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 e^2 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 e^2 studio Integrated Solution Development Environment (ISDE), v7.3.0 or later
- Synergy Software Package (SSP) v1.6.0 or later
- GUIX Studio v5.4.1.1 or later
- IAR Embedded Workbench® for Renesas Synergy™ v8.23.3 and SSC v7.3.0 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 e^2 studio, where you configure the drivers and framework included with the SSP. Project setup in e^2 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, 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 people 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* (r11an0023eu0121-synergy-ssp-import-guide.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.

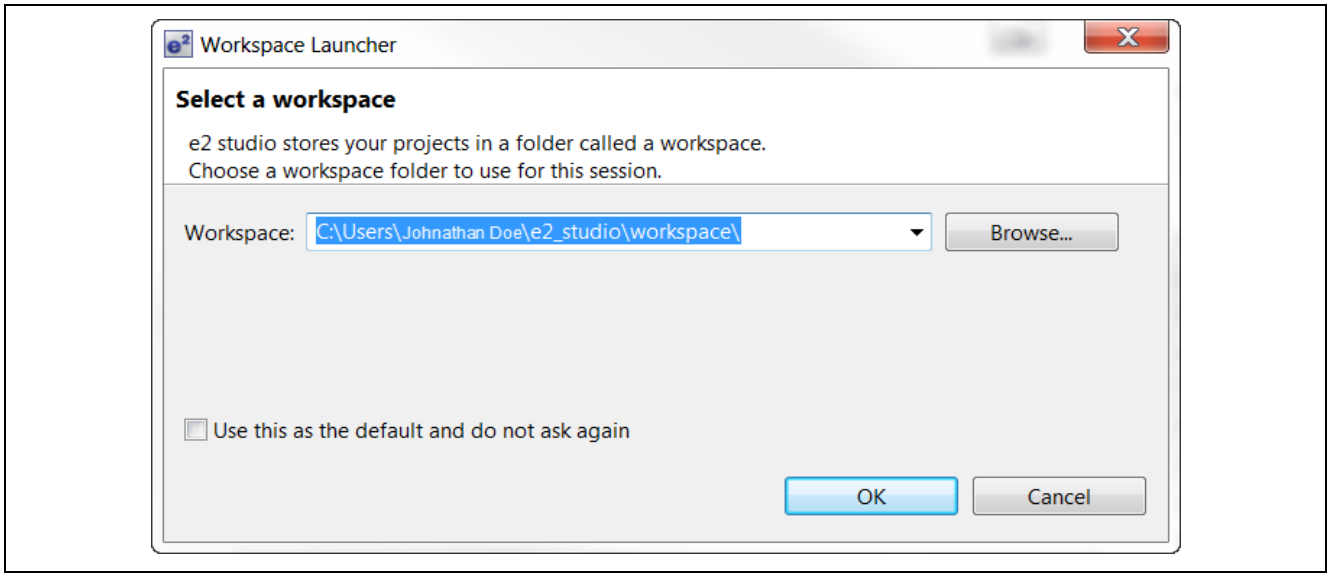

Figure 1. Workspace Launcher Dialog

3. Create a new workspace:
   1. 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.

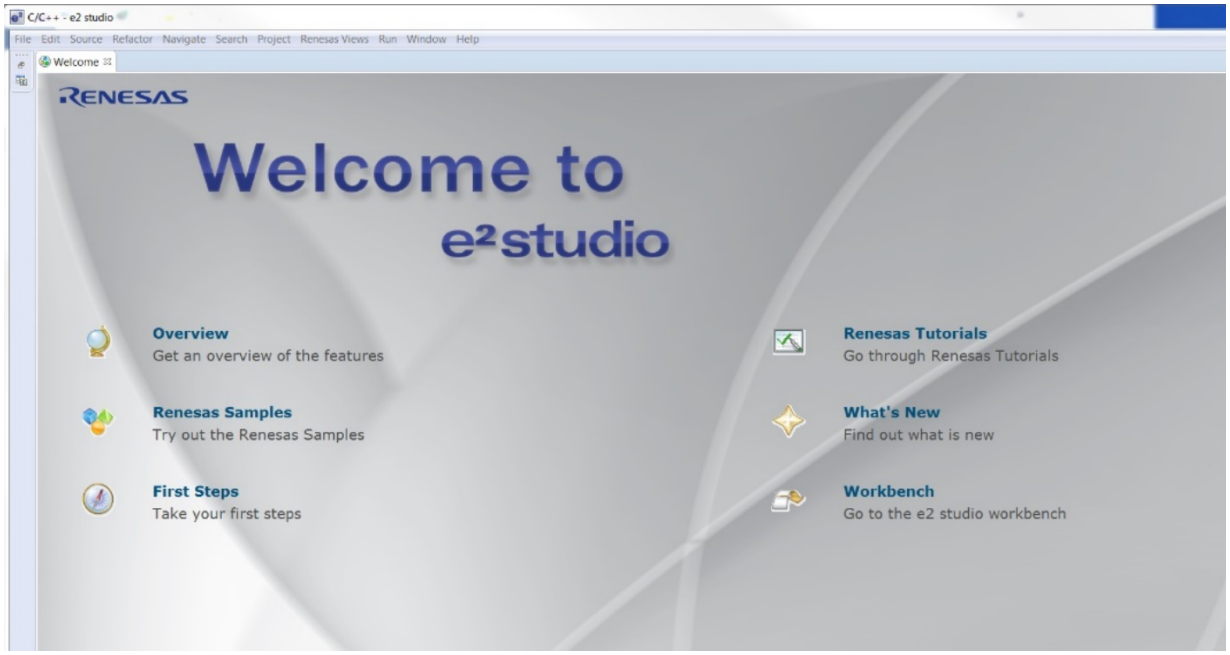

Figure 2. Close the Welcome Window by Clicking in the Workbench Area

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

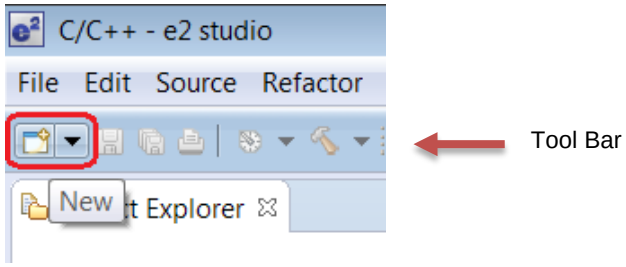

Figure 3. Start a New Project

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

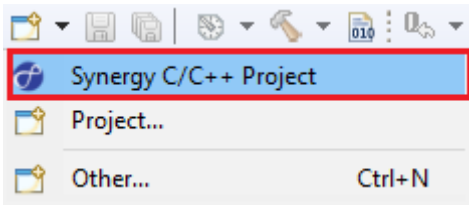

Figure 4. Select Synergy C/C++ Project in the Drop-down Menu

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

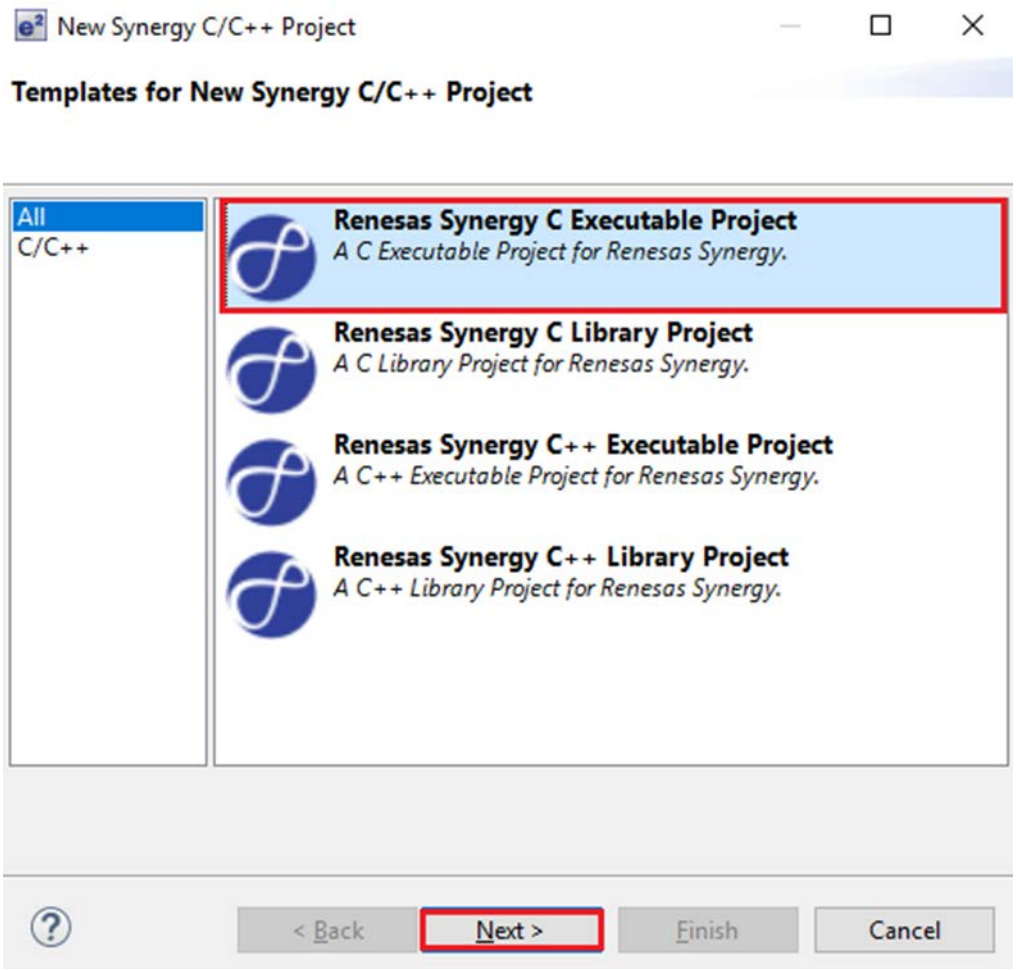

Figure 5. Project Type Selection

10. If the License file is configured, you see this area of the form.

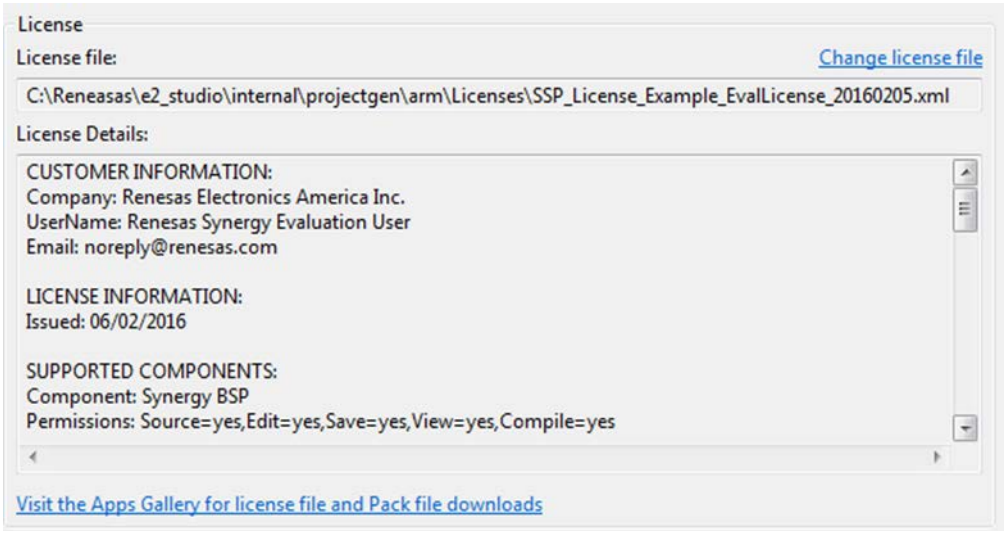

Figure 6. Configured License File

A. Click the **Change license file button.** e² studio will display the Preferences dialog box.

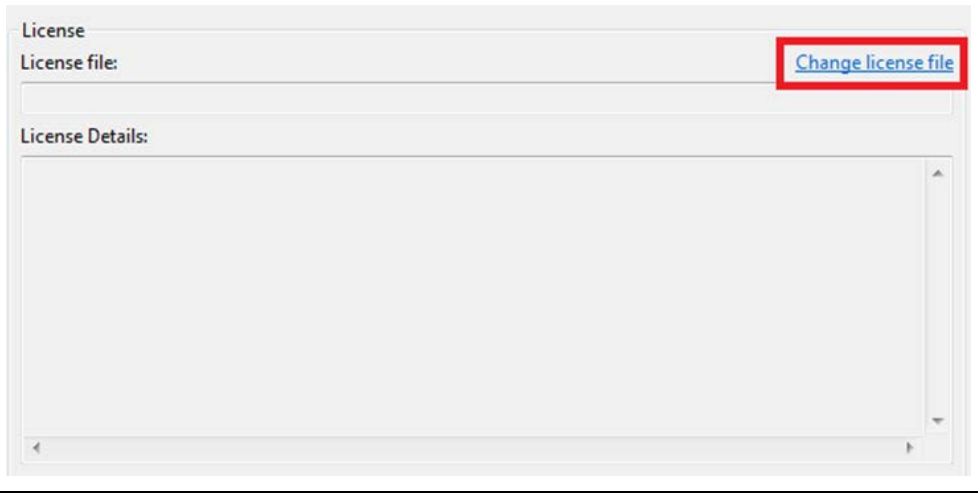

Figure 7. Unconfigured License File

B. Click the Browse […] button to open the **Specify Synergy License** dialog box.

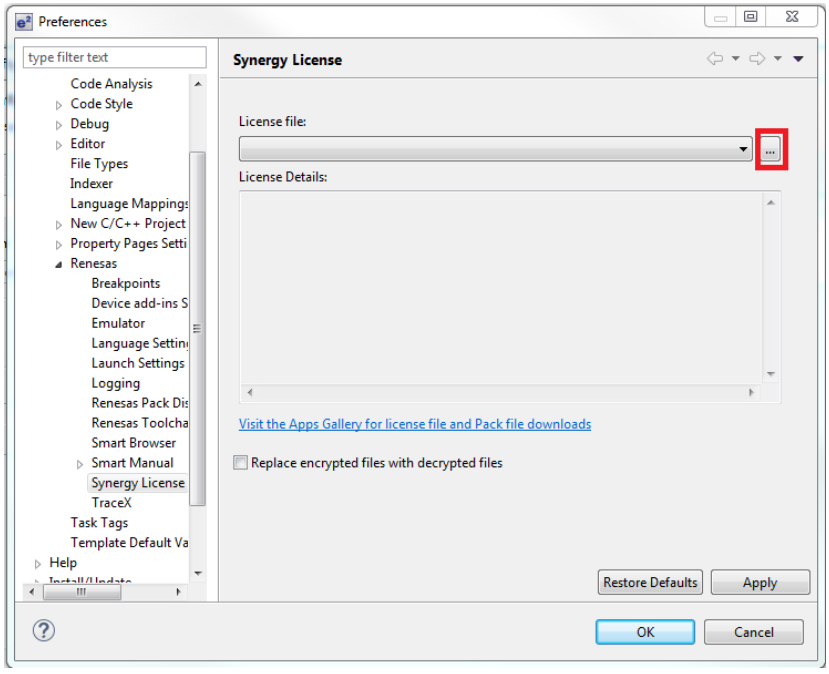

Figure 8. Preferences Dialog Box with Synergy License Configuration

C. Click the **Browse…** button.

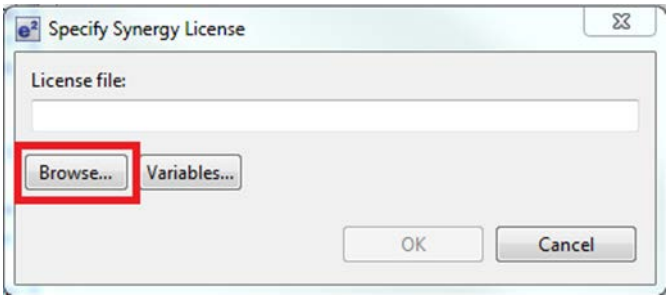

Figure 9. Synergy License Dialog Box

The e² studio **Open Dialog** box and Licenses directory displays.

Note: If you installed e² studio into the default location, the license file will be located in: **C:\Renesas\e2_studio\internal\projectgen\arm\Licenses** directory.

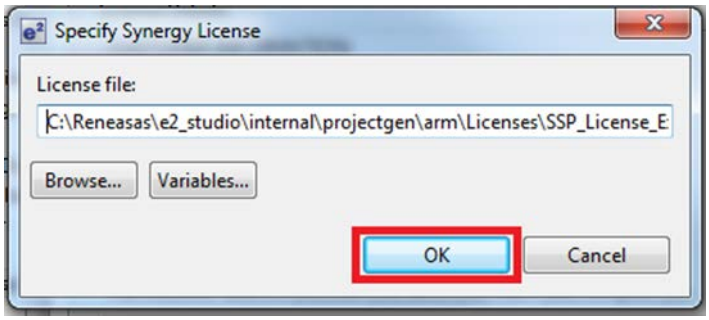

Figure 10. Confirm License File

D. Select SSP_License_Example_EvalLicense_*.xml located in the directory.
E. Click Open to select the License file.
F. Click OK to set the license and close the dialog.
G. Click Apply and then OK in the Preferences Dialog box.

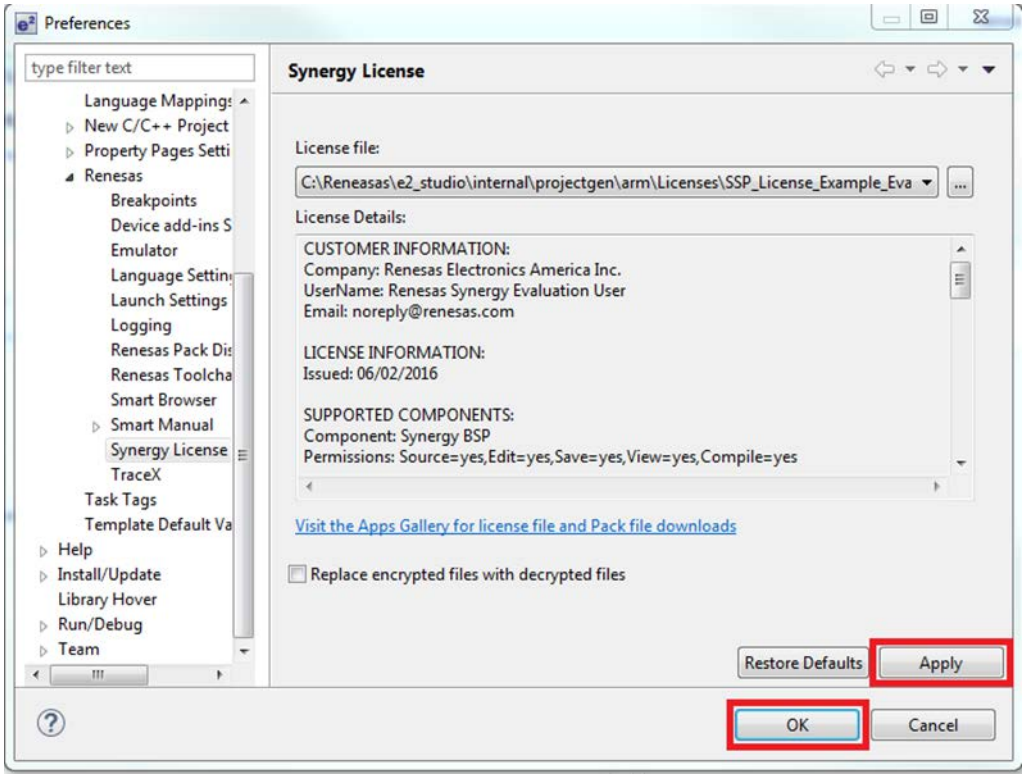

Figure 11. Apply and Confirm Synergy License File Selection

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

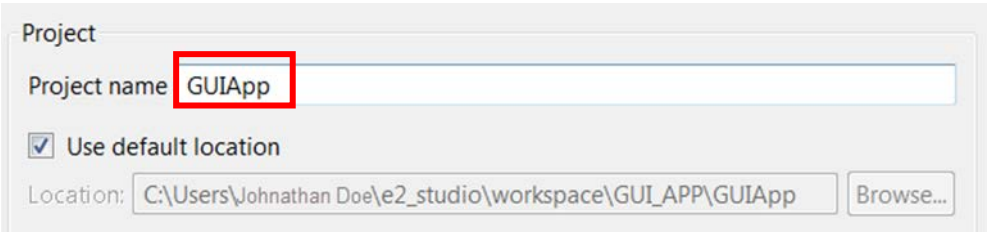

Figure 12. Enter a Project Name

12. On the top right of this page, verify that the Toolchain option is set to GCC ARM Embedded.

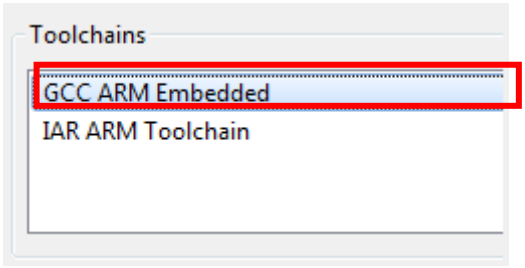

Figure 13. Verify GCC ARM Embedded Toolchain

13. Click the **Next** button to continue.
14. Under **Device Selection** (top left), select SSP version **1.5.0** (or later).
15. For the Board field, select **S7G2 DK**. The Device field updates automatically.

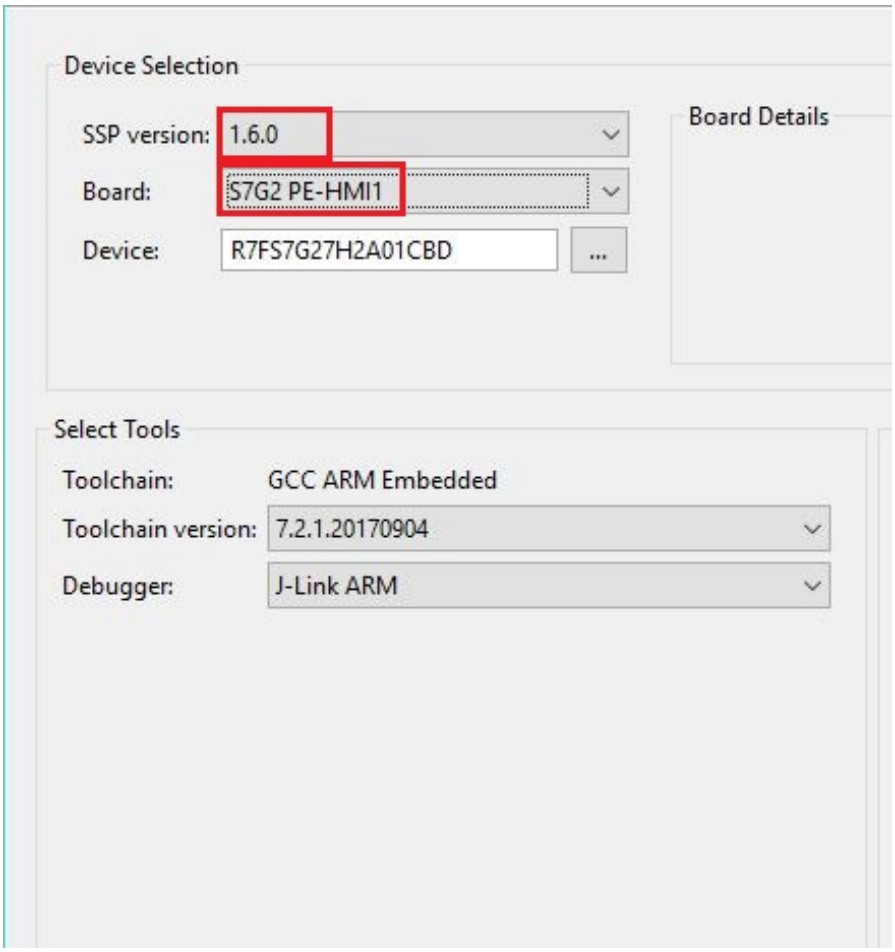

Figure 14. Device Selection

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

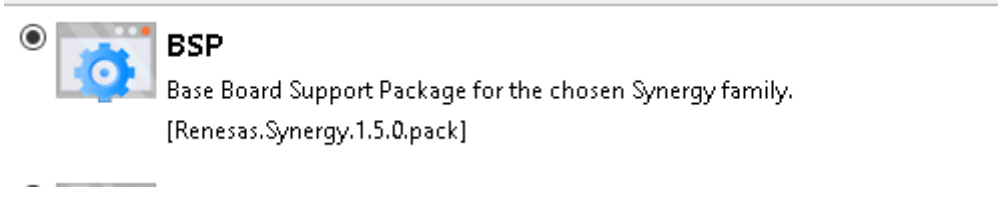

Figure 15. Select the BSP

18. Click the Finish button.
19. 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.

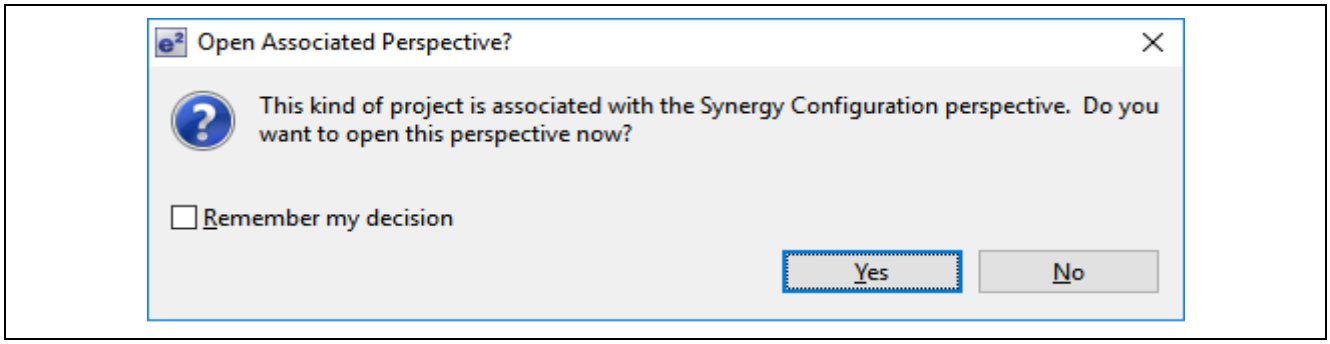

Figure 16. Open Perspective Dialog Box

When e² studio has finished creating the project, the following screen displays.

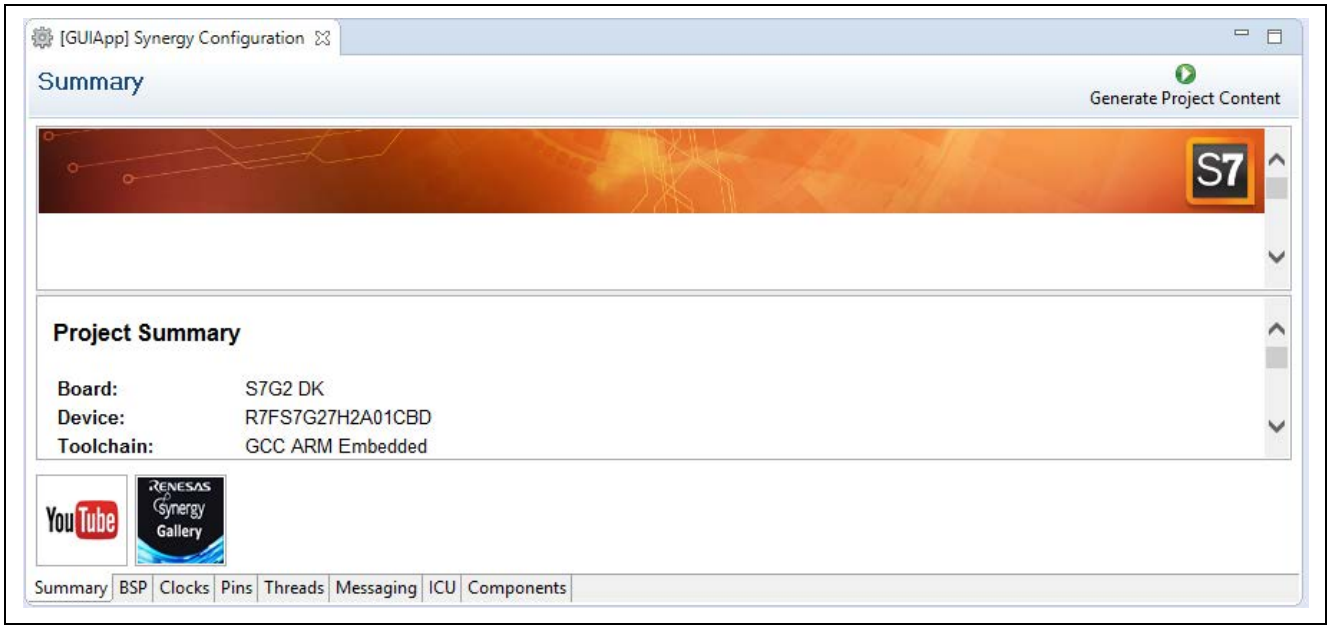

Figure 17. GUIApp Project

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**.

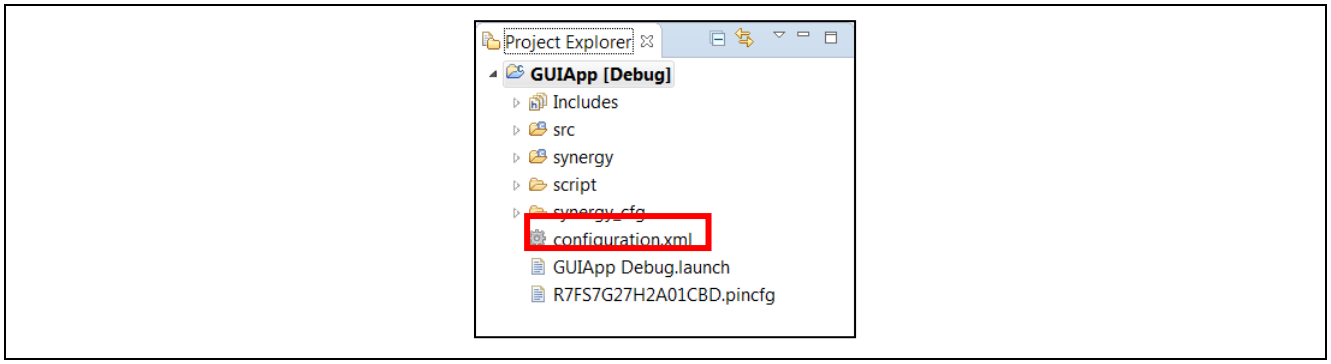

Figure 18. Selecting the Configuration.xml File in Project Explorer

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

Figure 19. Synergy Configuration Threads Tab

3. Select the **HAL/Common thread**.

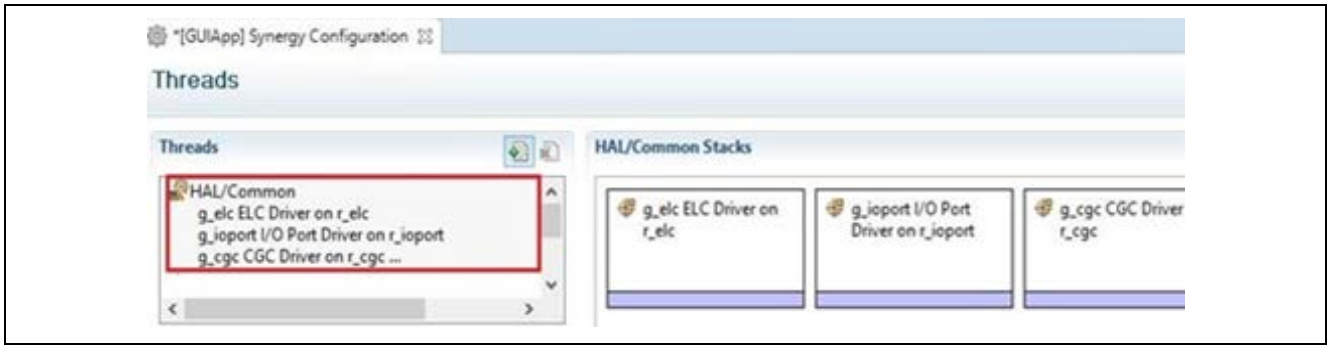

Figure 20. Threads

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

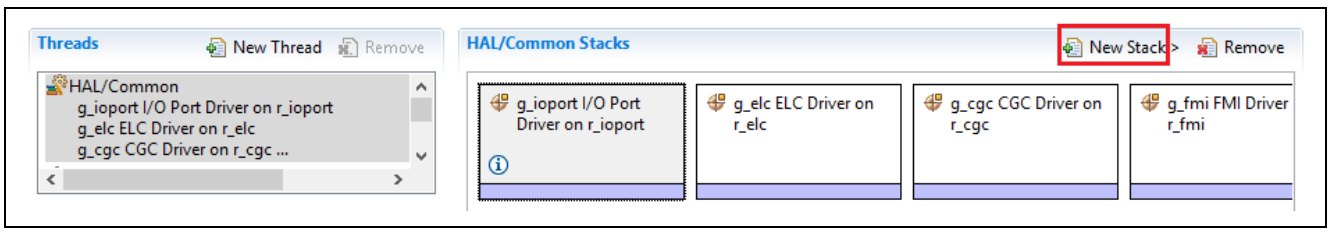

Figure 21. Add a Timer Driver Module to the HAL/Common Thread Part 1

5. In the **Menu** select **Driver** -> **Timers** -> **Timer Driver on r_gpt**.

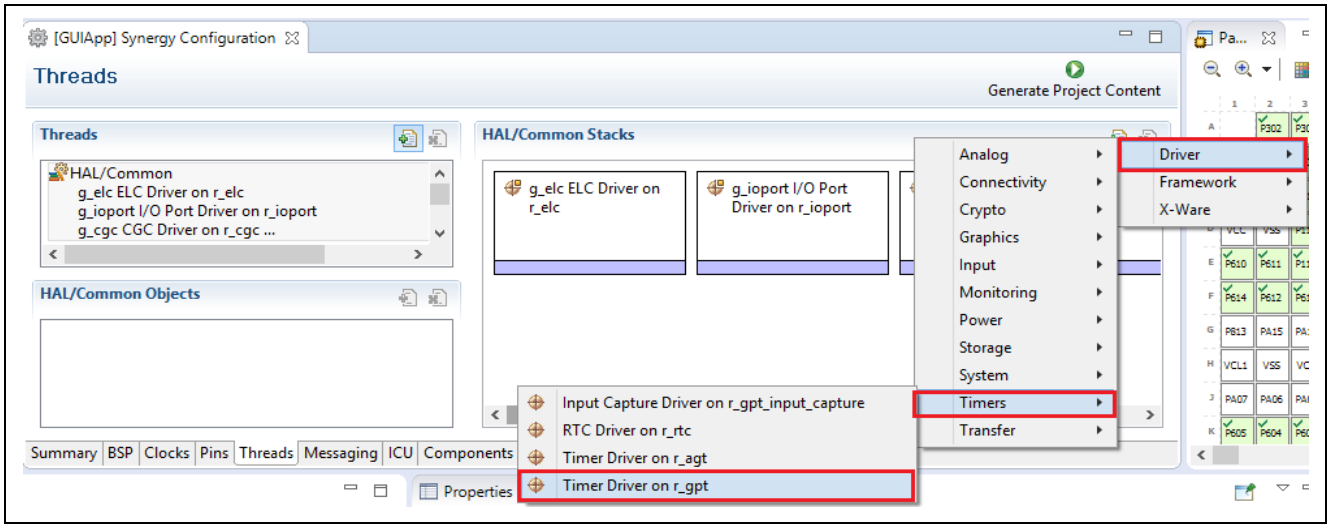

Figure 22. Add a Timer Driver Module to the HAL/Common Thread Part 2

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

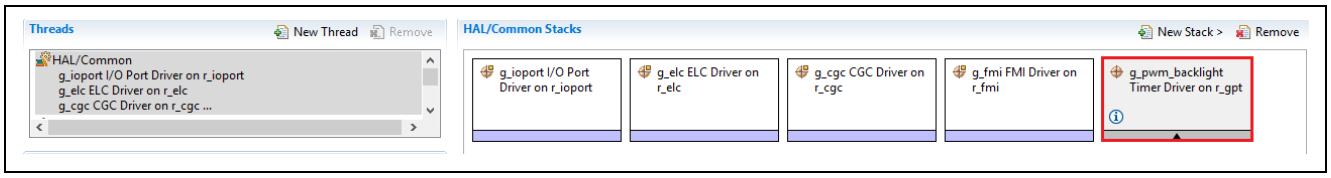

Figure 23. Select the Newly Created Timer Driver Module

7. In the **Properties Window**, change the **Properties** to match those in Figure 24. Hint: Change the channel to 2 first!

Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_pwm_backlight Timer Driver	
Name	g_pwm_backlight
Channel	2
Mode	PWM
Period Value	10
Period Unit	Milliseconds
Duty Cycle Value	100
Duty Cycle Unit	Unit Percent
Auto Start	True
GTIOCA Output Enabled	False
GTIOCA Stop Level	Pin Level Low
GTIOCB Output Enabled	True
GTIOCB Stop Level	Pin Level Low
Callback	NULL
Interrupt Priority	Disabled

Figure 24. 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 I²C 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.

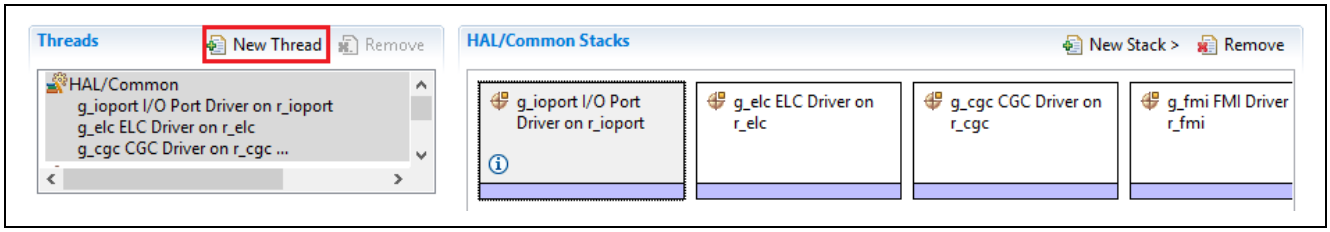

Figure 25. Creating a New Thread

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

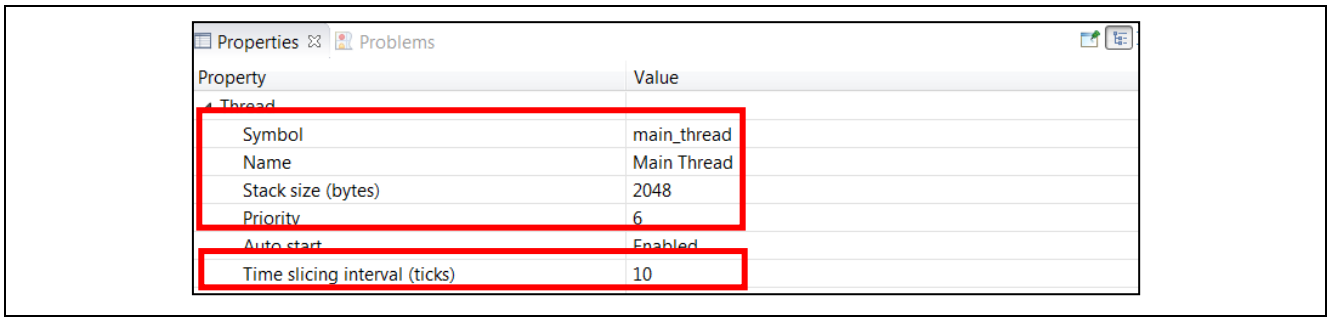

Figure 26. Configuring the Main Thread Properties

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.

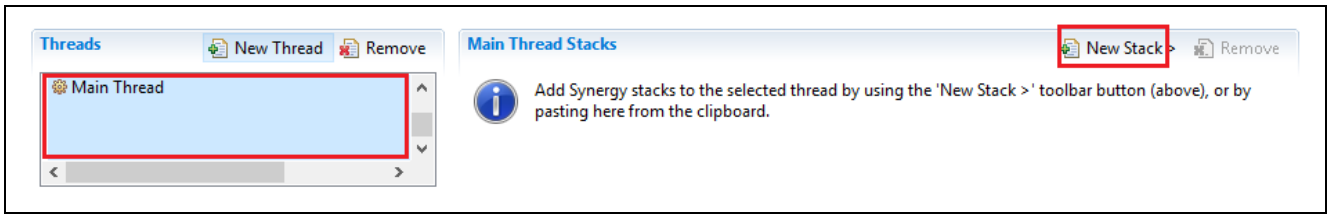

Figure 27 Main Thread Stacks

12. Add a framework for the touch panel by selecting **New,** then **Framework > Input > Touch Panel Framework** on **sf_touch_panel_i2c**.

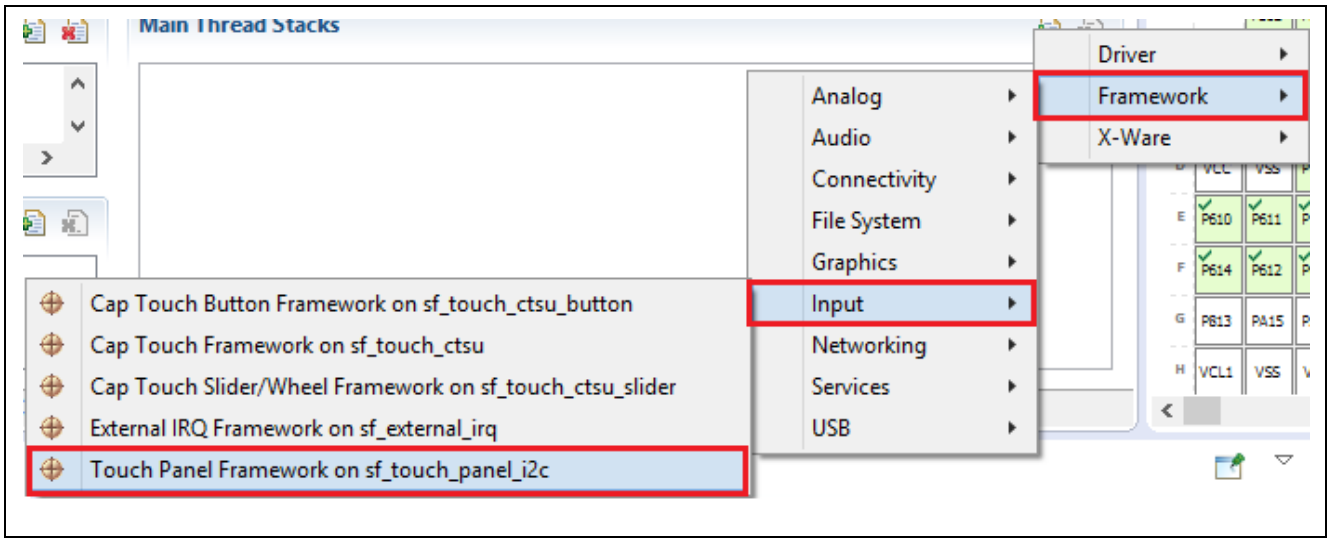

Figure 28. Adding the Touch Panel Framework

13. Configure the Touch Panel **Properties** as shown.

Property	Value
◢ Common	
Parameter Checking	Default (BSP)
◢ Module g_sf_touch_panel_i2c0 Touch Panel Framew	
Name	g_sf_touch_panel_i2c
Thread Priority	8
Hsize Pixels	480
Vsize Pixels	272
Update Hz	10
Reset Pin	IOPORT_PORT_07_PIN_11
Touch Event Class Instance Number	0
Touch Coordinate Rotation Angle(Clockwise)	0
Name of generated initialization function	sf_touch_panel_i2c_init0
Auto Initialization	Enable

Figure 29. Configuring Touch Panel Properties

The Synergy Configurator has created the message framework, external IRQ framework and driver, and has a placeholder for the I²C driver and Touch driver.
The messaging framework is used by other framework layers and tasks to pass messages around the system. This system passes data from the touch screen driver to the **Main Task** to handle touch inputs. Figure 30 shows the **SF External Interrupt** is a framework layer used by the **Touch Controller Driver**.

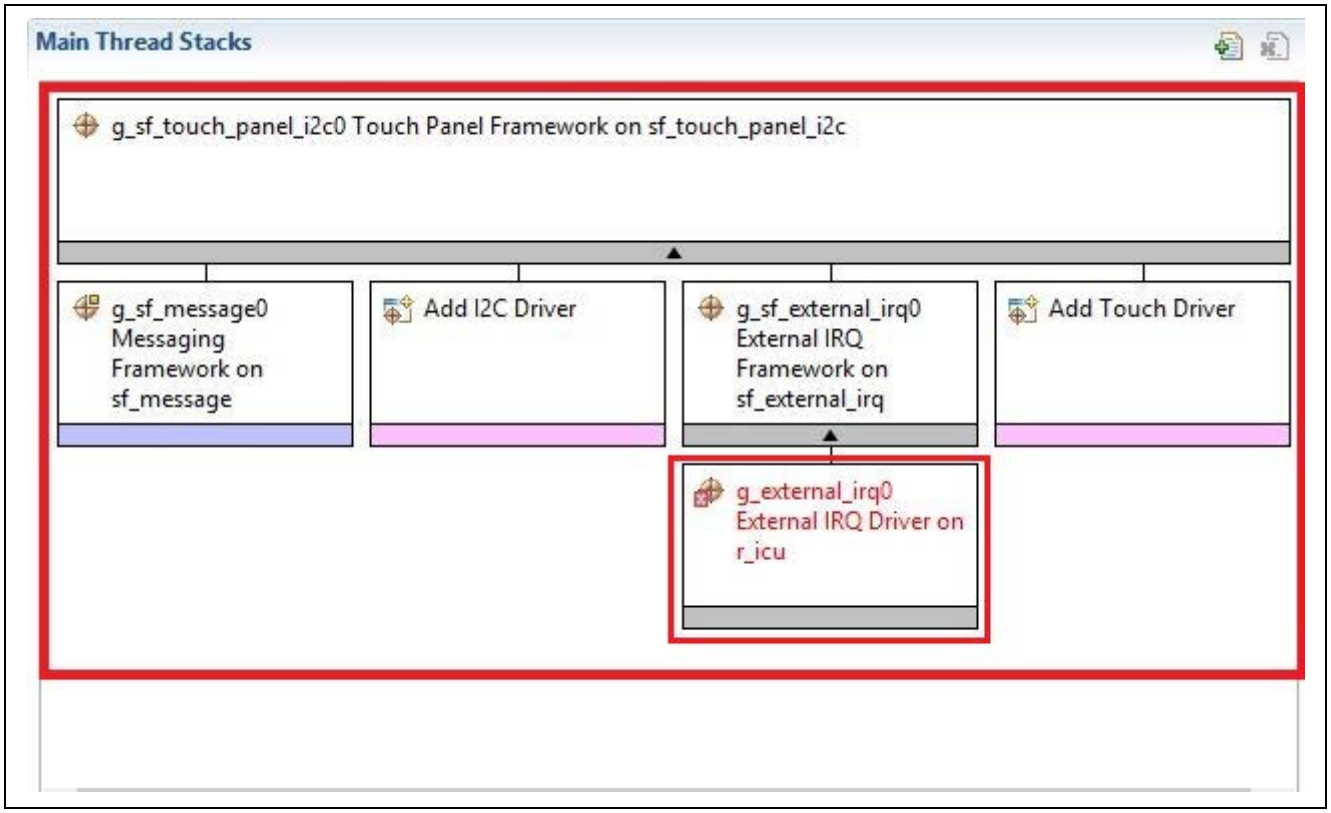

Figure 30. Touch Panel Framework Stack

14. Select the **External IRQ Framework** on **sf_external_irq** and configure the **Properties**.

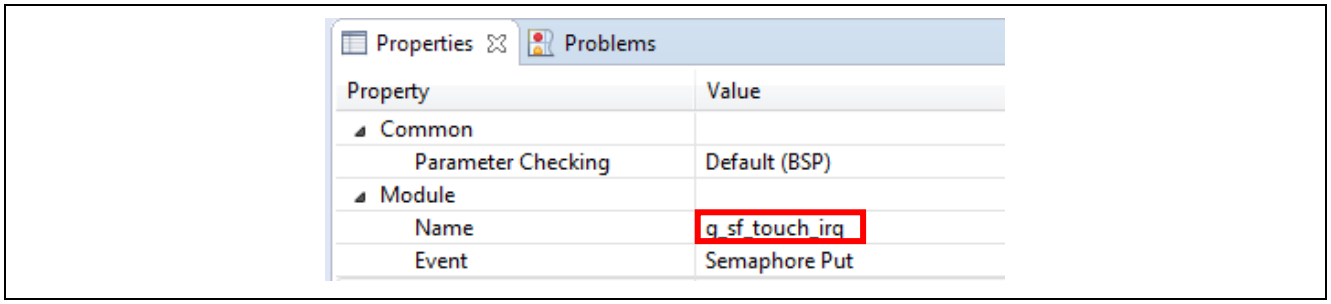

Figure 31. Configuring External Interrupts Properties

15. Select **External IRQ Driver** on **r_icu**. Configure the properties for the new module as shown. Hint: Change the **Channel** first!

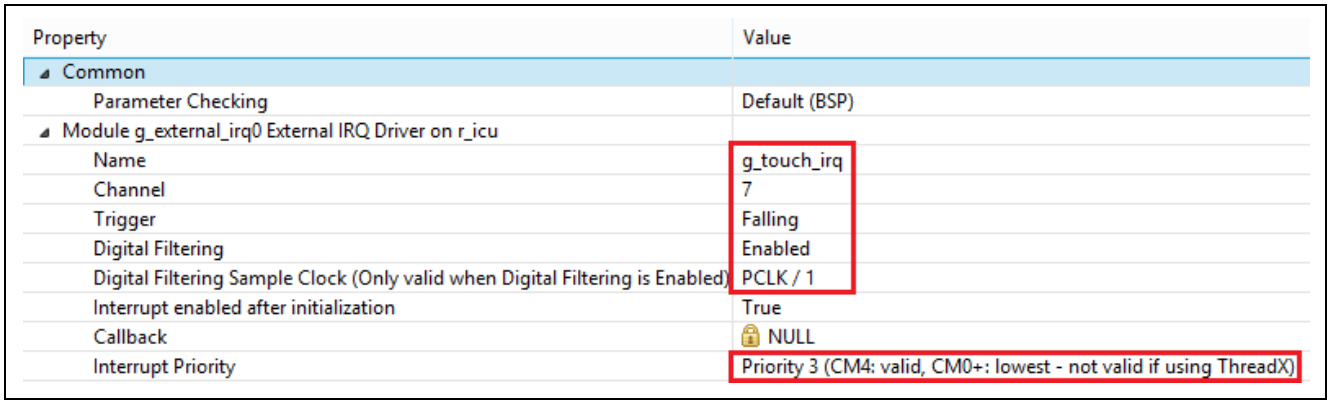

Figure 32. Touch Screen IRQ Properties

16. In the **Synergy Configuration Window** > **Threads tab** > **Main Thread Stacks** area, click **Add I2C Driver** and then select **New -> I2C Master Driver** on **r_sci_i2c**.

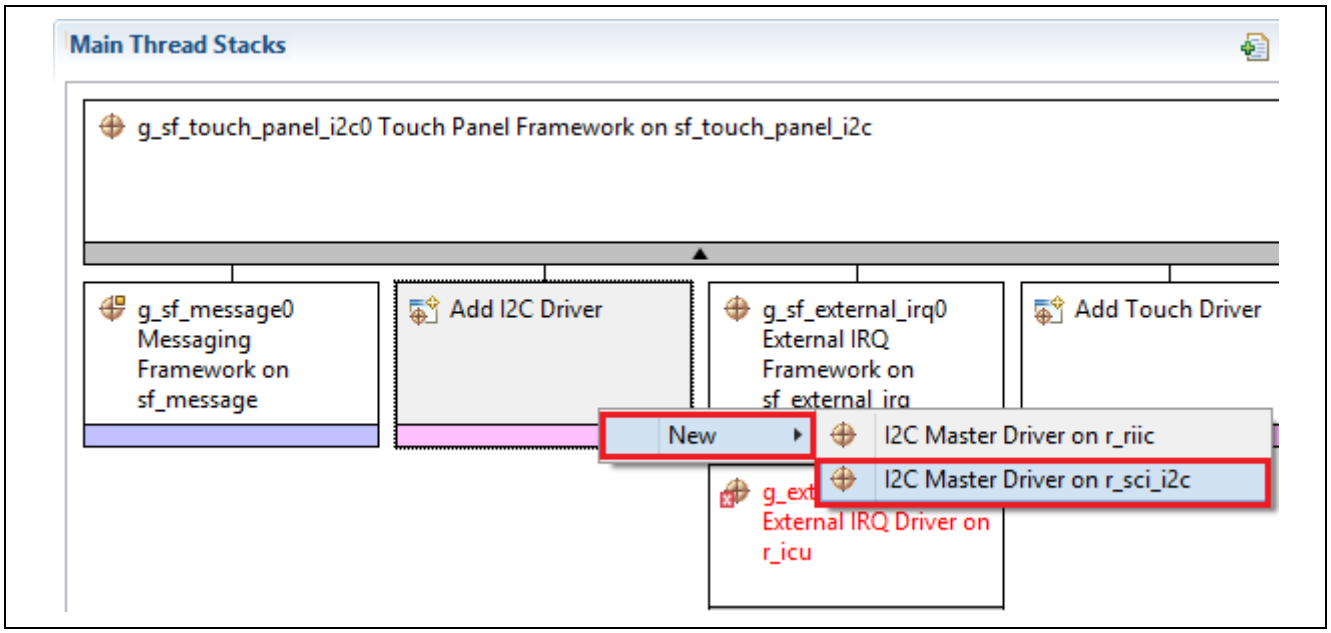

Figure 33. I2C Driver on SCI

17. Configure the properties for I²C Driver on **r_sci_i2c** as shown. Hint: Change the **Channel** first!

Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_i2c0 I2C Master Driver on r_sci_i2c	
Name	g_i2c
Channel	7
Rate	Standard
Slave Address	0x48
Address Mode	7-Bit
SDA Output Delay (nano seconds)	300
Bit Rate Modulation Enable	Enable
Callback	NULL
Receive Interrupt Priority	Priority 3 (CM4: valid, CM0+: lowest - not valid if using ThreadX)
Transmit Interrupt Priority	Priority 3 (CM4: valid, CM0+: lowest - not valid if using ThreadX)
Transmit End Interrupt Priority	Priority 3 (CM4: valid, CM0+: lowest - not valid if using ThreadX)

Figure 34. I²C on SCI Properties

18. In the **Synergy Configuration Window** > **Threads tab** > **Main Thread Stacks** area, click **Add Touch Driver** and then select **New -> Touch Panel Driver on touch_panel_sx8654**.

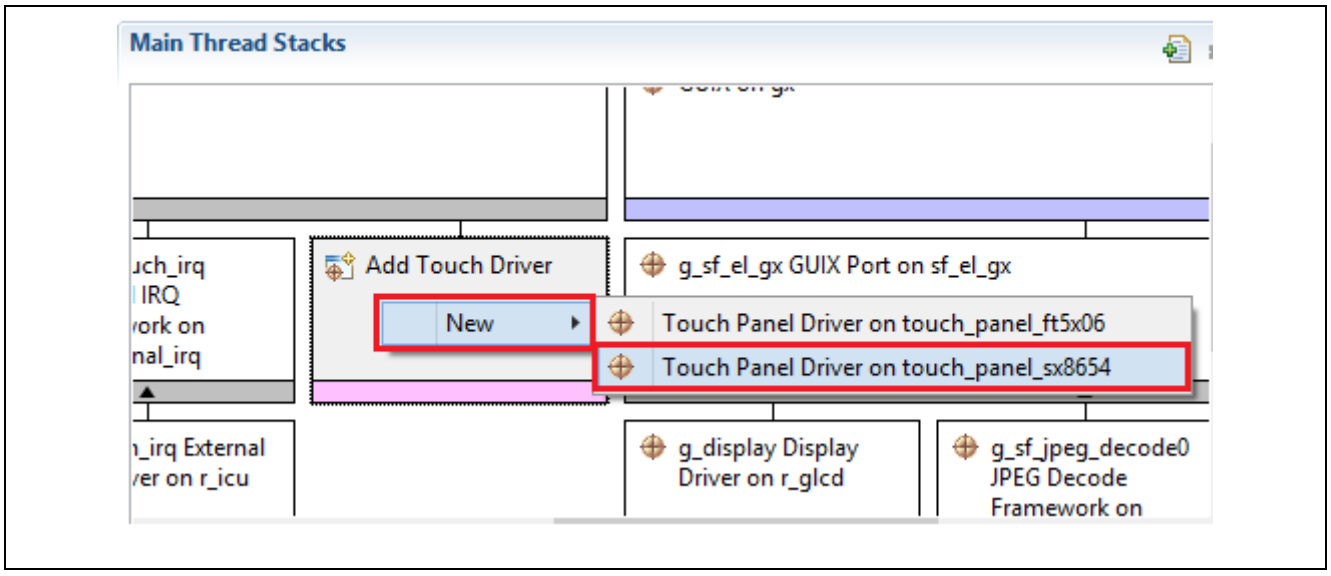

Figure 35. Adding Touch Driver

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

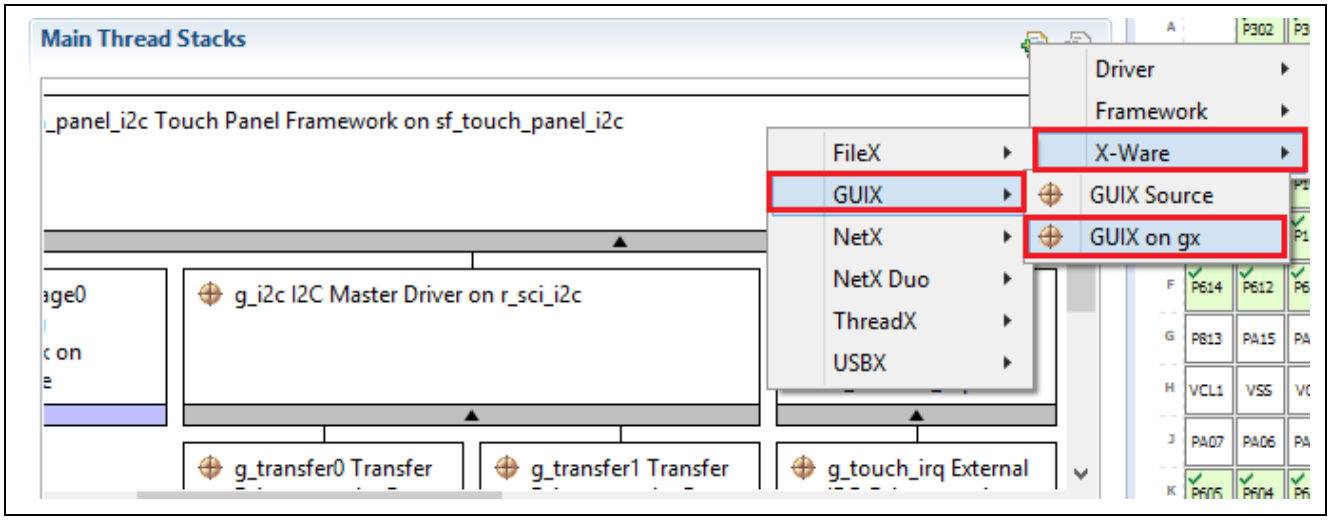

Figure 36. GUIX on gx

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.

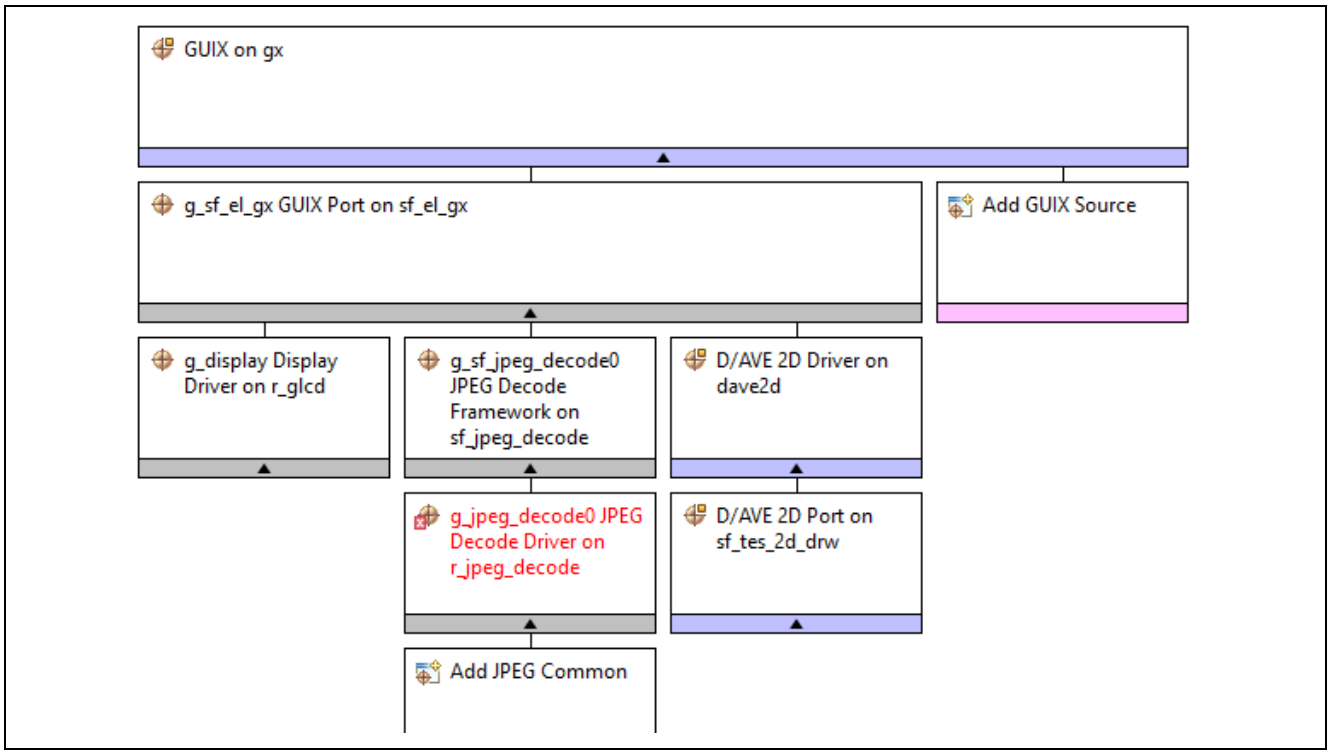

Figure 37. GUIX on gx

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

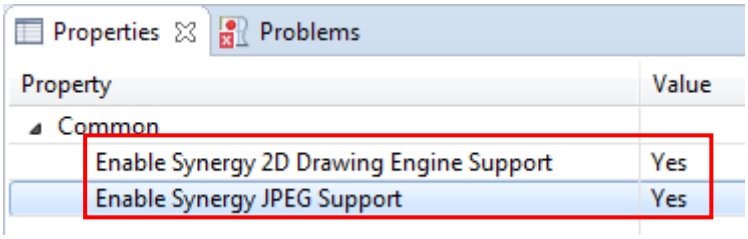

Figure 38. GUIX on gx Properties

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

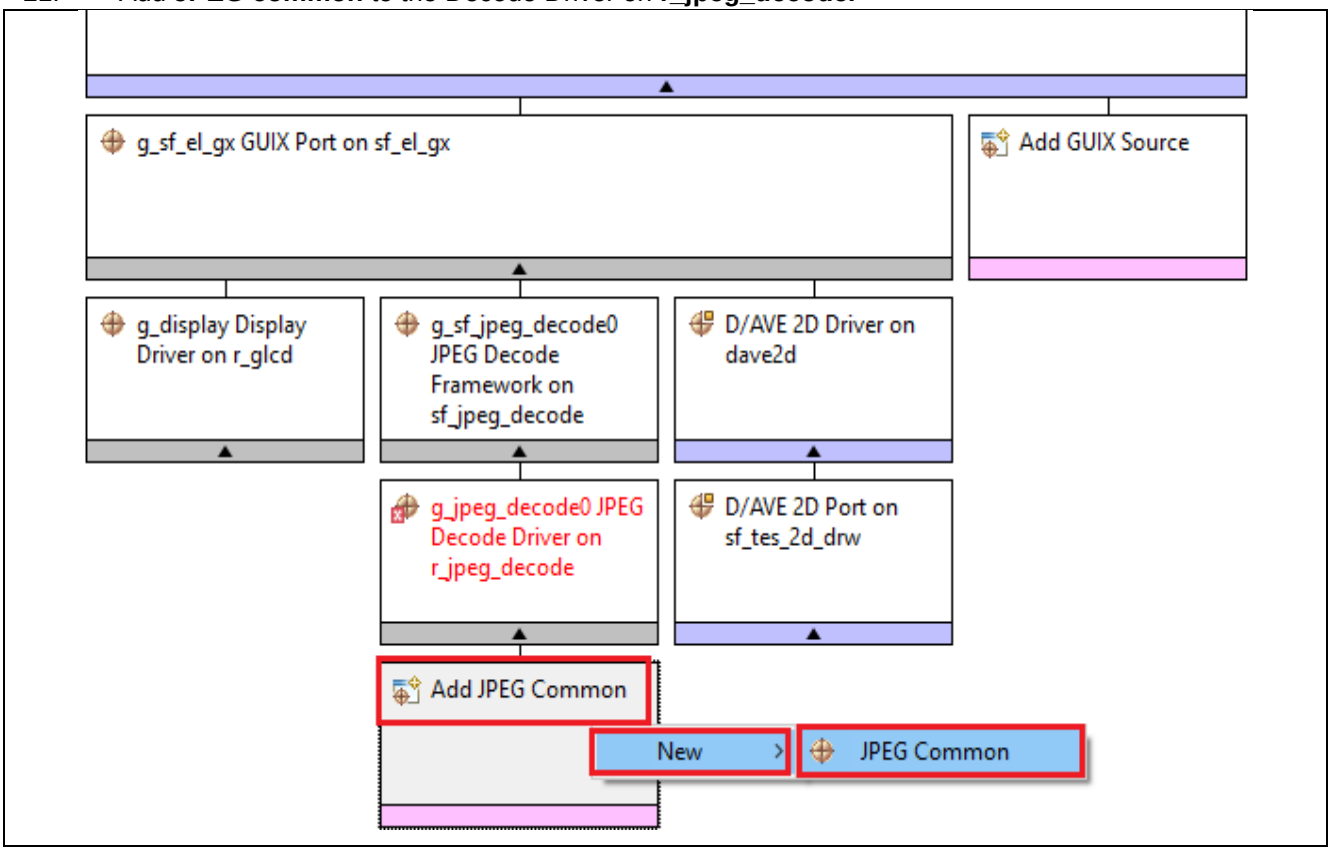

Figure 39. JPEG Common Module

22. Select **GUIX Port on sf_el_gx** and configure the Properties as shown.

Property	Value
◢ Common	
Parameter Checking	Default (BSP)
◢ Module g_sf_el_gx0 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 enabled)	1000
Memory section for GUIX Canvas Buffer	sdram
Memory section for JPEG Work Buffer	sdram

Figure 40. GUIX Port on sf_el_gx Properties

23. 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 41. JPEG Decode Driver on r_jpeg Properties

24. 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 42. D/AVE 2D Port Properties

25. 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 43. Interrupt Properties

26. 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 44. Graphics Screen 1 Properties

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

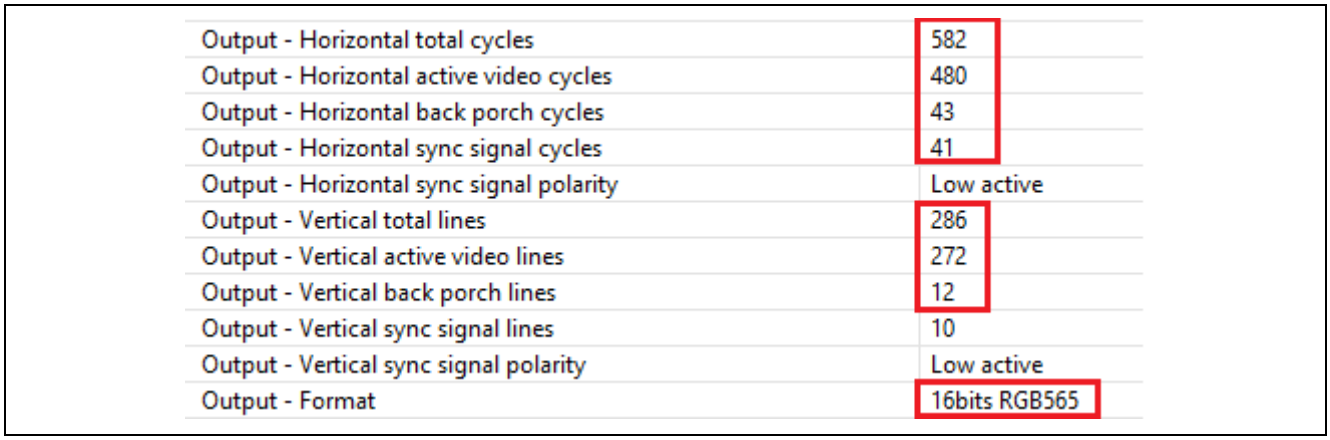

Figure 45. Output Screen 2 Properties

28. Configure the TCON pins and clock as shown.

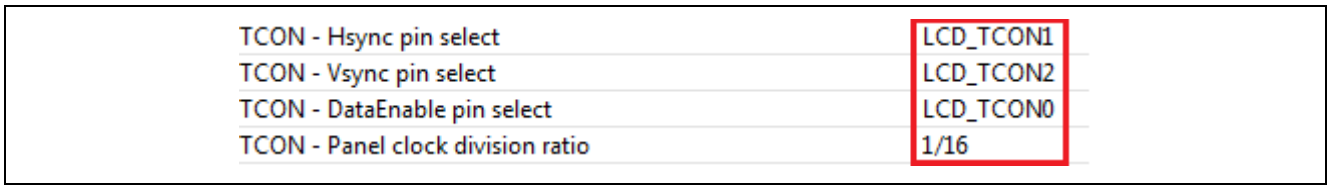

Figure 46. TCON Settings

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

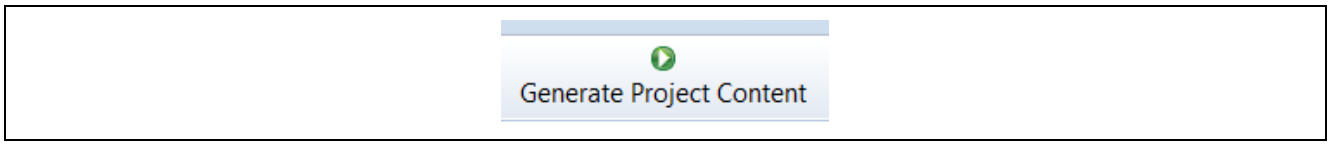

Figure 47. Generate Project Content

31. Close the Synergy Configuration window.
32. 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.

33. In the **Synergy Configuration** window, under the **Pins** tab **Select** the **Import the pin configuration**, as shown.

Figure 48. Synergy Pin Configuration

34. Click **File System** to access the pin configuration given in the source file as shown.

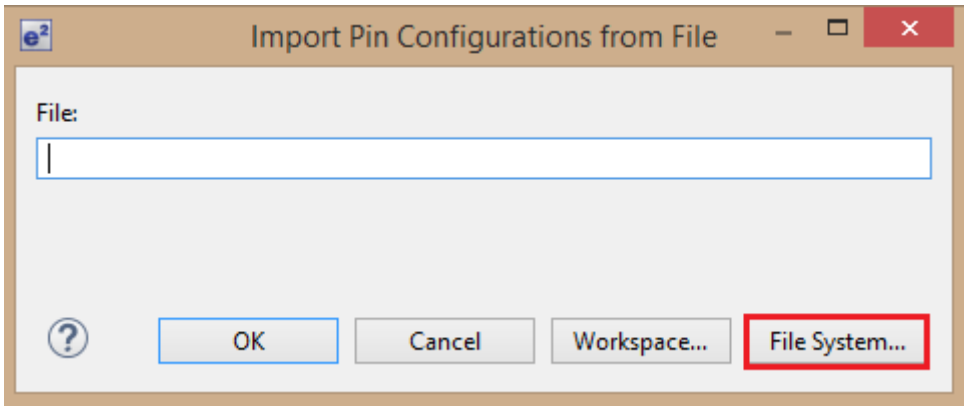

Figure 49. Importing Pin Configuration

35. Select the pin configuration in the source file and press **Open** as shown. Click **OK** to import the pin configurations.

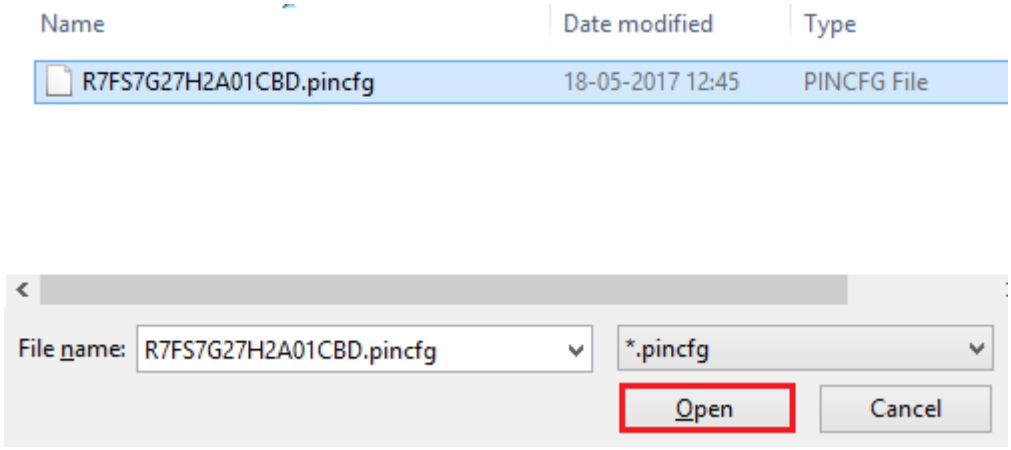

Figure 50. Synergy Configuration

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

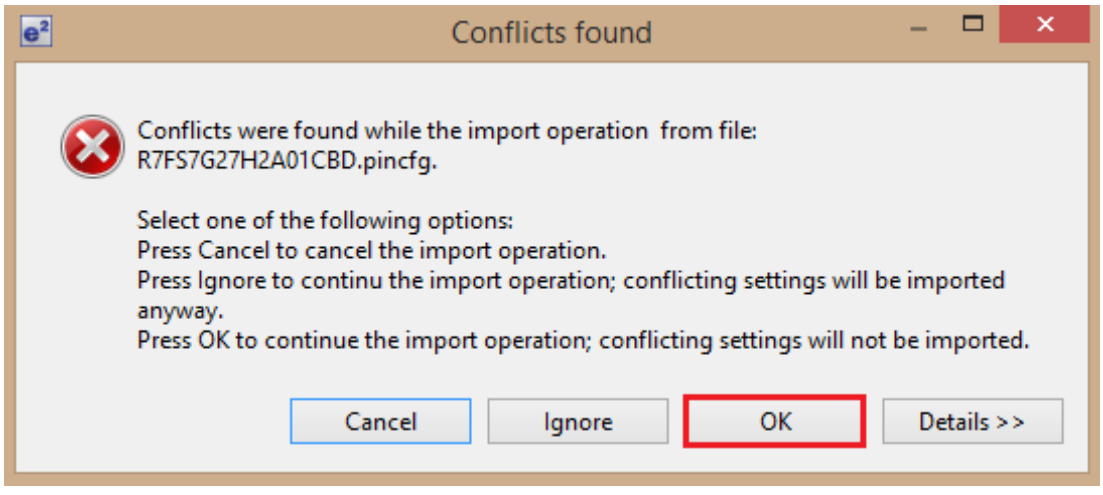

Figure 51. Conflict Found Error Box

37. Once you select the reference pin configuration, you have a pin error message, **Pin Dangling**. This means two functionalities are using the same PIN. As shown, the error is in the **Peripherals**.

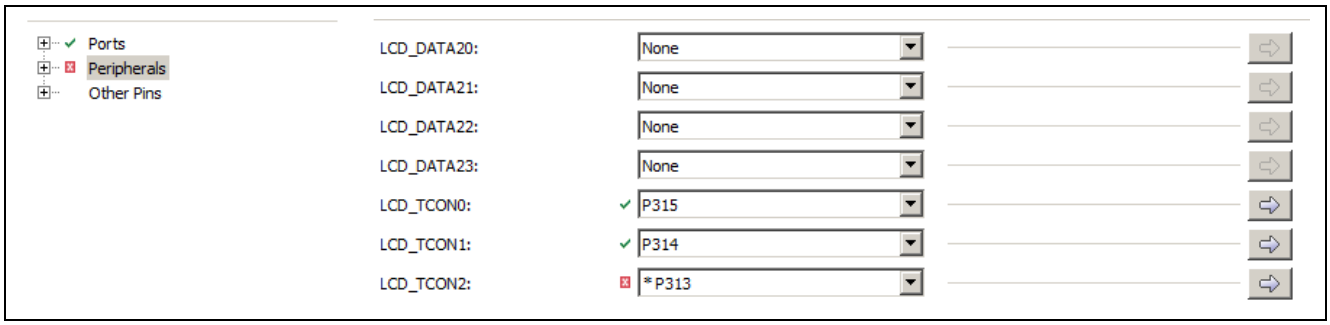

Figure 52. Selecting Pin Configuration

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

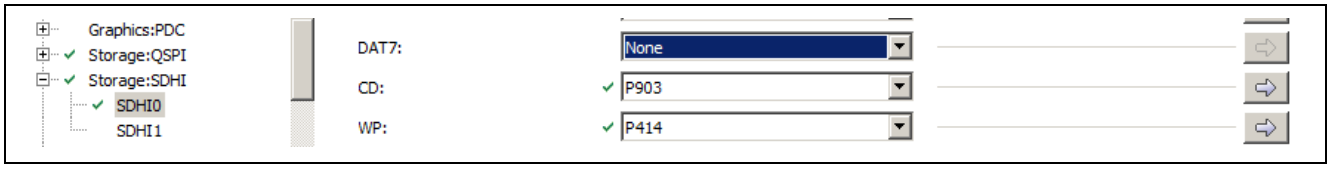

Figure 53. Making P313 Usable

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 42 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.

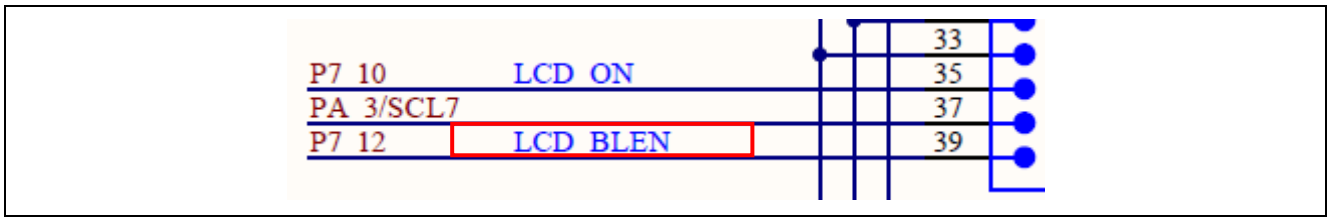

Figure 54. LCD Backlight Pin

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.

39. Select the **Pins** tab on the Synergy Configuration Window.

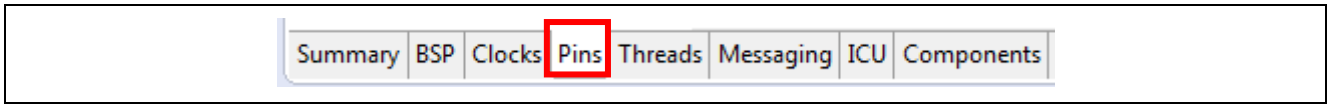

Figure 55. Configuration Pins

40. Expand **Ports** and **P7** to show the port 7 pins.

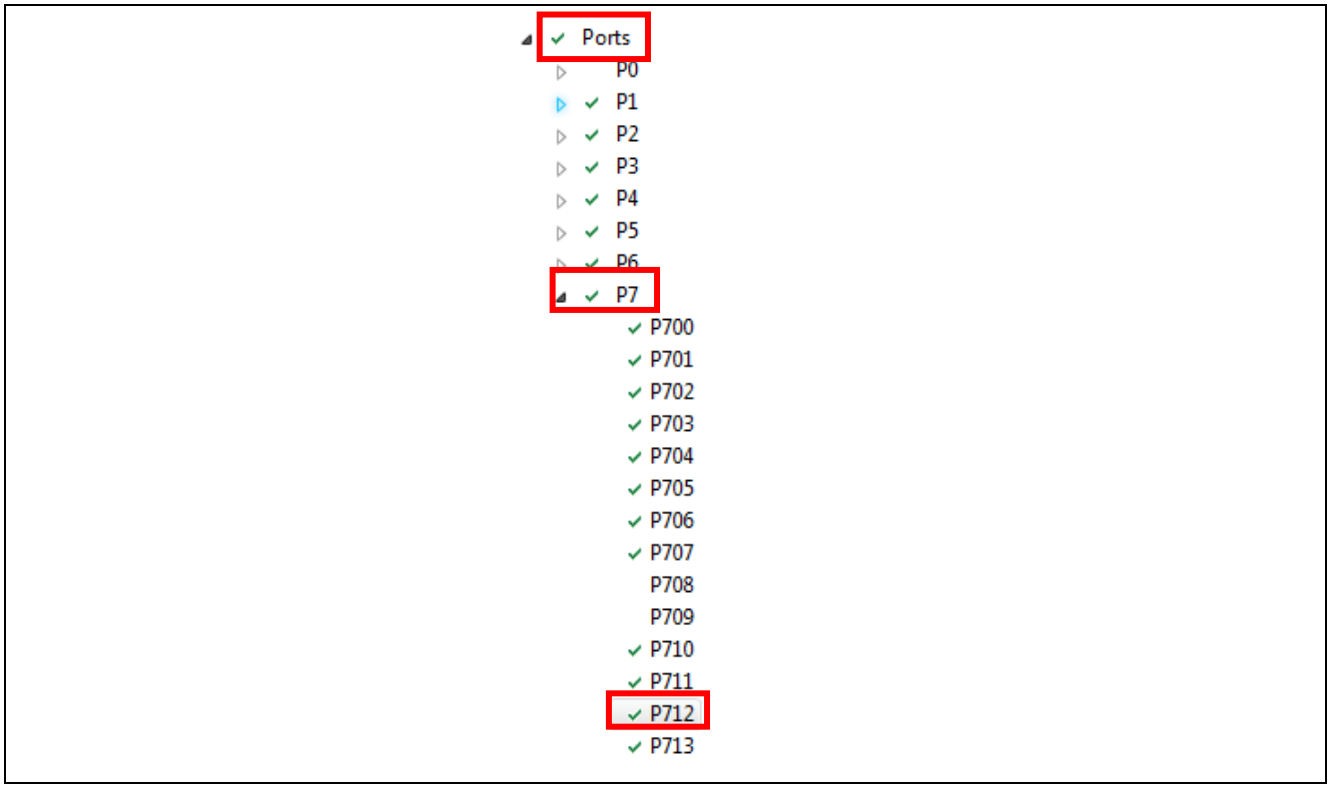

Figure 56. Port 7 Pins

41. Select **P712** to show the options for this pin.

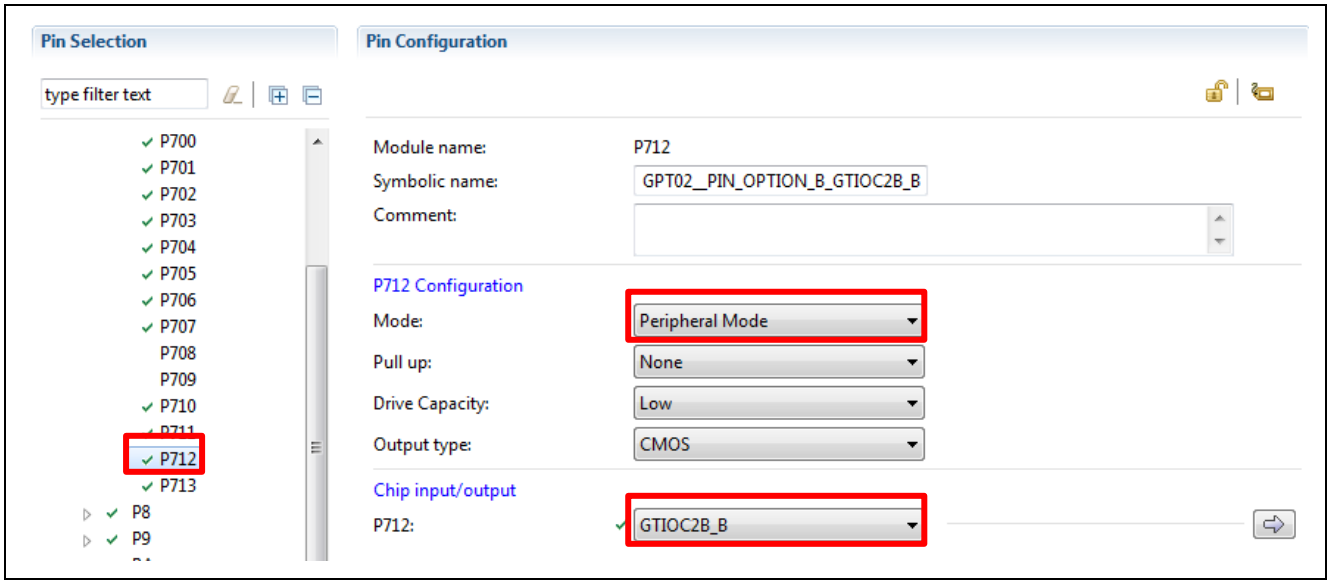

Figure 57. Pin Configuration for P7_12

- Module name: Synergy defined port and pin name
- Symbolic name: Optional symbolic name for code reference
- Comments: Optional description for the pin
- Mode: Pin's function; Input Mode, Output Mode, Peripheral Mode
- Pull up: Internal resistor pullup; None, input pull-up
- Drive Capacity: Output drive capability; Low, Medium, High
- Output type: CMOS, n-ch open drain
- P712: This option changes based on the mode setting. In this case, GTIOC2B_B is selected to use as the Timer 2 B output.

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

This view shows the pins that are available for different functions.

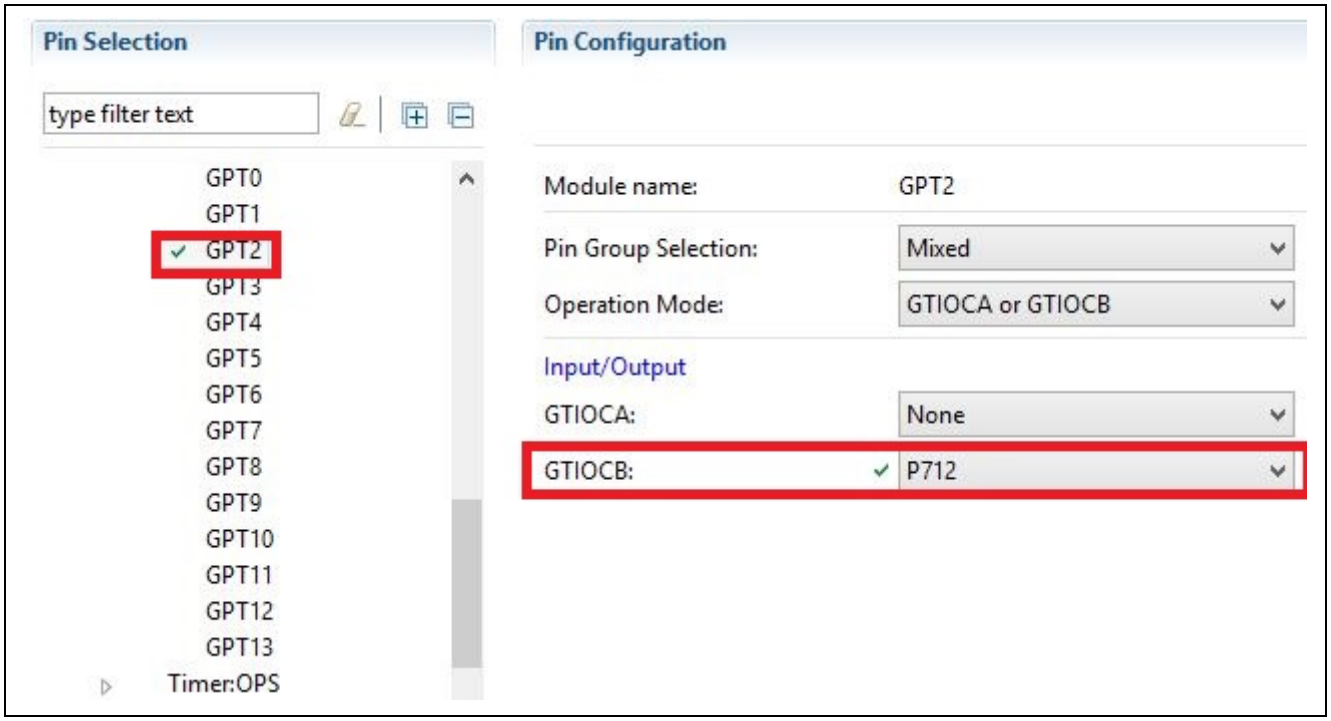

Figure 58. GPT02 Pin Configuration

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

The interrupt for the touch controller is located on pin P**0_1** as seen in the breakout board schematic.

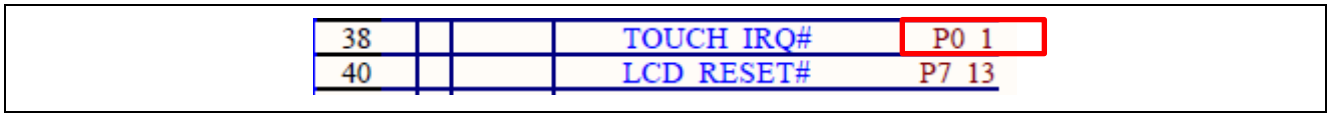

Figure 59. Touch IRQ

The touch panel pin is configured as an IRQ in the Pin Configuration window.

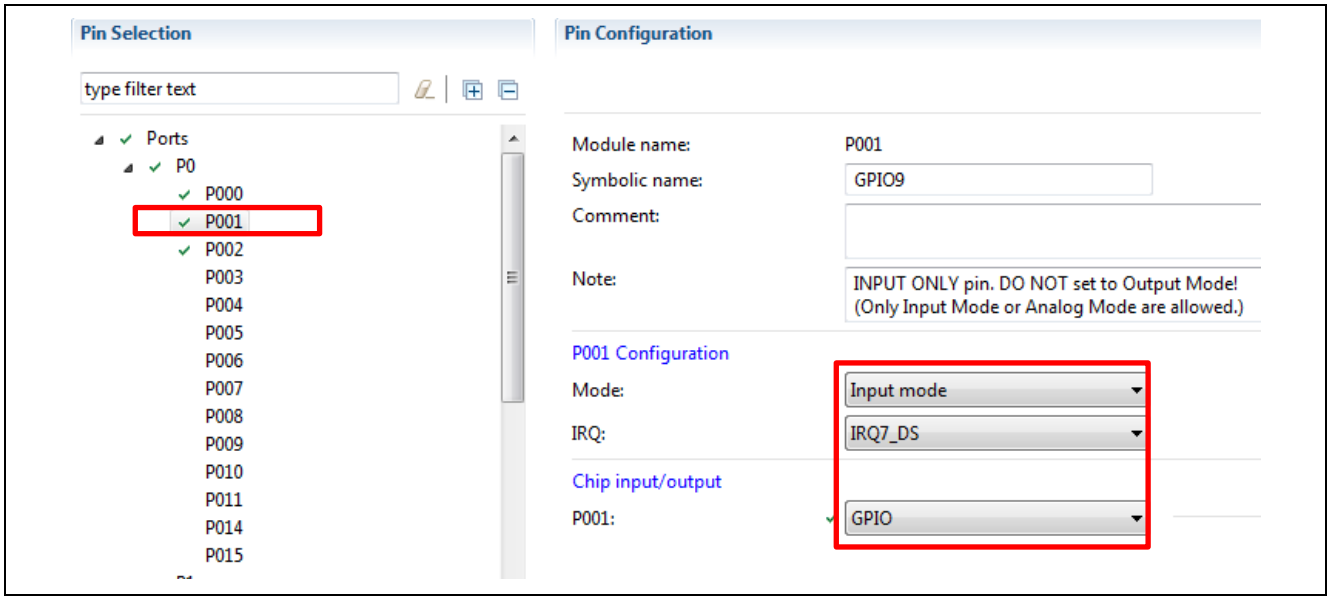

Figure 60. P0_01 Pin Configuration

For the LCD board schematic, see the touch controller is the SX8656.

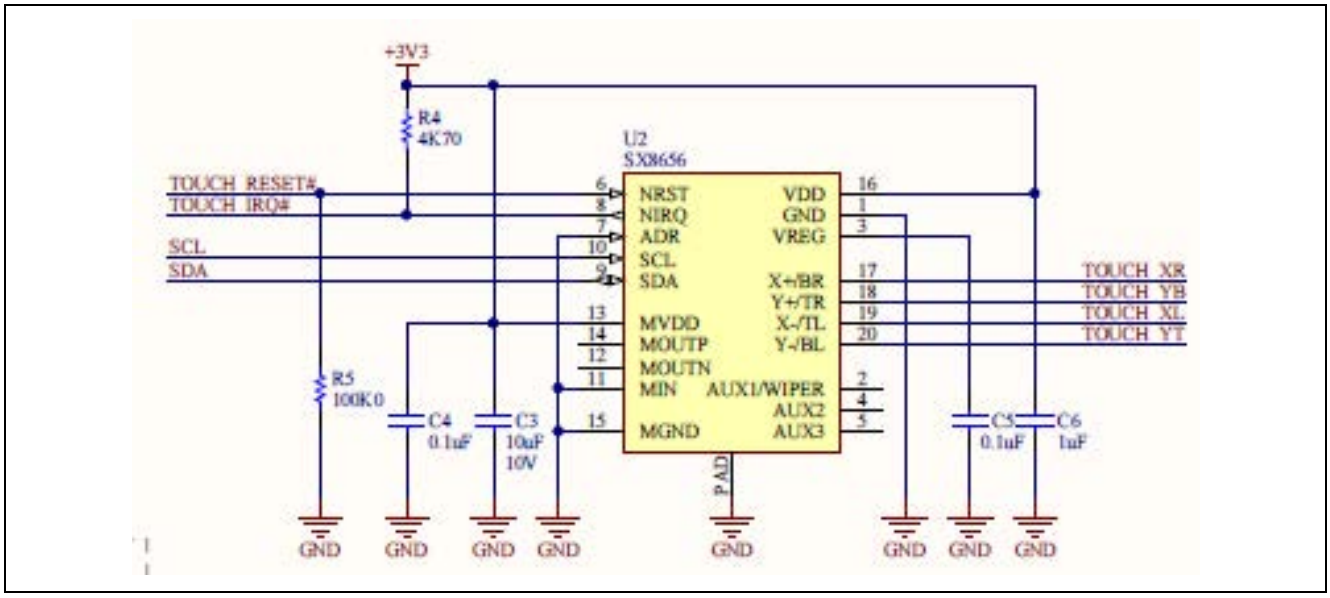

Figure 61. Touch Controller

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

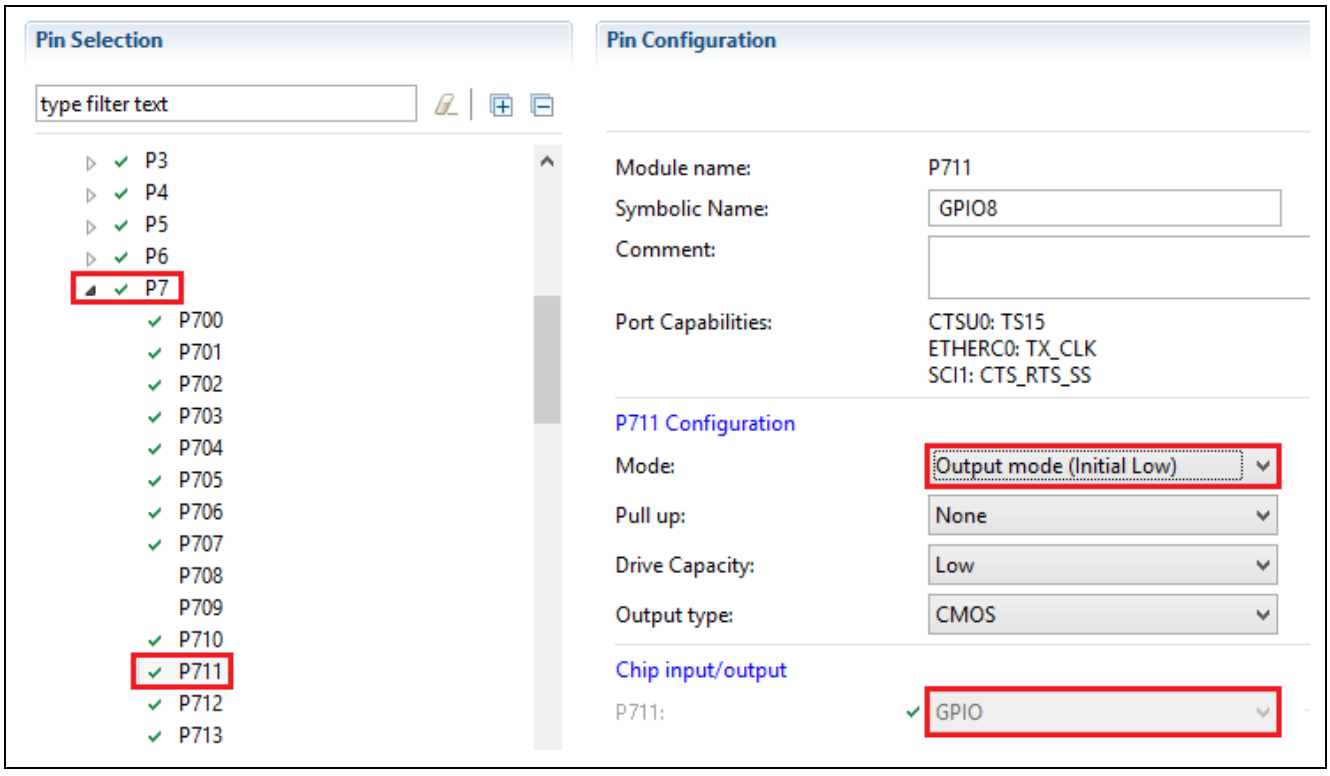

Figure 62. P7_11 Pin Configuration

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.

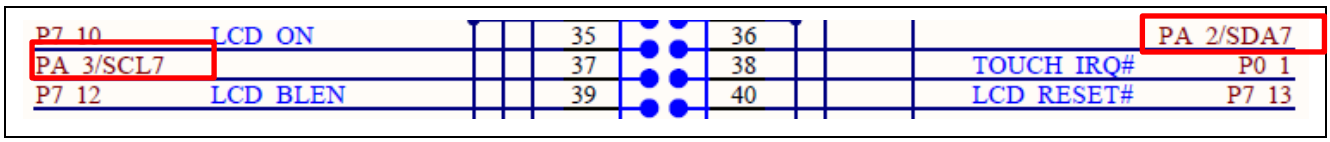

Figure 63. I²C Pins

The pins are configured in the Pin Configurator under the peripheral section.

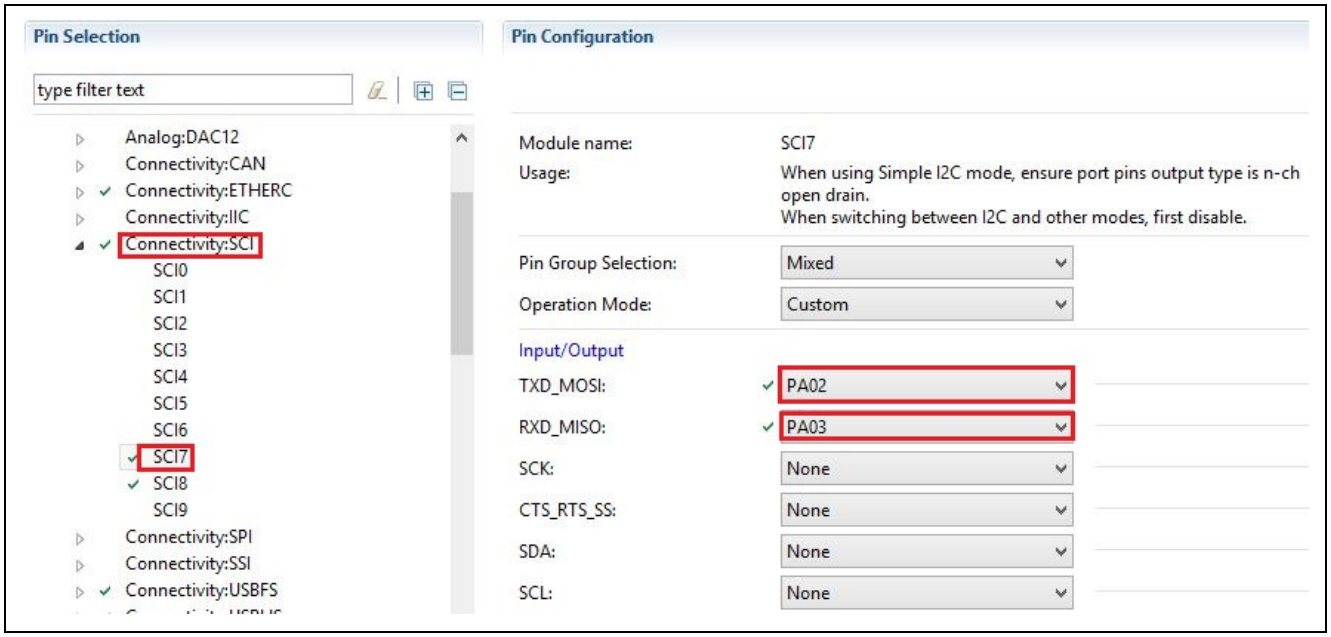

Figure 64. SCI7 Configuration

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

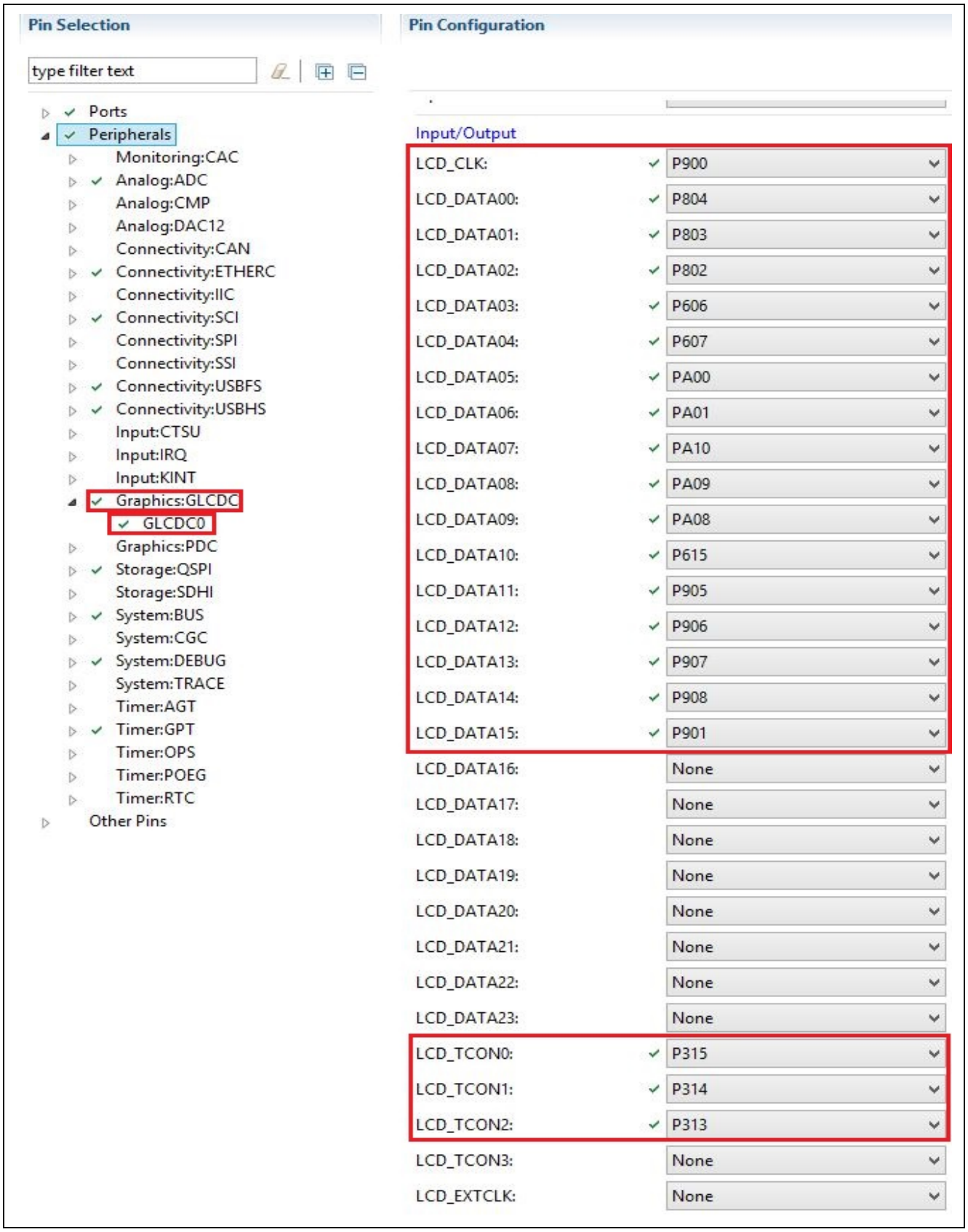

Figure 65. GLCD Pin Option B

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

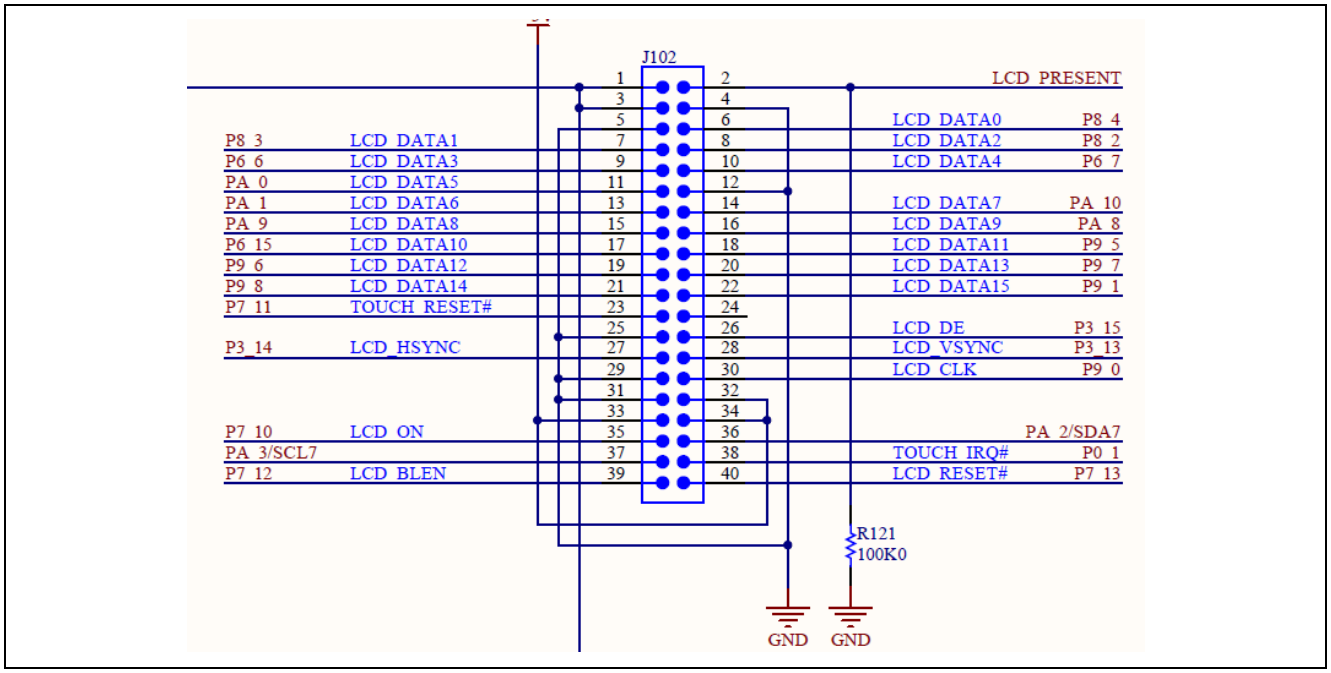

Figure 66. LCD Connector Pin Out

42. Select the **Messaging** tab on the **Synergy Configuration Window** as shown.

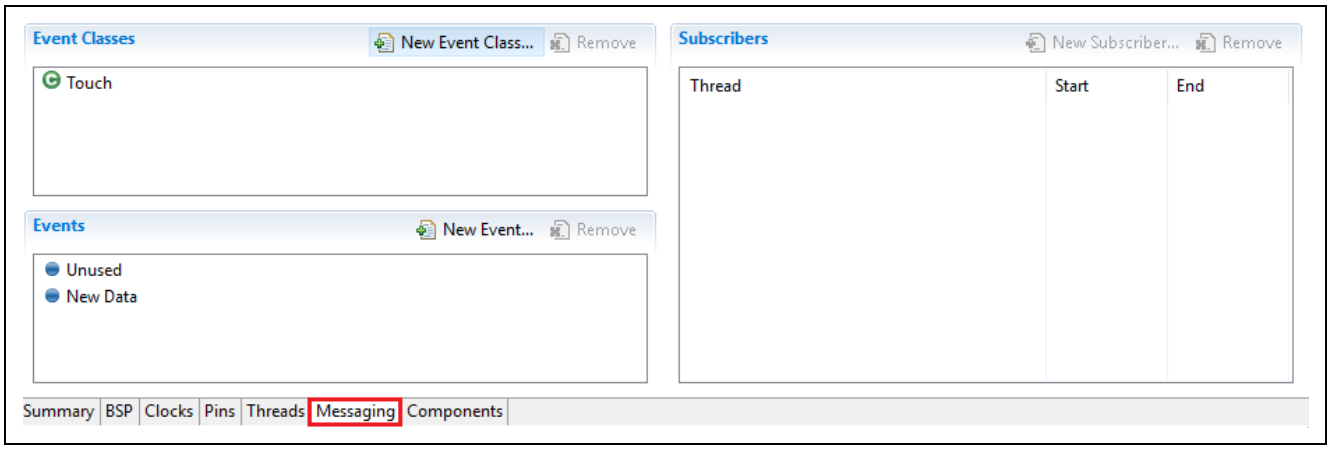

Figure 67. Messaging Tab

Note: This tab configures the event class definitions for the touchscreen events along with the event queue initialization and linking variables. The touch event was automatically generated when **Touch Panel Framework** on **sf_touch_panel_i2c** was added in the threads menu.

43. Select the **Touch Event** class.
44. On the touch subscribers' menu, click the **New Subscriber** button.
45. Inside of the **New Subscriber Dialog**, select the **Main Thread**.

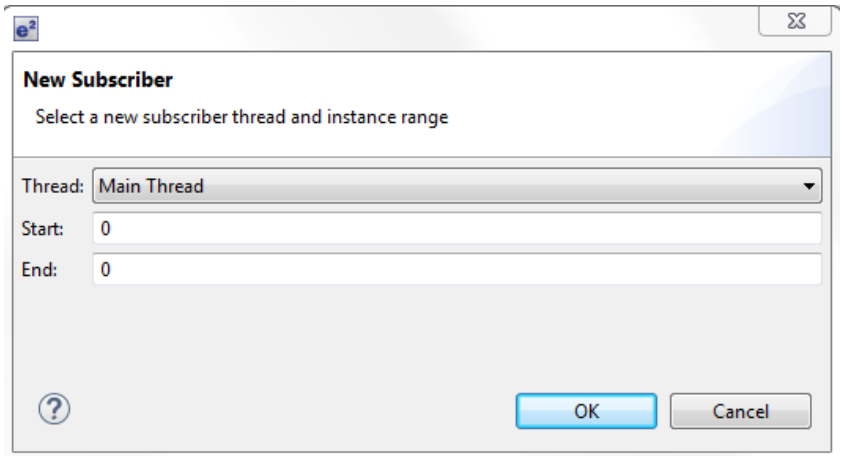

Figure 68. New Subscriber Dialog

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

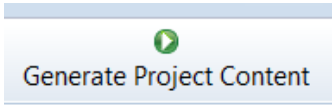

Figure 69. Generate Project Content

49. 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** by right clicking on the **src** folder and selecting **New -> Folder**.

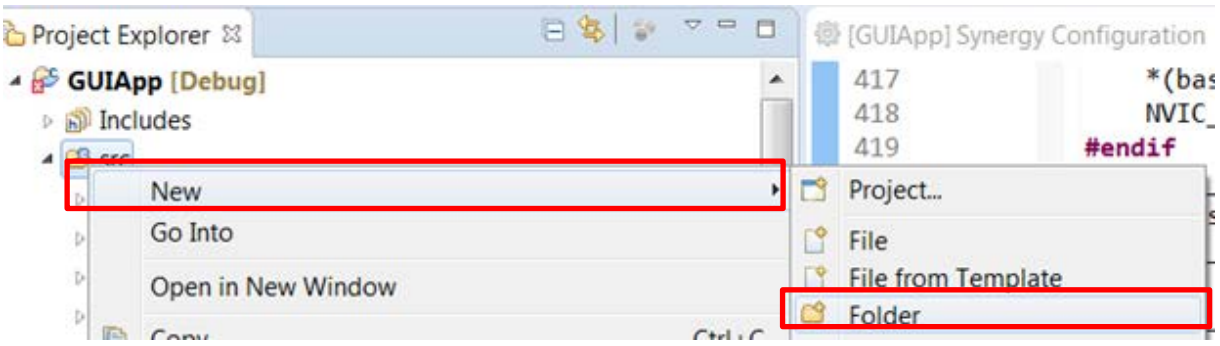

Figure 70. Creating a New Folder

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.

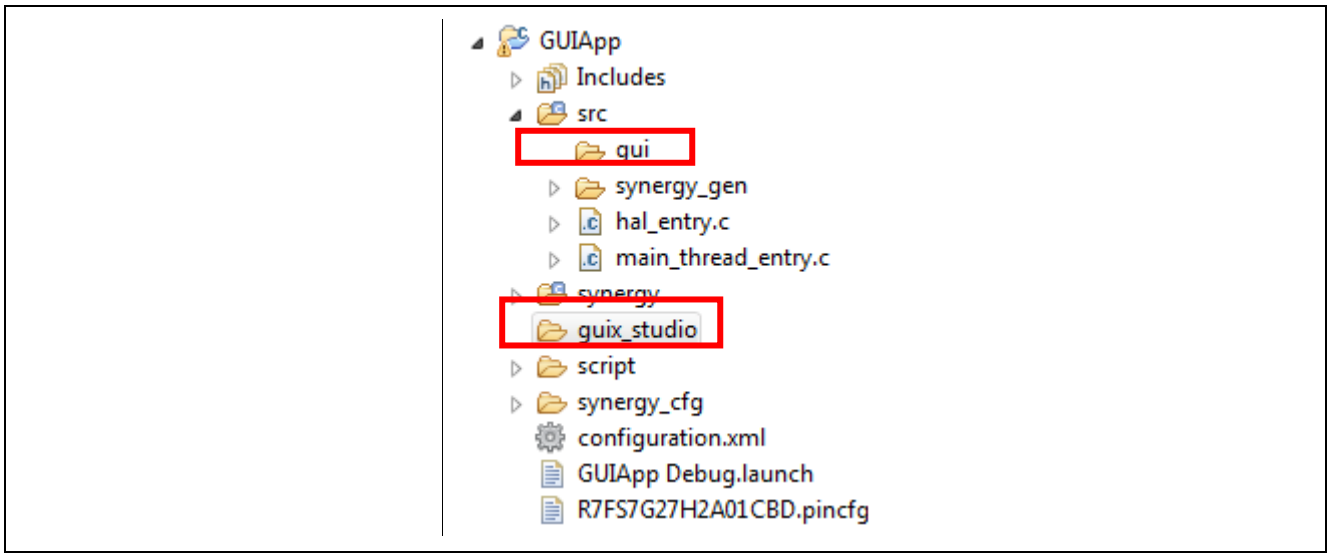

Figure 71. Final Folder List

3. Open GUIX Studio by clicking the desktop icon or by clicking on the GUIX icon in the Windows Start Menu, **All Programs > Express Logic > GUIX Studio 5.4.0.0** folder.

Figure 72. Start GUIX Studio

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

Figure 73. 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 74. Create a New GUIX Project

7. Click **Save**.
8. Change the Directories for all three options to be **..\src\gui**.

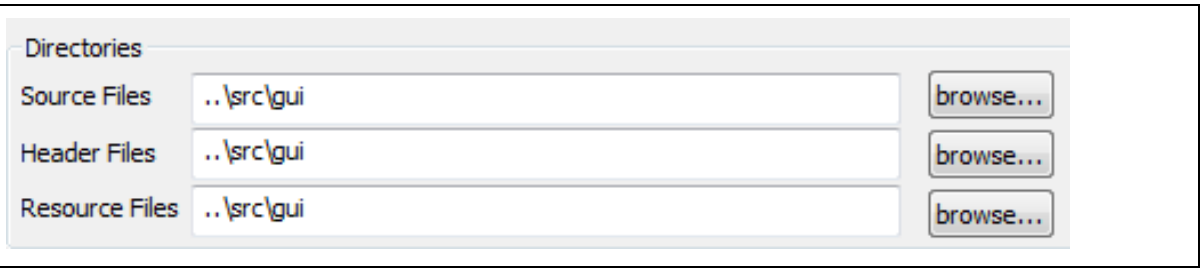

Figure 75. Correct the file locations

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 to **5.4.1**.

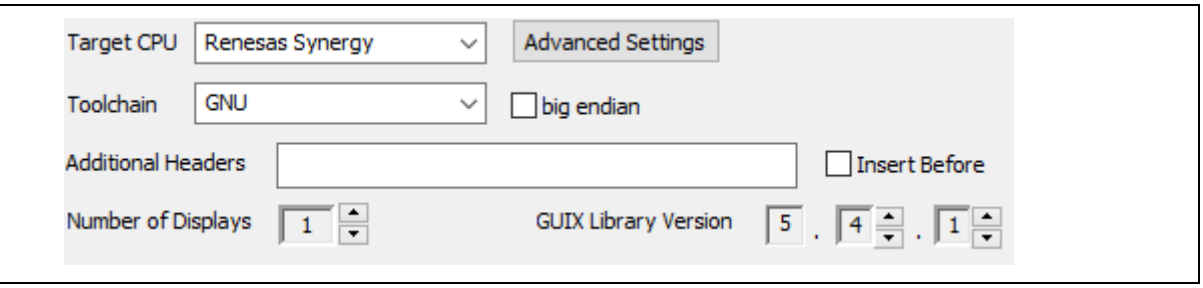

Figure 76. Target and GUIX version settings

11. Click the **Advanced Settings** button. A dialog window appears.
12. **Enable** the graphics accelerator and **Hardware JPEG Decoder** as shown.

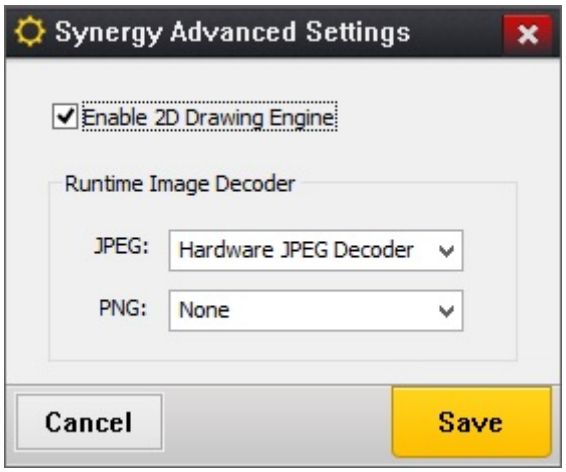

Figure 77. Synergy Advanced Settings

13. Click **Save**.
14. Setup the Display Configuration as shown.

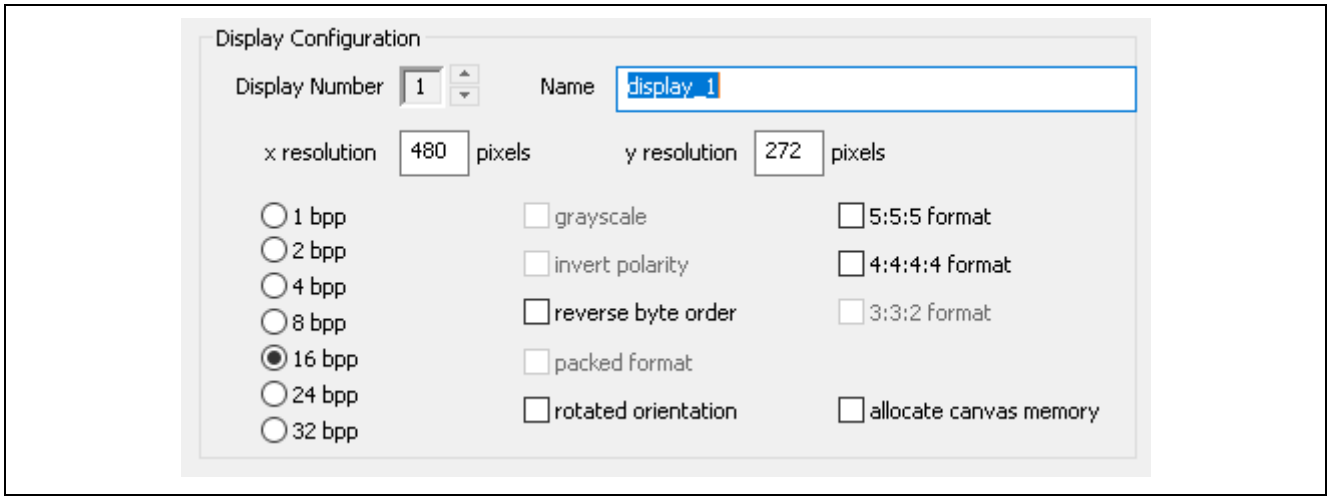

Figure 78. Configure the Display

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

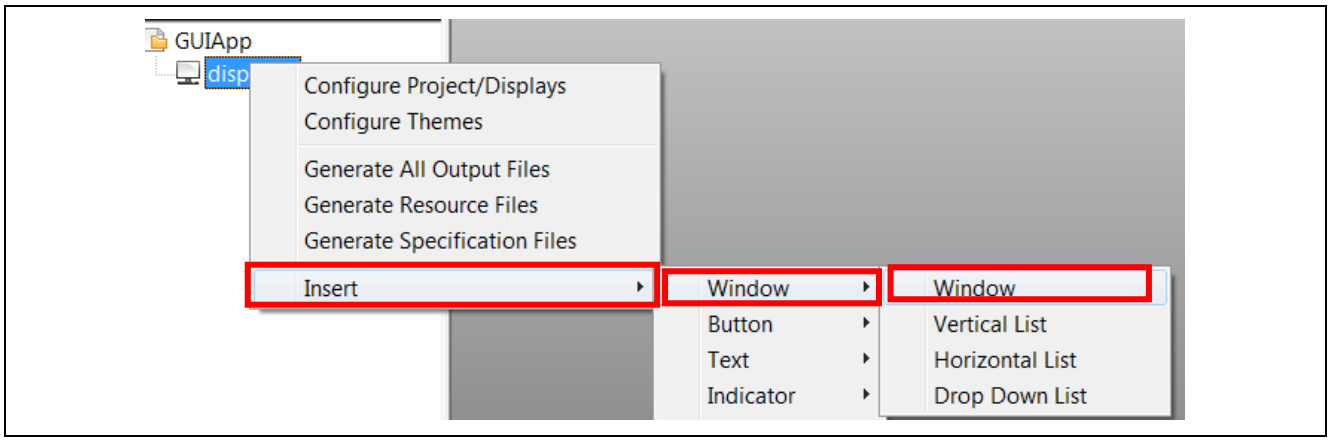

Figure 79. New Window

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

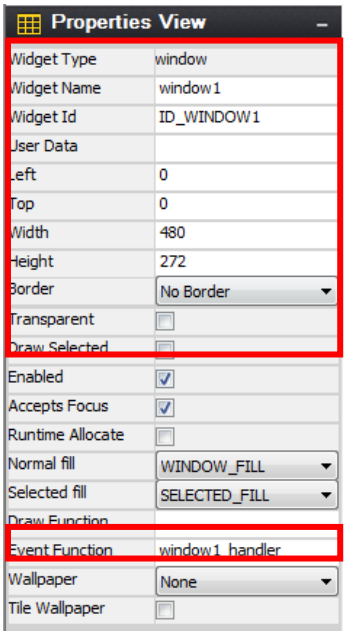

Figure 80. Configure Window1 Properties

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.

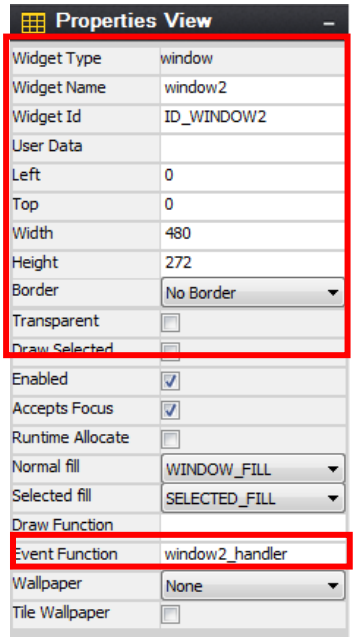

Figure 81. Configure Window2 Properties

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

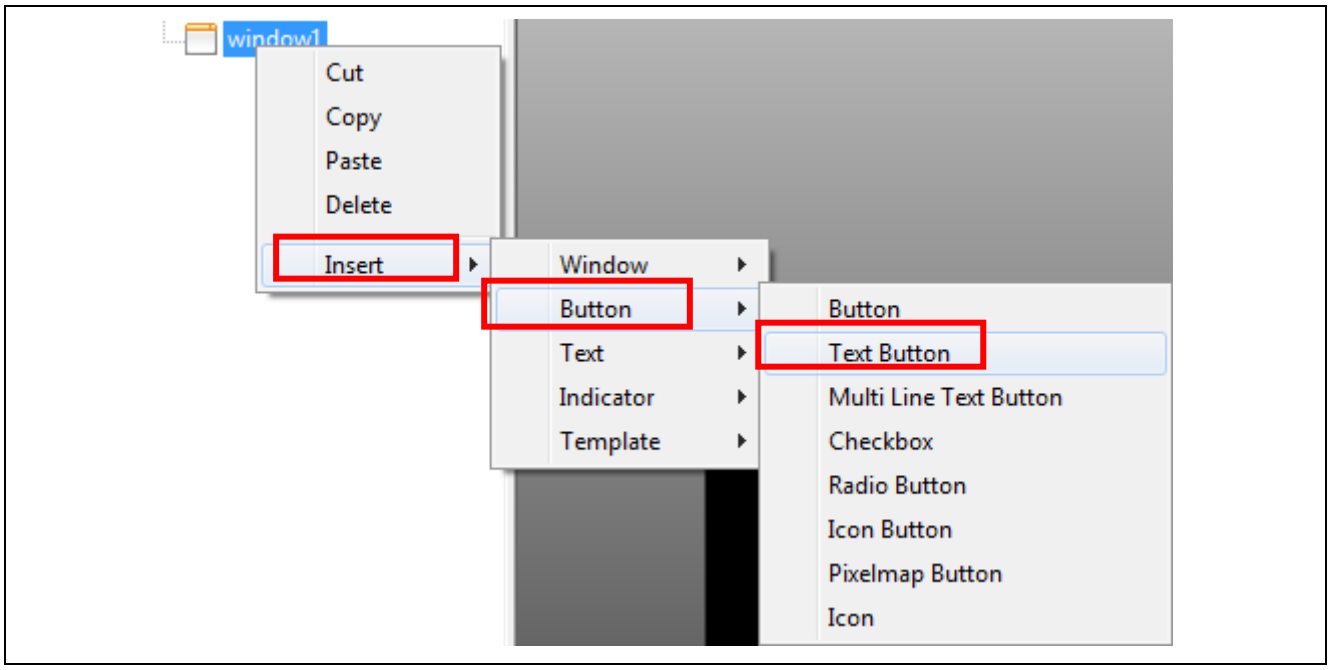

Figure 82. Add a New Text Button

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

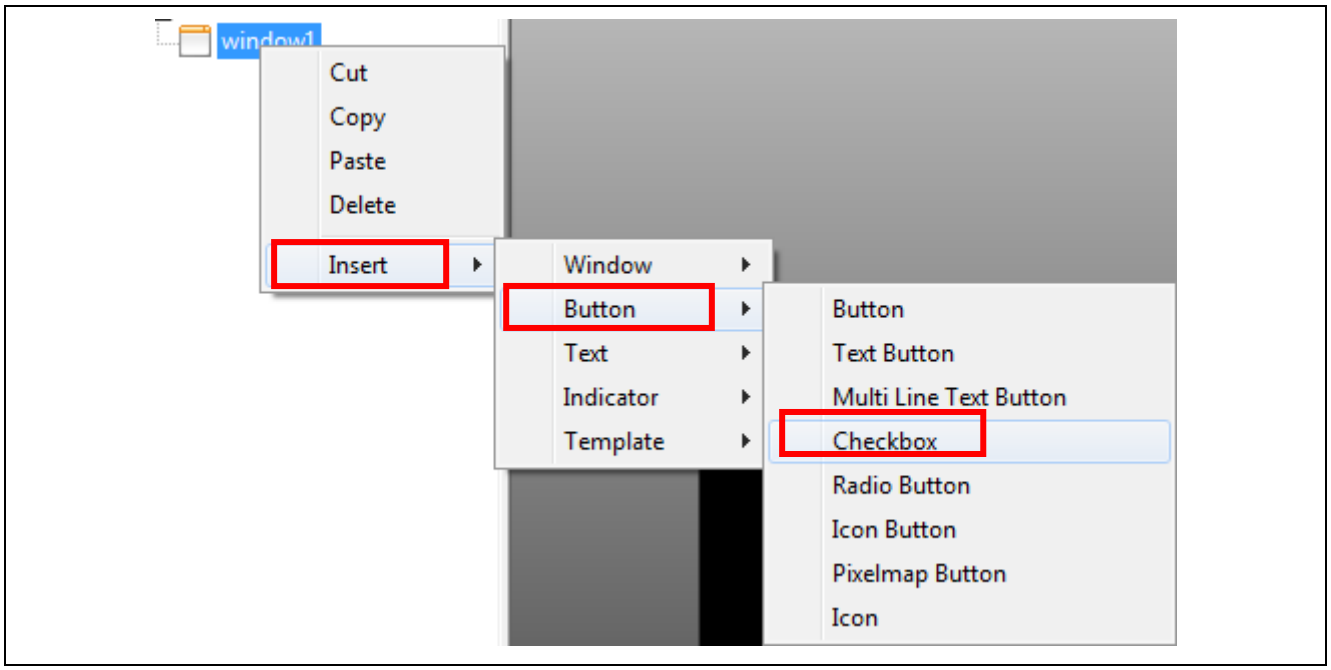

Figure 83. Add a New Checkbox

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

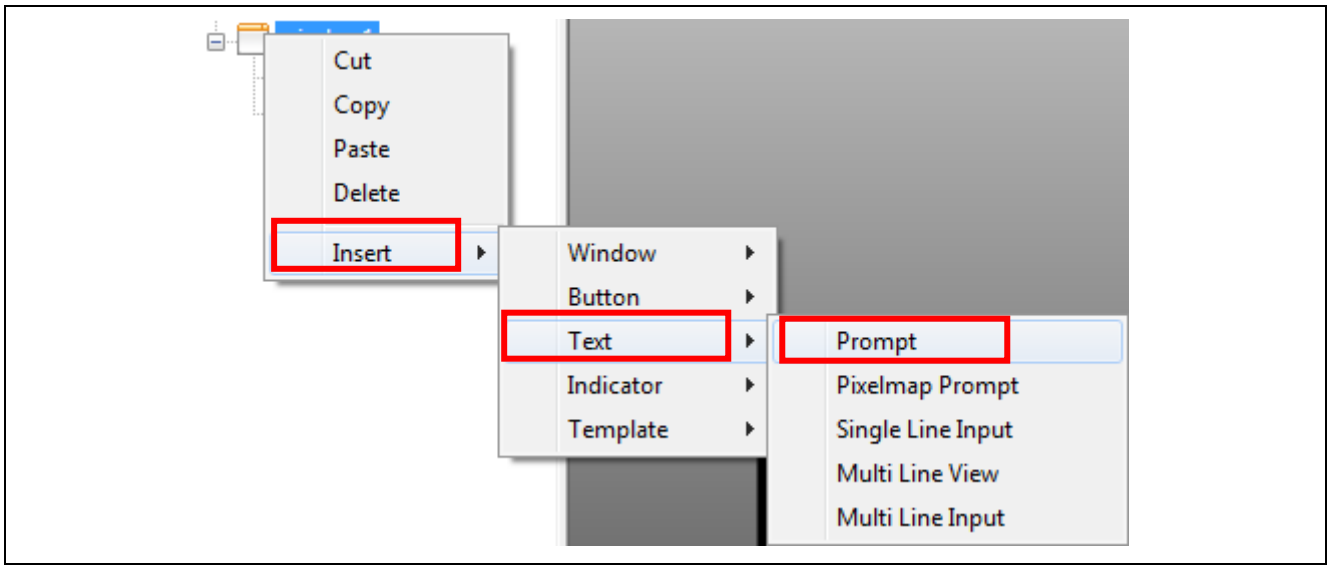

Figure 84. Adding New Prompt

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.

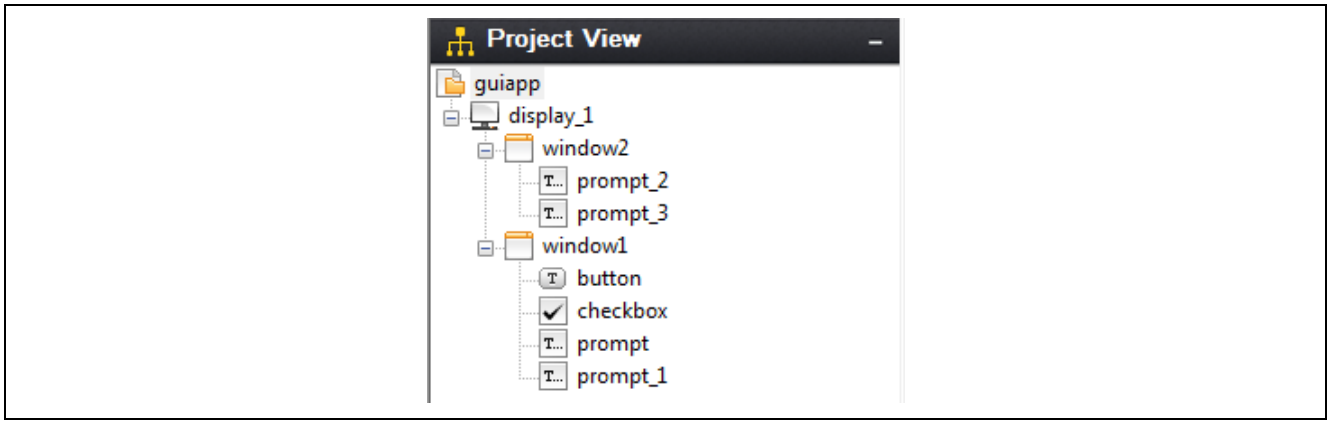

Figure 85. GUIX Project View

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

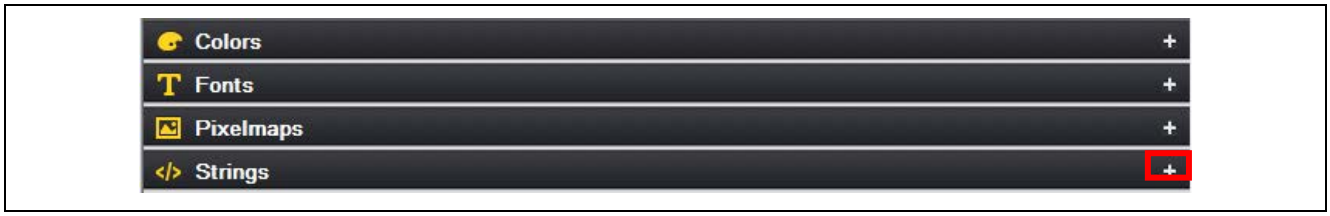

Figure 86. Strings Button

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 87. 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.

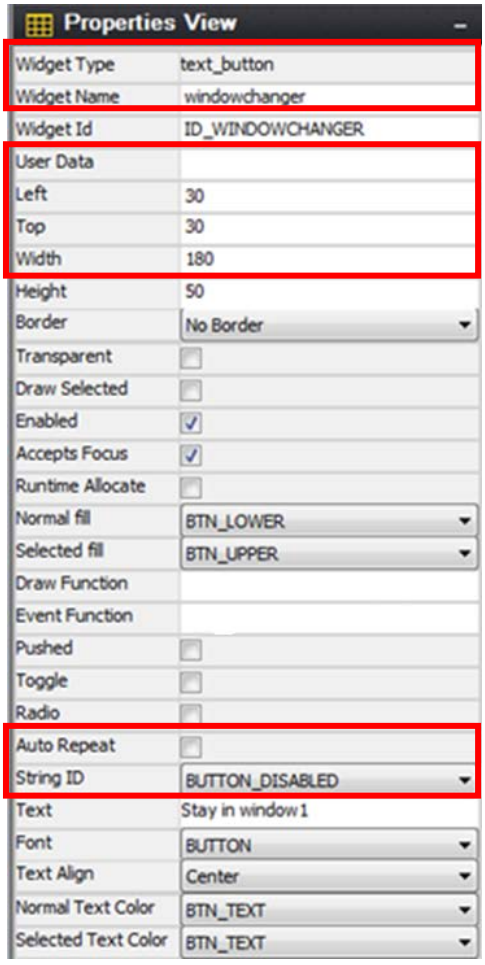

Figure 88. Configure Windowchanger Button Properties

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

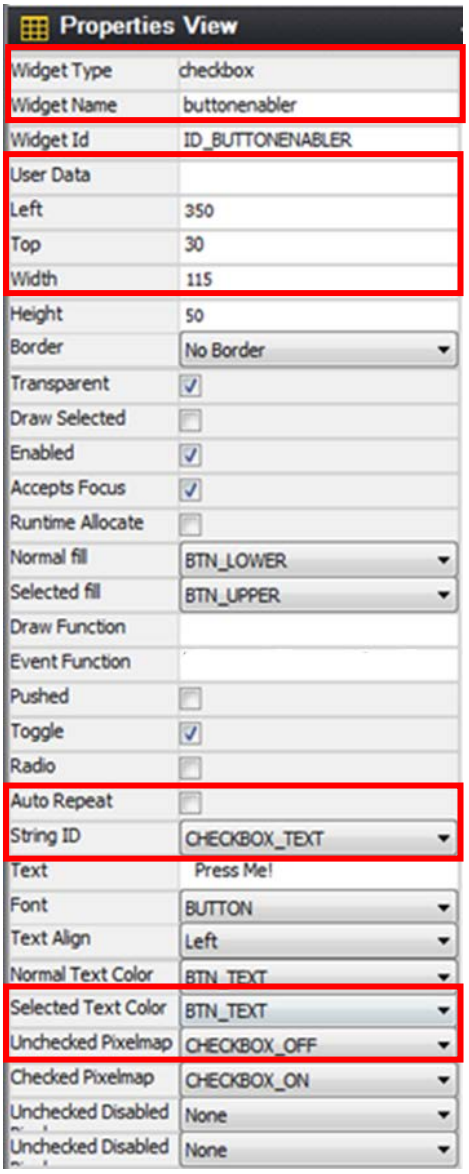

Figure 89. Configure Buttonenabler Checkbox Properties

34. In the **Project View** under window1, click the prompt, then modify the properties to match the window below.

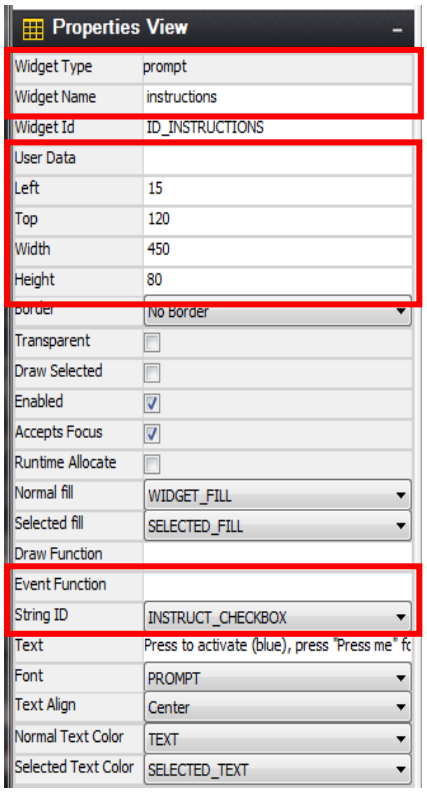

Figure 90. Configure Prompt Properties

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

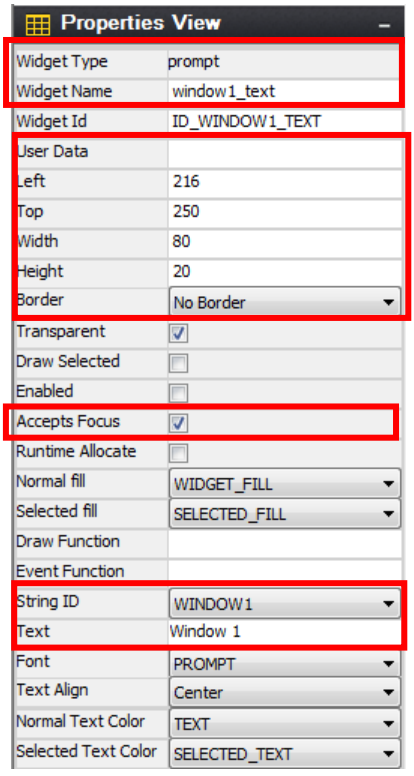

Figure 91. Configure Window Text Properties

36. In the **Project View** under window2, click prompt_2, then modify the properties to match the following figure.

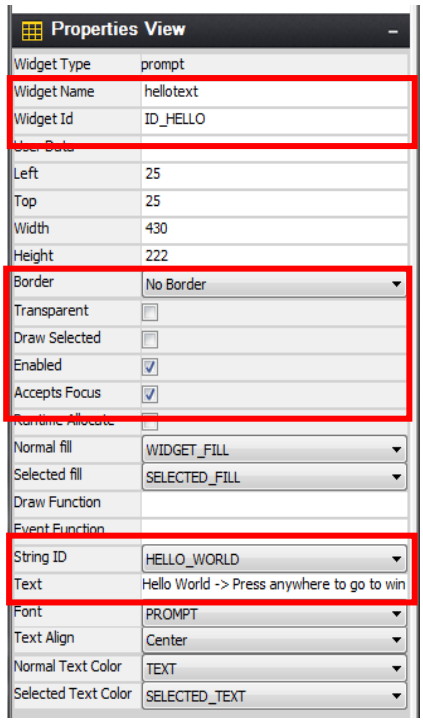

Figure 92. Configure Hello Text Prompt Properties

37. In the **Project View** under window2, click prompt_3, then modify the properties to the following figure.

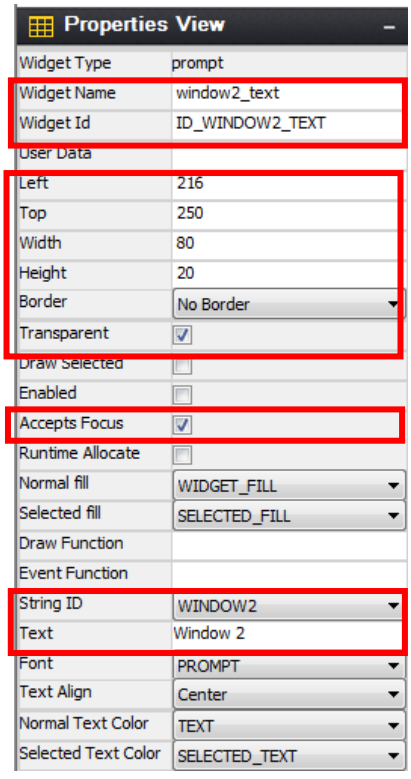

Figure 93. Configure Window Text Properties

After these configuration steps, the two windows should look like Figure 94 and Figure 95.

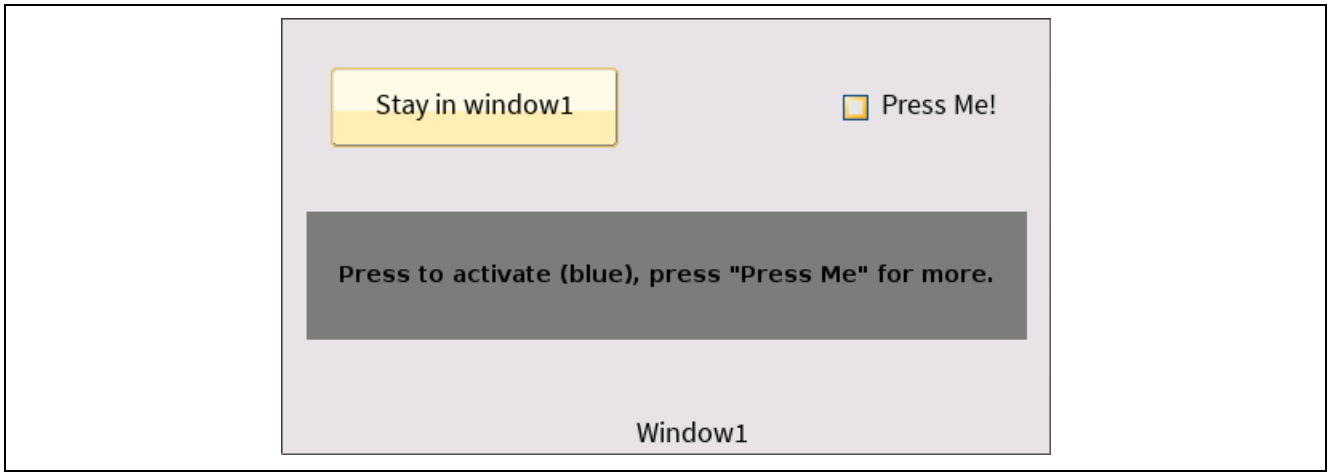

Figure 94. Configured Window1

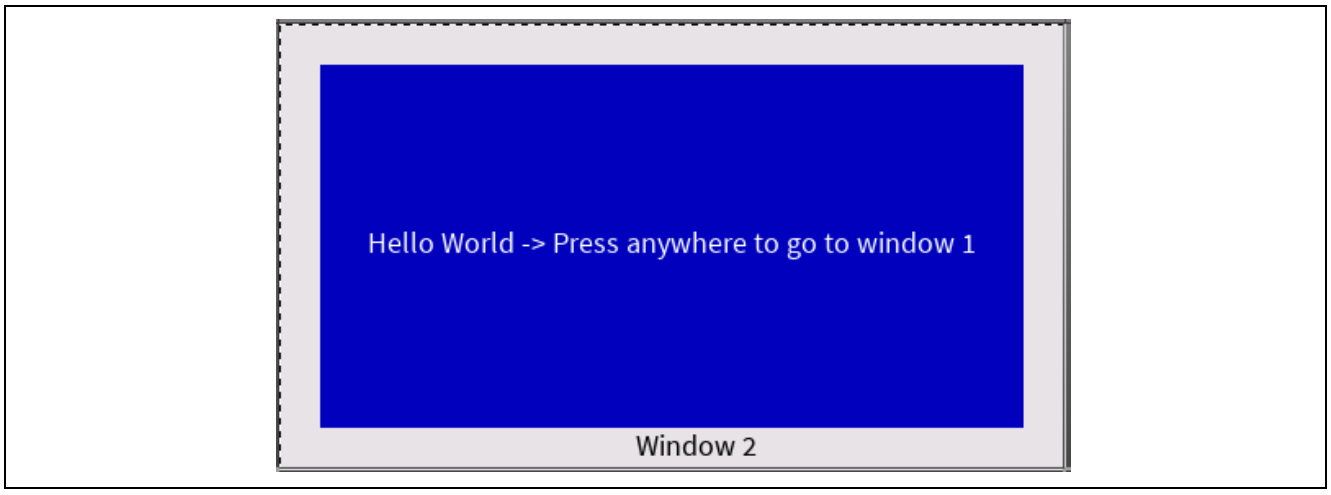

Figure 95. Configured Window2

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

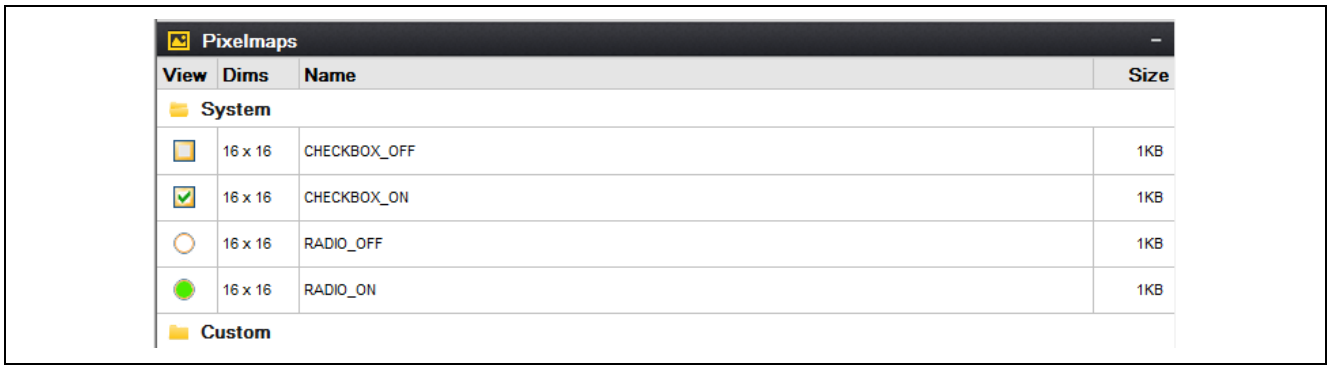

Figure 96. Configuration of Pixelmaps

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.

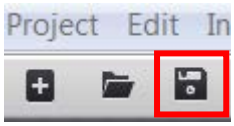

Figure 97. Save Project

45. From the pull-down menu select **Project > Generate all Output Files**.

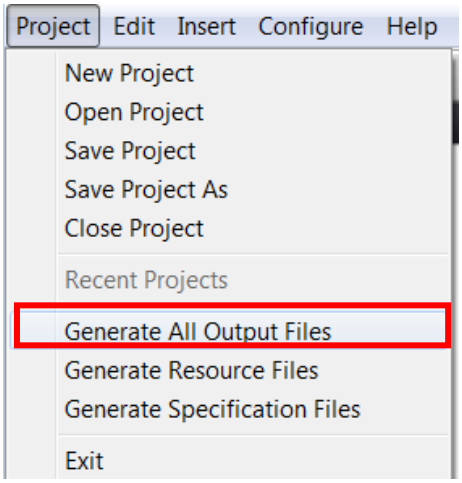

Figure 98. Generate All Output files

46. Click on **Generate**.
47. Return to e² 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 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 (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.

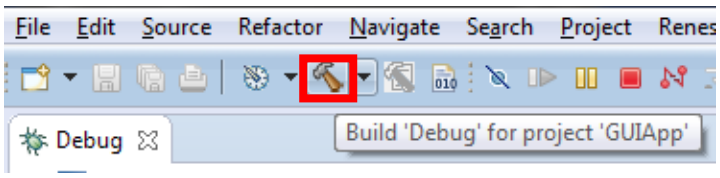

Figure 99. Build Button

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.

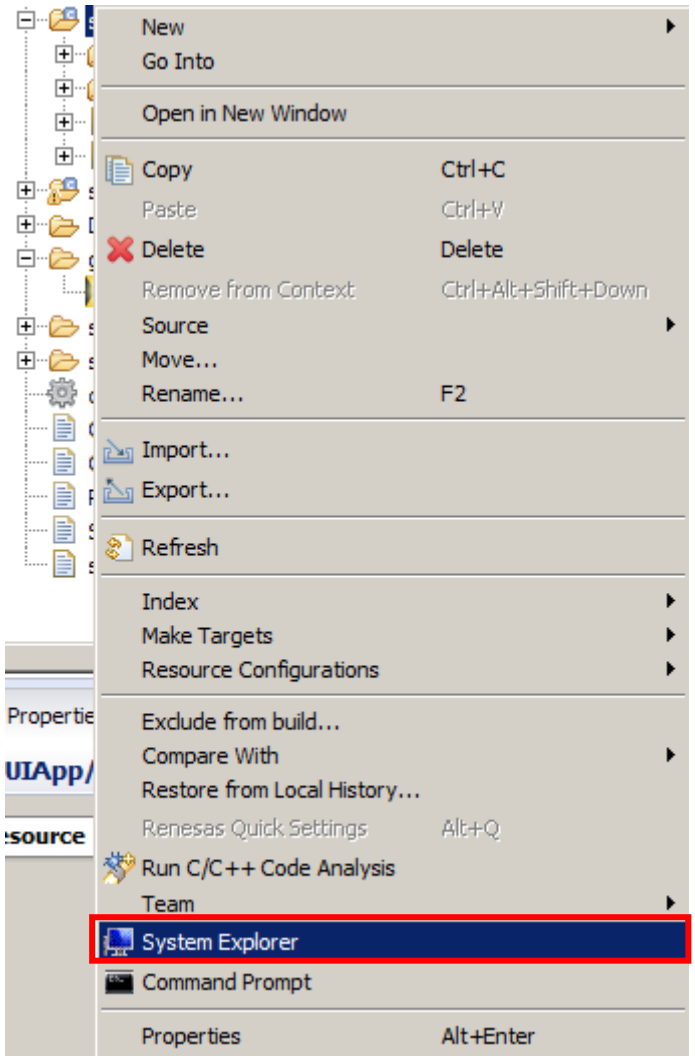

Figure 100. Adding handlers

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

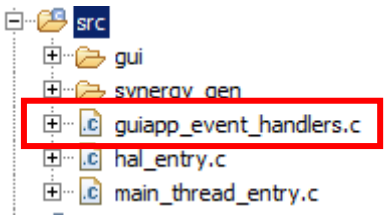

Figure 101. Final src structure

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_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 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.

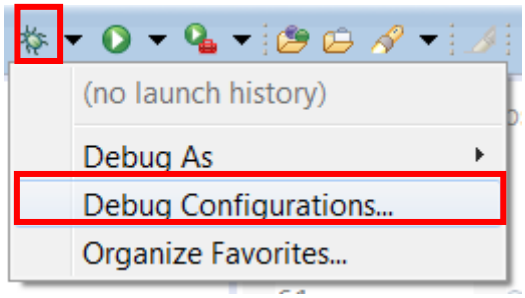

Figure 102. Debug Options

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.

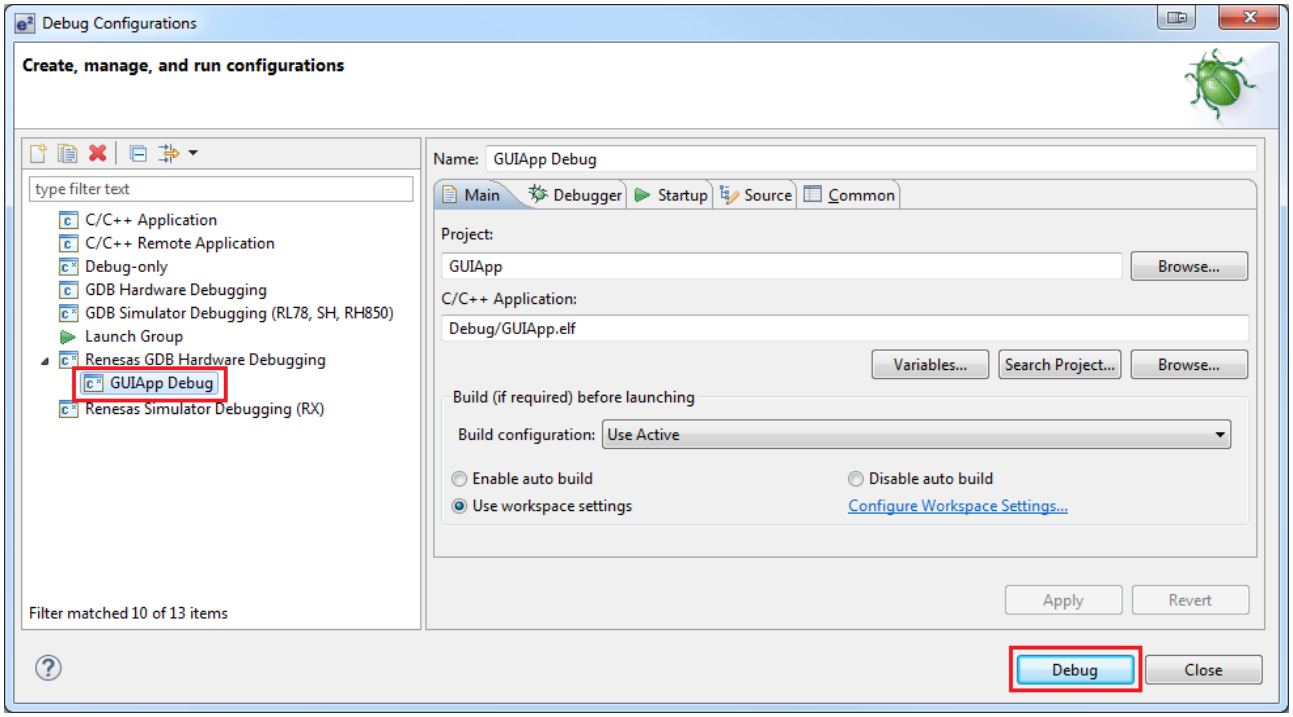

Figure 103. Debug Configurations

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.)

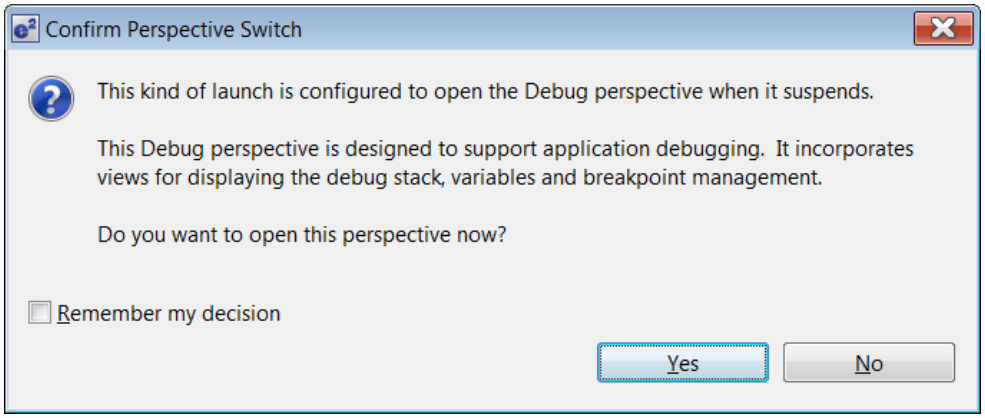

Figure 104. Perspective Switch Dialog

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

Figure 105. 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. Overview of the Demo is as follows.

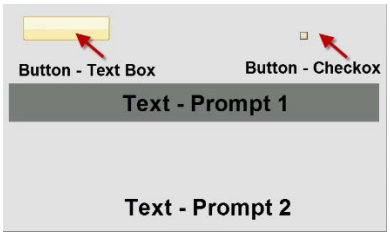

Figure 106. Window1

A. Figure 106 shows **Window1**. In this window are four elements:
   a. **Button – Text Box**: This box simply shows what window d 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 what window they are in. It never changes (always shows **window1**).

Figure 107. 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 they are in. It never changes (always shows **window2**).

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

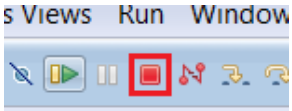

Figure 108. Stop Button

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

Synergy Software	www.renesas.com/synergy/software
Synergy Software Package	www.renesas.com/synergy/ssp
Software add-ons	www.renesas.com/synergy/addons
Software glossary	www.renesas.com/synergy/softwareglossary
Development tools	www.renesas.com/synergy/tools
Synergy Hardware	www.renesas.com/synergy/hardware
Microcontrollers	www.renesas.com/synergy/mcus
MCU glossary	www.renesas.com/synergy/mcuglossary
Parametric search	www.renesas.com/synergy/parametric
Kits	www.renesas.com/synergy/kits
Synergy Solutions Gallery	www.renesas.com/synergy/solutionsgallery
Partner projects	www.renesas.com/synergy/partnerprojects
Application projects	www.renesas.com/synergy/applicationprojects
Self-service support resources:	
Documentation	www.renesas.com/synergy/docs
Knowledgebase	www.renesas.com/synergy/knowledgebase
Forums	www.renesas.com/synergy/forum
Training	www.renesas.com/synergy/training
Videos	www.renesas.com/synergy/videos
Chat and web ticket	www.renesas.com/synergy/resourcelibrary

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	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	All	Updated for SSP v1.5.0
1.08	Mar.08.19	All	Updated for SSP v1.6.0

Notice

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

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

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

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

(Rev.4.0-1 November 2017)

Corporate Headquarters

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

Trademarks

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

Contact information

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