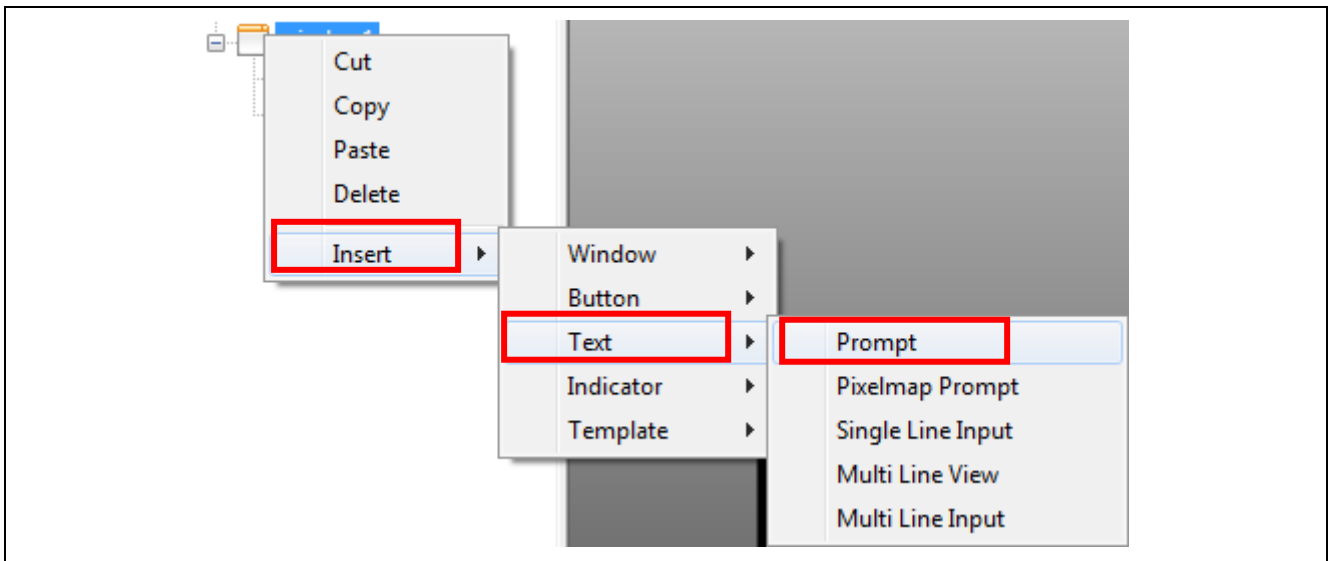
Figure 80.　Adding New Prompt

24. In the **Project View**, right-click on **window1 a**nd insert another **Text Prompt**.
25. In the **Project View,** right-click on **window2** and insert a **Text Prompt**.
26. In the **Project View**, right-click on **window2** and insert another T**ext Prompt**.
    After you have followed these directions, your Project View should look like the following screen.
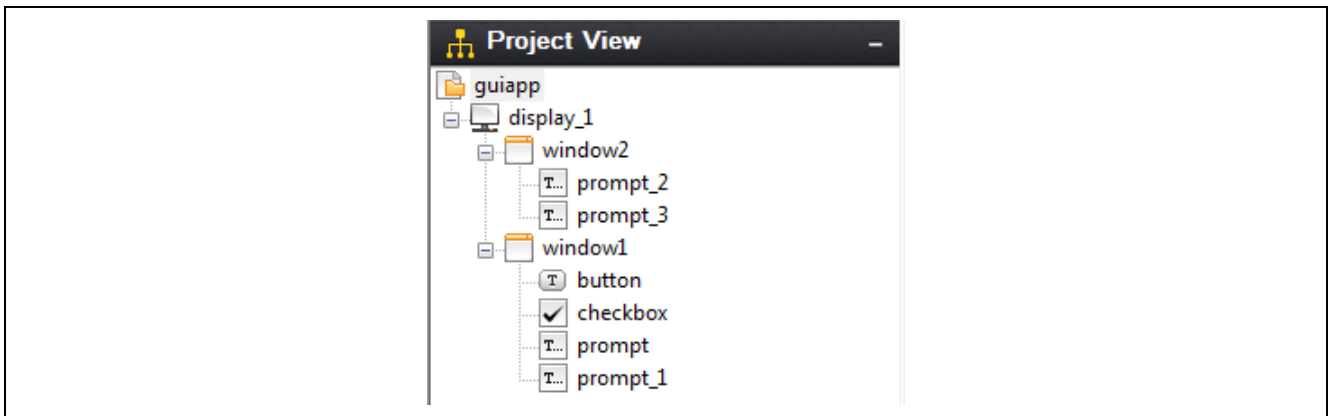
Figure 81.　GUIX Project View

27. Press the '**+**' character on right of the **</> Strings** to expand the Strings menu.
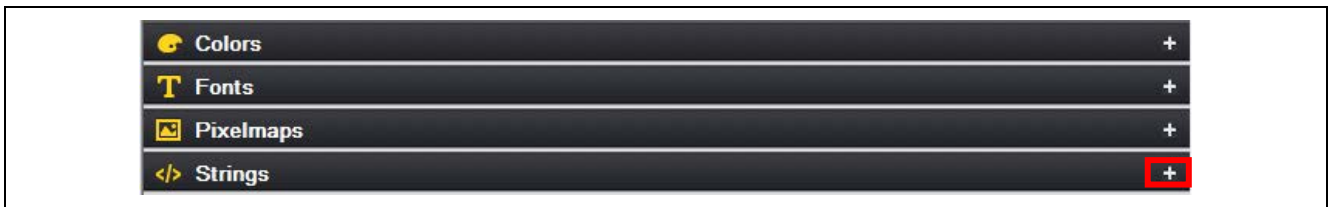
Figure 82.　Strings Button

28. Double-click on any of the strings to open the **String Table Editor**.
29. Delete the existing strings by selecting them, then click the **Delete String** button in the **String Table Editor**.
30. Add the strings using the **Add String** button as shown.

Figure 83.　New Strings

31. When correct, click the **Save** button.

32. In the **Project View** under window1, click on the button, then modify the properties in the **Properties View** to match the following figure.

Figure 84.　Configure Windowchanger Button Properties

33. In the **Project View** under **window1**, click the checkbox, then modify the properties in the **Properties View** to match the following figure.
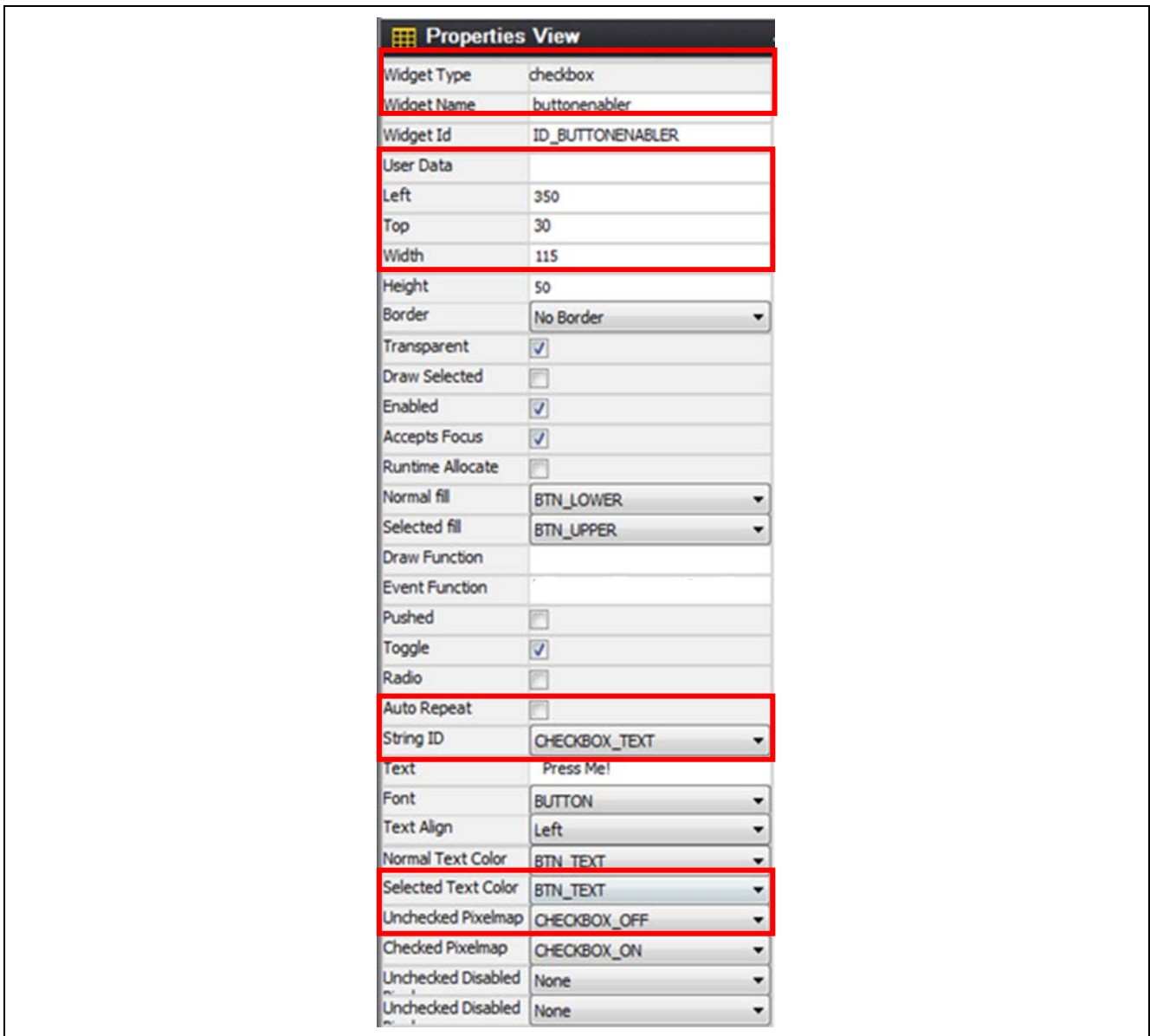
Figure 85.   Configure Buttonenabler Checkbox Properties

34. In the **Project View** under **window1**, click the prompt, then modify the properties to match the window below.
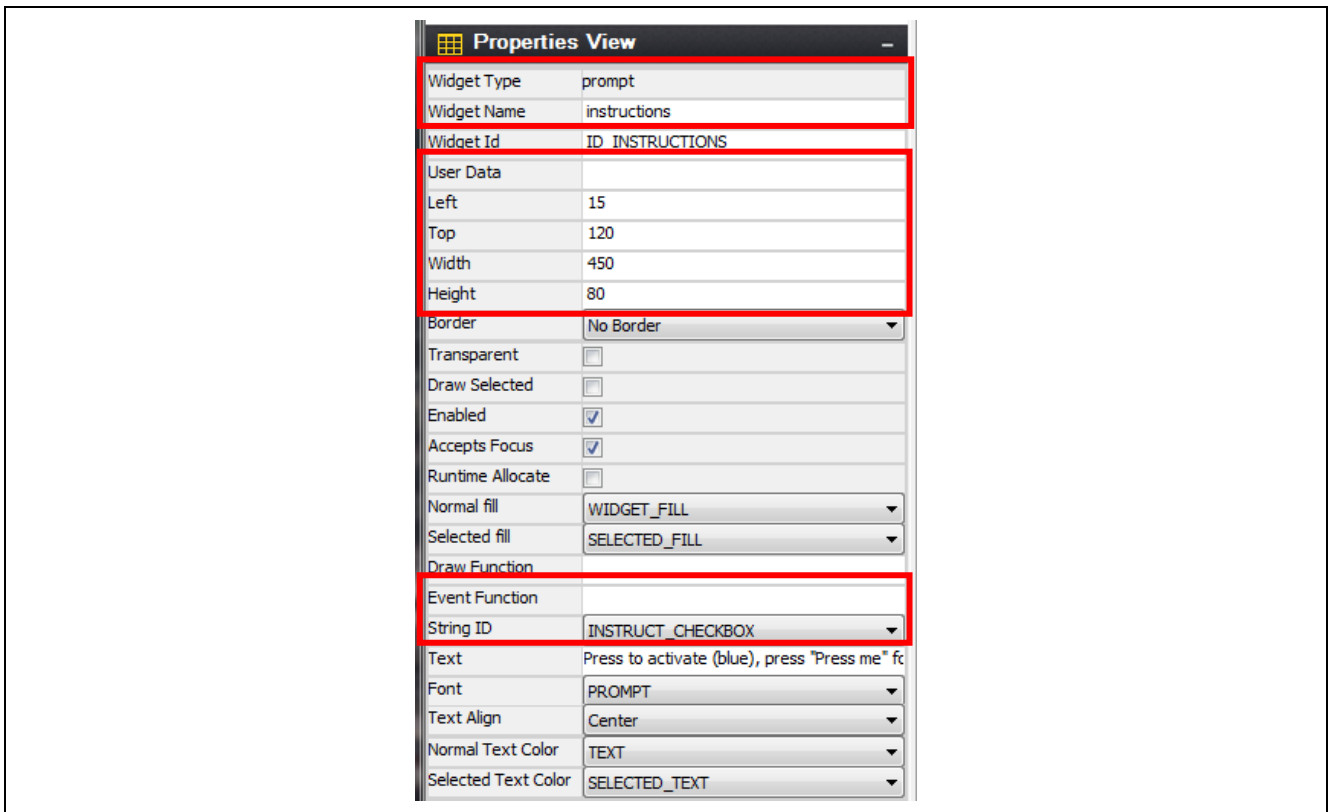
Figure 86.   Configure Prompt Properties

35. In the **Project View** under **window1**, click **prompt_1**, then modify the properties to match the following figure.
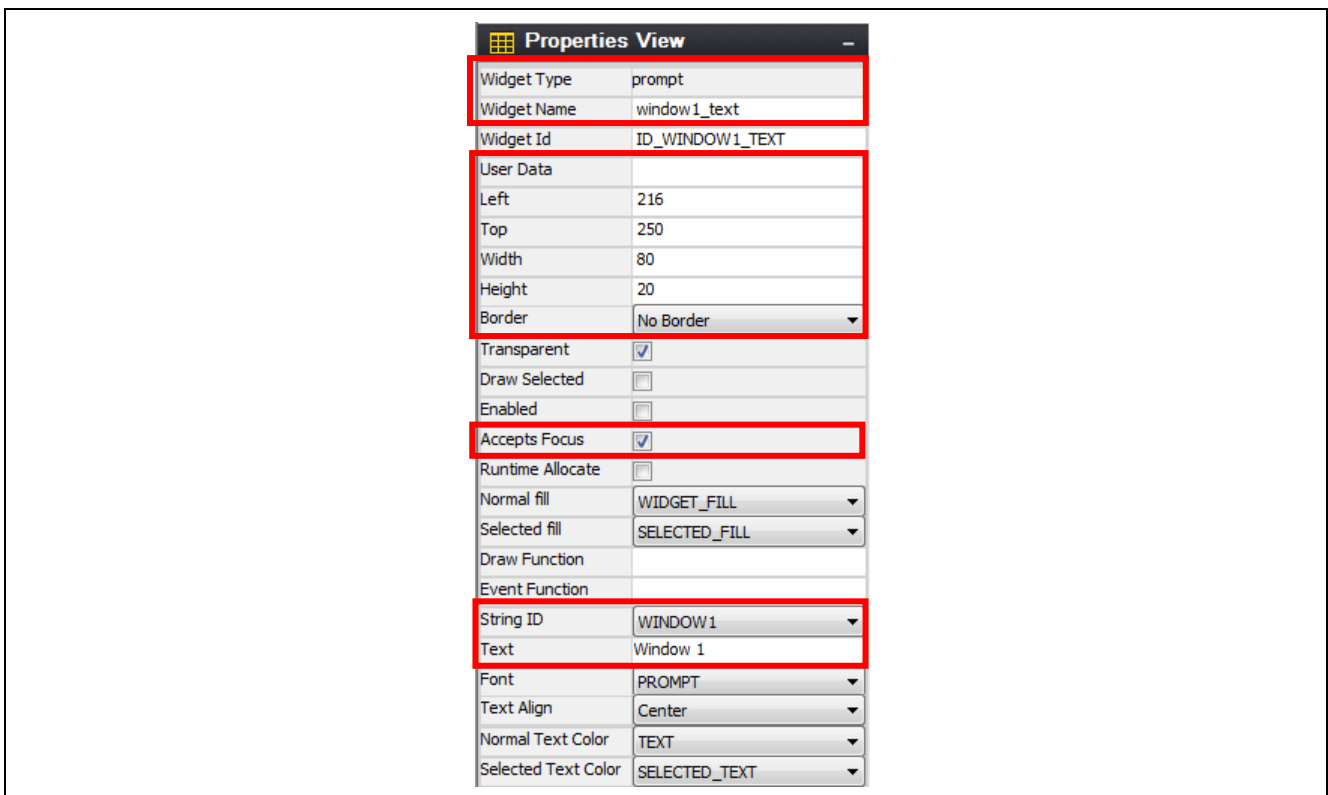
Figure 87.   Configure Window Text Properties

36. In the **Project View** under **window2**, click **prompt_2**, then modify the properties to match the following figure.
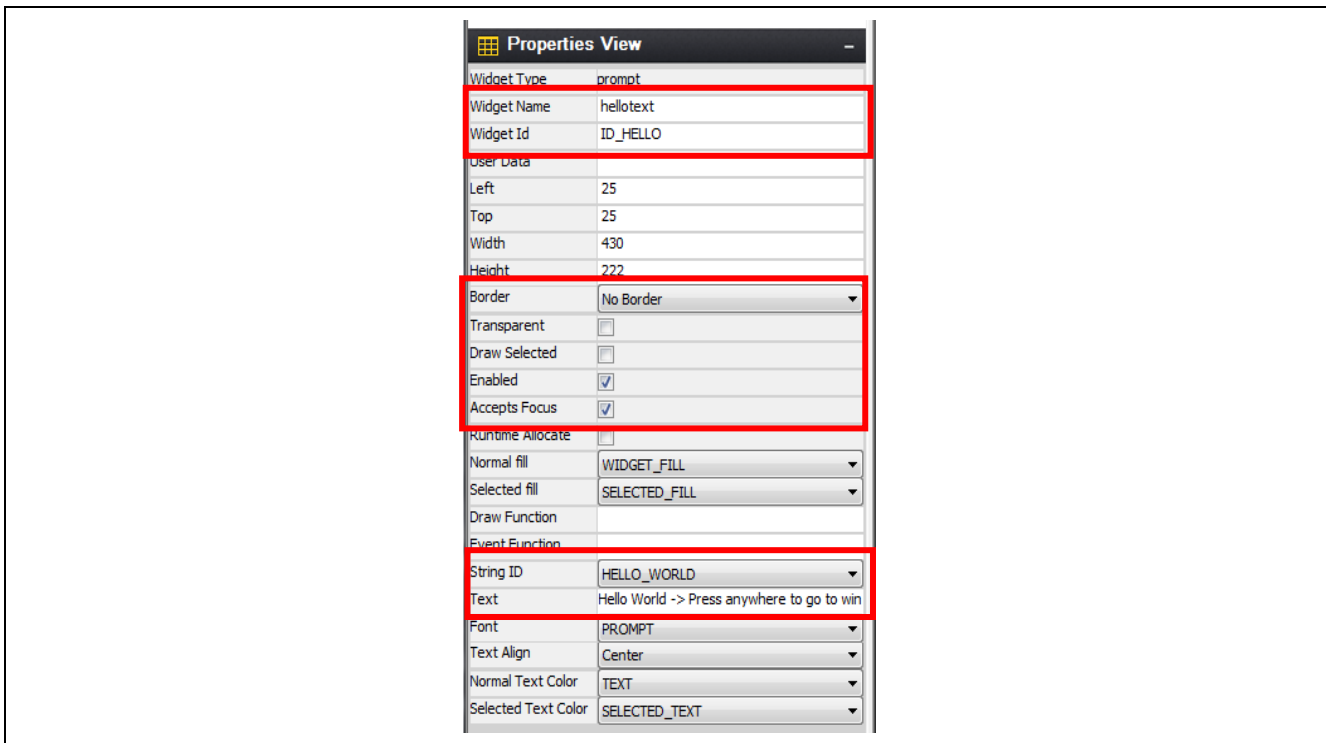
Figure 88.   Configure Hello Text Prompt Properties

37. In the **Project View** under **window2**, click **prompt_3**, then modify the properties to the following figure.
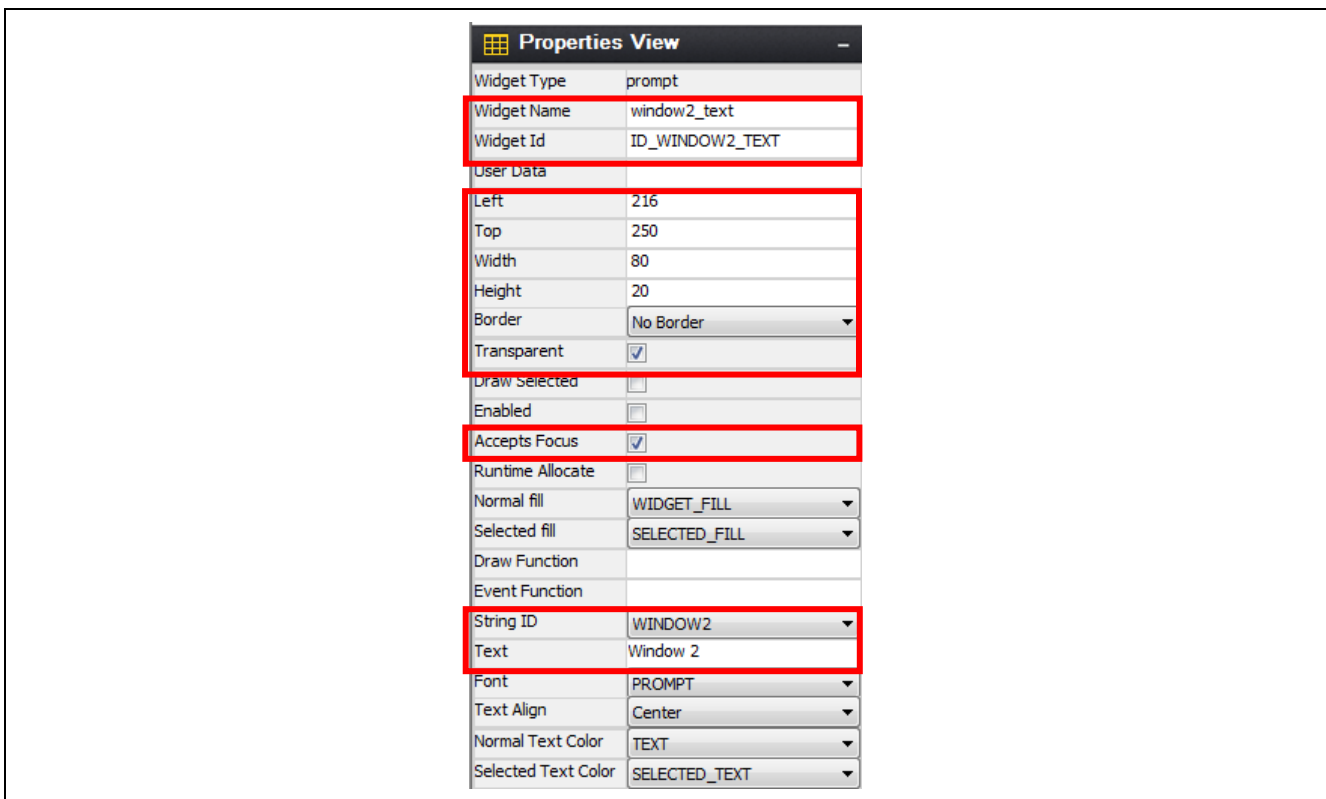
Figure 89.   Configure Window Text Properties

After these configuration steps, the two windows should look like Figure 90 and Figure 91.
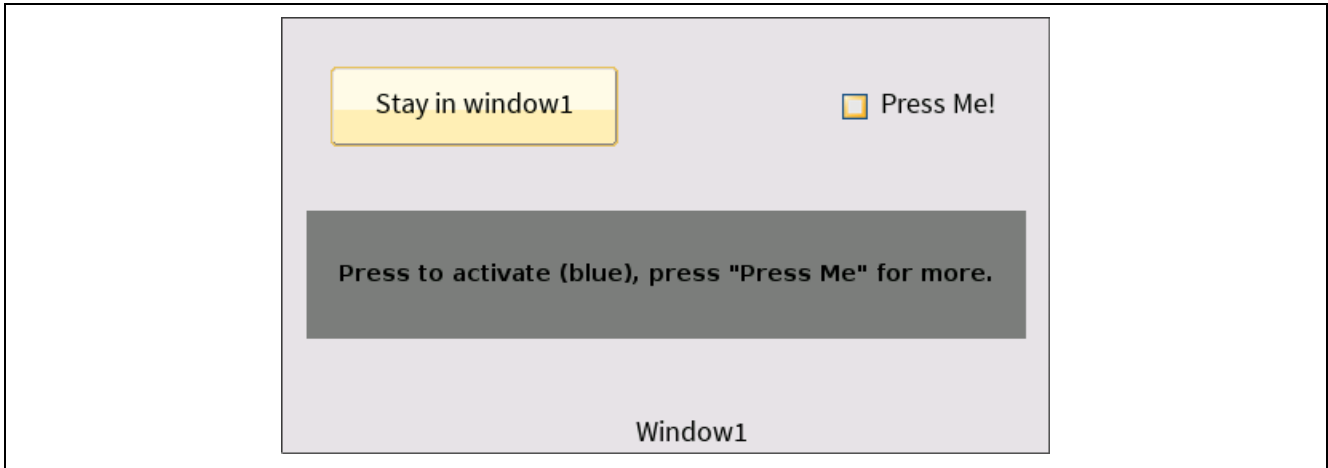
Figure 90.   Configured Window1
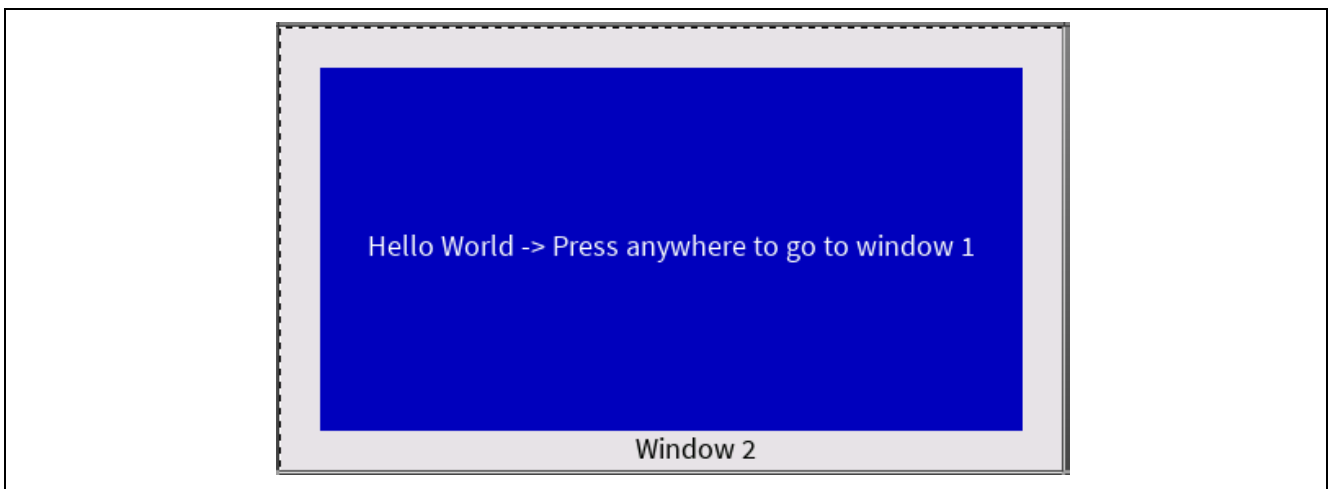
Figure 91.   Configured Window2

38. Expand the **Pixelmaps** section on the right by clicking the **+** symbol.
39. Click **System**.

Figure 92.   Configuration of Pixelmaps

40. Double-click **CHECKBOX_OFF** to edit the **Pixelmap**.
41. Deselect **Compress Output** and click **Save**.
42. Double-click **CHECKBOX_ON** to edit the **Pixelmap**.
43. Deselect **Compress Output** and click **Save**.
44. **Save** the project.
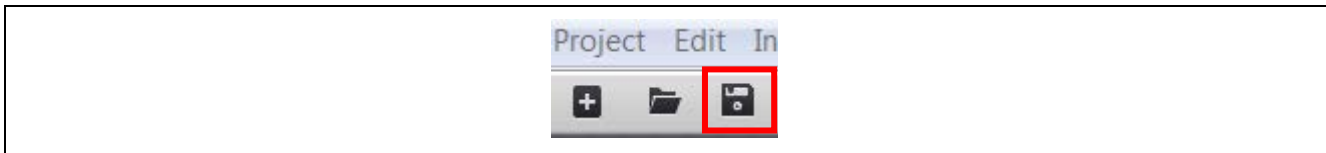
Figure 93.   Save Project

45. From the pull-down menu, select **Project > Generate all Output Files**.
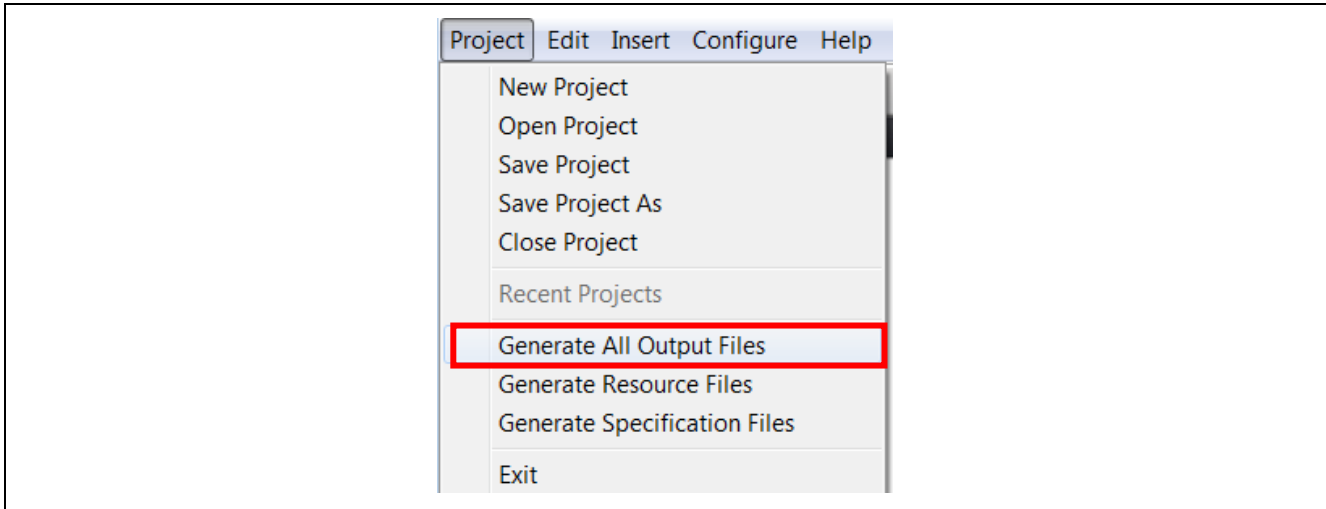
Figure 94.   Generate All Output files

46. Click on **Generate**.
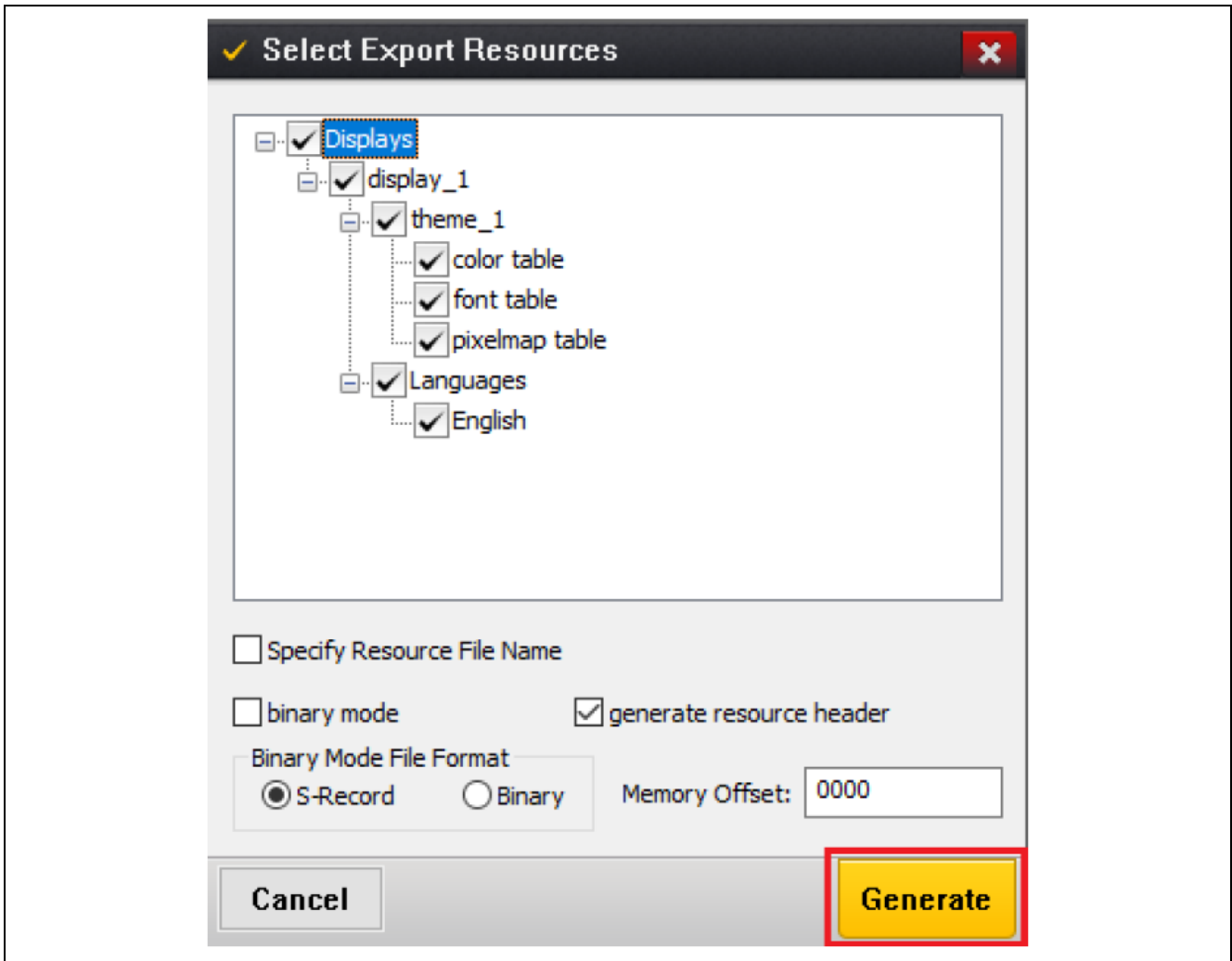
Figure 95.   Select Export Resources

47. Return to e[2] studio.

6.   Adding Code for Custom Interface Controls

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`. Drag the file from the **Windows Explorer** window into the **src** folder inside the e[2] 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 (button, checkbox, prompt).

Build the project by clicking the **Hammer icon**, below the **Menu Bar**. There should be no errors reported in the build output.
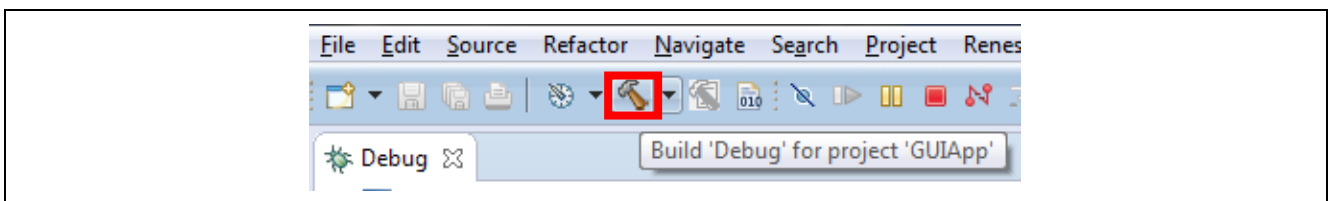
Figure 96   Build Button

Handlers can be found in the source file given with the application description. To add the handlers to your code, right click on **src**, then **System Explorer** as shown in the following figure.
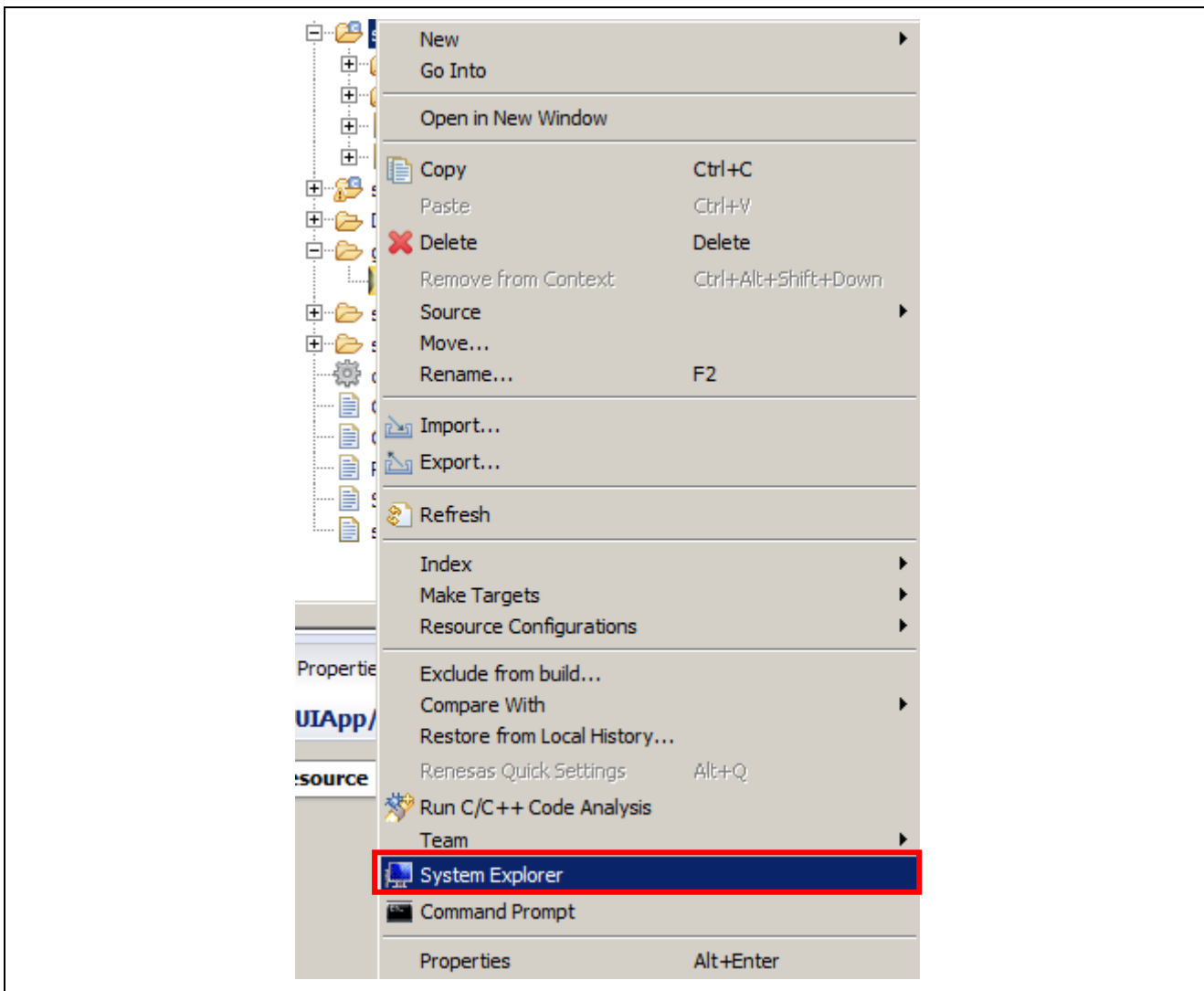
Figure 97.   Adding handlers

After you are done importing the file from the file source, you should have a structure similar to the following figure.
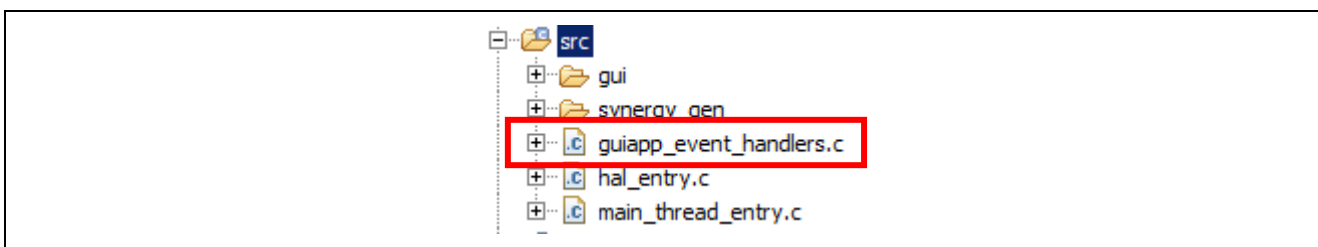
Figure 98.   Final src structure

GUIX handles events at a system level. To handle custom commands like screen transitions and button actions event handler need to be defined. The following event handler for window1 provides an example.

```
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_button_text_id(widget>gx_widget_parent, ID_WINDOWCHANGER,
GX_STRING_ID_BUTTON_ENABLED);
        update_promt_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_button_text_id(widget>gx_widget_parent, ID_WINDOWCHANGER,
GX_STRING_ID_BUTTON_DISABLED);
        update_promt_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 and the box will be checked.

7. Running the Application

1. On the DK-S7G2 Synergy MCU, perform the following steps:
   A. Set DIPSW S5 DRAM switch to on.
   B. Set DIPSW S5 EXP to off.
   C. Connect 5VDC Power to J1 (power plug).
   D. Connect the JLink-OB on J17 of the DK-S7G2 MCU main board to the PC using a micro USB cable.
   Note: The application is not yet ready to be run on the target hardware. The following steps are necessary to run it.

2. On the PC, click the dropdown menu for the debug icon.
3. Select the **Debug Configurations...** option.
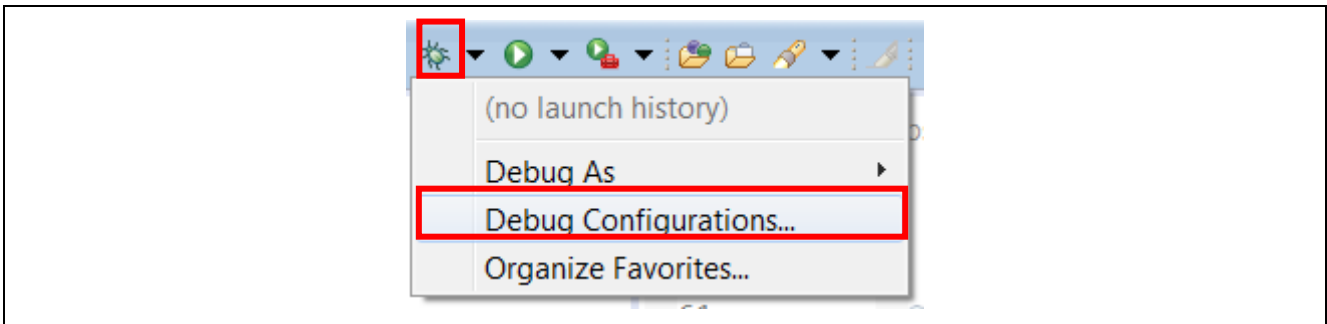
Figure 99.   Debug Options

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 an issue with the build. Check all steps from the document again for mismatched options.
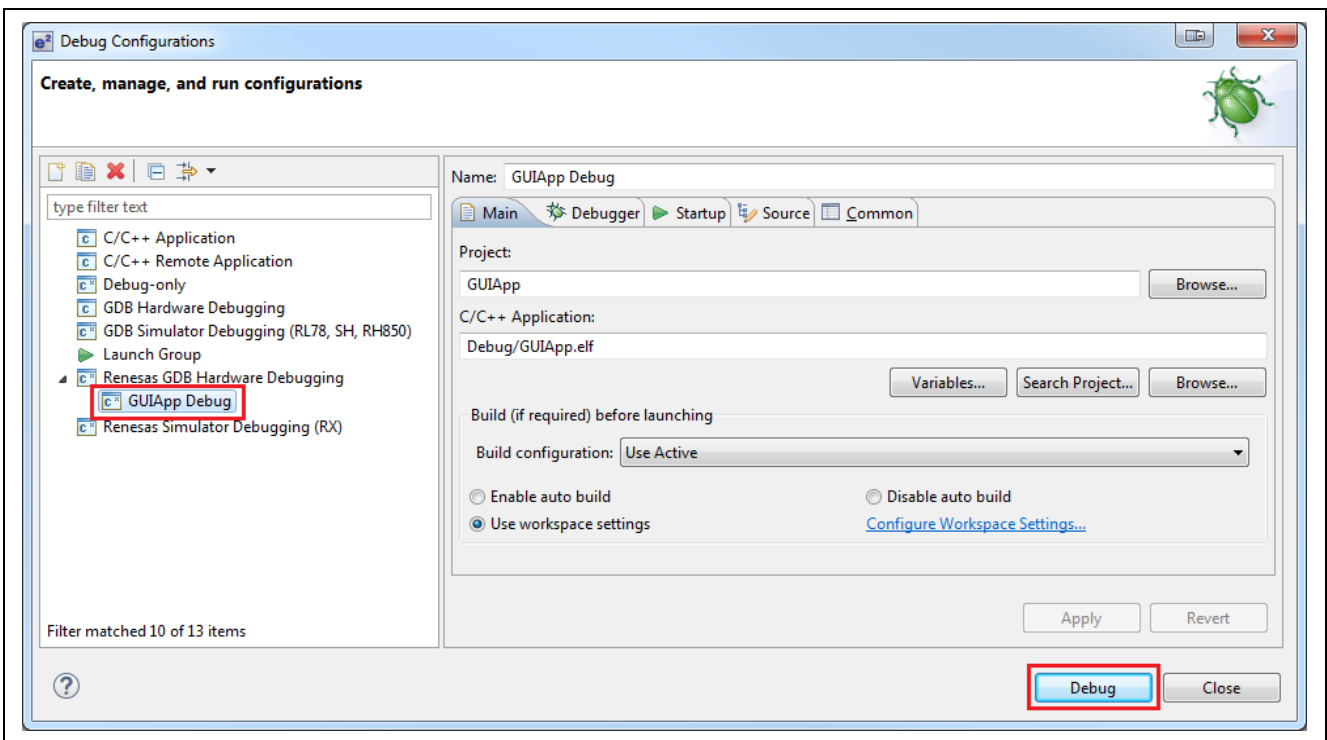
Figure 100.   Debug Configurations

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.)
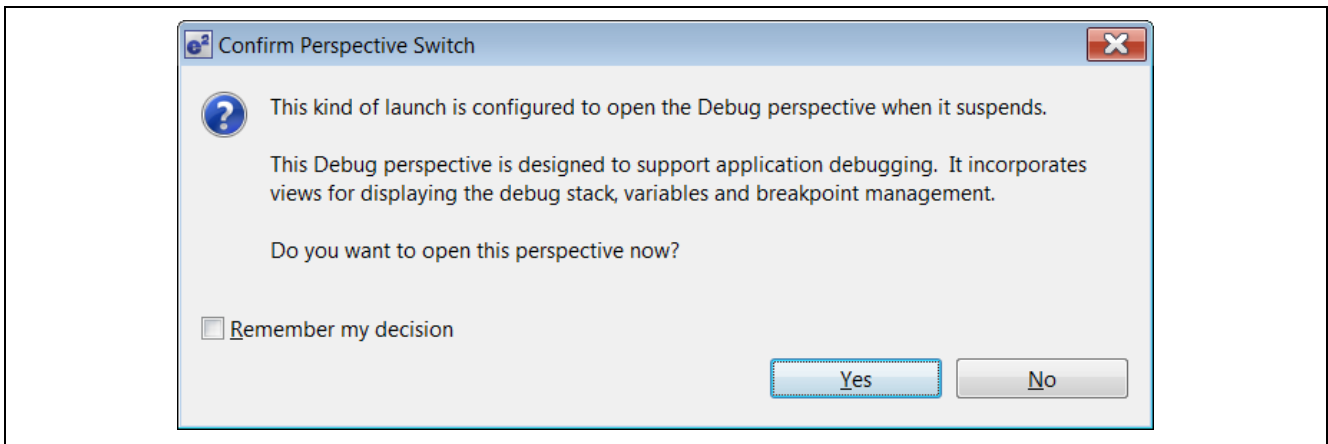
Figure 101.   Perspective Switch Dialog

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

Figure 109.   Resume Button

8. Press **F8** or the **resume button** to run the code.
   Note: The GUI created earlier should now be on the screen.
9. An overview of the Demo is as follows.
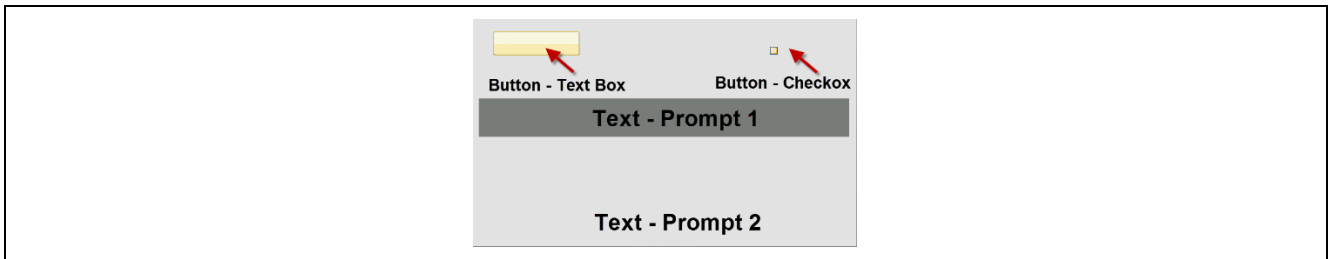
Figure 102.   Window1

A. Figure 102 shows **window1**. In this window are four elements:
   a. **Button – Text Box**: This box simply shows what the window does if you press outside the **Text – Prompt 1** area. (Refer to **Button – Checkbox** to see how it is changed.) Press in this area to activate the `window1 _handler` event that is picked up by `guiapp_event_handlers.c` where the code changes the window to window2.
   b. **Button – Checkbox**: This button is used to enable going to **window2**. Text is set to **Press Me!** and it is unchecked. When you press within the Checkbox active area you activate the event `window1_event_handler`. 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.
   c. **Text – Prompt 1**: This area instructs you how to control the demo. (Refer to **Button – Checkbox** to see how it is changed.)
   d. **Text – Prompt 2:** This Prompt is used to show you the window being displayed. It never changes (always shows **window1**).
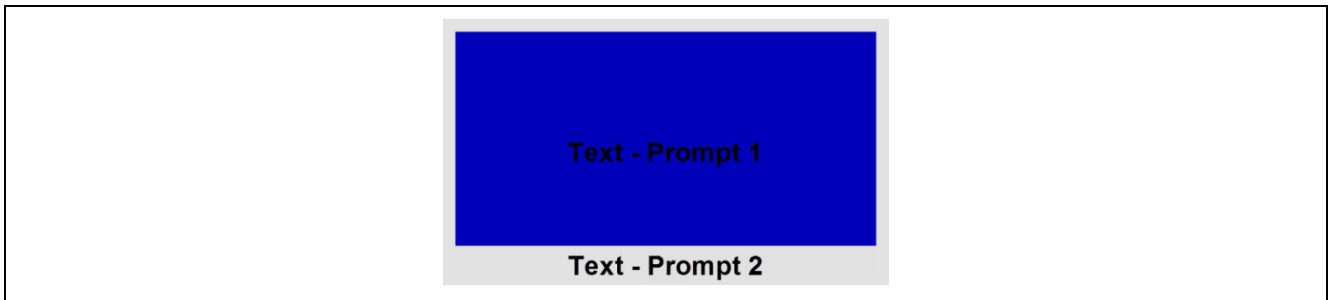
Figure 111. Window2

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

    a. **Text – Prompt 1**: This area presents **Hello World** and instructs you how to return to **window1**. Pressing in this area initiates the `window2_handler event` that is picked up by `guiapp_event_handlers.c` and changes the active window to **window1**.

    b. **Text – Prompt 2:** This prompt is used to show you the window being displayed. It never changes (always shows **window2**).

10. Press **Ctrl + F2** or the **Stop** button to end the debug session.
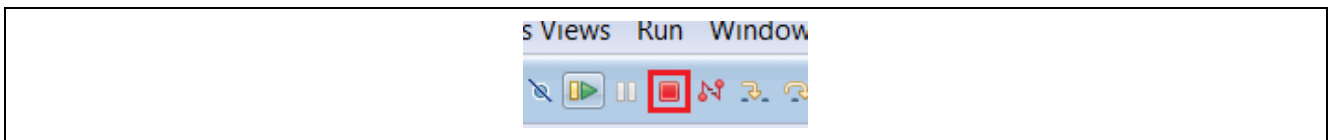
Figure 112. Stop Button

11. This concludes the **GUIX Hello World** for the DK-S7G2 Synergy MCU.

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) about using QSPI for storing the image resource files.

Note: Users are required to make the QSPI pins drive capacity to High instead of Low when QSPI is used for external storage (On DK-S7G2 Board).

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.22.15	All	Created Initial Document
1.01	Jul.06.16	All	Updated for SSP v1.1.0
1.02	May.15.17	All	Updated for SSP v1.2.1
1.03	Aug.11.17	All	Updated for SSP v1.3.0
1.04	Nov.13.17	All	Updated for SSP v1.3.1
1.05	Feb.27.18	All	Updated for SSP v1.4.0
1.06	Jun.18.18	—	Sample codes updated.
1.07	Oct.10.18	—	Updated for SSP v1.5.0
1.08	Mar.08.19	—	Updated for SSP v1.6.0
1.09	Oct.15.21	—	Updated for SSP v1.6.0 "Touch Panel V2 Framework"
1.10	Nov.10.21	—	Updated for SSP v1.6.0 to SSP v2.1.0
1.11	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)