Renesas Synergy™ Platform

GUIX "Hello World" for SK-S7G2 and PK-S5D9

Introduction

This application note guides you through the process of creating a simple two screen GUI using GUIX Studio™ for the SK-S7G2 and PK-S5D9 Synergy MCU Group. Its application project demonstrates how 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 within e^2 studio using the Renesas Synergy™ Platform.

Target Device

- SK-S7G2 Synergy MCU Group board v3.1
- PK-S5D9 Synergy MCU Group board v1.0

Minimum PC

- Microsoft® Windows® 7
- 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
- Internet connection.

Installed Software

- Synergy e^2 studio Integrated Solution Development Environment (ISDE) 6.2.0 or later
- Synergy Software Package (SSP) v1.4.0 or later
- Synergy Standalone Configurator v6.2.0 or later
- GUIX Studio v5.4.0 or later
- IAR Embedded Workbench® for Renesas Synergy™ v8.21.1 or later

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

Software Files Provided

- guiapp_event_handlers.c
- main_thread_entry.c
- lcd_setup.c
- lcd.h.

Purpose

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

Intended Audience

The intended audience are developers designing GUI applications

Contents

1. Overview

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

2. Importing the project into e² studio

Note: This step is included to give you the ability to skip the development steps and start at the point of verifying a working project on the SK-S7G2 Synergy MCU Group or the PK-S5D9 Synergy MCU Group. You can skip this step and proceed to section 3 to create a project in e² studio. If you do import the project, skip to section 7. Running the Application.

To skip the development walkthrough in this document and open a completed project in e² studio, see the. *Renesas Synergy™ Project Import Guide* (r11an0023eu0120-synergy-ssp-import-guide.pdf) in this package. It contains instructions on importing the project into e² studio and building the project. The included **GUIX_Hello_World_SK-S7G2.zip and GUIX_Hello_World_PK-S5D9.zip** files contain 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 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.

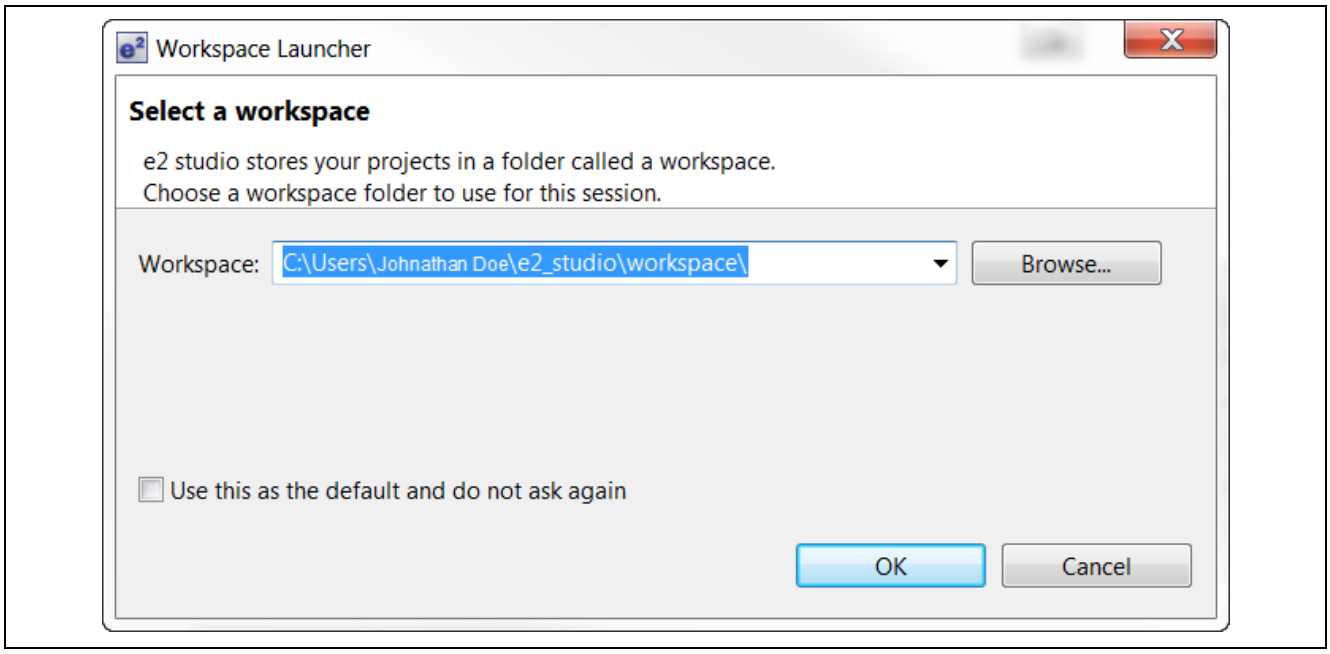

Figure 1. Workspace Launcher Dialog

3. Create a new workspace:

   From the **File** drop-down menu, select **Switch Workspace** > **Other**…
4. Append a workspace name:

   In the **Workspace Launcher** window, add text to the end of the workspace name to make it unique, such as **GUI_APP**. If you installed at 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. Proceed past the **Welcome** screen by closing it.

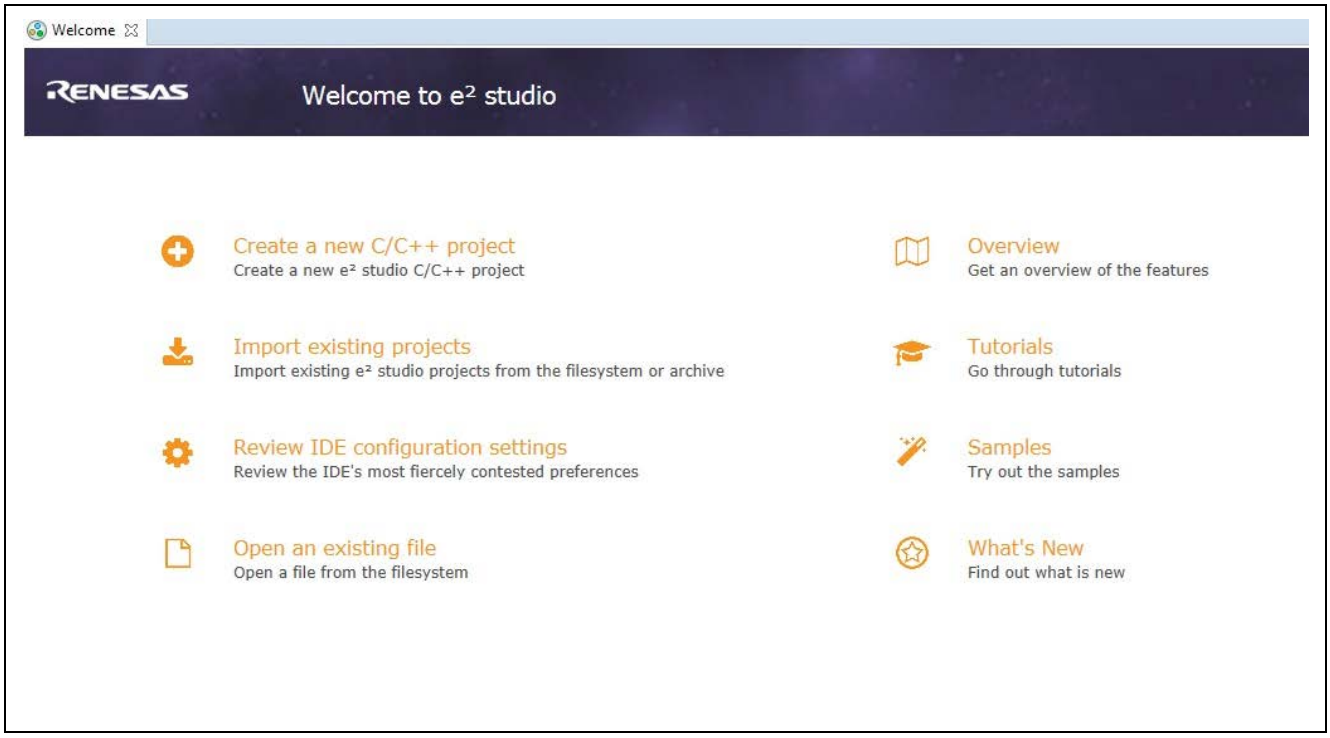

Figure 2. Close the Welcome Window by clicking in the Workbench Area

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

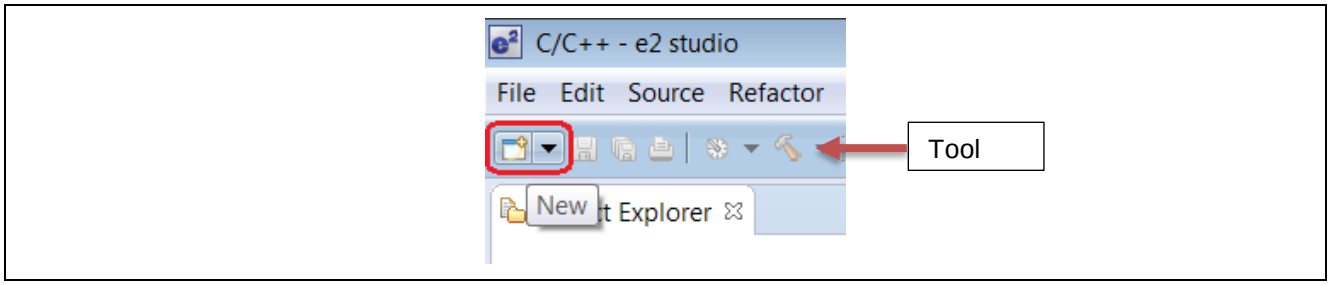

Figure 3. Start a New Project

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

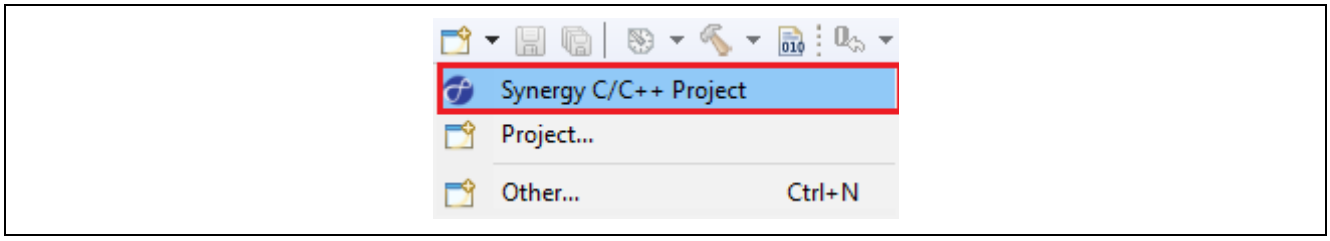

Figure 4. Select Synergy C/C++ Project in the drop-down menu

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

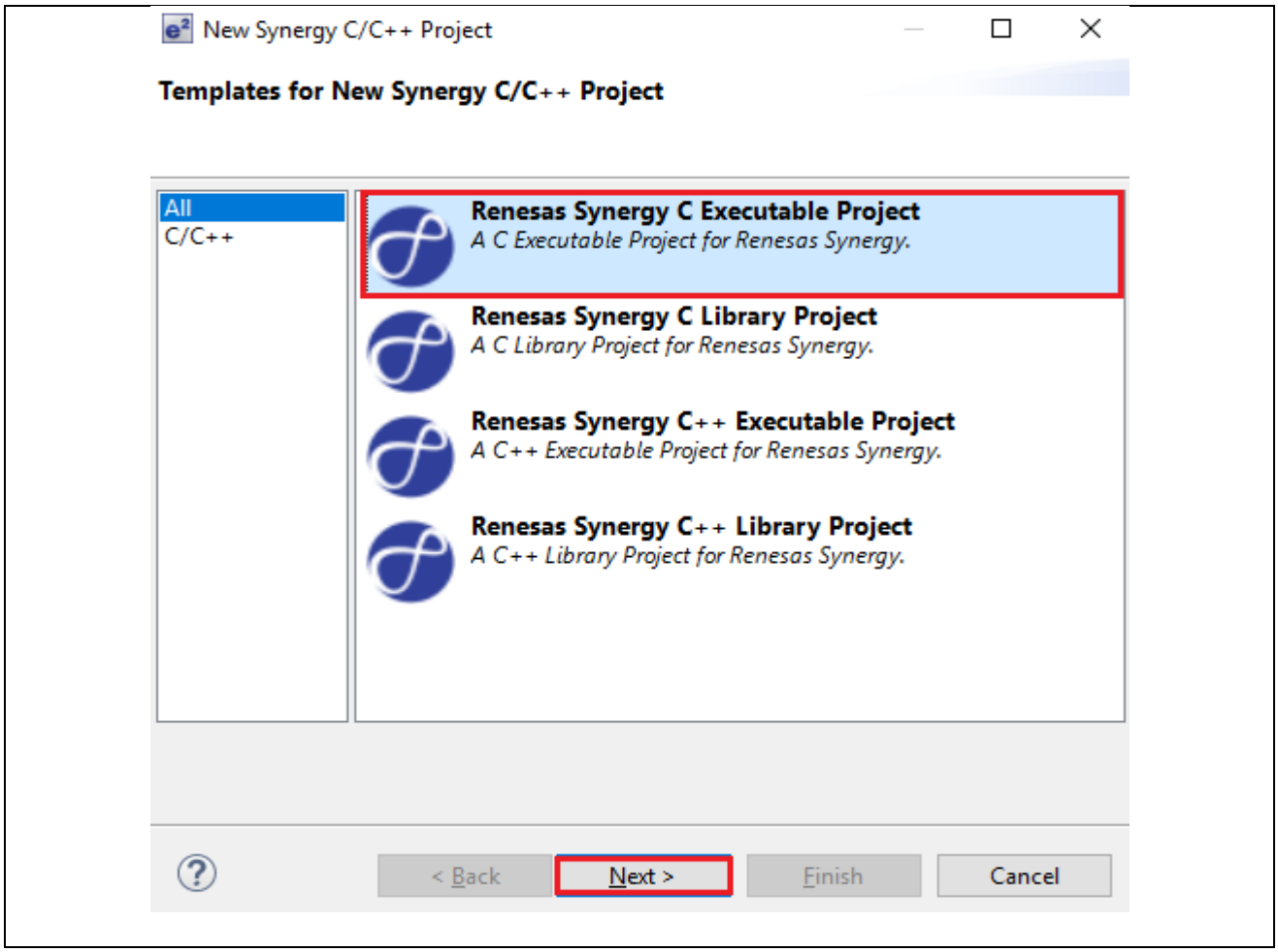

Figure 5. Project type selection

10. If the License file is configured, you see this area of the form. If the license displays, skip to step 11. If the form is empty, do the following steps (A to G).

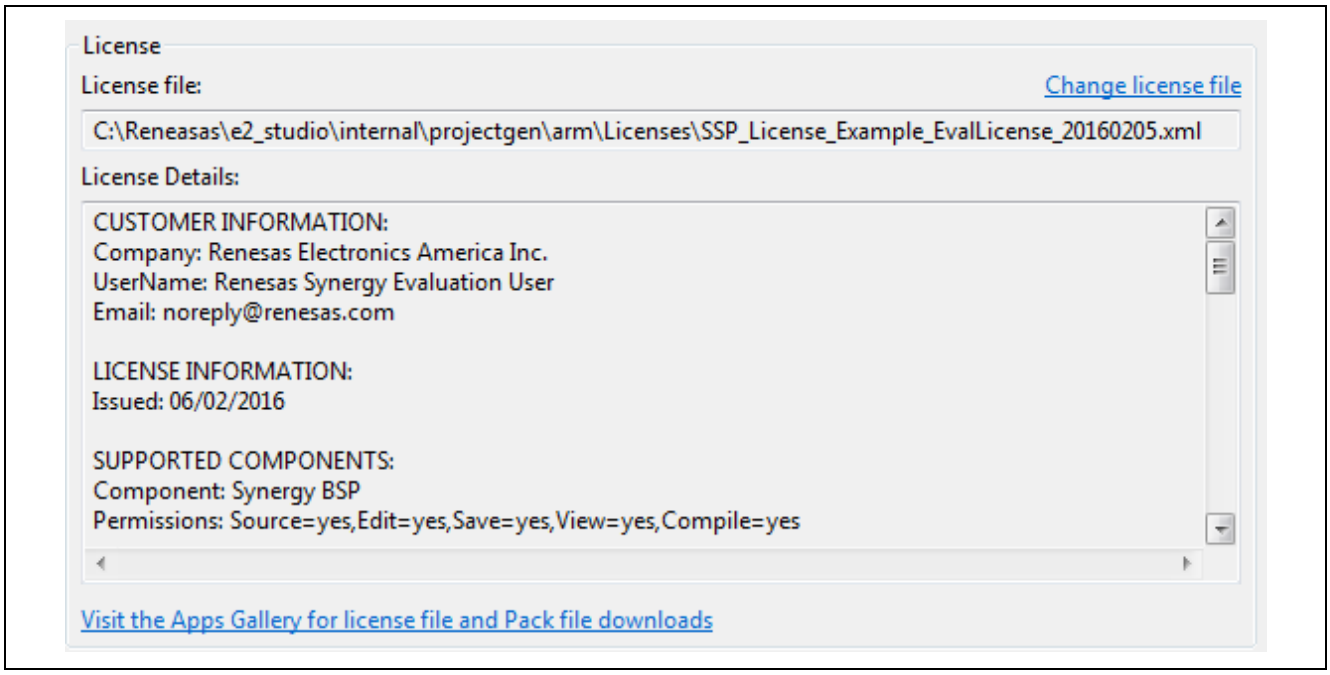

Figure 6. Configured License File

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

Figure 7. Unconfigured license file

B. Click the Browse **…** button. e² studio displays the **Specify License** dialog box.

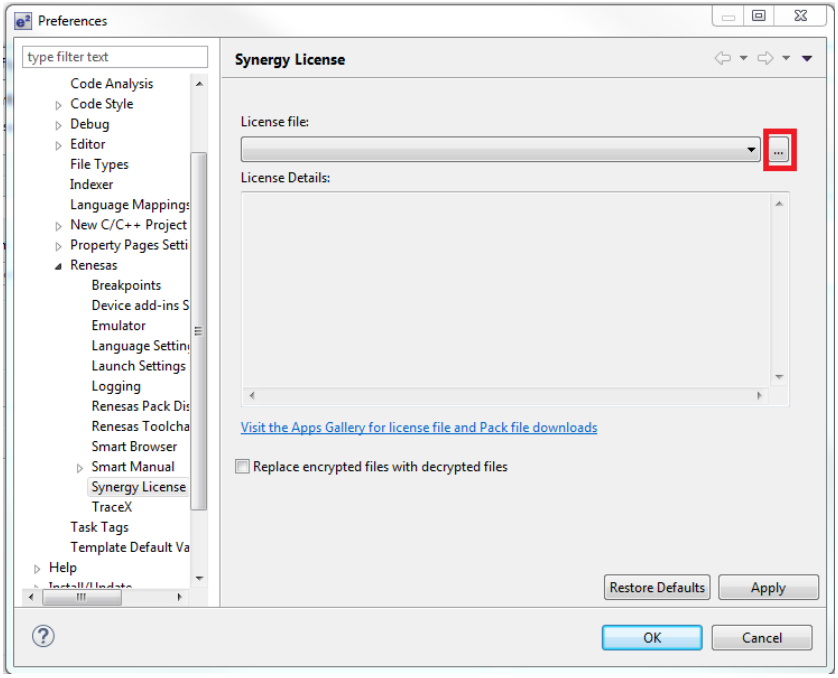

Figure 8. Preferences dialog box with Synergy license configuration

C. Click the Browse … button. The e² studio Open Dialog box and Licenses directory displays.

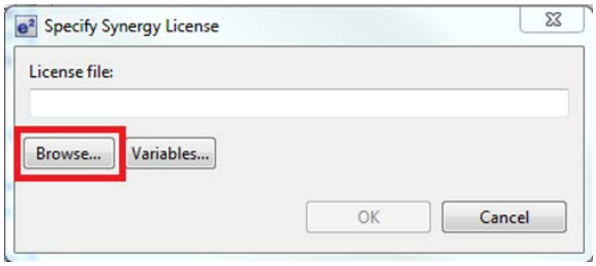

Figure 9. Synergy license dialog box

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

D. Select the **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.

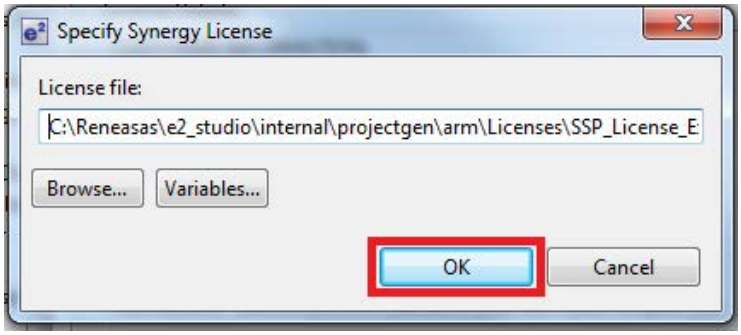

Figure 10. Confirm license file

G. Click **Apply** and then **OK** in the **Preferences** dialog box.

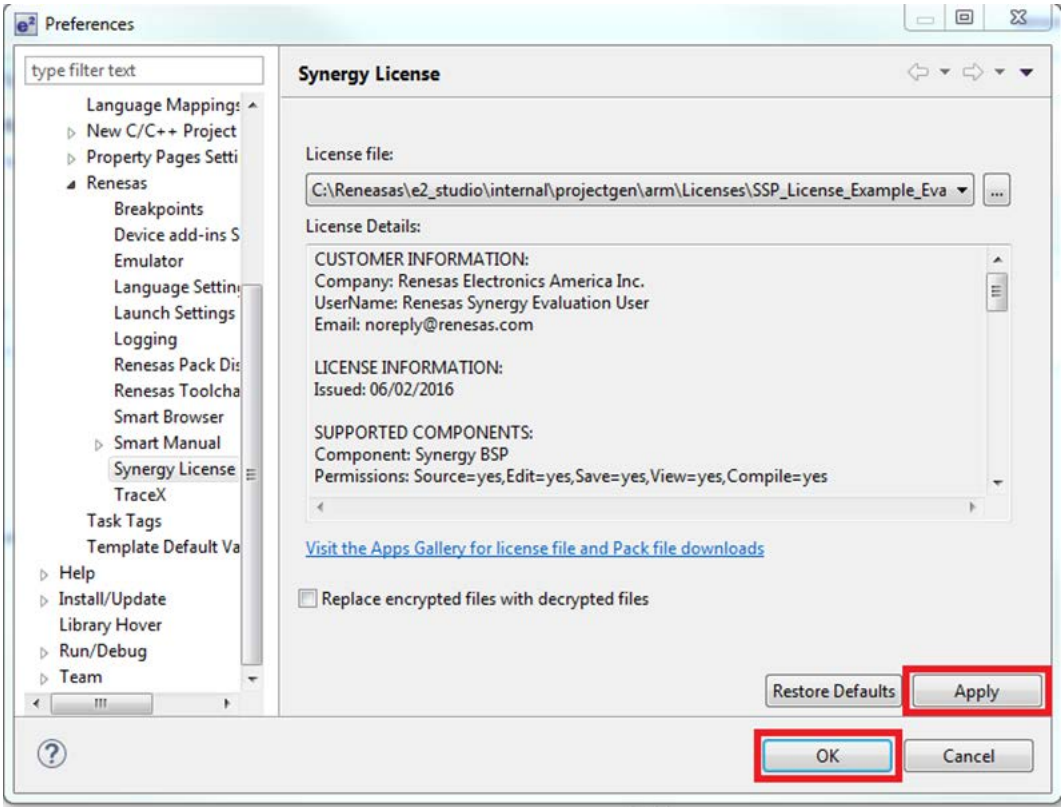

Figure 11. Apply and confirm Synergy license file selection

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

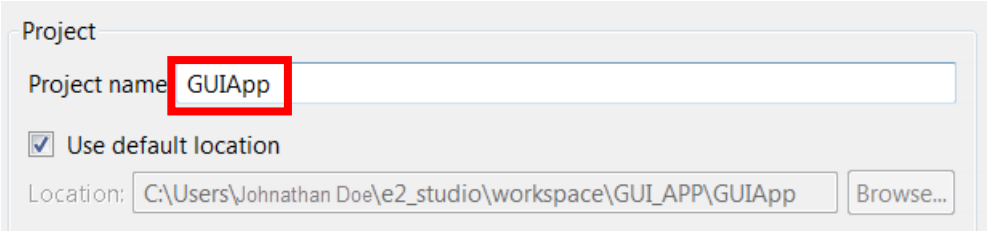

Figure 12. Enter a project name

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

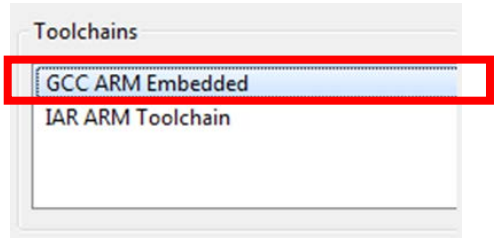

Figure 13. Verify GCC ARM Embedded Toolchain

13. Click the **Next** button to continue.
14. Under **Device Selection** (top left), select **SSP version** as v1.4.0 (or later).
15. For **Board** field, select **S7G2 SK**. The **Device** field updates automatically.

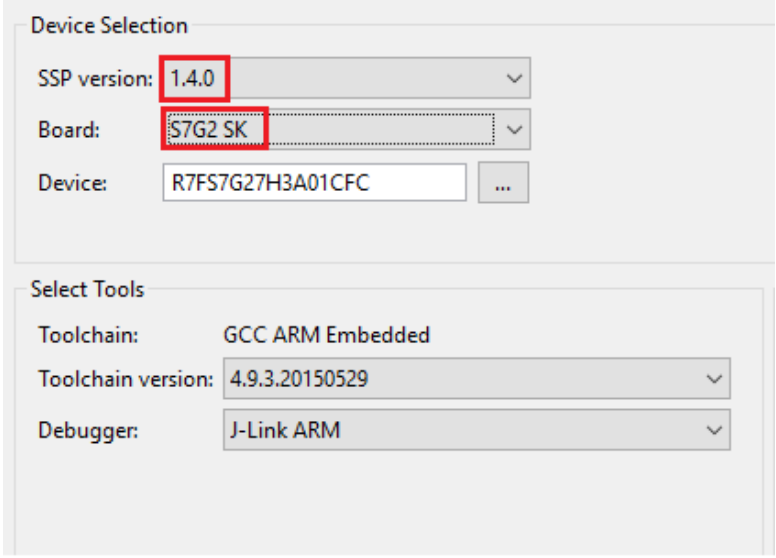

Figure 14. SK-S7G2 Device Selection

16. For the Board field, select **S5D9 PK** if using PK-S5D9 board. The Device field updates automatically.

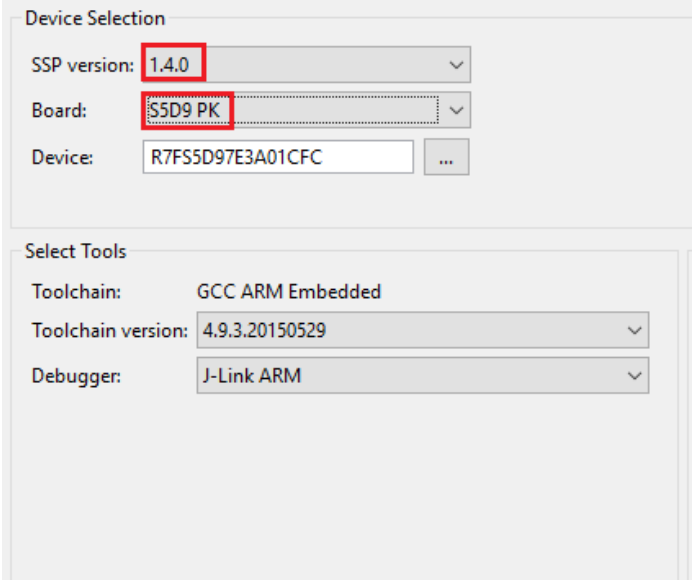

Figure 15. PK-S5D9 Device selection

17. Click the **Next button** to continue.
18. In the **Project Configuration** dialog, select the option **BSP**.

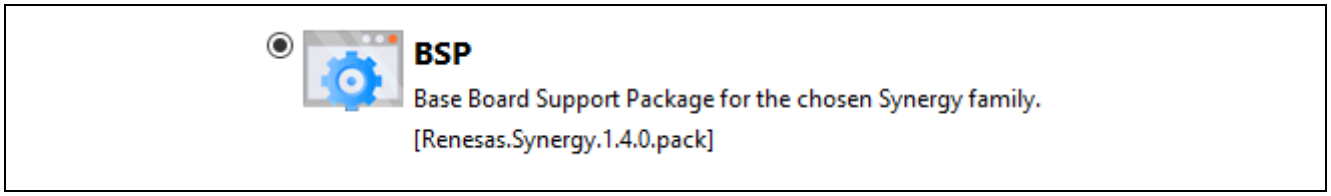

Figure 16. Select BSP

19. Click the **Finish** button.
20. If you have not directed e² studio to remember your perspectives, e² studio displays the **Open Associated Perspective?** dialog box. If opened, click **Yes** to acknowledge and close.

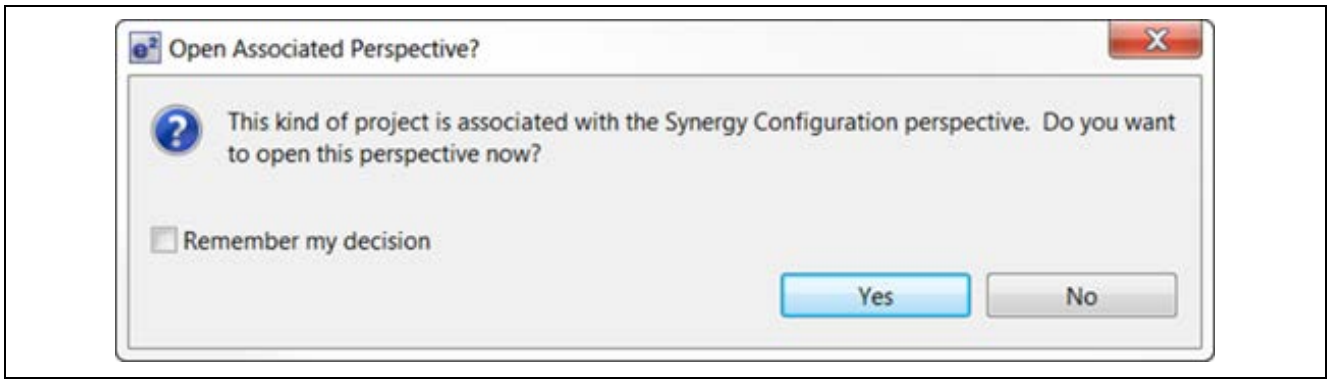

Figure 17. Open Associated Perspective dialog box

When the project is created, the following screen displays.

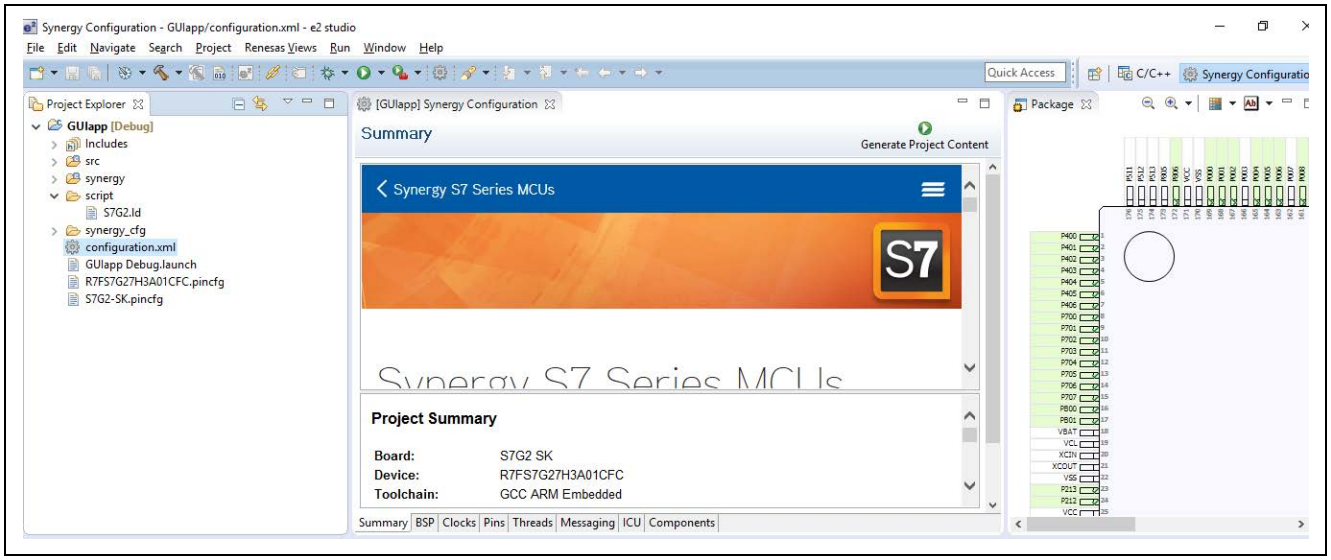

Figure 18. GUIApp project

Note: The settings applicable for PK-S5D9 Synergy MCU Group are the same as SK-S7G2 Synergy MCU Group unless explicitly specified.

4. Configuring the project in e² studio

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

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

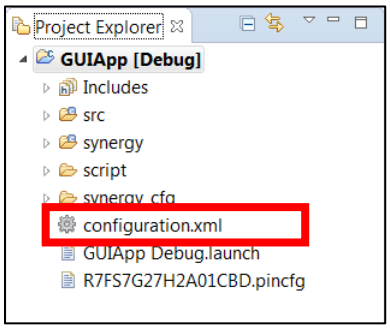

Figure 19. Selecting the configuration.xml file in Project Explorer

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

Figure 20. Synergy Configuration Threads tab

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

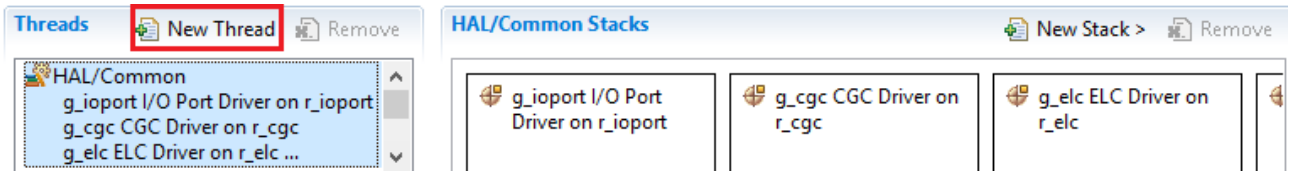

Figure 21. Create a New Thread

4. Click **New Thread** to display the properties.
5. Edit the **Properties** to match the following.

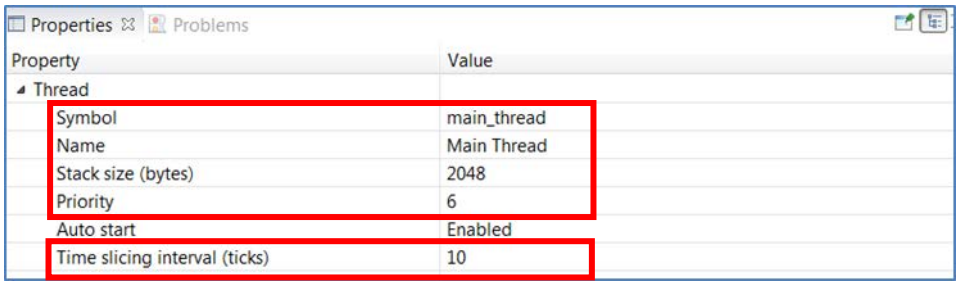

Figure 22. Configure Main Thread Properties

6. Back in the Synergy **Configuration** window, **Threads** tab, **Main Thread Stacks** area, click **New**.

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

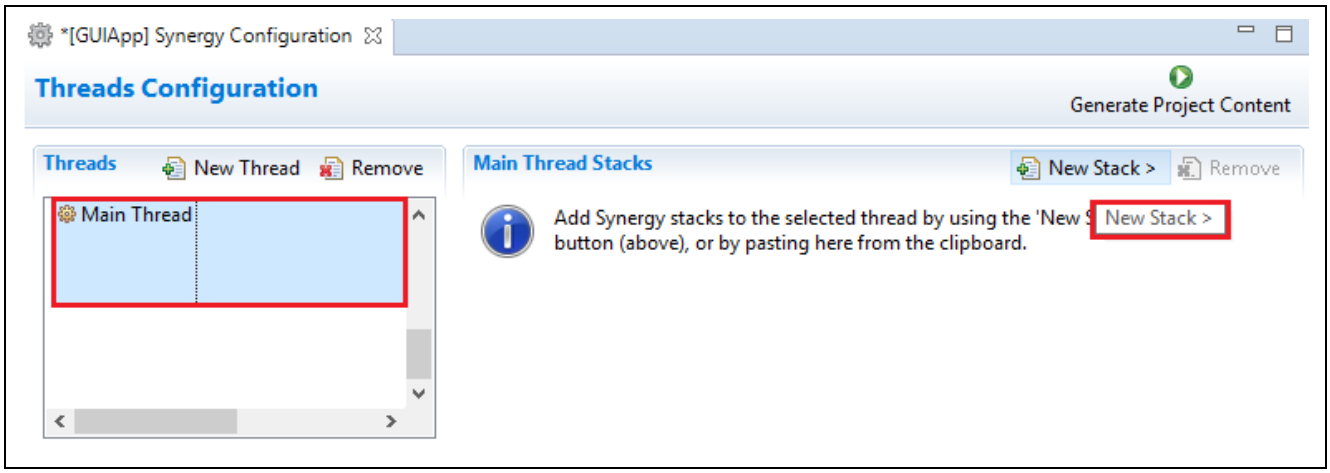

Figure 23. Main Thread Stacks

7. Add a framework for the **Touch Panel** by selecting **New Stack** > **Framework** > **Input** > **Touch Panel Framework on sf_touch_panel_i2c**.

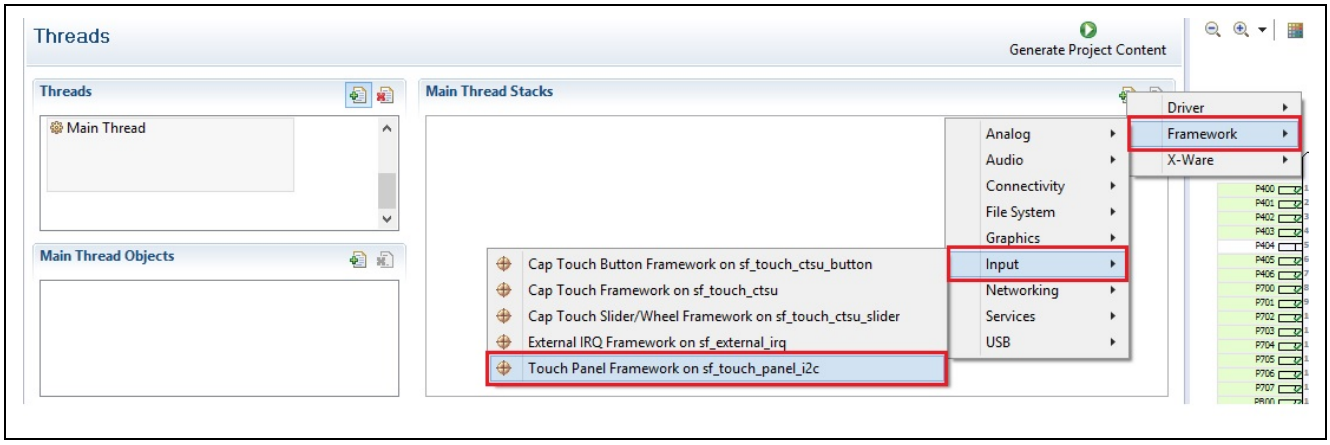

Figure 24. Adding Touch Panel framework

8. Configure the following properties.

Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_sf_touch_panel_i2c Touch Panel Framework on sf_touch_panel_i2c	
Name	g_sf_touch_panel_i2c
Thread Priority	3
Hsize Pixels	240
Vsize Pixels	320
Update Hz	10
Reset Pin	IOPORT_PORT_06_PIN_09
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 25. Configure Touch Panel properties

Notice that the Synergy Configurator has already created the message framework, external IRQ framework, and has a placeholder for the external IRQ and I²C driver stacks (see Figure 25). The messaging framework is used by other framework layers and tasks to pass messages around the system. This system is used to pass data from the touch screen driver to the Main Task to handle touch inputs. The SF External Interrupt is a framework layer used by the touch controller driver as shown below.

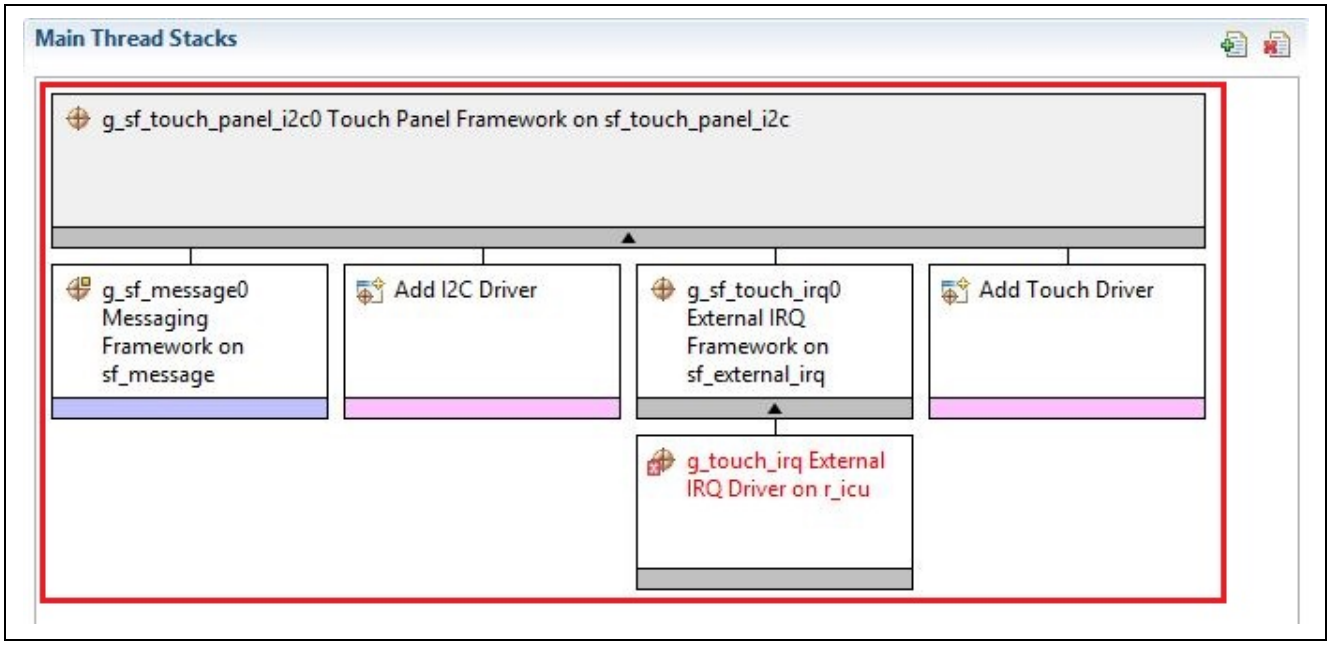

Figure 26. Touch Panel Framework Stack

9. Select the **External IRQ Framework on sf_external_irq** and configure the following properties.

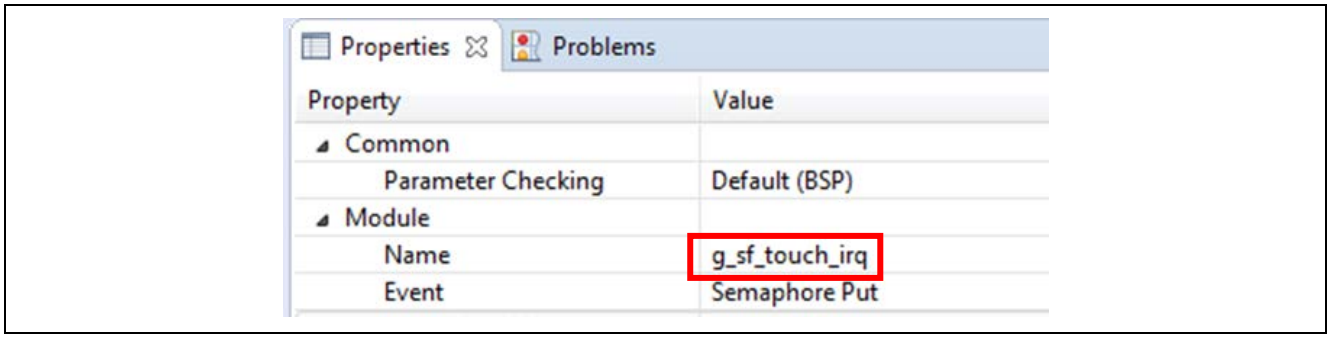

Figure 27. Configuring External Interrupts Properties

10. Select **External IRQ Driver on r_icu**. Configure the following properties for the new module. Change the **Channel** first.

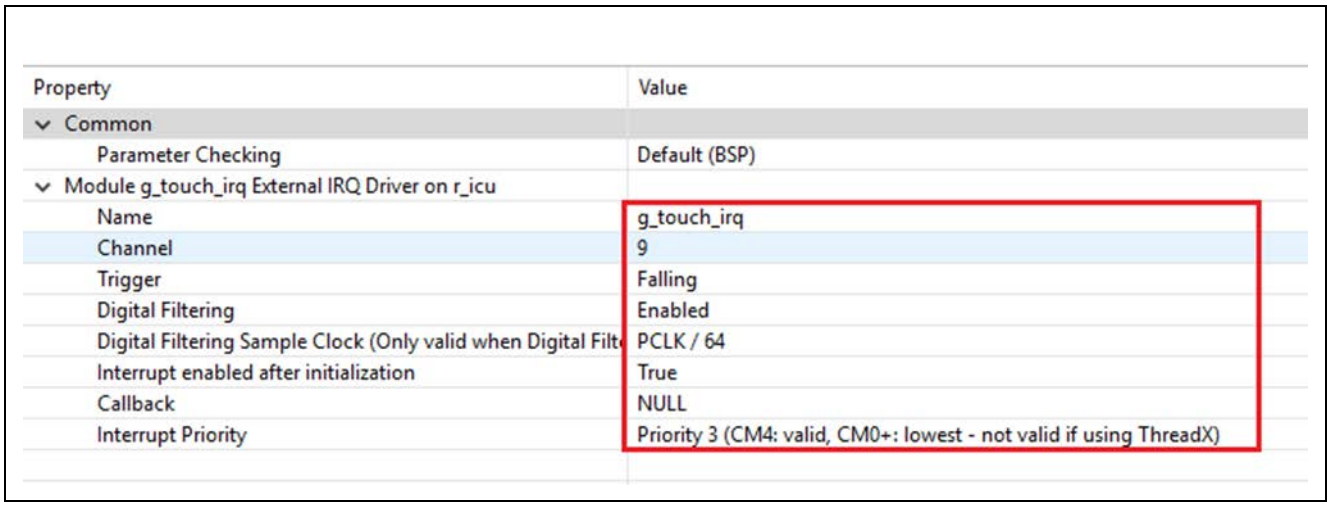

Figure 28. Touch screen IRQ properties

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

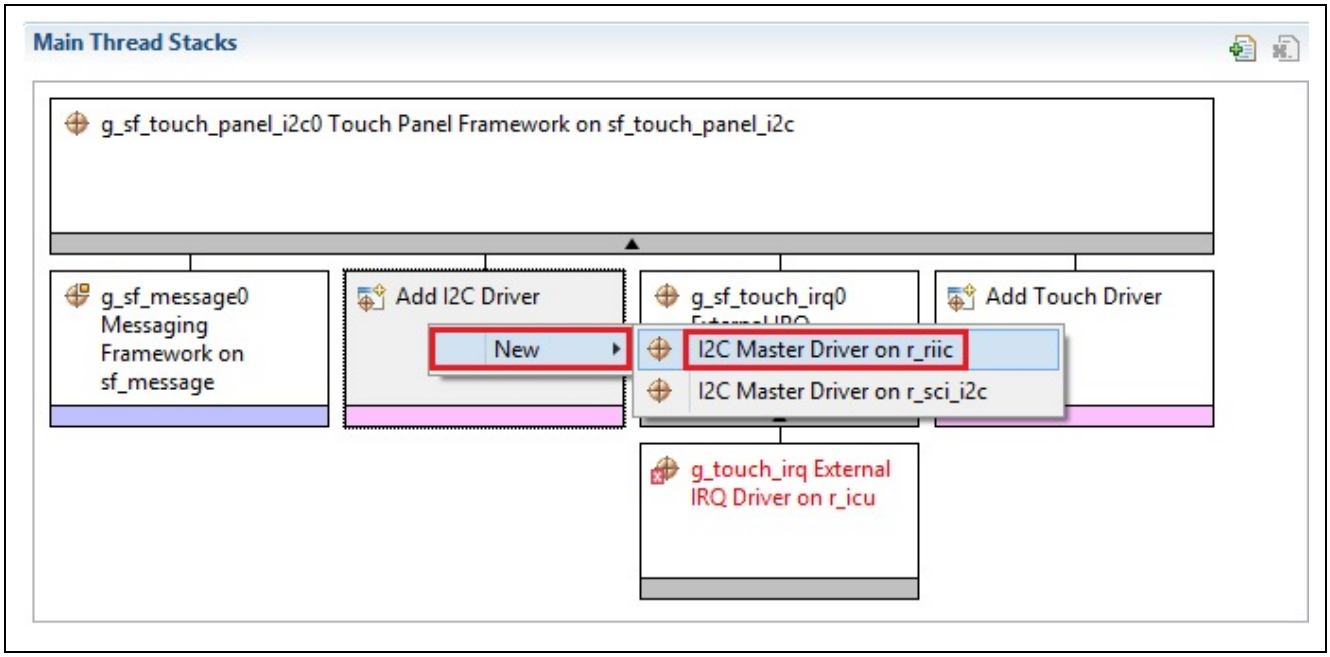

Figure 29. Adding I²C Driver

12. Configure the following properties for **I2C Master Driver on r_riic**. **Hint: Change the Channel option first!**

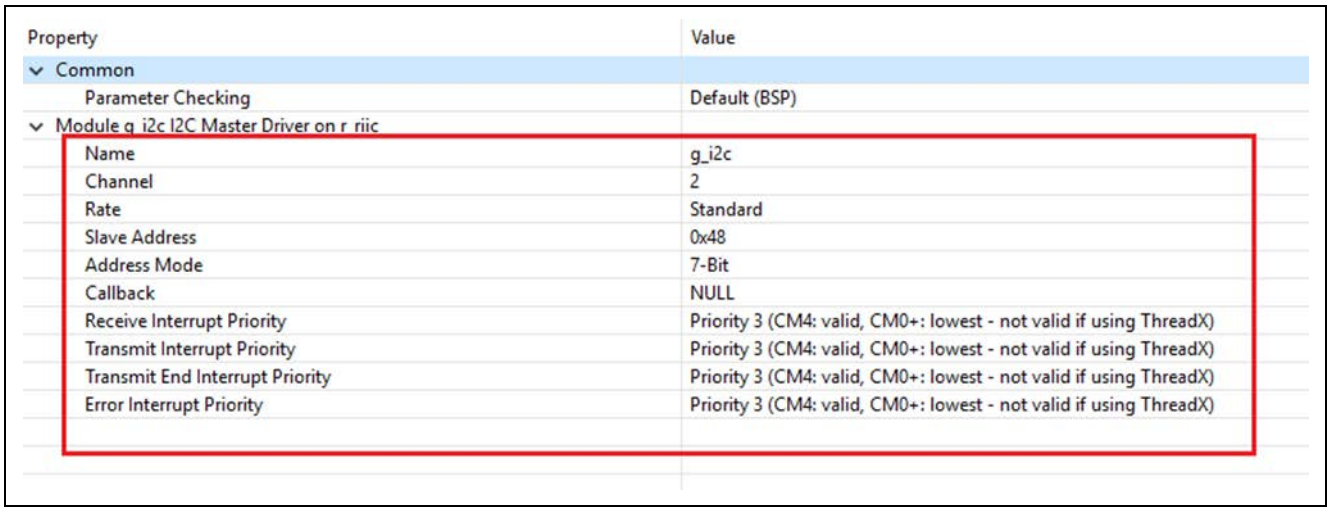

Figure 30. Configuring I²C Driver

13. Configure the following properties of **Touch Driver**.

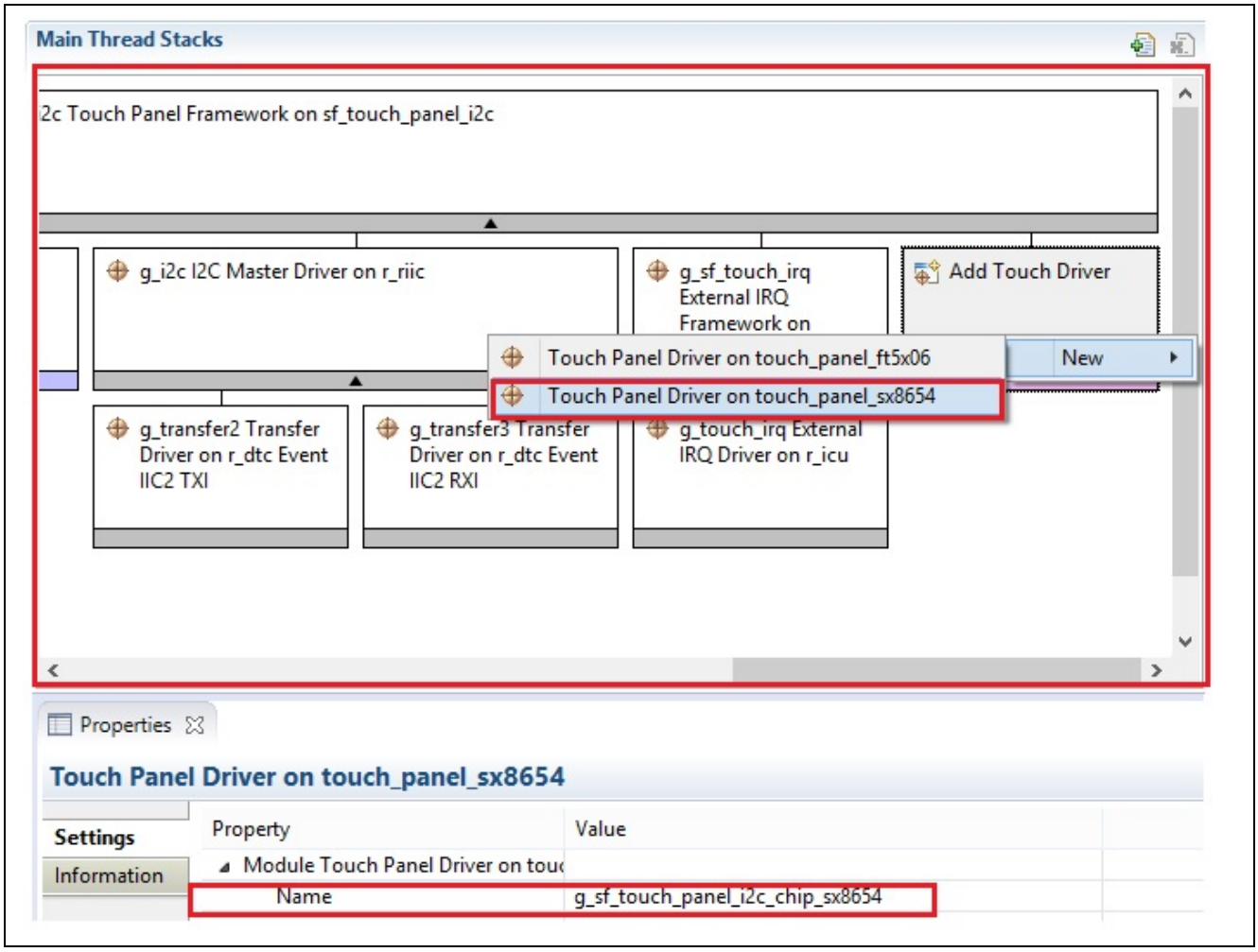

Figure 31. Configuring Touch Driver

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

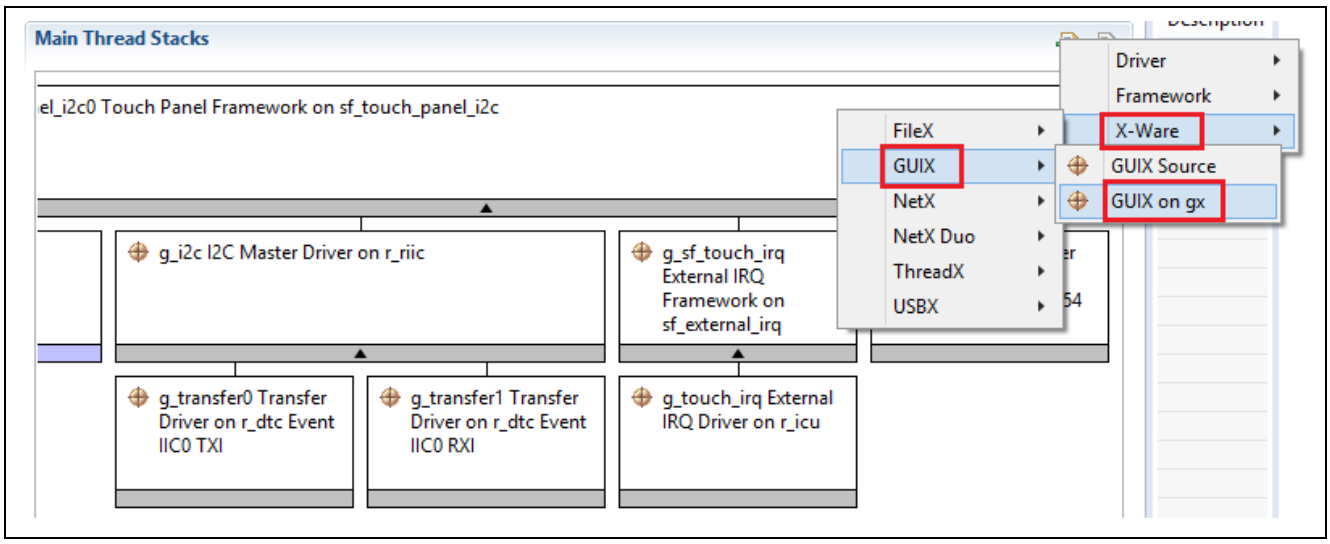

Figure 32. Adding Framework for GUIX on gx

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

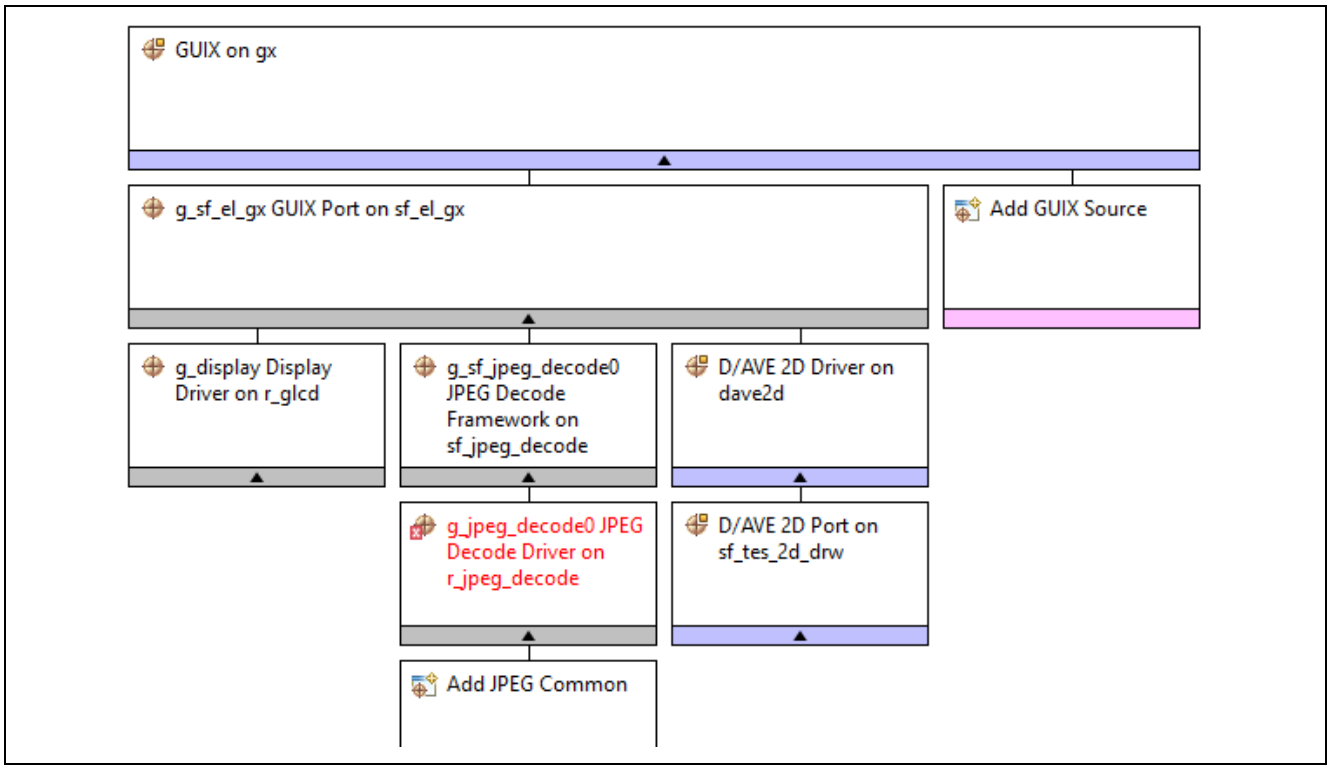

Figure 33. GUIX on gx stack

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

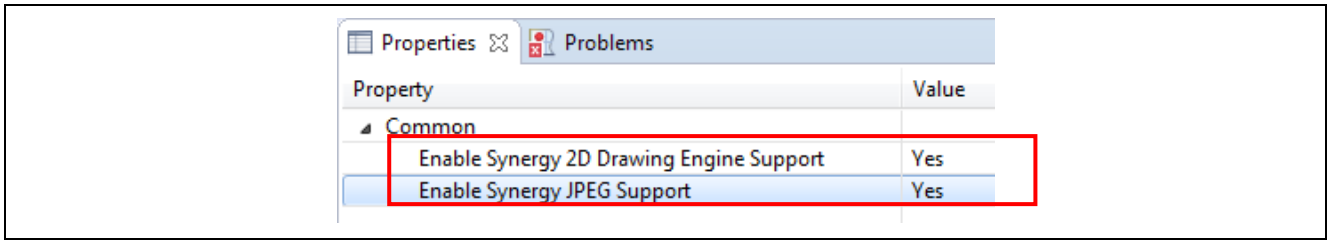

Figure 34. GUIX on gx Properties

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

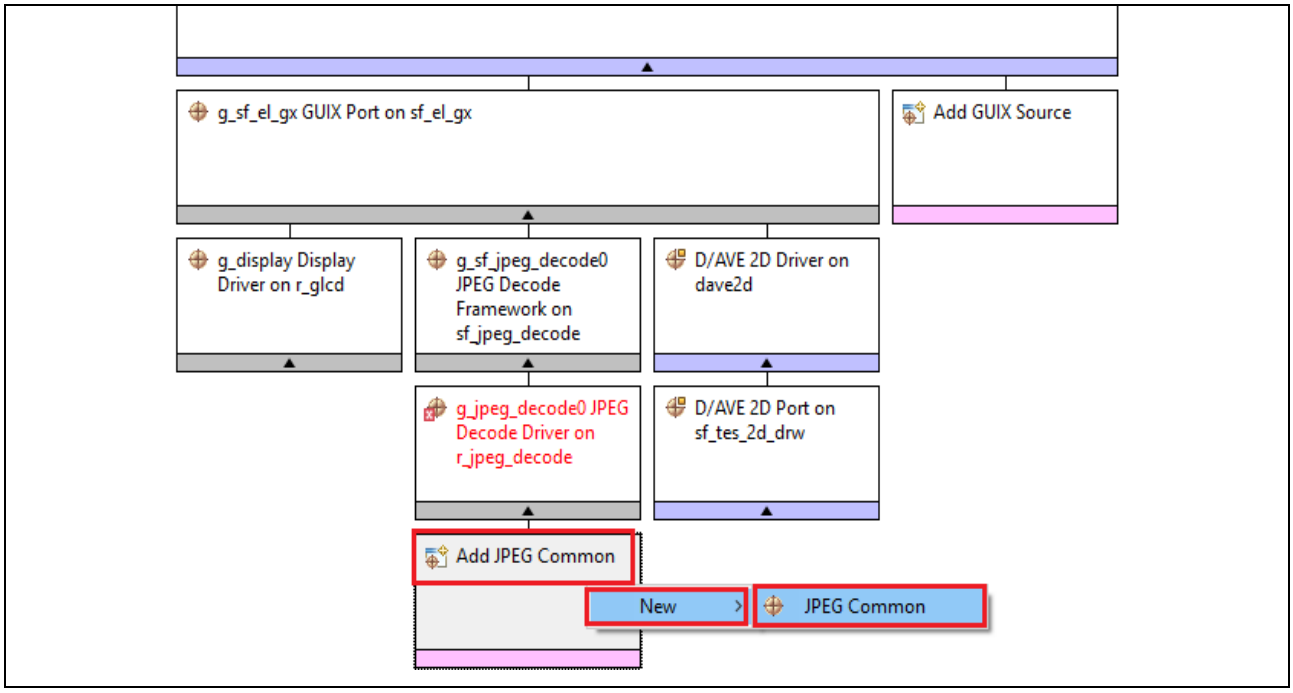

Figure 35. JPEG Common module

17. Select **GUIX Port** on **sf_el_gx** and configure the following **Properties**.

Property	Value
Common	
Parameter Checking	Enabled
Module g_sf_el_gx GUIX Port on sf_el_gx	
Name	g_sf_el_gx
Display Driver Configuration Inheritance	Inherit Graphics Screen 1
Name of User Callback function	NULL
Screen Rotation Angle(Clockwise)	0
GUIX Canvas Buffer (required if rotation angle is not zero)	Not used
Size of JPEG Work Buffer (valid if JPEG hardware acceleration enabled)	81920
Memory section for GUIX Canvas Buffer	sdram
Memory section for JPEG Work Buffer	bss

Figure 36. Configure GUIX Port property

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

Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_jpeg_decode0 JPEG Decode Driver on r_jpeg	
Name	g_jpeg_decode0
Byte Order for Input Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8)
Byte Order for Output Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8)
Output Data Color Format	Pixel Data RGB565 format
Alpha value to be applied to decoded pixel data(only valid for 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 37. JPEG Decode Driver on r_jpeg properties

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

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

Figure 38. D/AVE 2D Port Properties

20. Under **Main Thread Stacks**, select **Display Driver on r_glcd** and configure the following **Interrupt Properties**.

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	Priority 3 (CM4: valid, CM0+: lowest - not valid if using ThreadX)

Figure 39. Interrupt Properties

21. Configure the **Graphics Screen 1** following properties.

Module g_display 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	bss
Input - Graphics screen1 input horizontal size	256
Input - Graphics screen1 input vertical size	320
Input - Graphics screen1 input horizontal stride(not bytes but pixels)	256
Input - Graphics screen1 input format	16bits RGB565
Input - Graphics screen1 input line descending	Not used
Input - Graphics screen1 input lines repeat	Off
Input - Graphics screen1 input lines repeat times	0
Input - Graphics screen1 layer coordinate X	0
Input - Graphics screen1 layer coordinate Y	0
Input - Graphics screen1 layer background color alpha	255
Input - Graphics screen1 layer background color Red	255
Input - Graphics screen1 layer background color Green	255
Input - Graphics screen1 layer background color Blue	255
Input - Graphics screen1 layer fading control	None
Input - Graphics screen1 layer fade speed	0

Figure 40. Graphics Screen 1 properties

22. Configure the Output following properties.

Output - Horizontal total cycles	320
Output - Horizontal active video cycles	240
Output - Horizontal back porch cycles	6
Output - Horizontal sync signal cycles	4
Output - Horizontal sync signal polarity	Low active
Output - Vertical total lines	328
Output - Vertical active video lines	320
Output - Vertical back porch lines	4
Output - Vertical sync signal lines	4
Output - Vertical sync signal polarity	Low active
Output - Format	16bits RGB565
Output - Endian	Little endian
Output - Color order	RGB
Output - Data Enable Signal Polarity	High active
Output - Sync edge	Rising edge
Output - Background color alpha channel	255
Output - Background color R channel	0
Output - Background color G channel	0
Output - Background color B channel	0

Figure 41. Output Screen 2 properties

23. Configure the following **TCON** pins and **clock**.

TCON - Hsync pin select	LCD_TCON2
TCON - Vsync pin select	LCD_TCON1
TCON - DataEnable pin select	LCD_TCON0
TCON - Panel clock division ratio	1/32

Figure 42. TCON settings

24. Under **Main Thread Stacks**, select **New Stack > Driver > Connectivity > SPI Driver on r_sci_spi**.

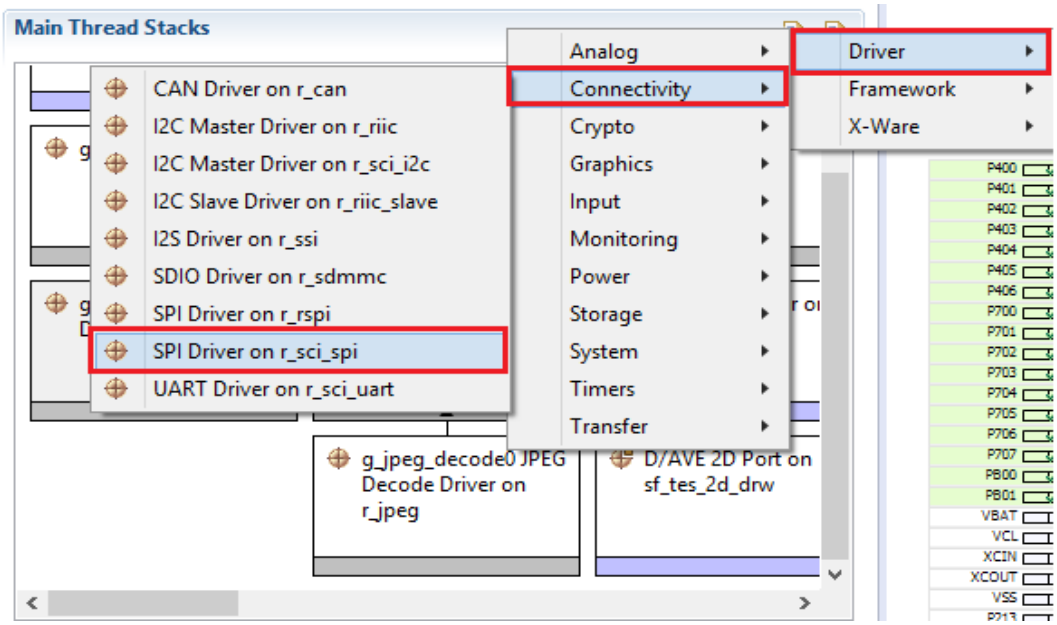

Figure 43. Adding Simple SPI (on SCI) Driver

25. Configure the following properties.

Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_spi_lcdc SPI Driver on r_sci_spi	
Name	g_spi_lcdc
Channel	0
Operating Mode	Master
Clock Phase	Data sampling on even edge, data variation on odd edge
Clock Polarity	High when idle
Mode Fault Error	Disable
Bit Order	MSB First
Bitrate	100000
Bit Rate Modulation Enable	Enable
Callback	g_lcd_spi_callback
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)
Error Interrupt Priority	Priority 3 (CM4: valid, CM0+: lowest - not valid if using ThreadX)

Figure 44. Configure Simple SPI (on SCI) properties

26. Click each **g_transfer** drive and remove it by clicking **Remove** since it is not needed for the LCD.

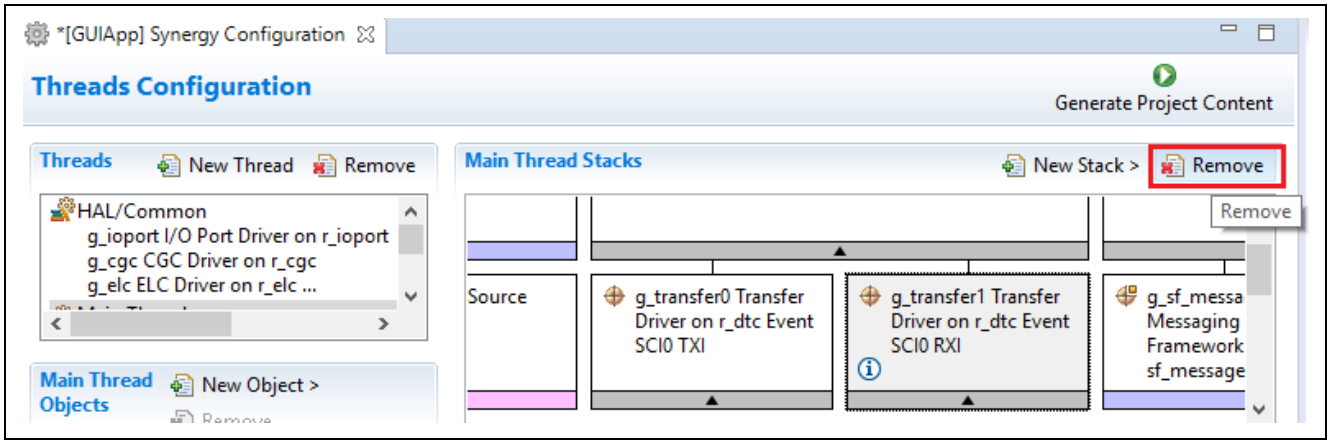

Figure 45. Remove Transfer Drivers

27. After removing the drivers, the placeholders for adding drivers remain as shown in the following figure.

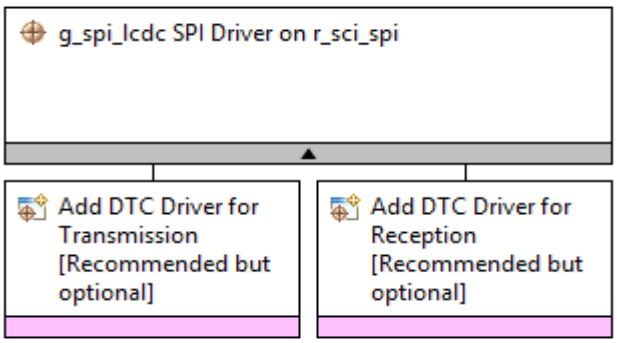

Figure 46. Transfer Drivers Placeholders

28. In the Synergy Configuration window, **Threads** tab, make sure the **Main thread** is still selected.

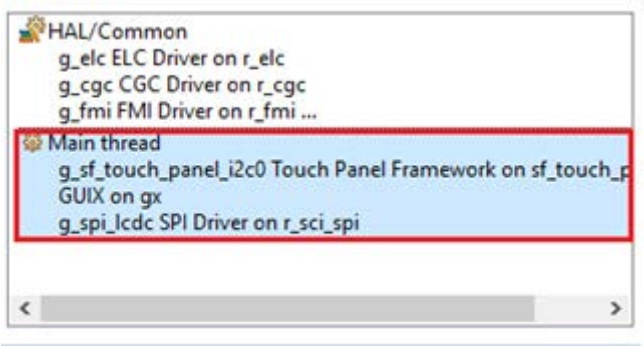

Figure 47. Click on Main thread

29. Under the **Main thread Objects**, click **New Object > Semaphore**.

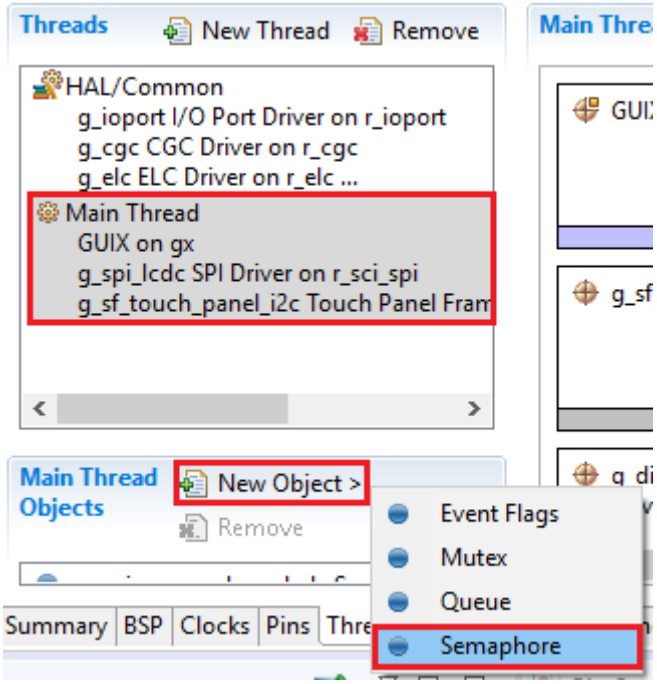

Figure 48. Add a Semaphore

30. Configure the following properties.

Figure 49. Configure Semaphore

31. In the Synergy Configuration window, select the **Pins** tab

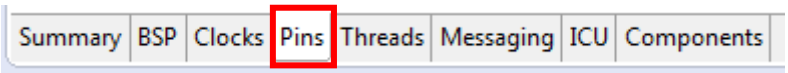

Figure 50. Configuration pins

32. Select **Peripherals** > **Connectivity:SPI** > **SPI0** in **Pin Selection**, and change **Operation Mode** to **Disabled** in Pin Configuration of SPI0. This must be disabled to free the pins it shares with the SCI module.

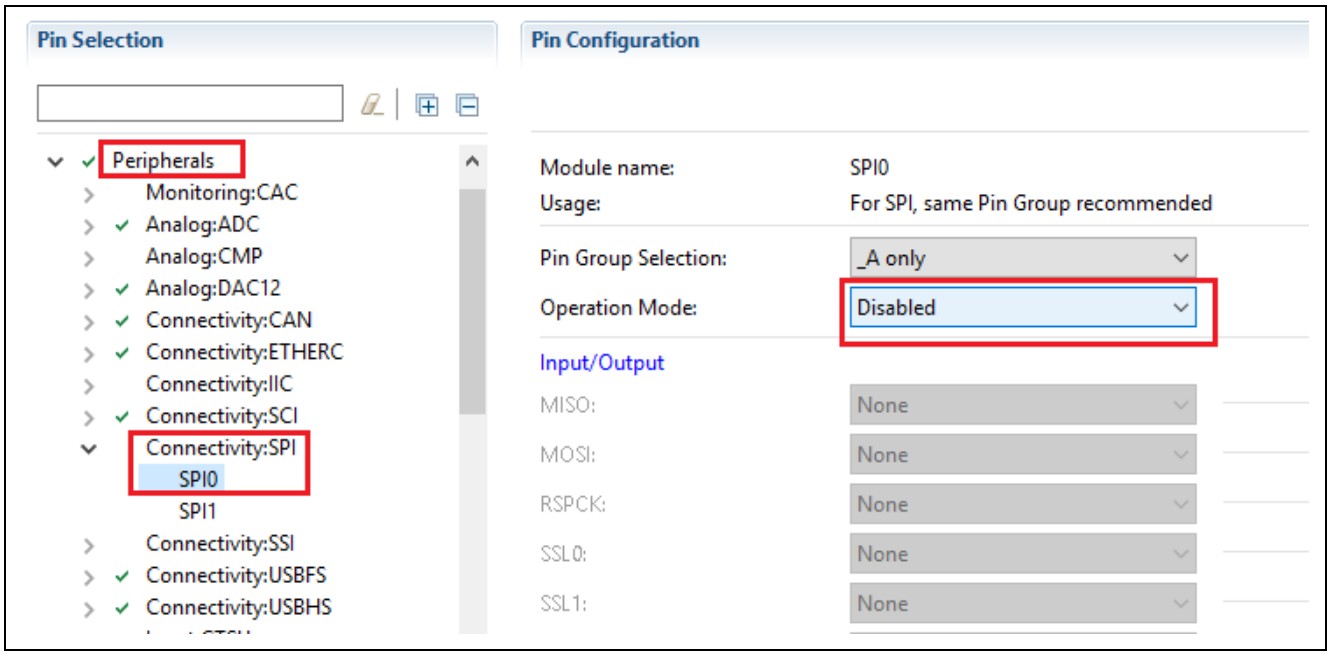

Figure 51. Disable SPI0_Pin_Option_A in Pin Configuration

33. Select **Peripherals** > **Connectivity:SCI**> **SCI0** in **Pin Selection**, and make the following configuration in **Pin Configuration** of the **SCI0** module.

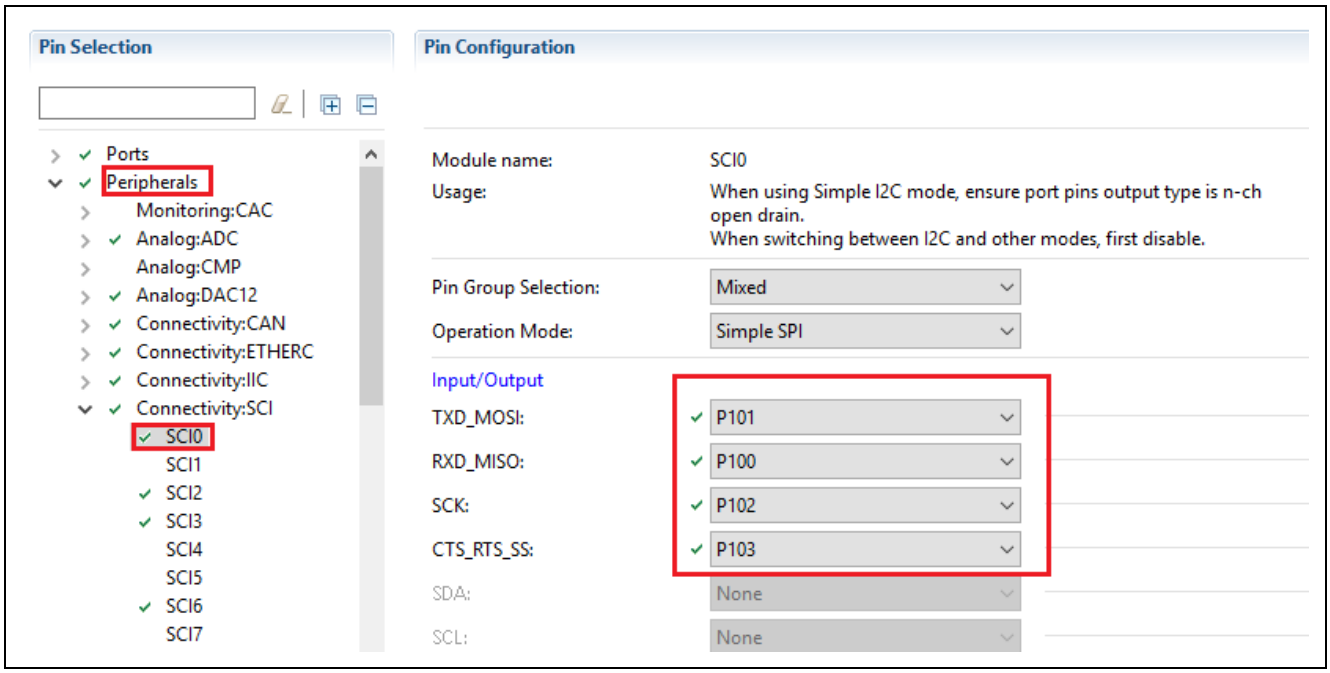

Figure 52. Configure SCI0 Pin Configuration

34. Select **Peripherals** > **Connectivity: IIC** > **IIC2** as the Pin Selection and enable the **IIC2** module in the Pin Configuration.

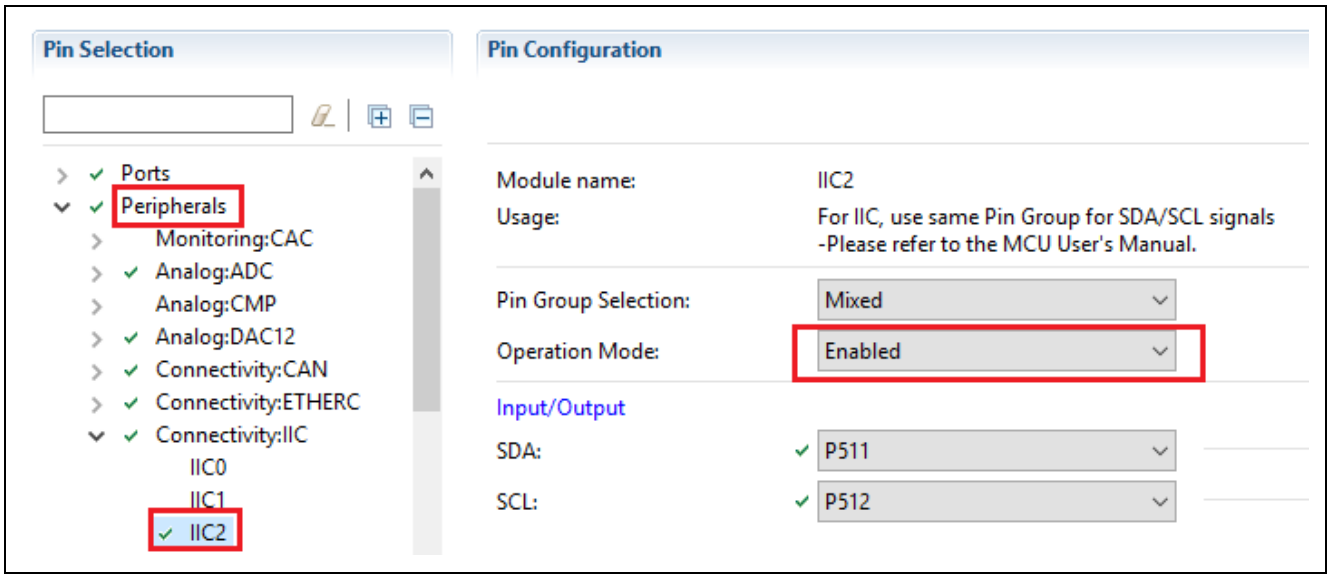

Figure 53. Configure IIC2 Pin Configuration

35. Select **Ports** > **P1** > **P115** in **Pin Selection**, and configure GPIO in **Pin Configuration**. This pin is connected with the LCD panel on the SK-S7G2 board to control data access timing from LCD_WR signal.

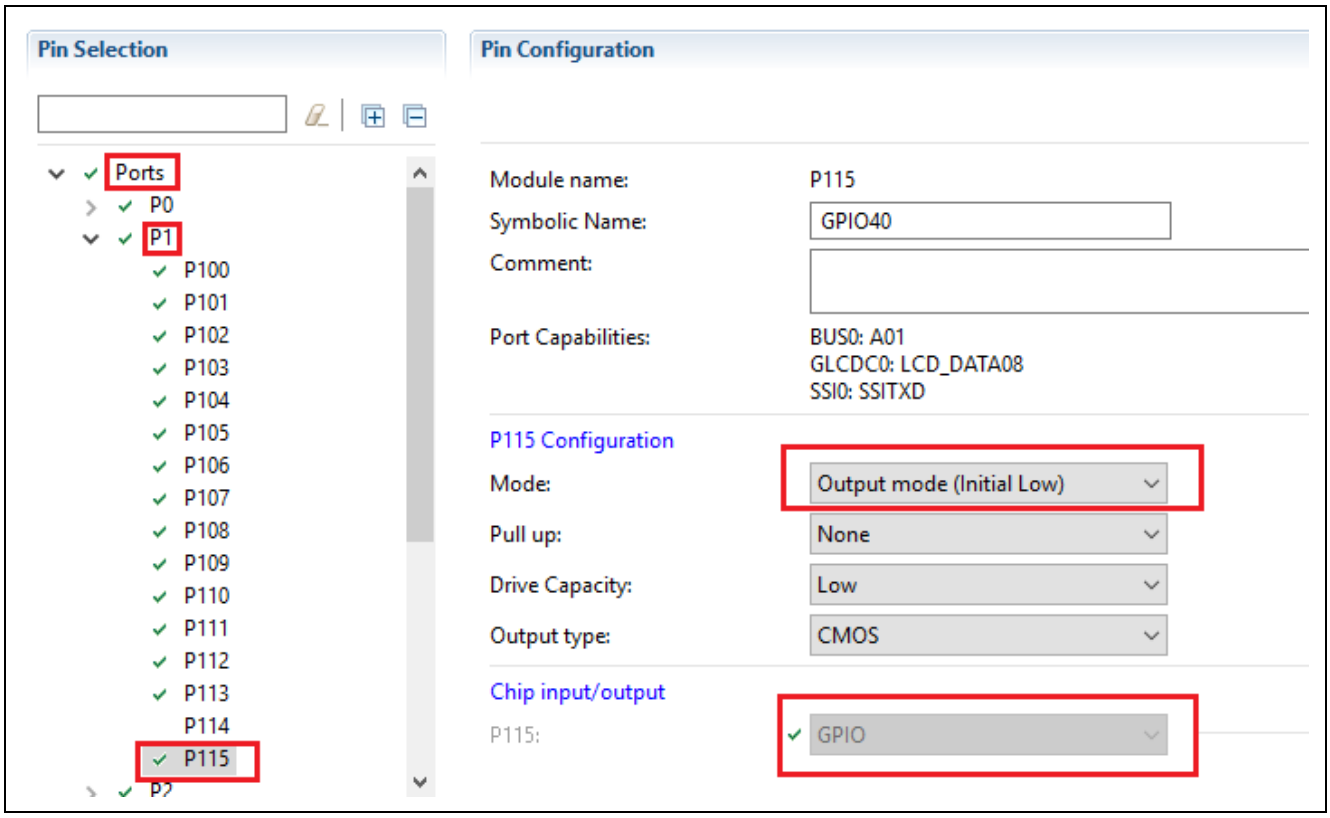

Figure 54. P115 configuration

36. Select **Ports** > **P6** in **Pin Selection** and configure **P609** (RESET# for Touch Panel), **P610** (LCD_RESET), and **P611** (LCD_CS) with output mode of GPIO.

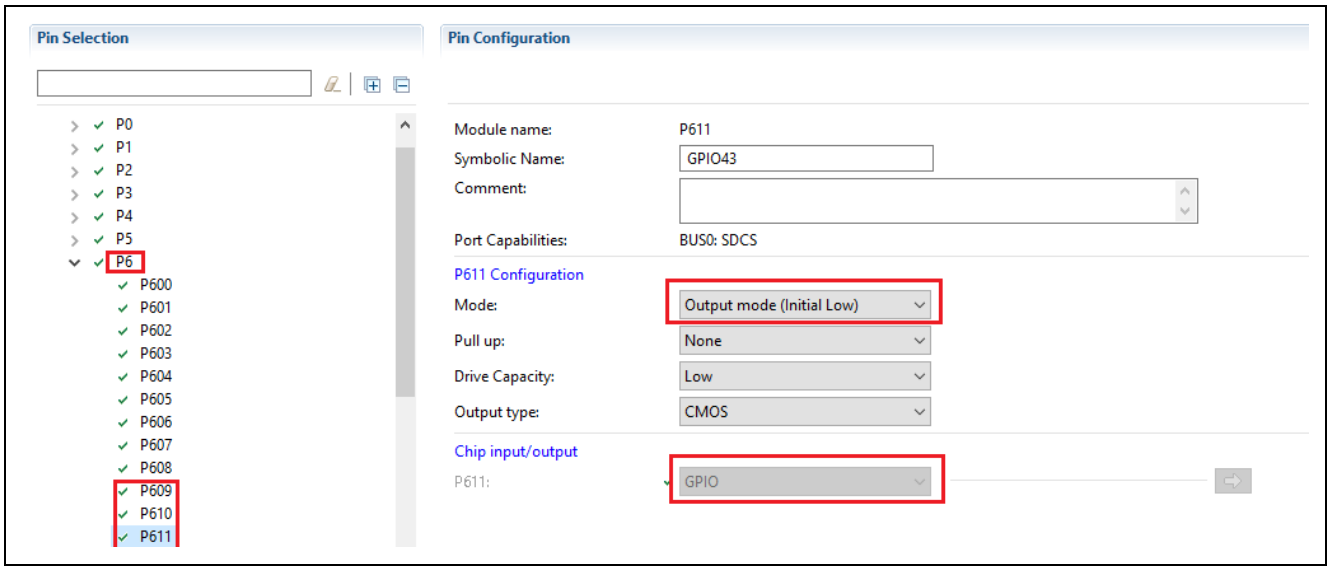

Figure 55. P609, P610 and P611 configurations

37. Configure **Drive Capacity** into **High** for all pins related to **GLCD_Controller_Pin_Option_B** as shown in Figure 56.

    There are two methods for setting the **Drive Capacity to High**. You may pick either one (A or B).

    A. You can confirm which pins would be used for **GLCD_Controller_Pin_Option_B** by referring to Figure 56 through Figure 58.

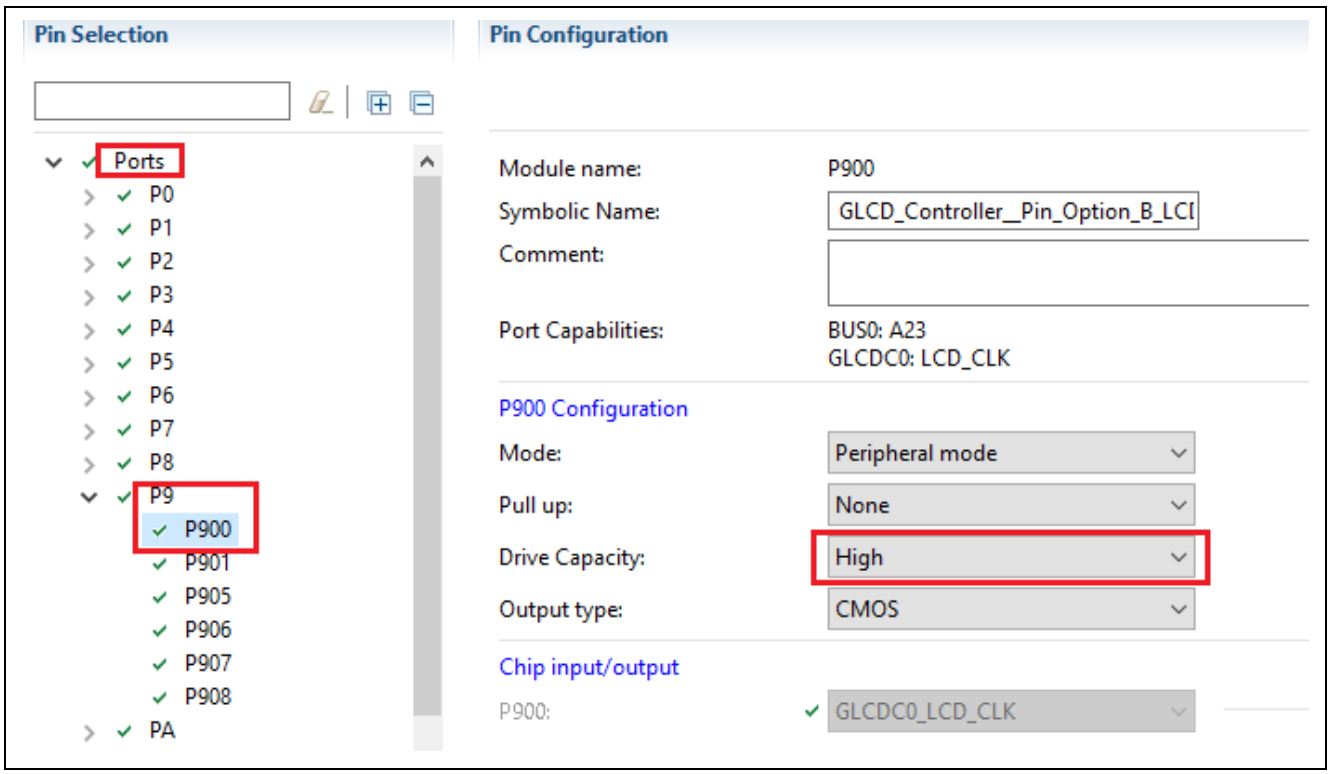

Figure 56. Example of Drive Capability configuration for GLCDC

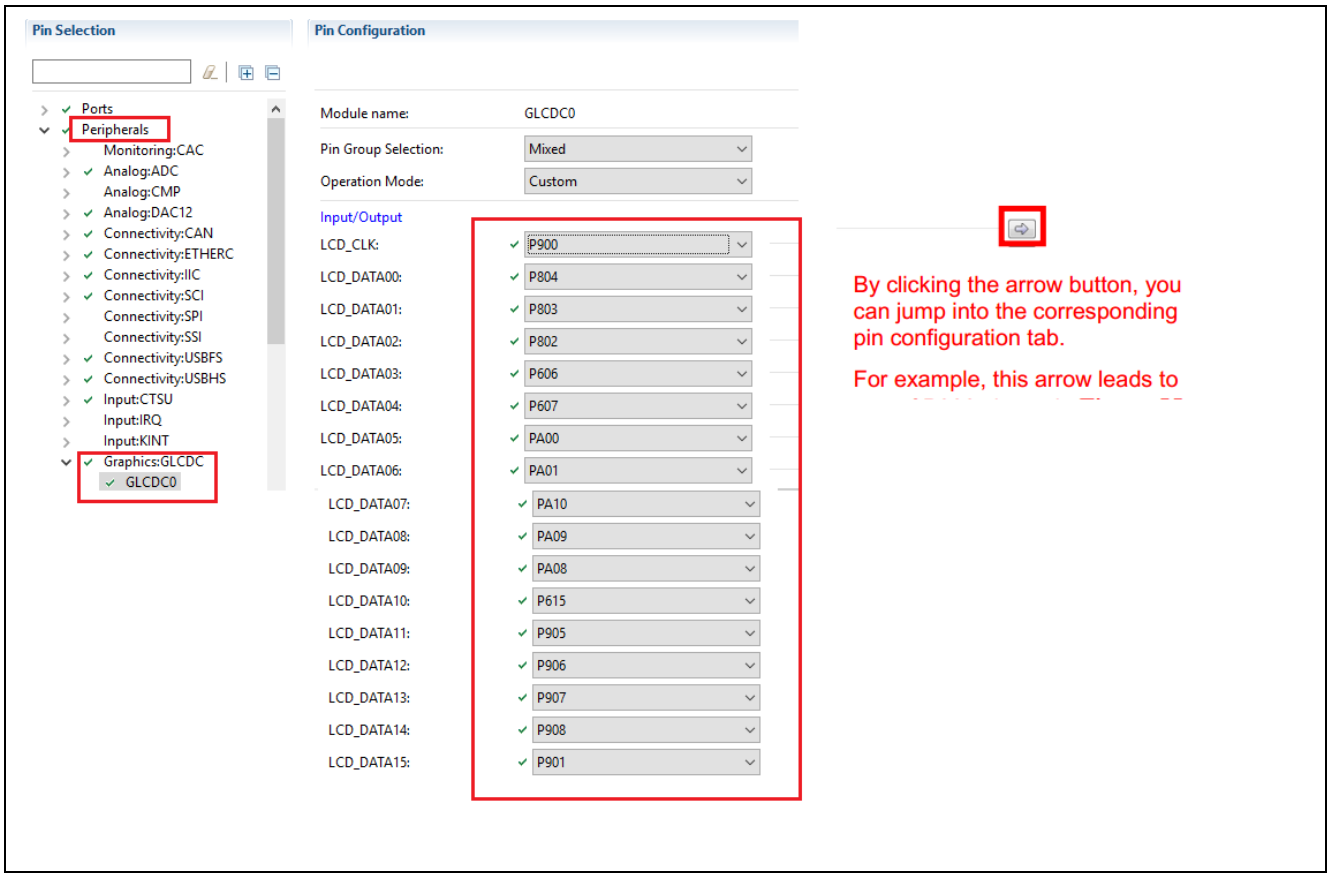

Figure 57. Pin assignment for GLCD_Controller_Pin_Option_B

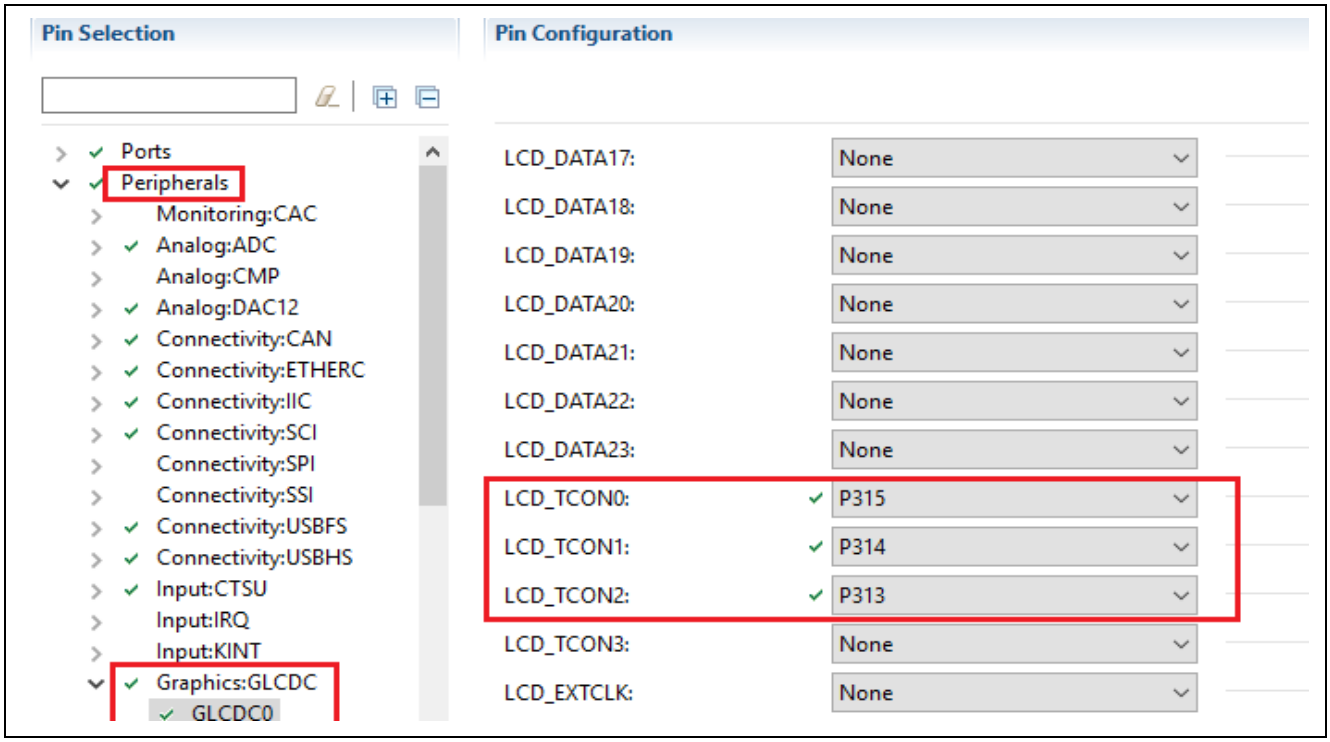

Figure 58. Pin assignment for GLCD_Controller_Pin_Option_B (continued)

B. You can also set the pins by port. Below is an ordered list of the pins that the **Drive Capacity** needs to be set on **High**. You can access these ports by going to **Ports** > **PX** > **PXYZ**. Where X is the second digit of the port from the list, and PXYZ is the entire port. Once the port is selected, set the **Drive Capacity** to **High** as shown in Figure 59.

S7G2 Pin
P313
P314
P315
P606
P607
P615
P802
P803
P804
P900
P901
P905
P906
P907
P908
PA00
PA01
PA08
PA09
PA10

Figure 59. Ordered list of ports to configure as high drive capacity

38. Select the **Messaging** tab on the **Synergy Configuration** window as shown in the following figure.

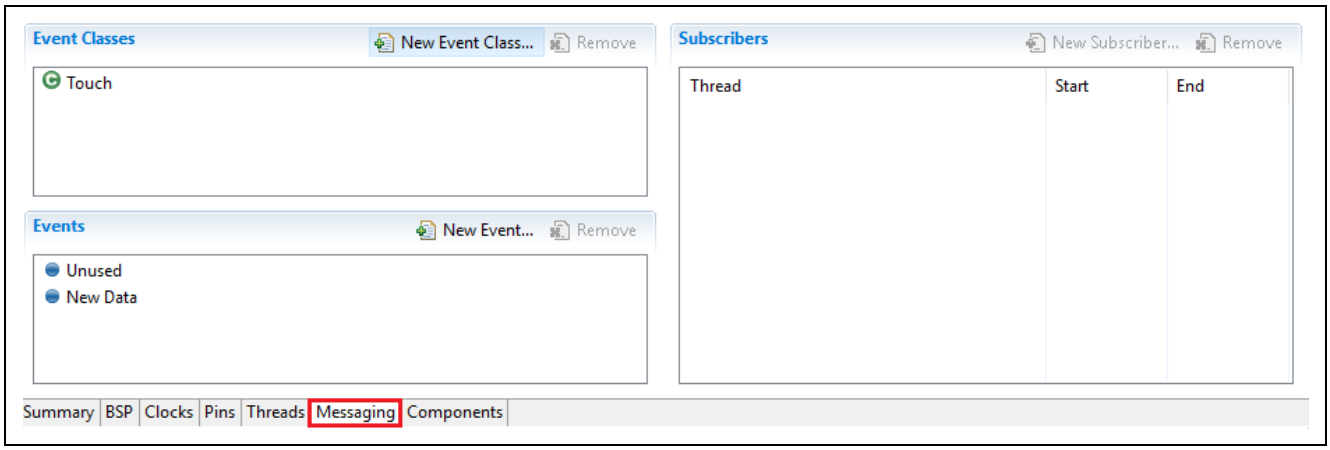

Figure 60. Messaging Tab

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

39. Select **Touch**, under the **Event Classes** window.

40. On the **Touch Subscribers** menu, click the **New +** button.

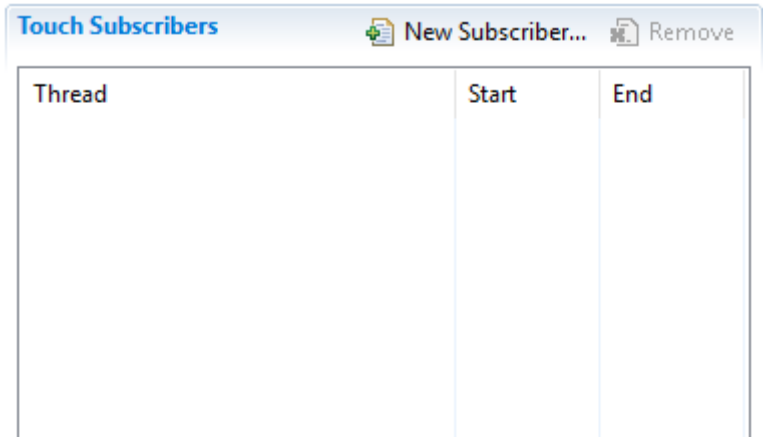

Figure 61. Messaging tab

41. From the **New Subscriber** dialog, select **Main Thread** from the **Thread drop-down list**.

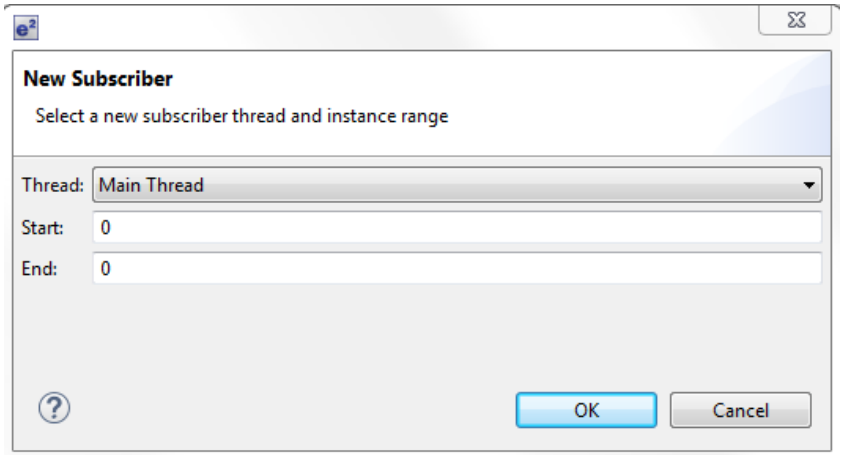

Figure 62. New Subscriber dialog

42. Click **OK**.
43. Save the project by pressing **Ctrl + s** on the keyboard.
44. Click the **Generate Project Content** button to update the project files.

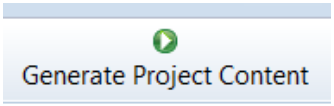

Figure 63. Generate Project Content

45. In the **Project Explorer** window, right-click **src** and select **New** > **Folder** to bring up the **New Folder** dialog box.

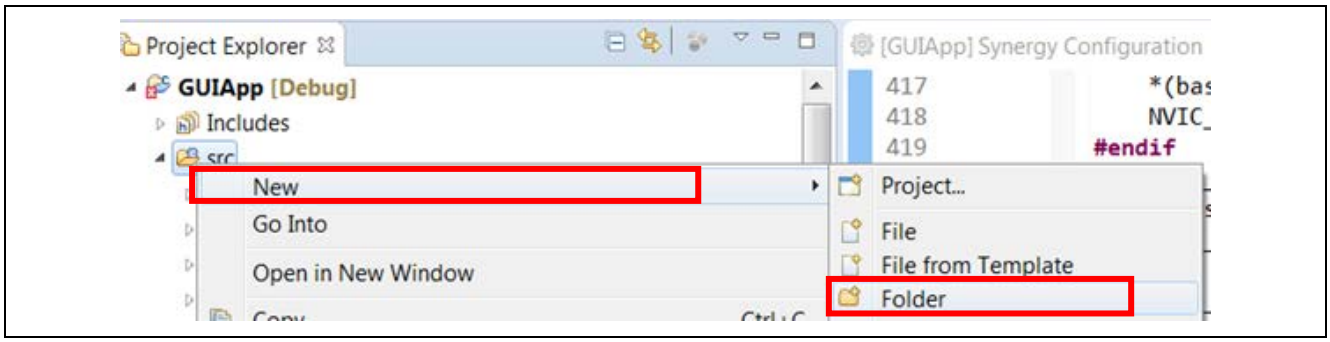

Figure 64. Creating a New Folder

46. Enter the name of the new folder, **hardware**, in the **Folder name:** text box.

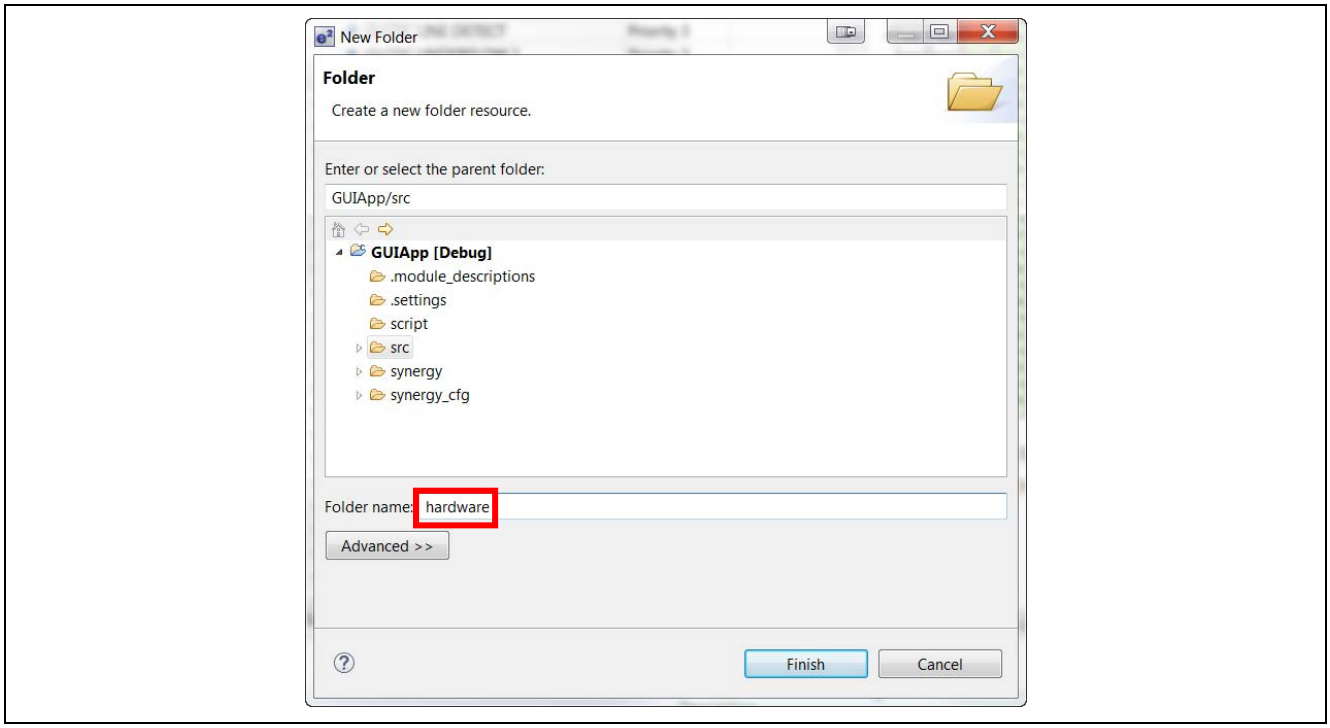

Figure 65. New Folder Dialog

47. Click the **Finish** button.

48. The folder appears in **Project Explorer** shown below.

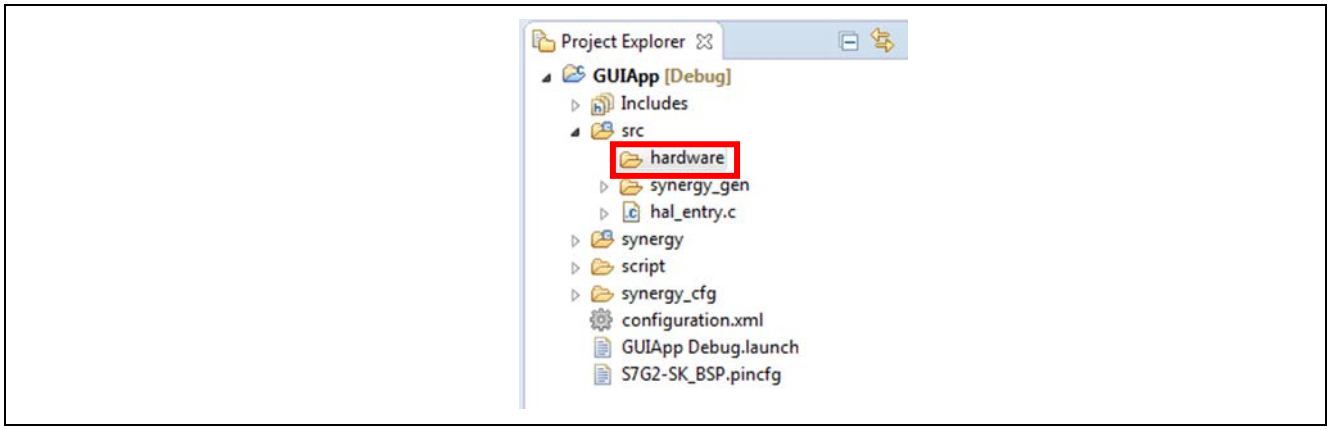

Figure 66. Hardware folder

49. Open **Windows Explorer** and navigate to where you put the files included with this application note. Locate the file `Source Files\lcd.h`. Now drag the file from the **Windows Explorer** window into the new **hardware** folder inside the e² studio **Project Explorer** window.

50. When prompted to import the selected files, click **OK** to copy the files.

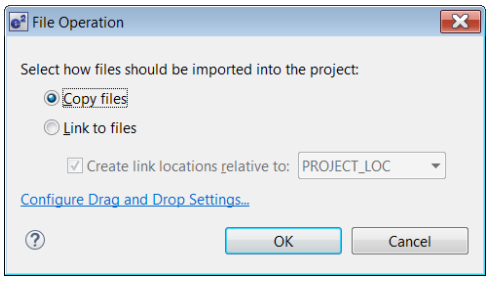

Figure 67. File Operation dialog

Note: This file contains the command definitions to control LCD panel.

51. Open **Windows Explorer** and navigate to where you put the files included with this application note. Locate the file `Source Files\ lcd_setup.c`. Now drag the file from the Windows Explorer window into the **hardware** folder inside the e² studio **Project Explorer** window.

52. When prompted to import the selected files, click **OK** to copy the files.

Note: This file contains command protocol through SPI to LCD panel and the initialization sequence.

53. Open **Windows Explorer** and navigate where you put the files included in 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.

54. When prompted to import the selected files, click **OK** to copy the files.

55. When prompted to overwrite, click **Yes**.

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

5. Creating the GUIX Interface using GUIX Studio

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

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

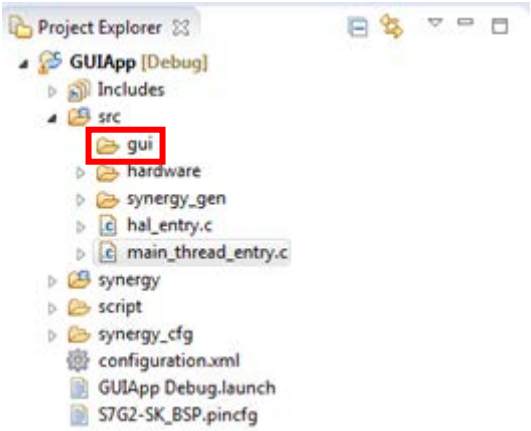

Figure 68. Creating a gui folder under the src folder

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

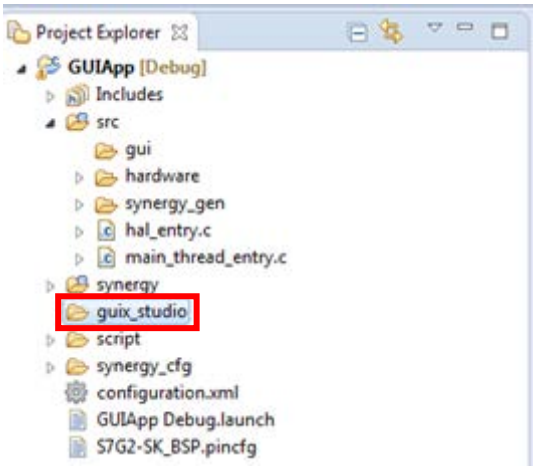

Figure 69. Final Folder list

3. Open GUIX Studio by clicking the desktop icon or by clicking the **GUIX Studio** icon in the **Windows Start** menu, **All Programs** > **Express Logic** > **GUIX Studio 5.4** folder.

Figure 70. Start GUIX Studio

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

Figure 71. Create New Project

5. Name the project **guiapp**.

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

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\GUIAPP\GUIApp\guix_studio`.

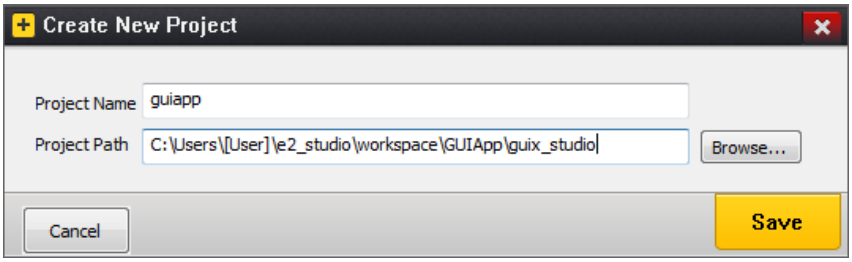

Figure 72. Create a New GUIX project

7. Click **Save**.

8. Change the **Directories** for all three options to be `..\src\gui`.

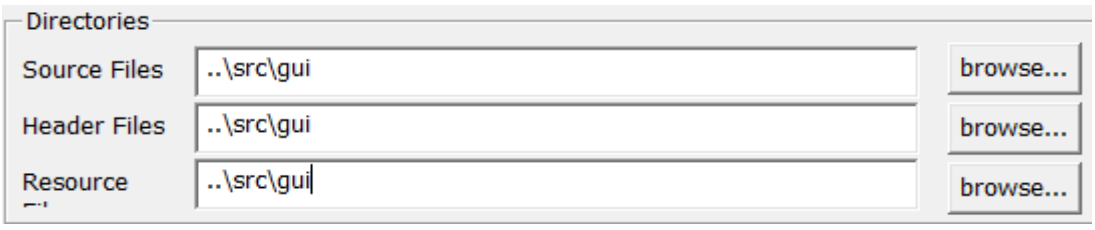

Figure 73. Correct the file Locations

Important:**.** Make sure you put in two periods **..** in the directories above.

9. Change the **Target CPU** setting to **Renesas Synergy**.
10. Change the **Toolchain** setting to **GNU**.

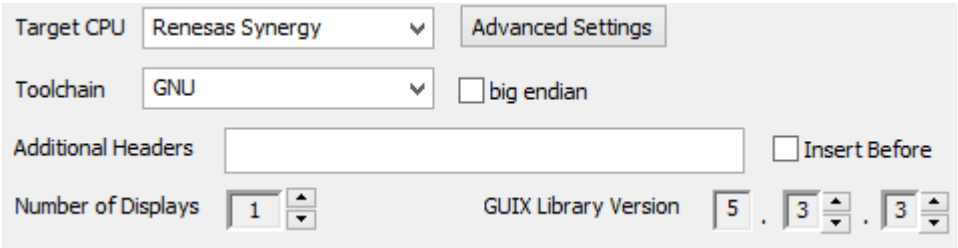

Figure 74. Target and GUIX version settings

11. Click **Advanced Settings**. A dialog will appear.
12. Enable the **2D Drawing Engine** and **Hardware JPEG Decoder** as shown in the following screen.

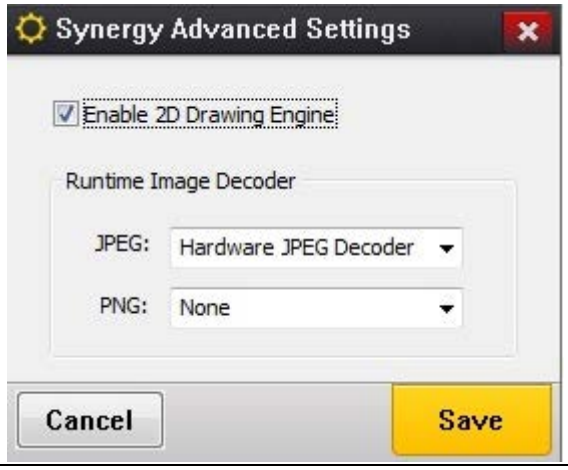

Figure 75. Synergy Advanced Settings

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

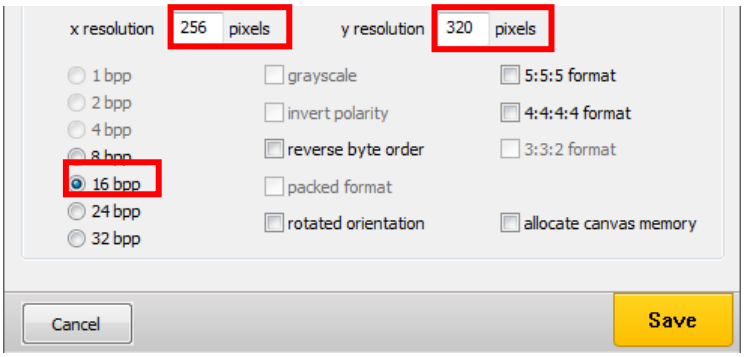

Figure 76. Configure Project

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

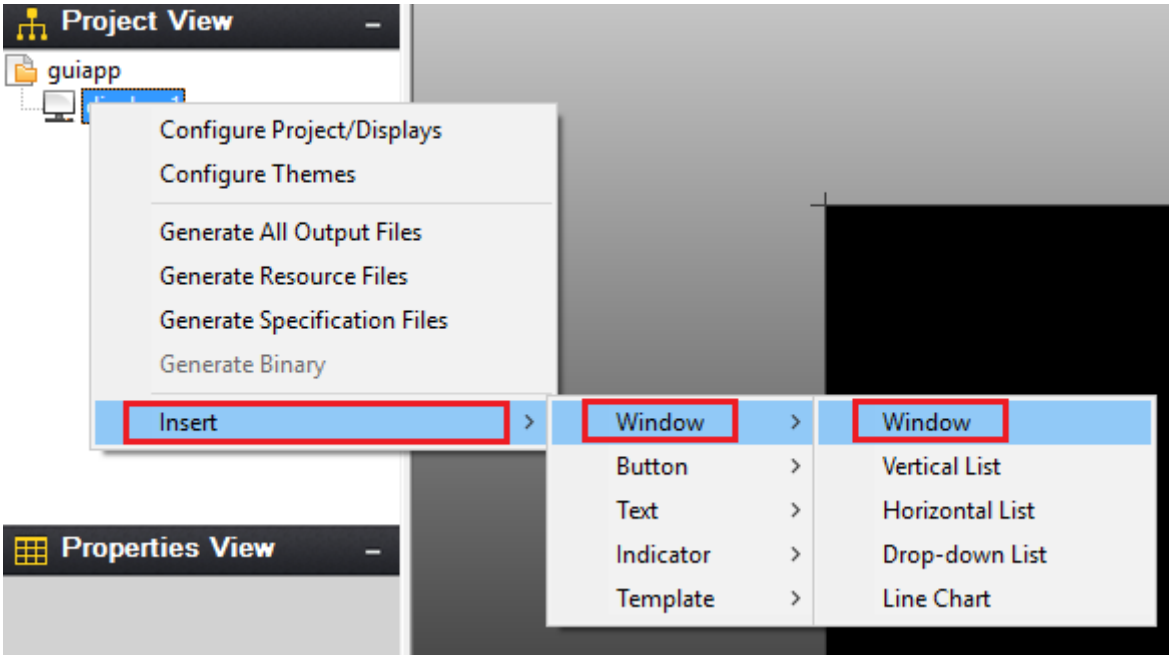

Figure 77. New Window

18. Modify the properties by selecting the new window and editing the **Properties View**. Update the current settings to match the following. Notice the **Event Function** field. This is the event that will be initiated when the **touch screen is pressed in window1**.

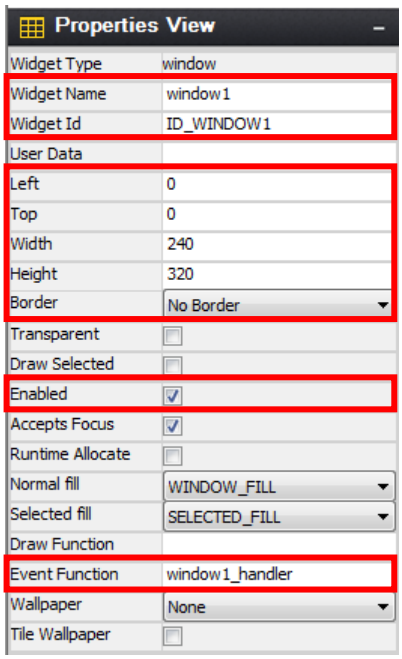

Figure 78. Configure window1 properties

19. Notice the window does not occupy the entire display. This is expected when working with GUIX with small screens and does affect the display once the application is running.
20. In the **Project View** window, right-click **display_1** and create another window by selecting **Insert > Window > Window**.
21. Modify the properties to match the following. Notice the **Event Function** field. This is the event that will be initiated when the **touch screen** is pressed in **window2**.

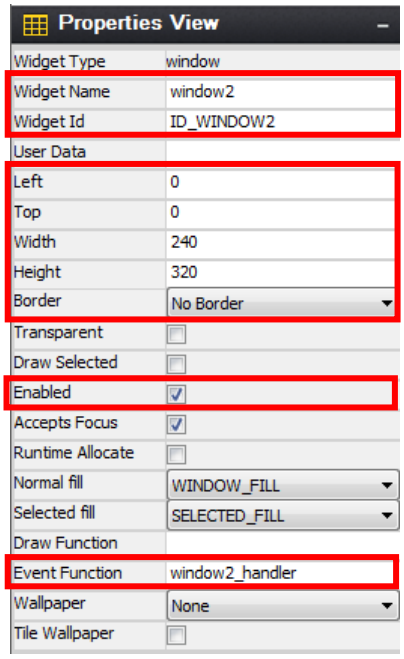

Figure 79. Configure window2 properties

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

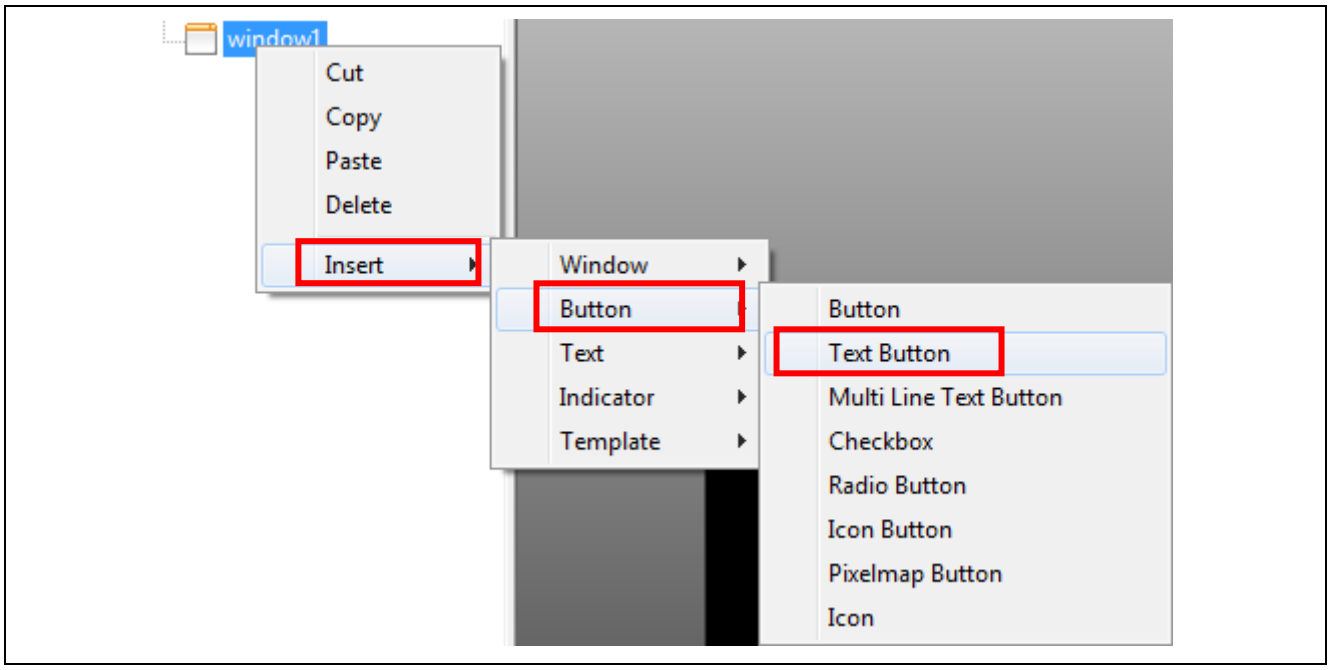

Figure 80. Add a New Text Button

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

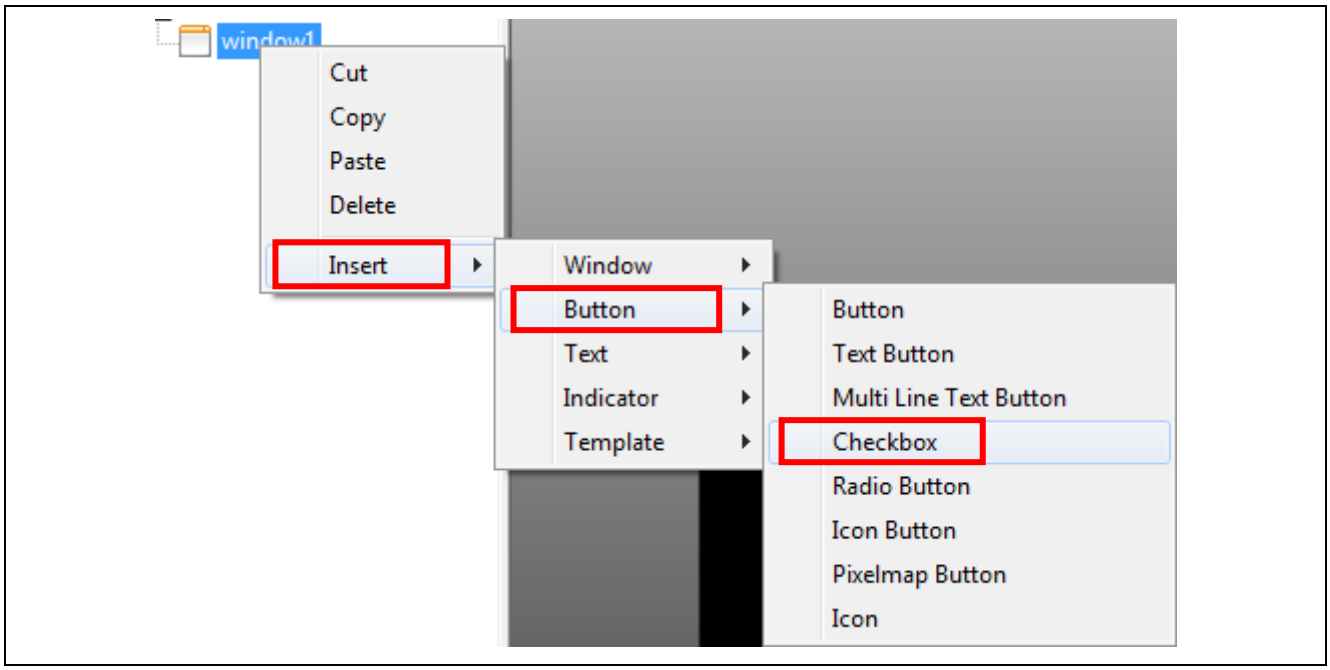

Figure 81. Add a New Checkbox

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

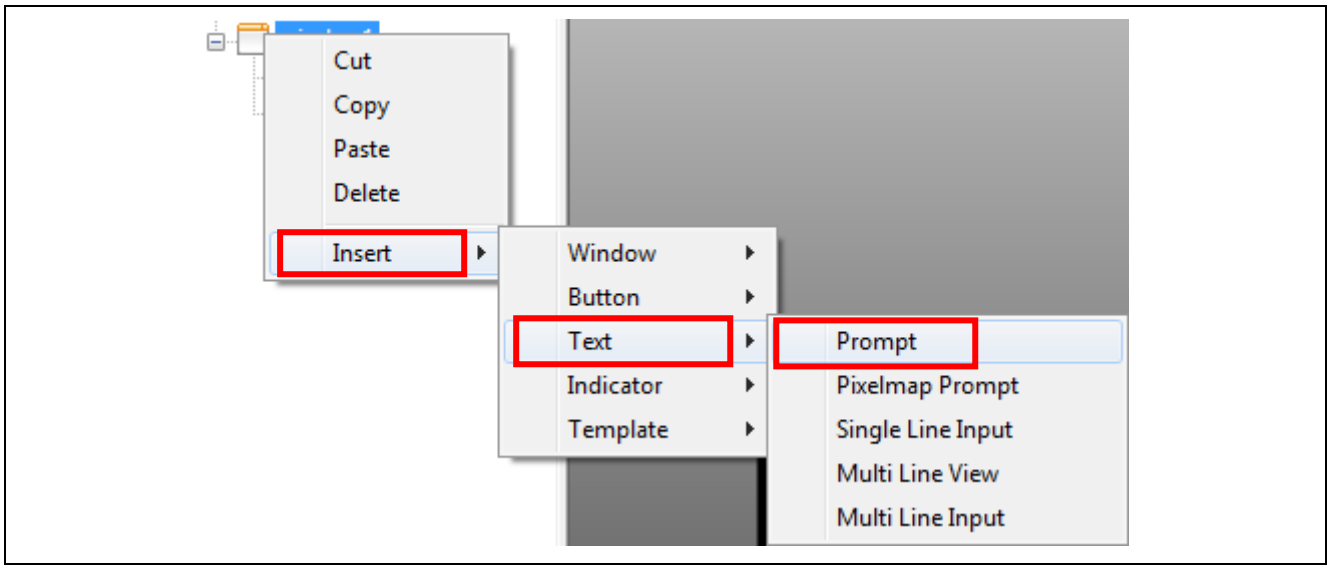

Figure 82. Adding New Prompt

25. In the **Project View**, right-click **window1** and **Insert** another **Text Prompt**.
26. In the **Project View**, right-click **window2** and **Insert** a **Text Prompt**.
27. In the **Project View**, right-click **window2** and **Insert** another **Text Prompt**.
28. If you have followed these directions correctly, your **Project View** should look like the following screen.

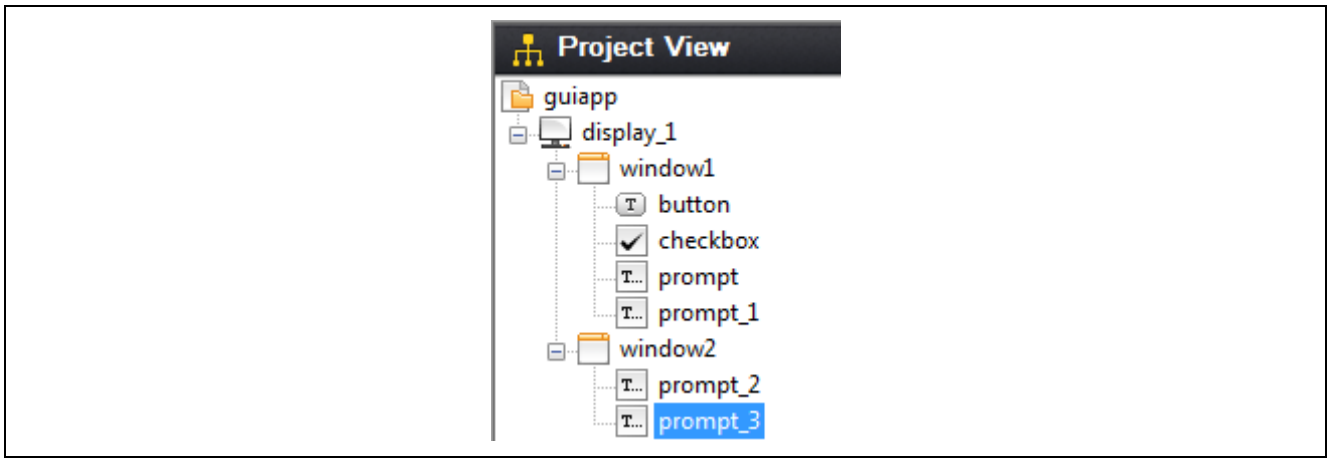

Figure 83. GUIX Project View

29. Expand the **Strings** menu by clicking **+**.

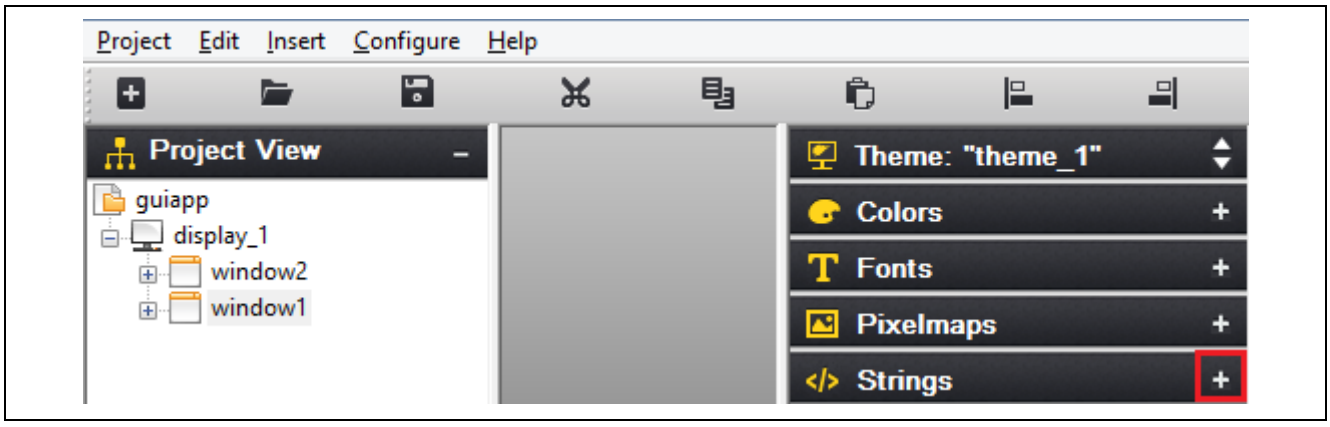

Figure 84. Strings Button

30. Double-click any of the strings to open the **String Table Editor**.

31. Delete the existing strings by selecting them, then click the **Delete String** button in the **String Table Editor**.

32. Add the following **Strings** using the **Add String** button:

Strings

String ID	English
HELLO_WORLD	Hello World (to window1)
CHECKBOX_TEXT	Press Me!
BUTTON_DISABLED	Stay in window 1
BUTTON_ENABLED	Goto window2
INSTRUCT_CHECKBOX	Press "Press Me!" for more.
WINDOW1	Window1
WINDOW2	Window2
INSTRUCT_BUTTON	Press to go to window2

Figure 85. New Strings

33. When completed, click **Save**.

34. In the **Project View** under **window1**, click the button and then modify the properties in the Properties View to match the following.

Properties View

Widget Type	text_button
Widget Name	windowchanger
Widget Id	ID_WINDOWCHANGER
User Data	
Left	30
Top	100
Width	180
Height	50
Border	No Border
Transparent	☐
Draw Selected	☐
Enabled	☑
Accepts Focus	☑
Runtime Allocate	☐
Normal fill	BTN_LOWER
Selected fill	BTN_UPPER
Draw Function	
Event Function	
Pushed	☐
Toggle	☐
Radio	☐
Auto Repeat	☐
String ID	BUTTON_DISABLED
Text	Stay in window1
Font	BUTTON
Text Align	Center
Normal Text Color	BTN_TEXT
Selected Text Color	BTN_TEXT

Figure 86. Configure windowchanger Button properties

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

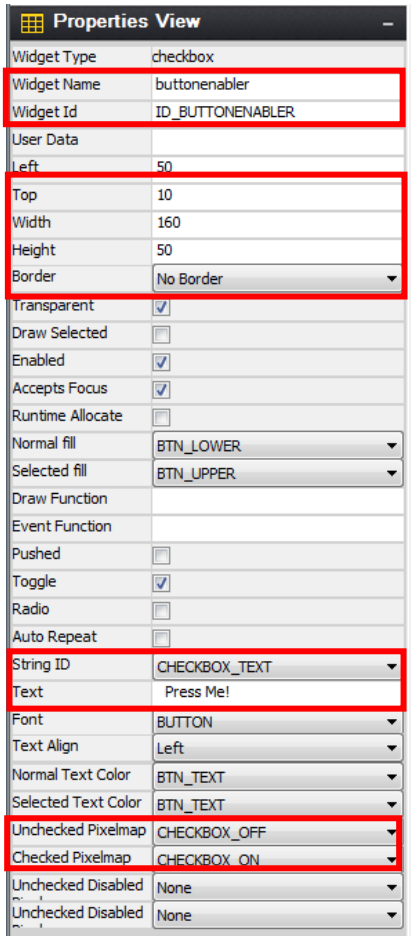

Figure 87. Configure Buttonenabler Checkbox properties

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

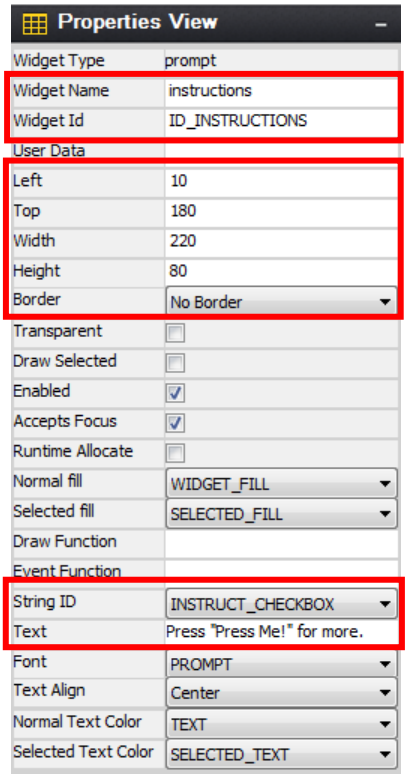

Figure 88. Configure Prompt properties

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

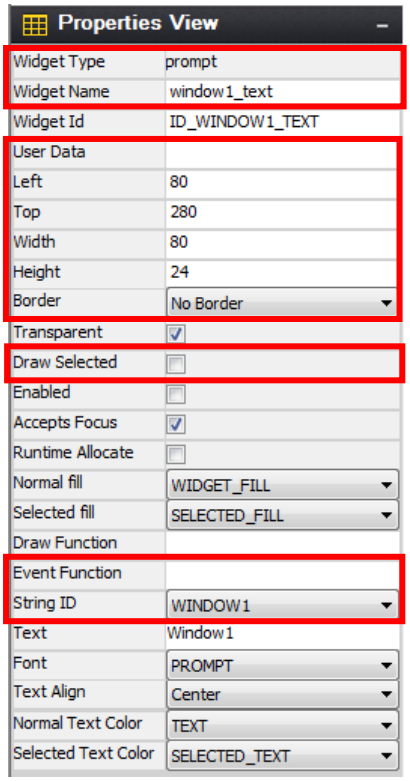

Figure 89. Configure Window Text properties

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

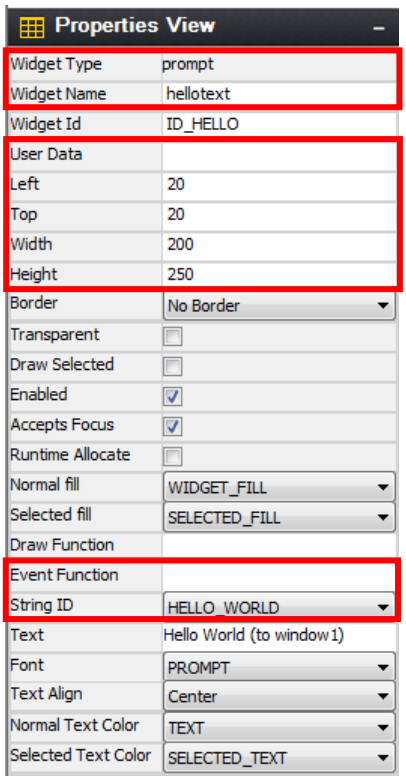

Figure 90. Configure Hello Text Prompt properties

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

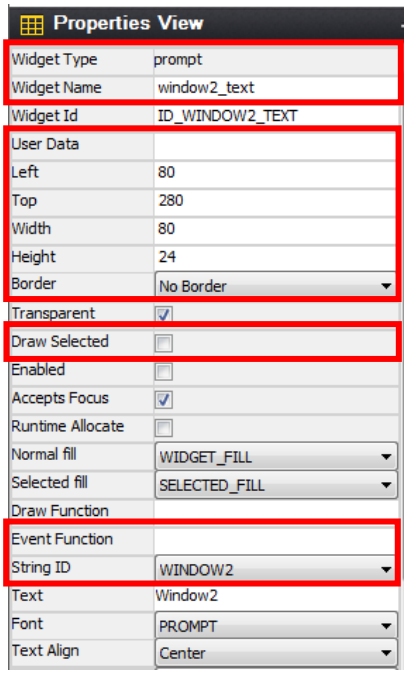

Figure 91. Configure Window Text properties

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

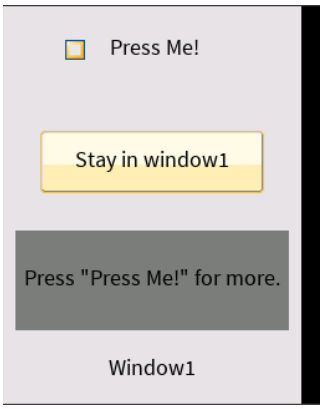

Figure 92. Configured window1

Figure 93. Configured window2

40. **Save** the project.

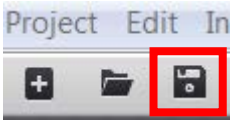

Figure 94. Save project

41. From the **Project** tab select **Generate all Output Files**.

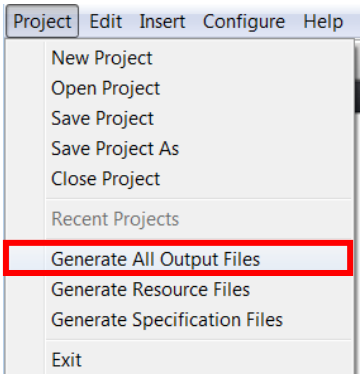

Figure 95. Generate All Output files

42. Return to e² studio.

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

1. Open **Windows Explorer** and navigate to where you put the files included with this application note. Locate the file `Source Files\guiapp_event_handlers.c`. Drag the file from the **Windows Explorer** window into the **src** folder inside the e[2] studio **Project Explorer** window.
2. When prompted to import the selected files, click **OK** to copy the files.

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

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

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

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

.    return result;
}
```

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

3. Turn optimization off:
   A. Right-click **GUIApp** in the **Project Explorer** window and select **Properties** from the context menu.
   B. Within the properties window, expand the C/C++ Build tree element.
   C. Select **Settings**.

D. In the **Tool Settings** tab, click **Optimization**.

E. Change the **Optimization Level** to None (**-O0**).

F. Click **OK** to save these changes.

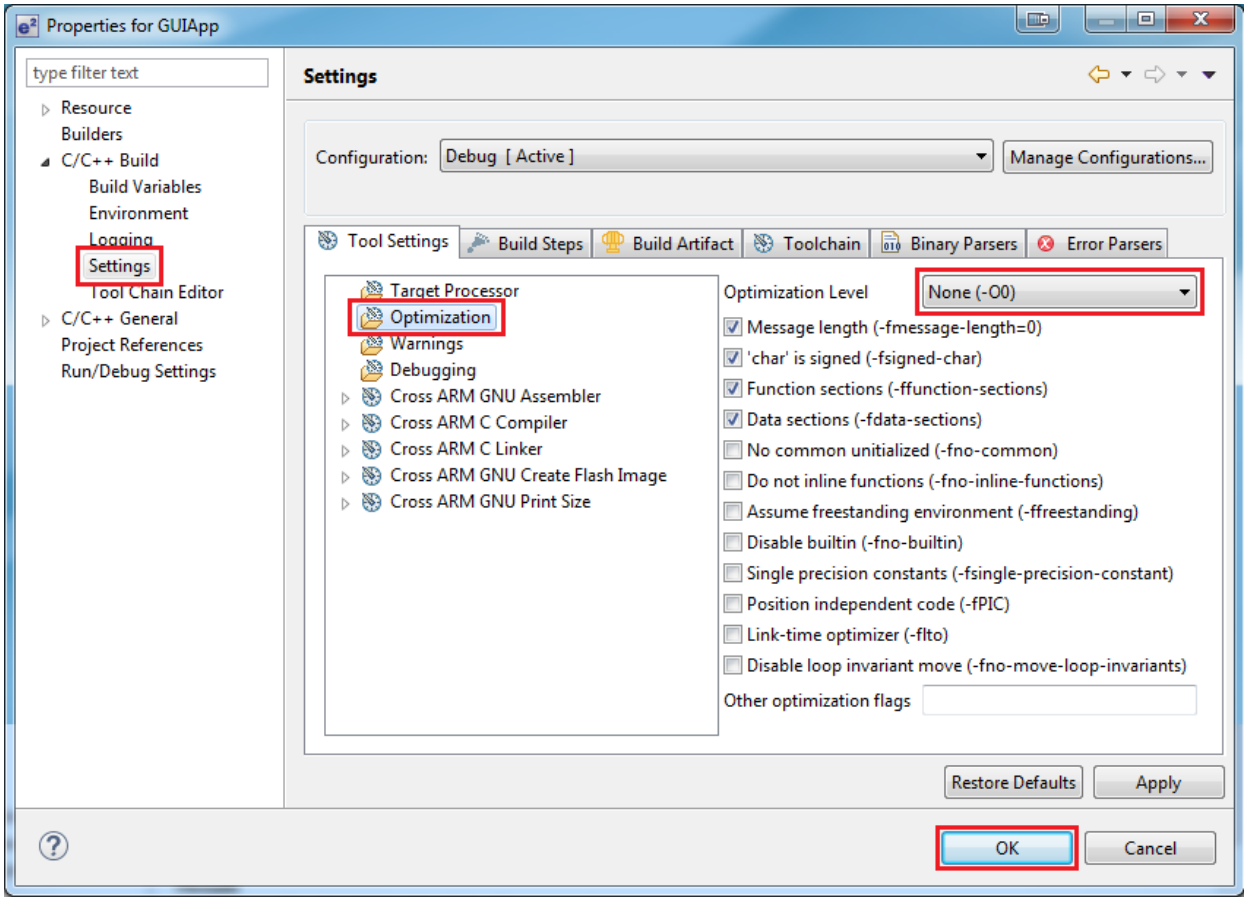

Figure 96. Disabling Compiler Optimizations

4. Build the project by clicking the **Hammer** icon below the menu bar.

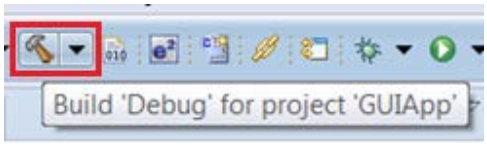

Figure 97. Build the project

Following these steps, there will be no errors reported in the build output, as the following figure shows.

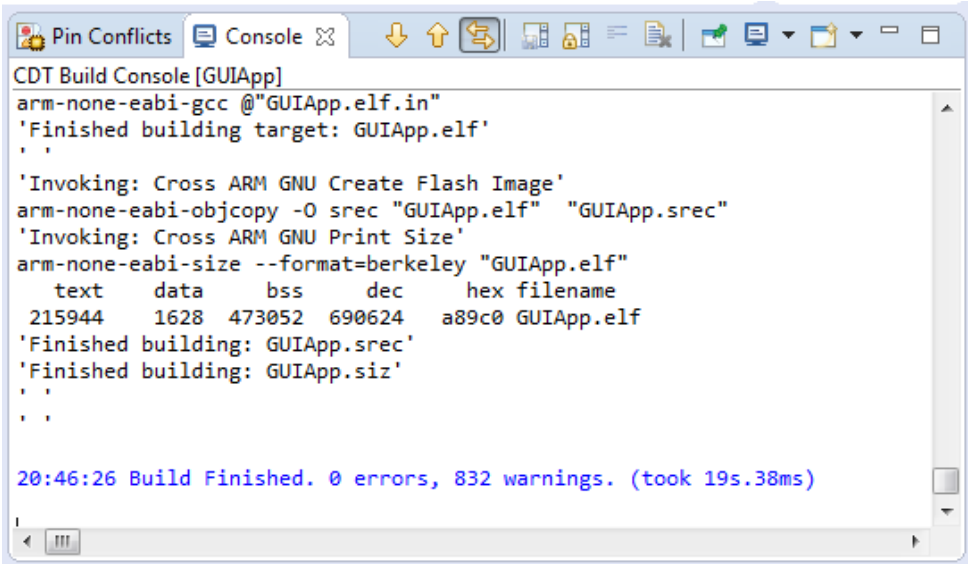

Figure 98. Build finished with 0 errors

7. Running the application

1. Connect the **SK-S7G2** or **PK-S5D9** Synergy MCU Groups (J19) to the PC with the 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. Click the **drop-down menu** for the **debug** icon.
3. Select the **Debug Configurations...** option.

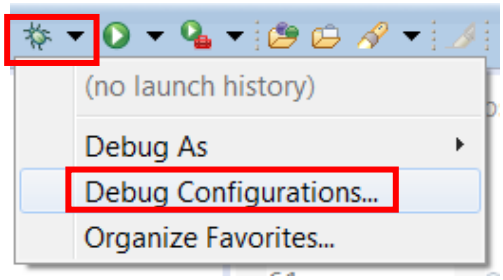

Figure 99. Debug options

4. Under the **Renesas GDB Hardware Debugging** section, select **GUIApp Debug**.
5. Click the **Debug** button to start debugging.

Note: If the **Debug** button is greyed out, then it is likely that there is an issue with the build. Check all steps for mismatched options.

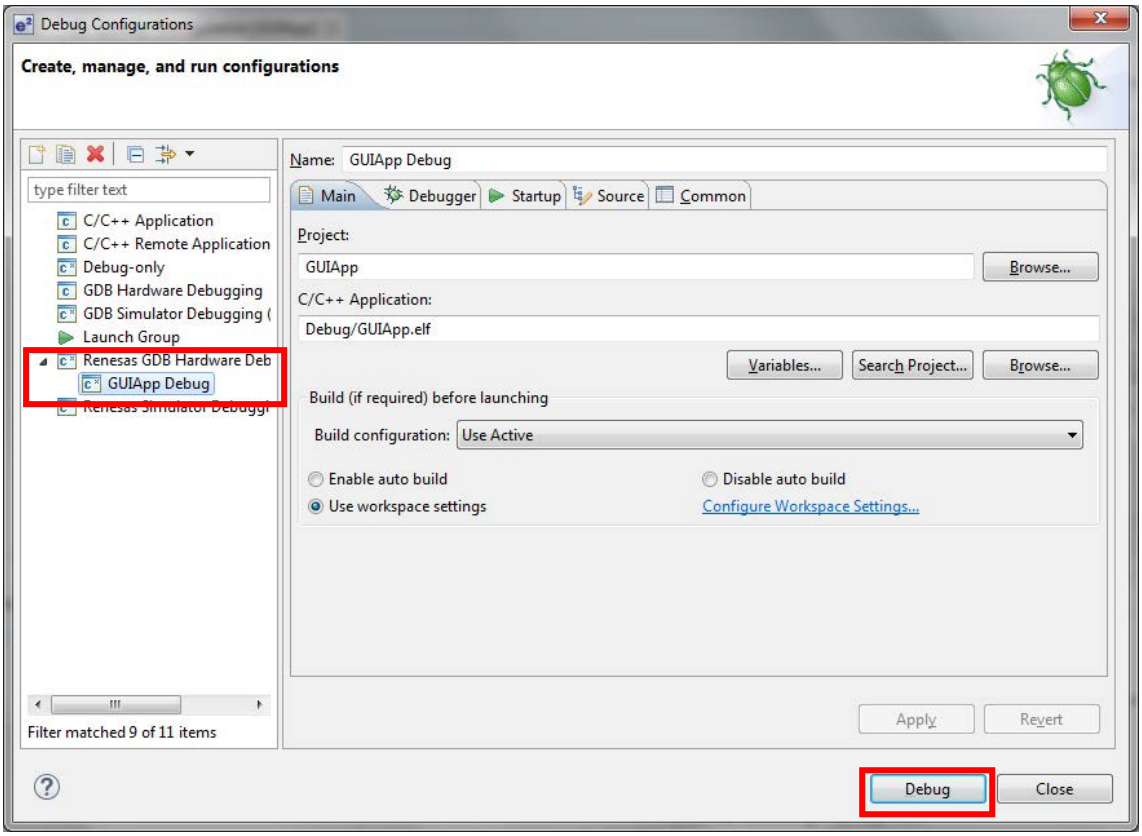

Figure 100. Debug Configurations

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.).

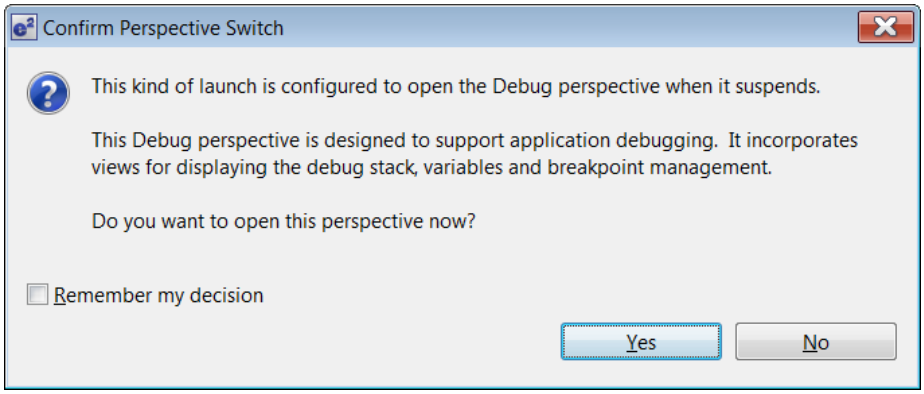

Figure 101. Perspective Switch Dialog

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

Figure 102. Resume Button

8. Press F8 or the **Resume** button to run the code.

Note: The GUI created earlier should display on the screen.

9. Overview of the Demo.

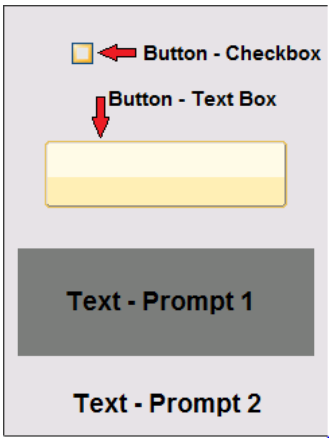

Figure 103. Window1

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

— **Button – Checkbox**: Use this button to enable navigating to **Window2**. Text is set to **Press Me!** and it is unchecked. When you click within the **Checkbox** active area, the event **window1_handler** is activated. This event is picked up inside `guiapp_event_handlers.c,` where the code toggles the checkbox then sets the text in **Text –Prompt 1** and **Button – Text Box** to the appropriate message.

— **Button – Text Box**: This box shows which window you will go to if you press outside the **Text – Prompt 1** area. (See **Button – Checkbo** for how it is changed.) Click 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**.

— **Text – Prompt 1**: This area instructs you how to control the demo. (See **Button – Checkbox** for how it is changed.)

— **Text – Prompt 2**: This Prompt is used to show you what window you are in. It never changes (always shows **window1**).

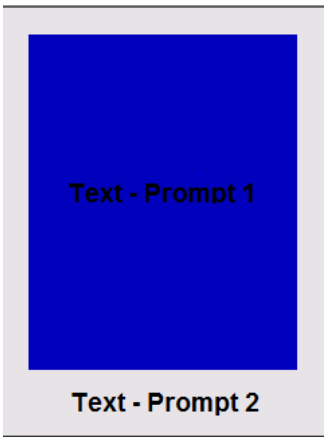

Figure 104. Window2

B. The preceding figure shows **Window2**. In this window are two elements:

— **Text – Prompt 1**: This area presents **Hello World**. Clicking in this area initiates the **window2_handler** event which is picked up by `guiapp_event_handlers.c` and changes the active window to **window1**.

— **Text – Prompt 2**: This Prompt is used to show you which window you are in. It never changes (always shows **window2**).

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

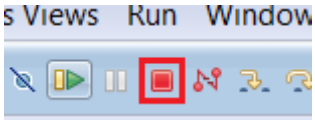

Figure 105. Stop Button

11. This concludes the GUIX **Hello World** for SK-S7G2 and PK-S5D9 Synergy MCU Groups.

8. Appendix:

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

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.22.16	—	Initial version
1.01	Apr.12.16	—	Updated lcd_setup.c to correct semaphore naming issue
1.10	Aug.30.16	—	Update to SSP v1.1.0
1.11	Nov.18.16	—	Minor Format Changes
1.12	Jan.06.17	—	Updated to SSP v1.2.0.b.1
1.13	Feb.28.17	—	Updated to SSP v1.2.0
1.14	Sep.20.17	—	Updated to SSP v1.3.0
1.15	Feb.28.18	—	Updated to SSP v1.4.0
1.16	Jun.18.18	—	Sample codes updated
1.17	Sep.07.18	—	Updated to SSP v1.5.0
1.18	Mar.22.19	—	Updated to 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/.