Renesas RA Family

GUIX Hello World for EK-RA8D1 MIPI LCD Display

Introduction

This application note describes the process of creating a simple two-screen GUI using Azure RTOS GUIX Studio for the EK-RA8D1 kit. This application demonstrates how easily a user can create and configure a new application using the Renesas Flexible Software Package (FSP).
The Renesas Flexible Software Package includes Azure RTOS ThreadX® real-time operating system, the Azure RTOS GUIX library and hardware drivers are unified under a single robust software package. This powerful suite of tools provides a comprehensive integrated framework for rapid development of complex embedded applications.

Required Resources

Development tools and software

- e² studio IDE Version: 2023-10 (23.10.0) or greater
- Renesas Flexible Software Package (FSP) v5.1.0 or greater
- Azure RTOS GUIX Studio V6.3.0.1 or greater

Hardware

- Renesas EK-RA8D1 kit (RA8D1 MCU Group)
- Renesas EK-RA8D1's "SW1" switches setting.
  - Switch #6 for GLCDC set "ON", switch #7 for SDRAM set "ON" and all the other switches set "OFF".

Reference

- RA Flexible Software Package Documentation Release v5.1.0
- Azure RTOS GUIX and GUIX Studio v6.3.0.1
- Renesas RA8D1 Group User's Manual Rev.1.1.0
- EK-RA8D1-v1.0 Schematics

Provided Software Files

- A `Source.zip` folder that has 2 `touch_gt911 and SEGGER_RTT` folders and 4 `*.c` with 2 `*.h` files inside
- `hal_entry.c`, `system_thread_entry.c`, `touch_thread_entry.c`, `windows_handler.c`, `system_thread_entry.h`, `common_utils.h`

Purpose

This document will guide you through the setup of an Azure RTOS GUIX touch screen interface Hello World application in e² studio. This document will show how to configure the drivers and library included with the FSP. The instructions in this document will allow you to set up the MIPI Display Controller, the touch screen driver, and semaphores to communicate with application tasks. It also shows the steps necessary to create a simple GUI interface using the Azure RTOS GUIX Studio editor. In addition, this app note will also cover project setup along with basic debugging operations. When it is running, the application will respond to touchscreen actions, presenting a basic graphical user interface (GUI).

Intended Audience

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

Contents

1. Downloading and Installing Tools

1.1 Overview

In this section you will copy materials to your PC and install the required software: e² studio v2023-10.0 with FSP v5.1.0, and Azure RTOS GUIX studio v6.3.0.1.

1.2 Procedural Steps

1. If you already have e² studio v2023-10 with FSP v5.1.0 or later installed, you can skip this step. Otherwise, you can download it from https://www.renesas.com/us/en/software-tool/flexible-software-package-fsp.
   Detailed installation instructions for the e² studio and the FSP are available on the Renesas website https://www.renesas.com/fsp. Review the release notes for e² studio to ensure that the e² studio version supports the selected FSP version. The starting version of the installer includes all features of the RA MCUs.
2. You can get Azure RTOS GUIX Studio v6.3.0.1 or greater from this link.
   You should see the window in the next step on the web browser.
   Note: Microsoft Store must be working on your PC to install Azure RTOS GUIX studio.
3. Click **Install Button** and New window popup. Click **Open Microsoft Store**.

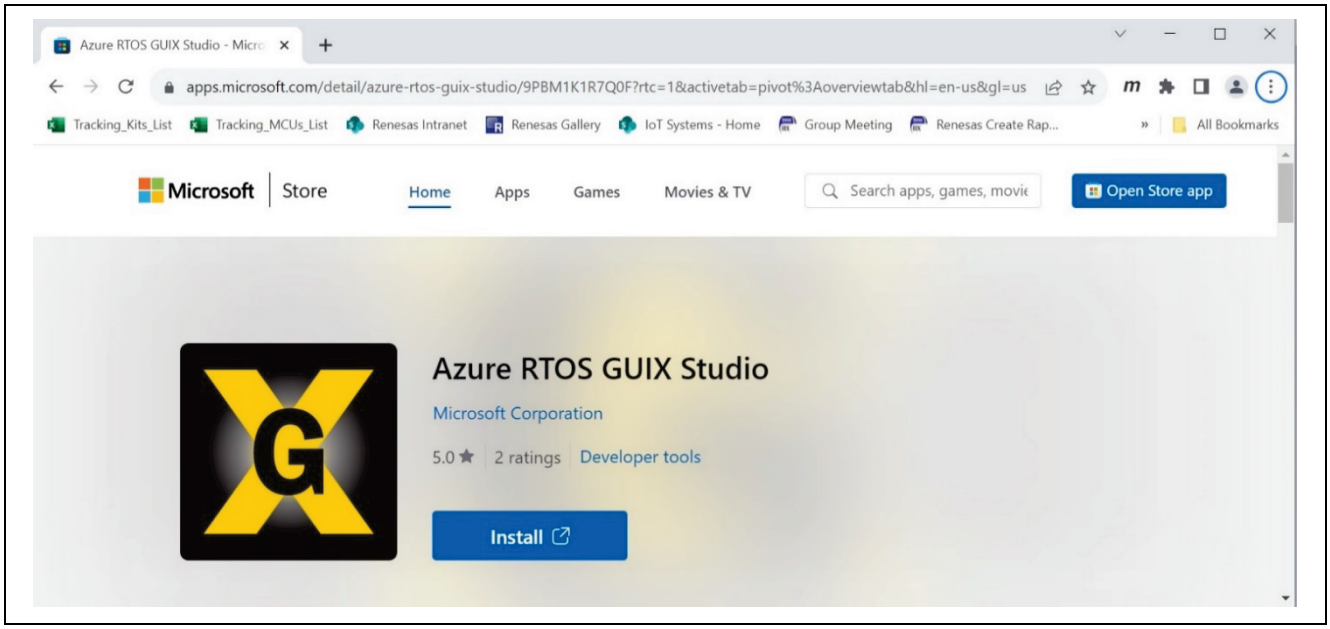


Figure 1. Get Azure RTOS GUIX Studio

4. Click **Open Microsoft store** to continue installing Azure RTOS GUIX Studio.

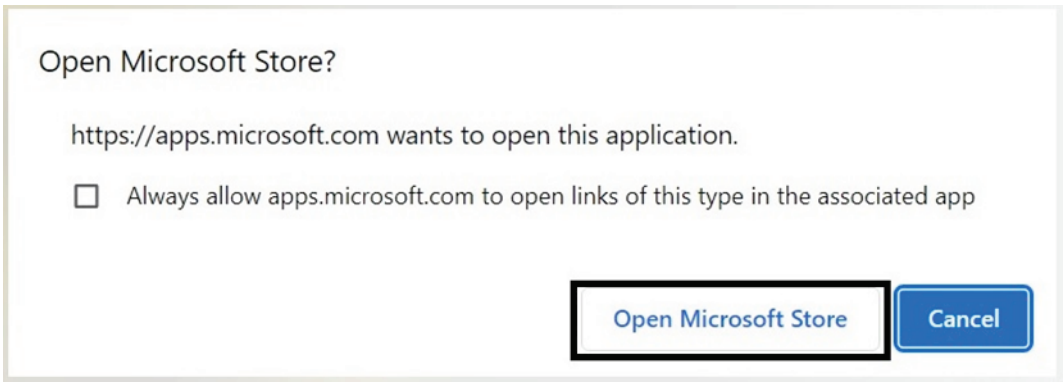
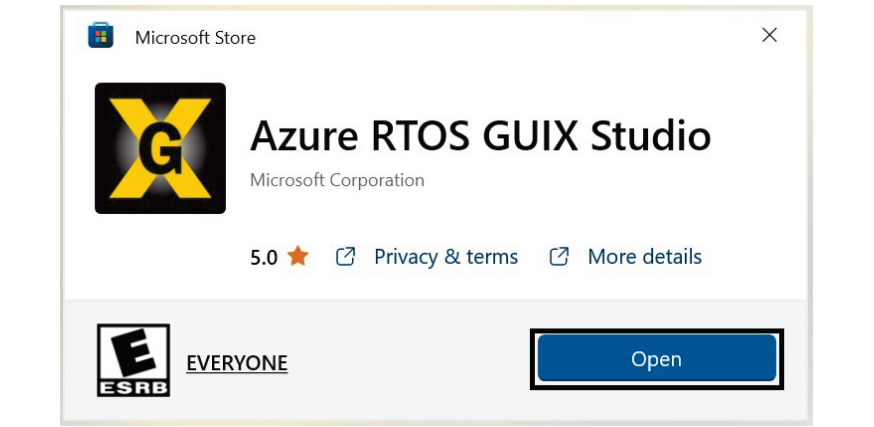


Figure 2. Open Microsoft Store

5. Click **Open** to open the “Azure RTOS GUIX Studio” App.



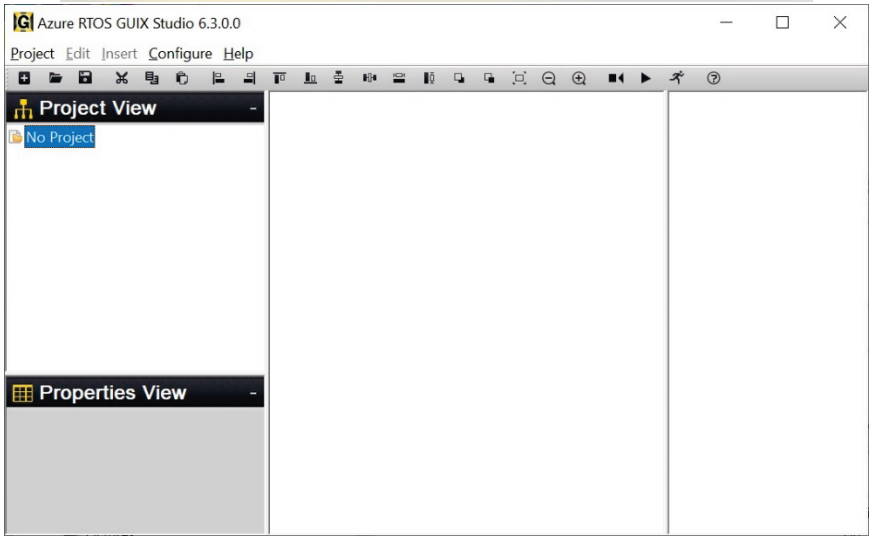


Figure 3. Click Open to start “Azure RTOS GUIX Studio”.

2. Create Application Note Project and Enabled Backlight

2.1 Overview

In this section, you will create a project to which you will add pre-written source code and integrate it with a pre-created Azure RTOS GUIX studio project.

2.2 Procedural Steps

1. Create a new **Renesas RA C/C++ project**. Name it **mipi_ek_ra8d1_guix_hello_world**.

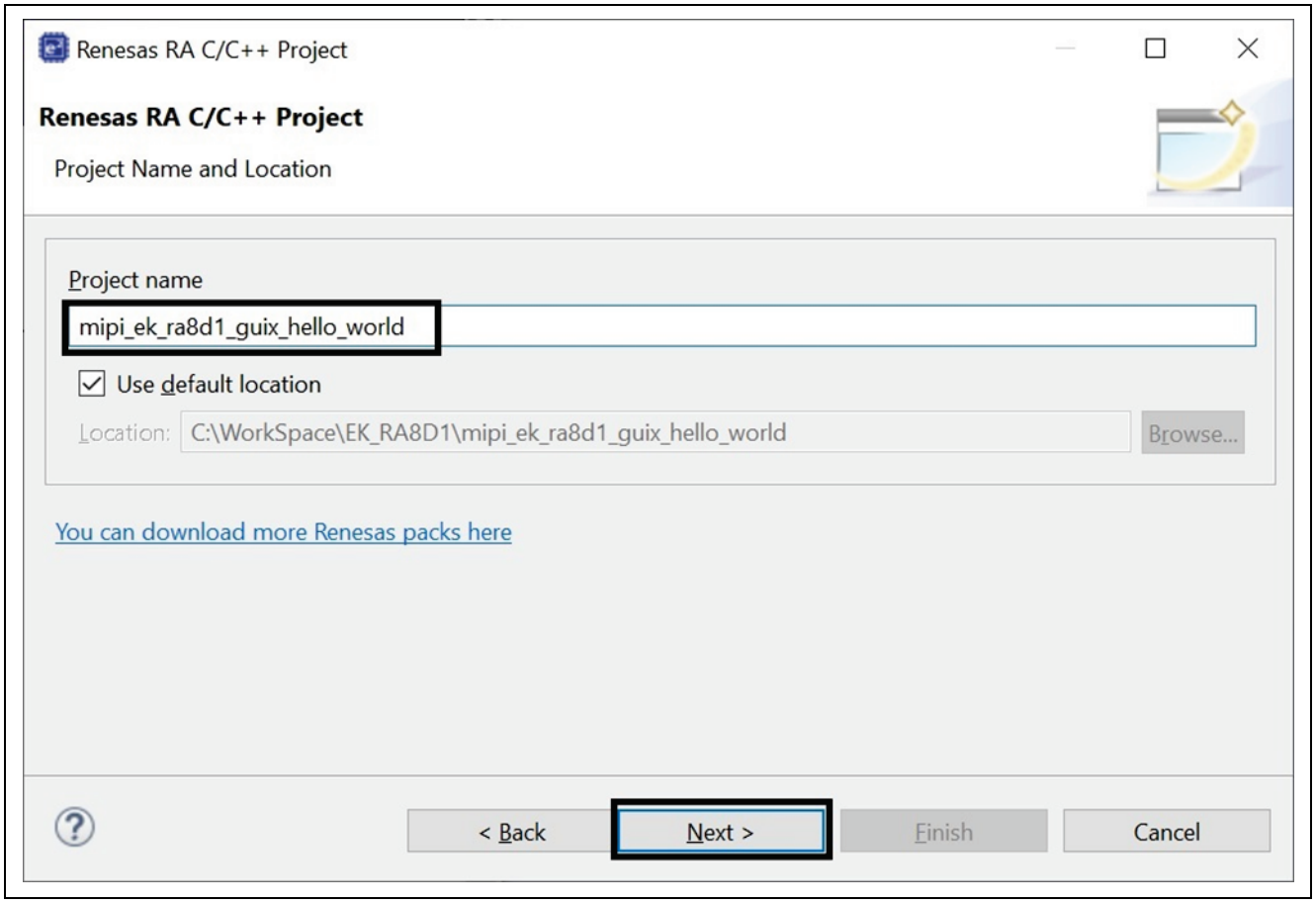


Figure 4. Create New Project

2. Select and set board to EK-RA8D1.

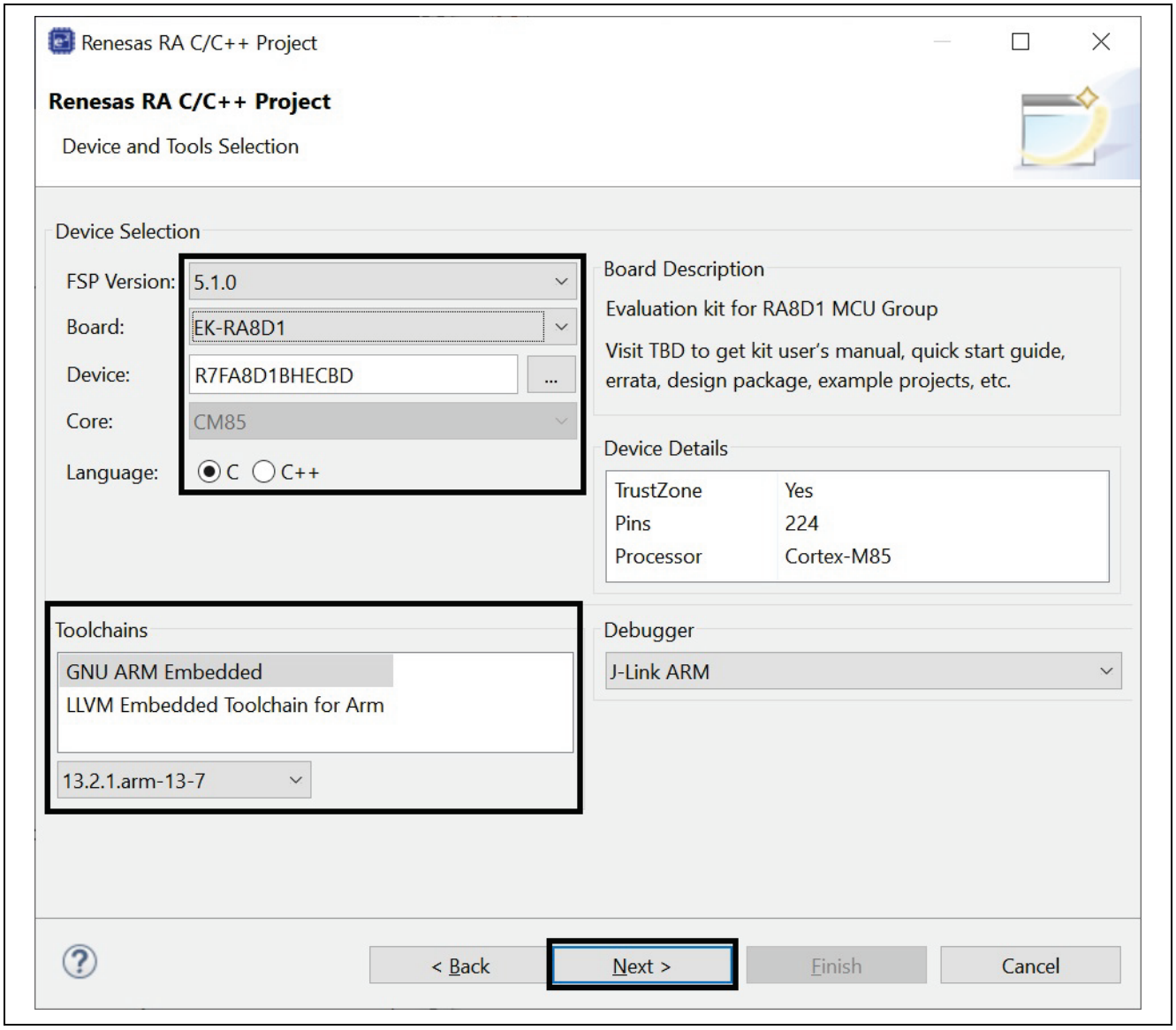


Figure 5. Select and Set board to EK-RA8D1

3. Select **Flat** (**Non-TrustZone**) and **Executable** with **Azure RTOS ThreadX** (**v6.2.1+FSP.5.1.0**).

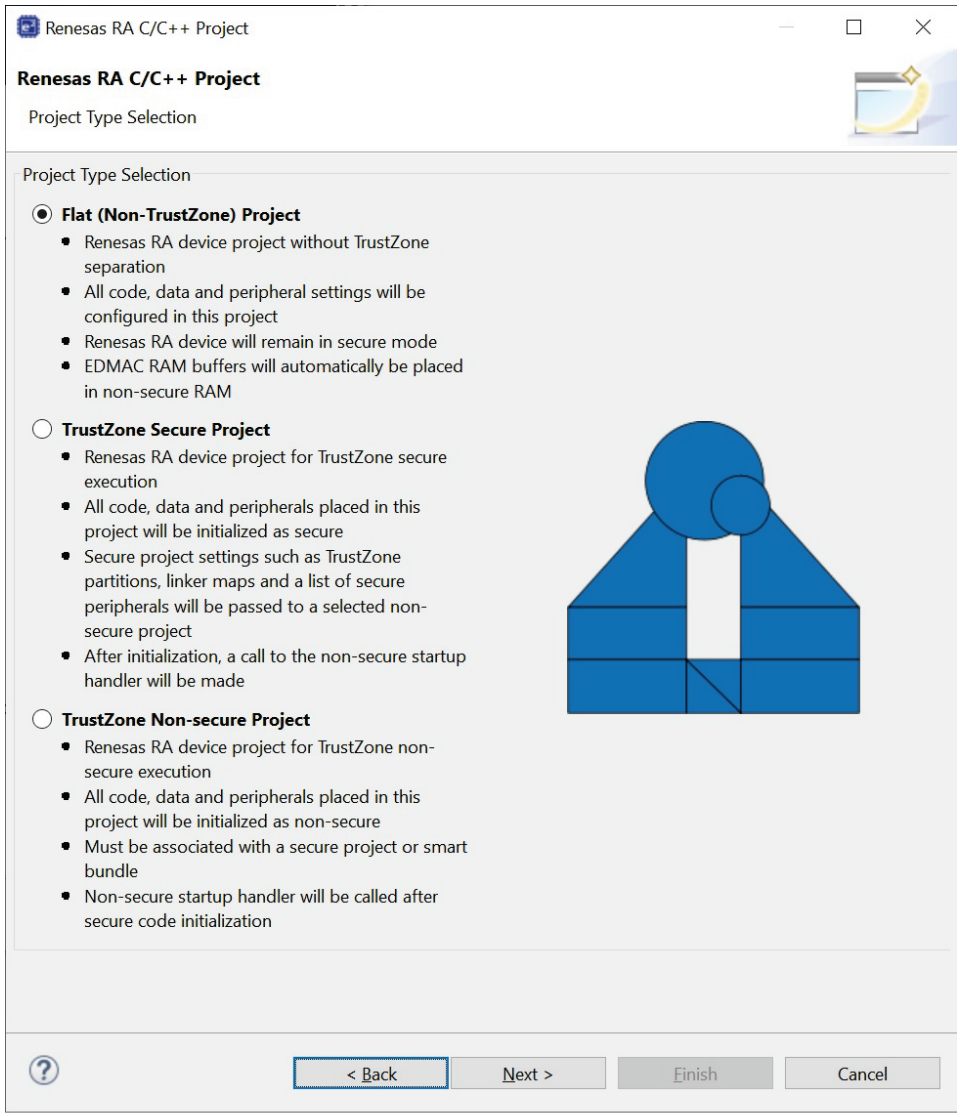


Figure 6. Select Azure RTOS ThreadX

4. Use **Azure RTOS ThreadX – Minimal** template.

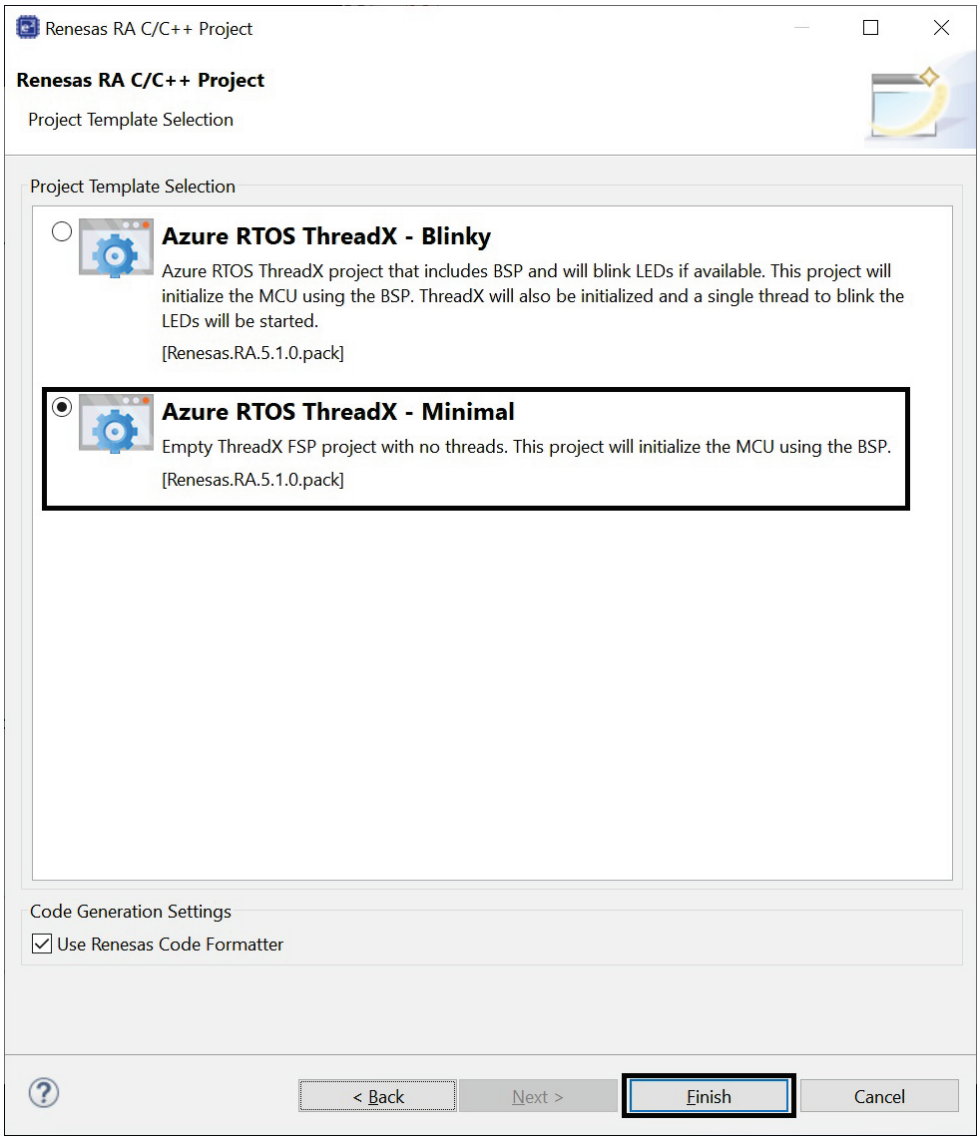
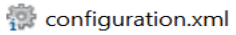


Figure 7. Select Minimal Template

5. Open the project configuration, by double clicking the **configuration.xml**  file and changing the e^2 studio view to FSP Configuration. Go to the **BSP** tab and select the **Properties** tab. Here, change **Main stack size (bytes)** and **Heap size (bytes)** to **0x2000**.

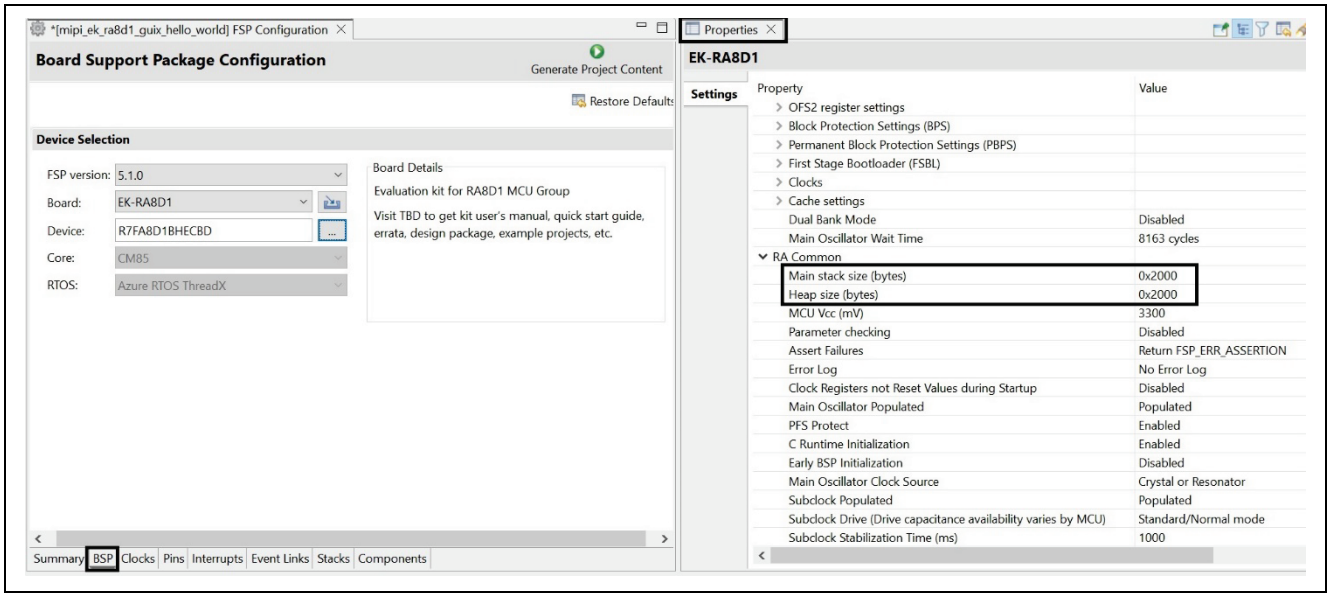


Figure 8. Change Heap Size

6. Click the **Clocks** tab and set the **LCDCLK** Source Input to **PLL1P**.

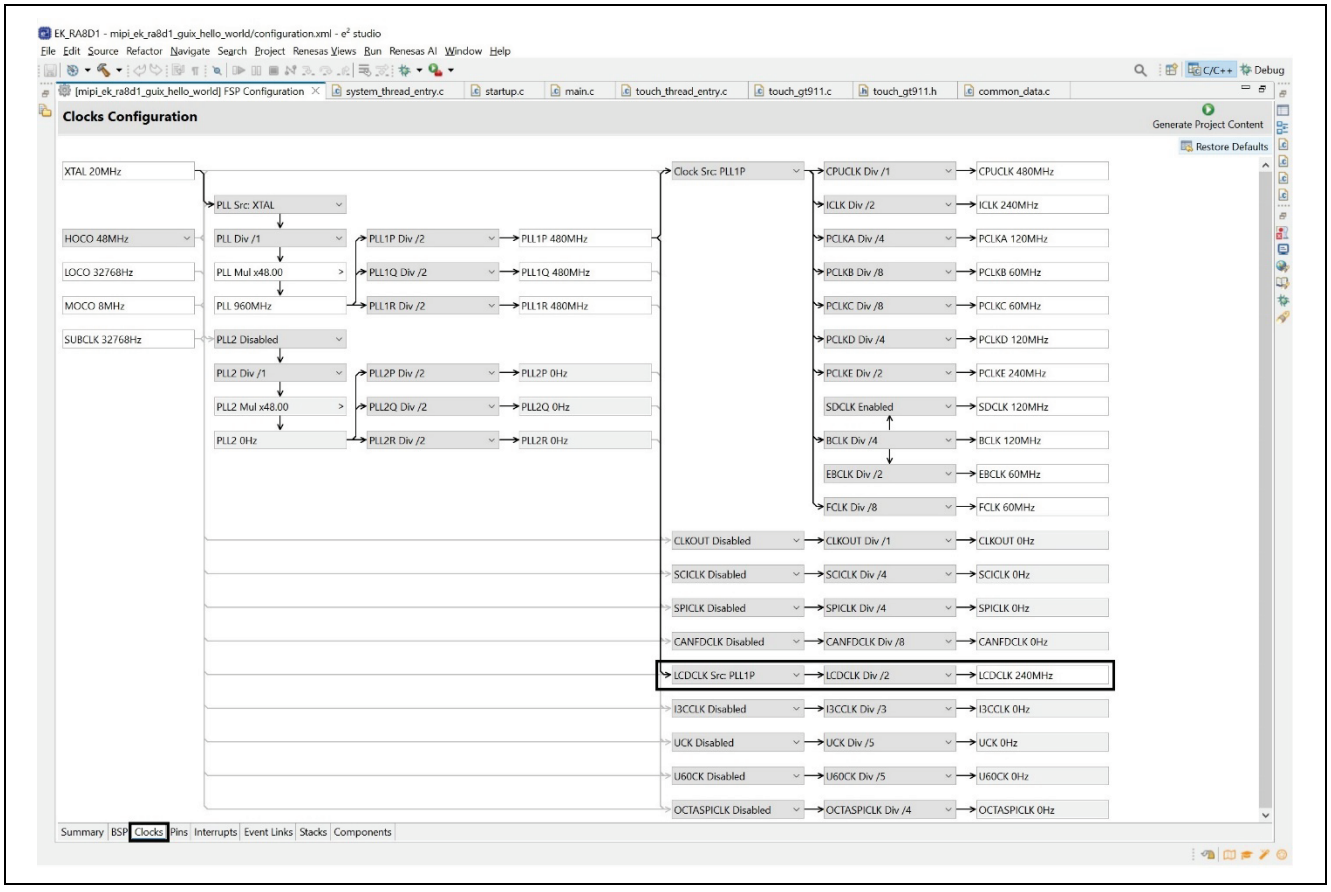


Figure 9. Enabling the LCD Clock to PLL1P

7. Go to the **Stacks** tab to view the current project's threads and modules. Add a **New Thread** and name it as **System Thread** with the **Property** settings shown below.

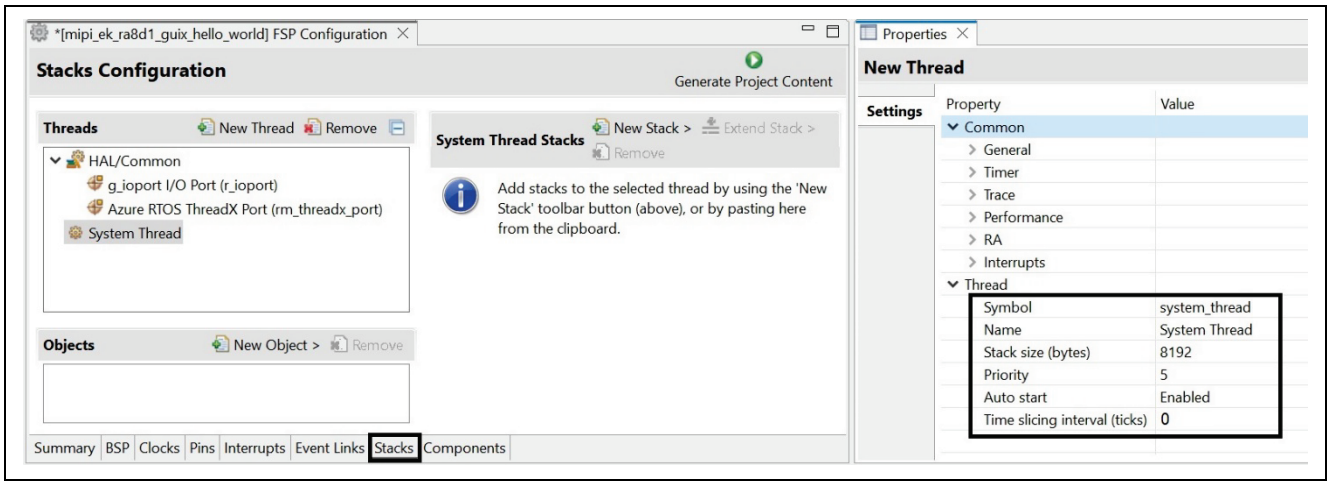


Figure 10. Add System Thread

8. Make sure the System Thread is highlighted in the Threads window. Click "**New Stack**" and add **Graphics > Azure RTOS GUIX** to System thread.

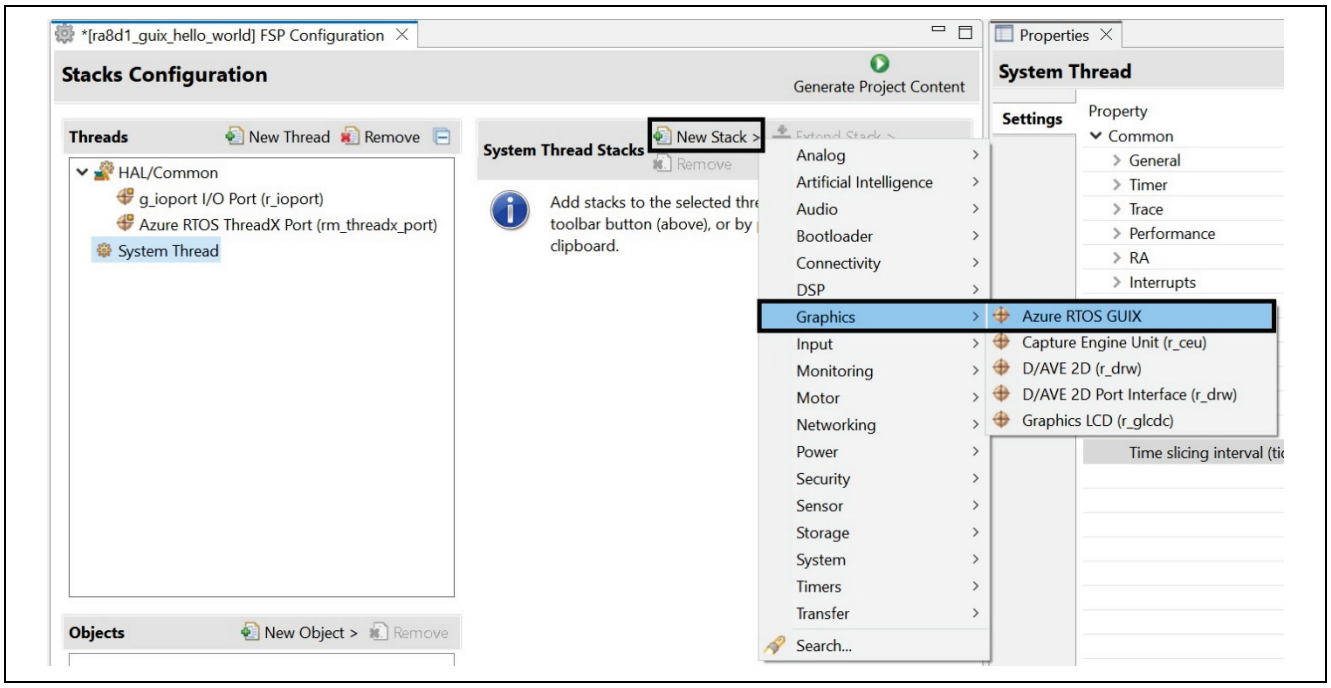


Figure 11. Add Azure RTOS GUIX

9. Settings properties for **Azure RTOS GUIX (rm_guix_port)** and **GLCDC Display.**

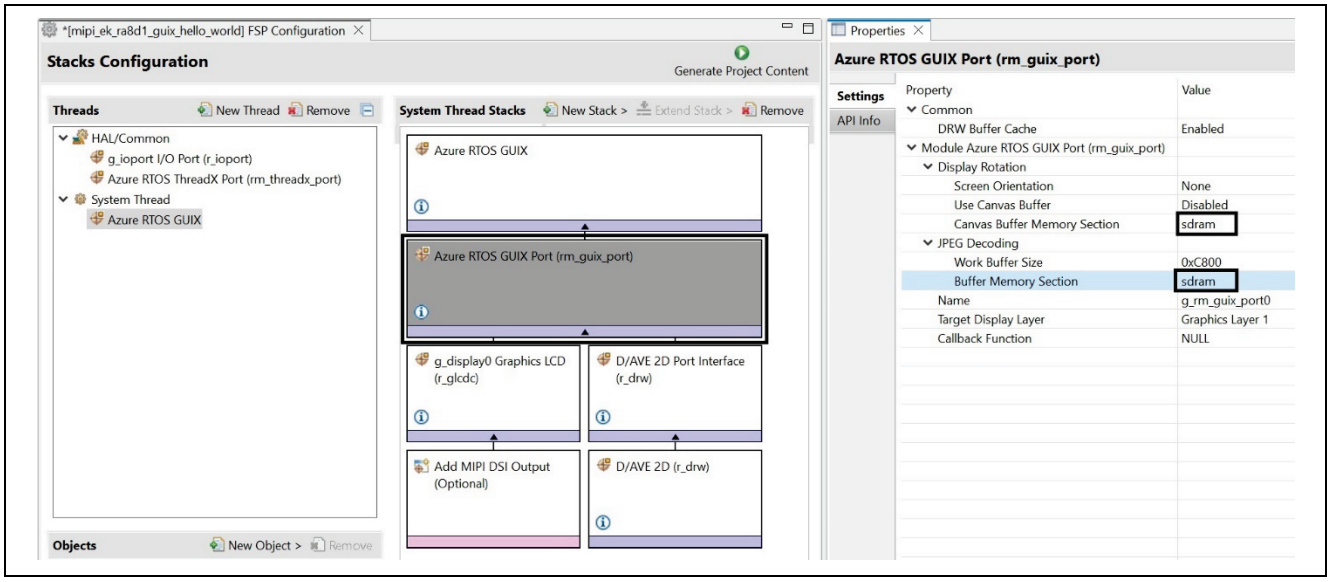


Figure 12. Setting and checking Properties for Hardware

10. Settings properties for **Graphics LCD**

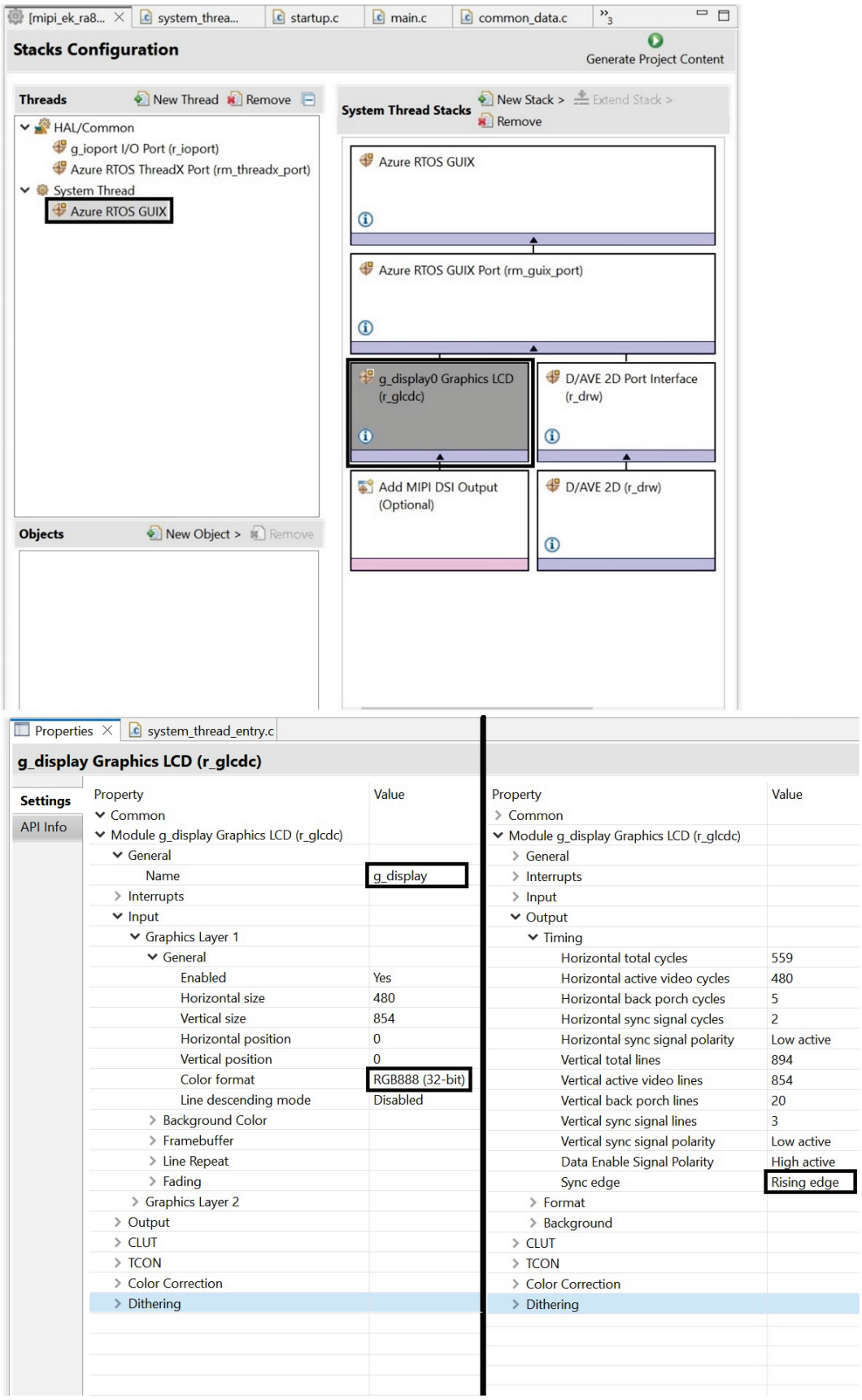


Figure 13. Setting Properties for GLCDC DISPLAY

11. Add the MIPI DSI Output by clicking on the “Add” option attached to the r_glcdc stack, and fill out the settings properties for **MIPI Graphics LCD (r_mipi_dsi)** module based on the following image:
    Note: All the other settings are set to the default value for the LCD pixel 480x854 so the details are not shown here.

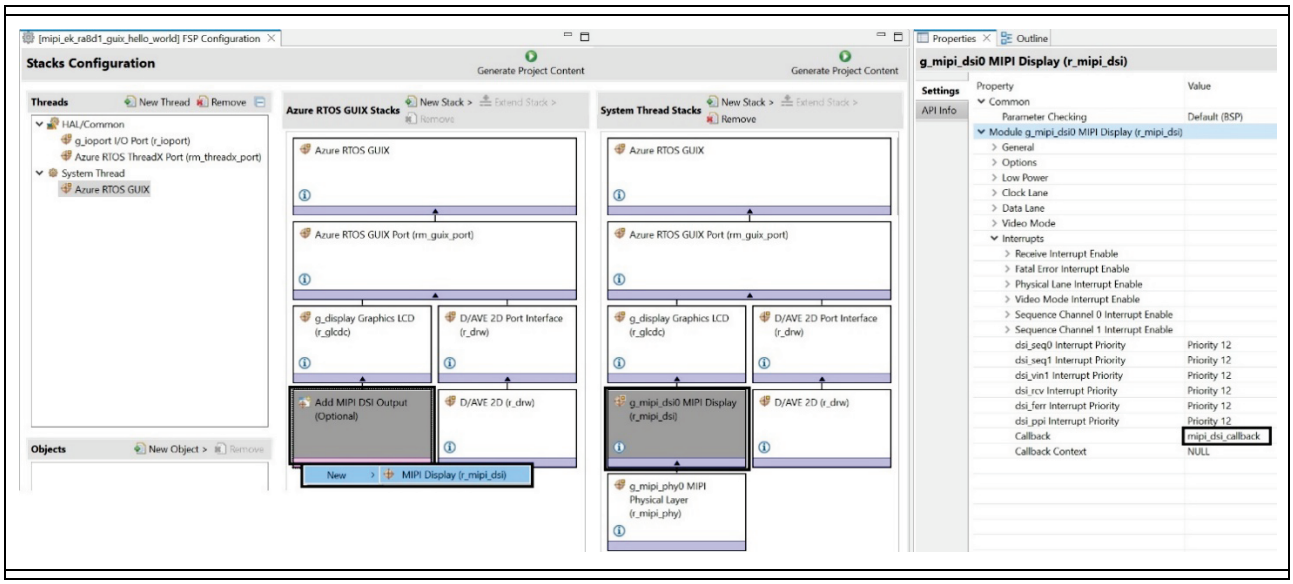


Figure 14. Adding and Setting Properties for MIPI Graphic LCD

12. Click “**New Stack**” and add **Timers > PWM Timer (r_gpt)**.

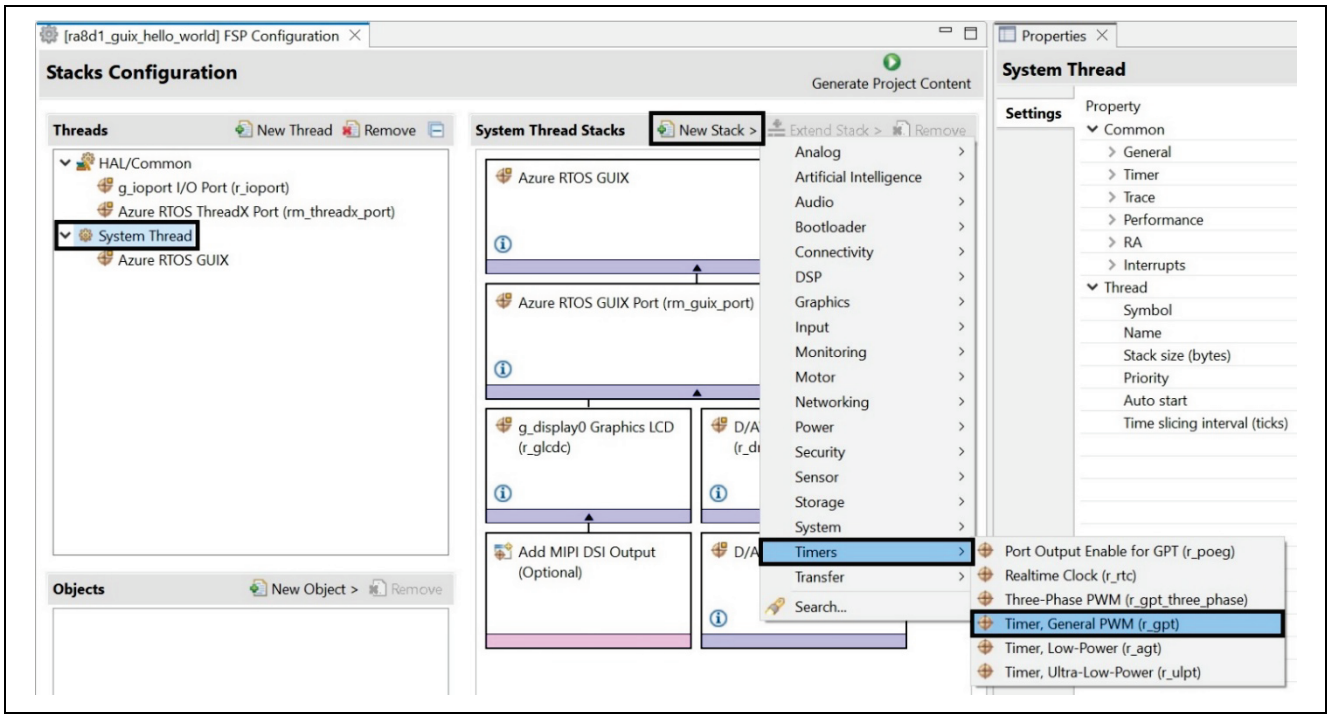


Figure 15. Add the Timer

13. Set the **Timer, General PWM (r_gpt)** module properties as shown below

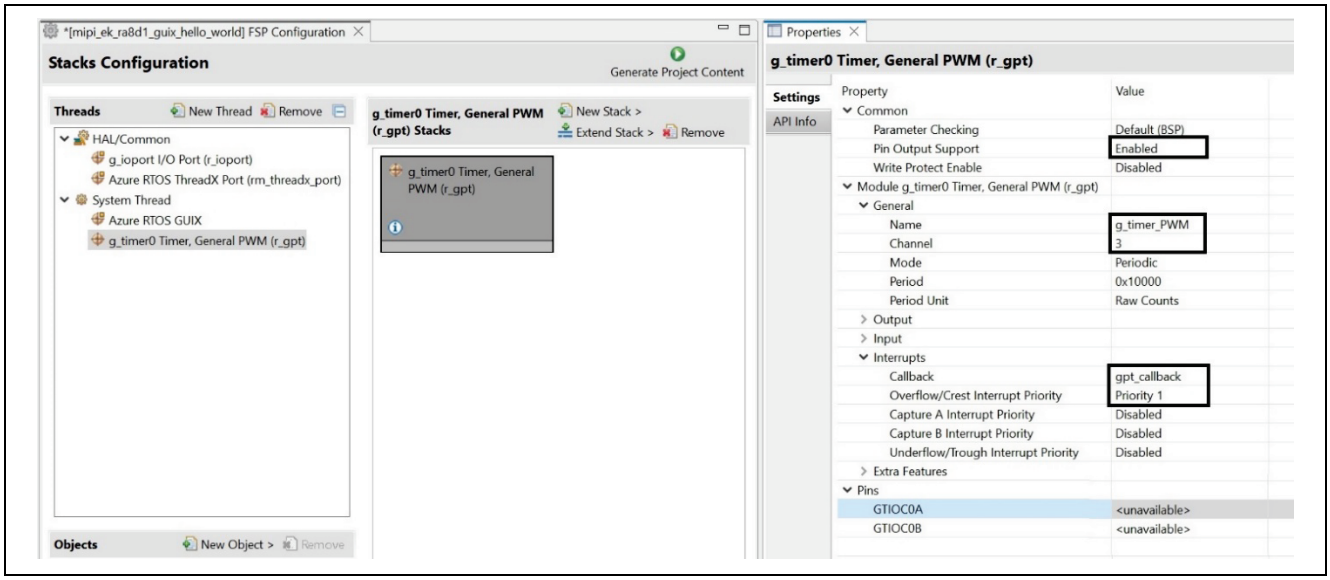


Figure 16. Settings Timer properties

14. Now we will set the output pins for the project. Navigate to the **Pins** tab. Configure pin **P404,** the **DISP_BLEN** signal of the LCD panel, and set the **Mode** to **Output mode (Initial High)**.

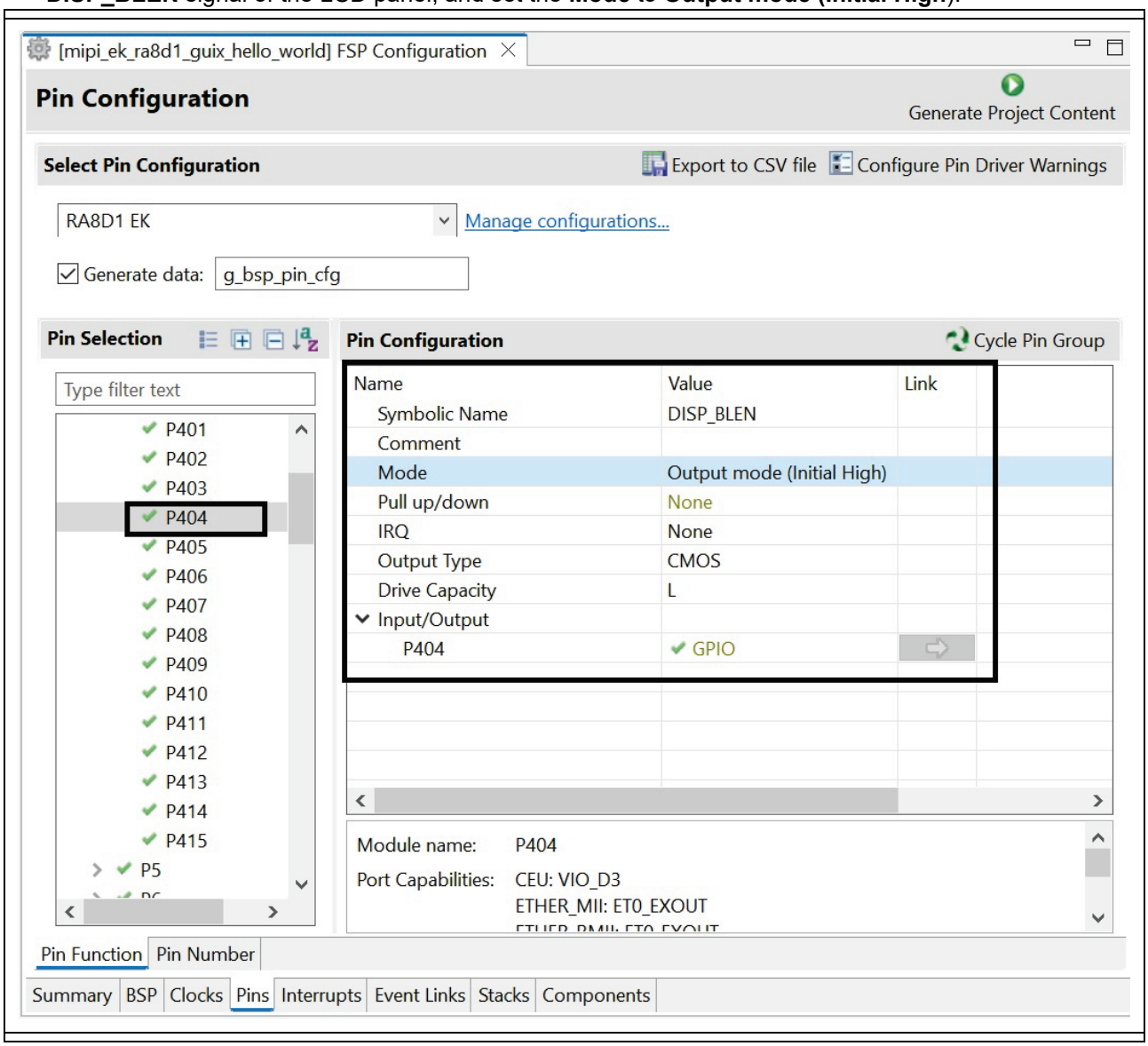


Figure 17. Settings pin P404 for DISP_BLEN

15. LCD_BLEN pin from the LCD panel will connect to DISP_BLEN pin P404 of the board RA8D1

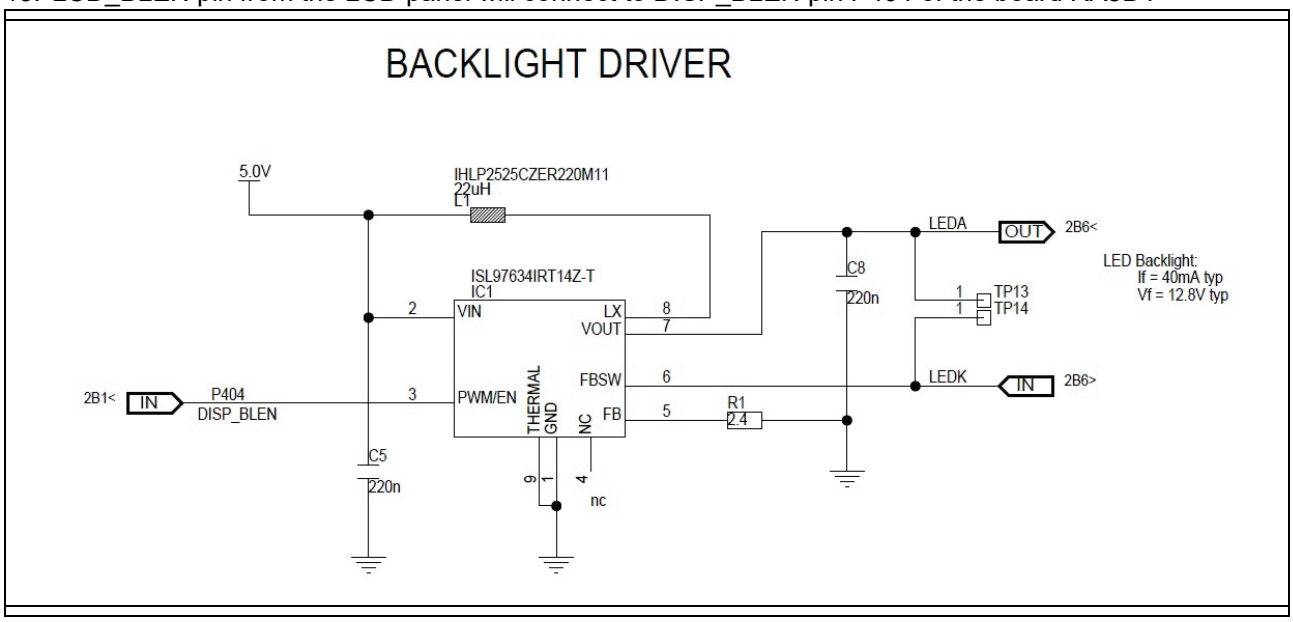


Figure 18. LCD_BLEN will connect to DISP_BLEN pin P404

16. Refer to "`system_thread_entry.c`" file in "**Source.zip**" folder for more information on what the system thread does. This function shown below controls the PWM output.

```
/* Initialize GPT module */
err = R_GPT_Open(&g_timer_PWM_ctrl, &g_timer_PWM_cfg);

/* Handle error */
handle_error(err, "** R_GPT_Open API failed ** \r\n");

/* Enable GPT Timer */
err = R_GPT_Enable(&g_timer_PWM_ctrl);

/* Start GPT timer */
err = R_GPT_Start(&g_timer_PWM_ctrl);
```

Figure 19. gpt_timer_PWM function located in "system_thread_entry.c"

3. Add and Configure the gt911 Touch Driver

1. Click “**New Thread**” and name “**Touch Thread**” with setting as shown below.

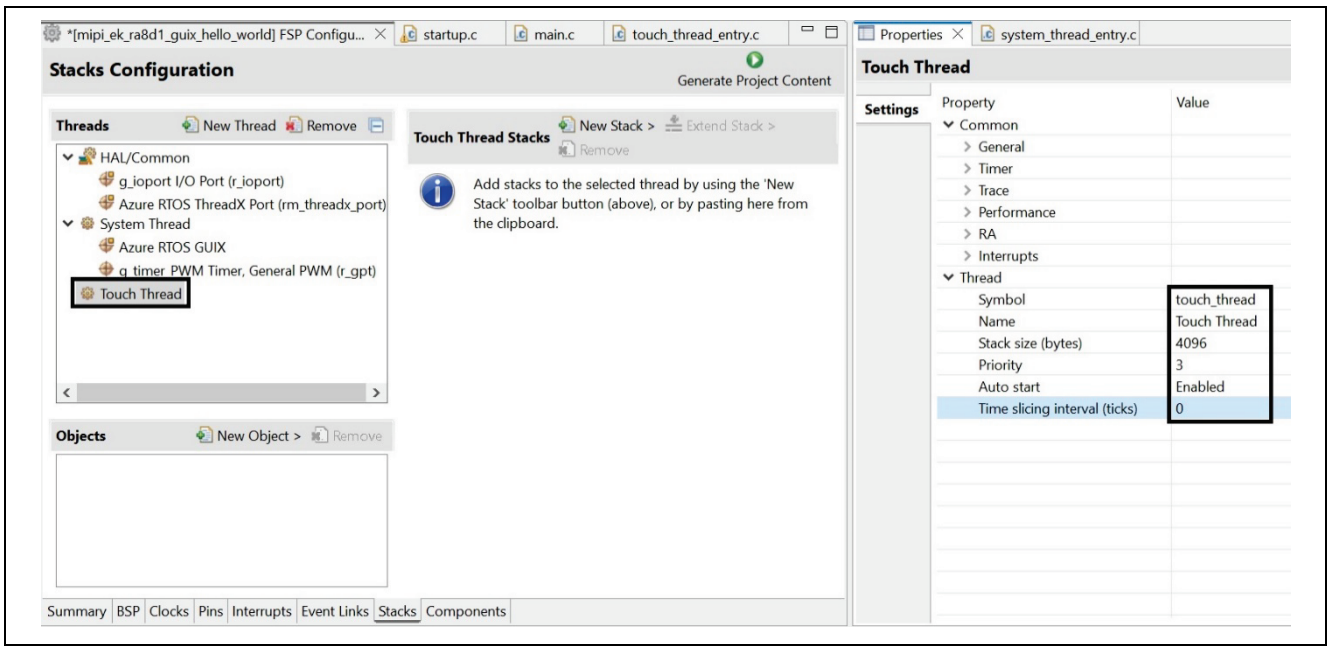


Figure 20. Add and Set Touch Thread Properties

2. From Threads window, select HAL/Common and click “**New Stack**” and add **Input > External IRQ (r_icu)** module.

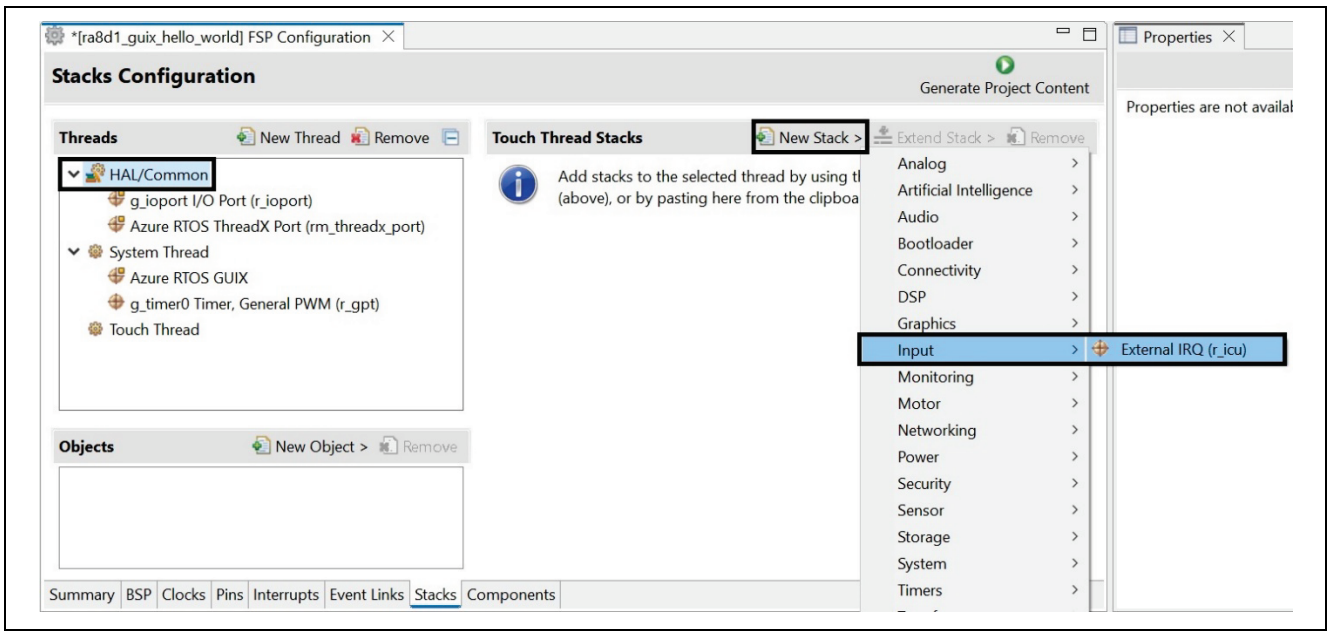


Figure 21. Add External IRQ

3. Name the module "**g_int_gt911**" and change the default settings based on the properties illustrated in the following image. Click "arrow"  to go to the **Pins** tab automatically for setting pin P510.

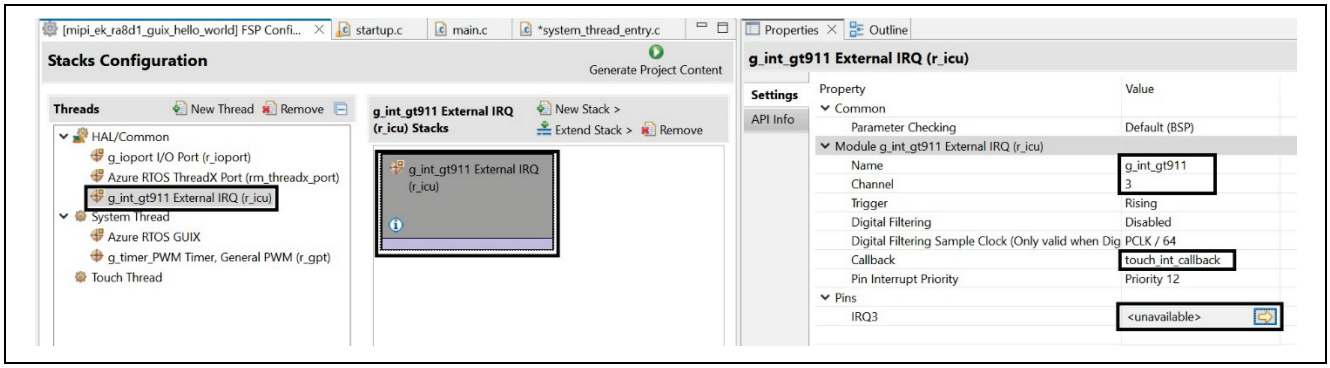


Figure 22. Setting External IRQ Properties

4. Change the configuration pin **P510** for **DISP_INT** signal pin so the Operation Mode is **Custom** and the Input/Output of IRQ3 is **P510**.

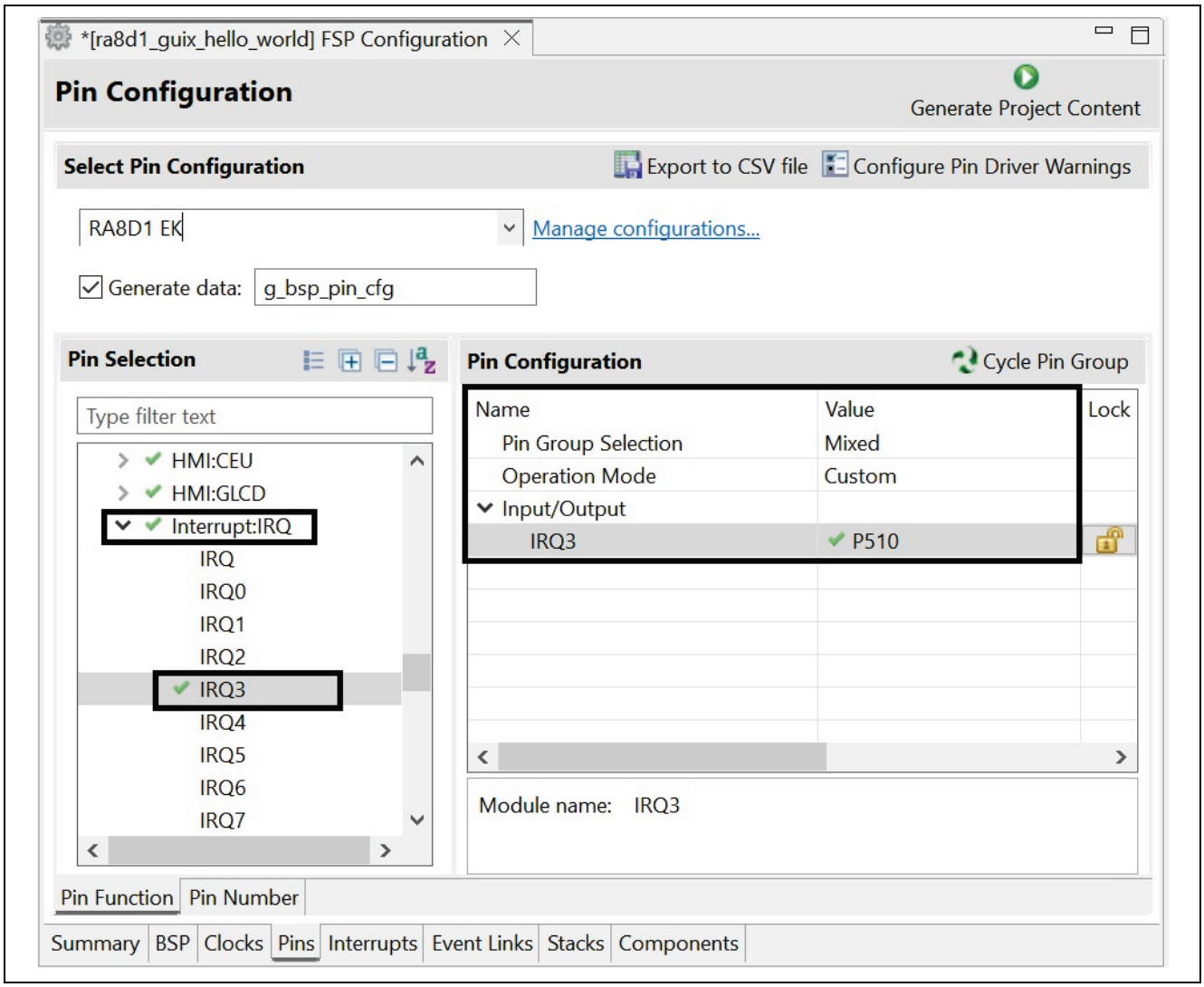


Figure 23. Configuration Pin P510 "IRQ3"

5. Go back to the **Stacks** tab. From the Threads window, select HAL/Common and click "**New Stack**" and add the **Connectivity > I2C Master (r_iic_master)** module**.**

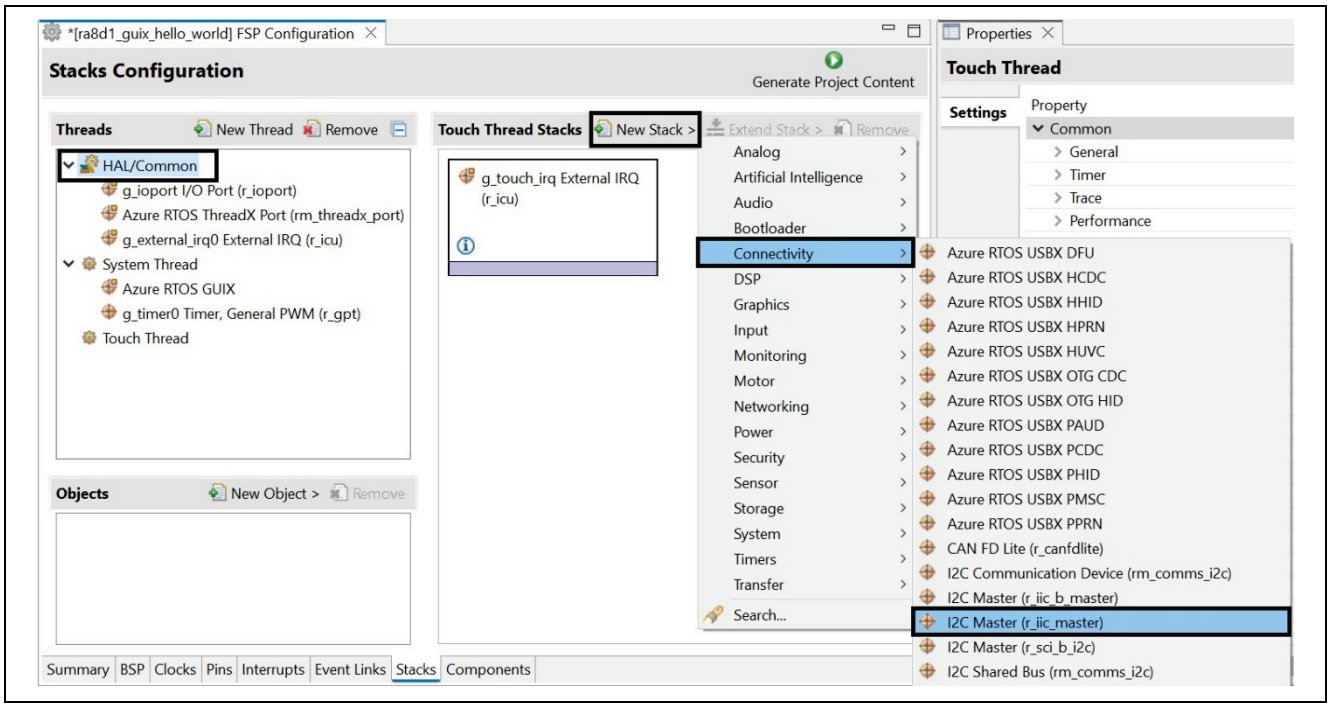


Figure 24. Add Touch I2C Master

6. Name the module “**g_i2c_gt911**” and change the default settings based on the properties illustrated in the following image. Click the “arrow”  to go to the **Pins** tab automatically for setting pins **SCL1** and **SDA1**.

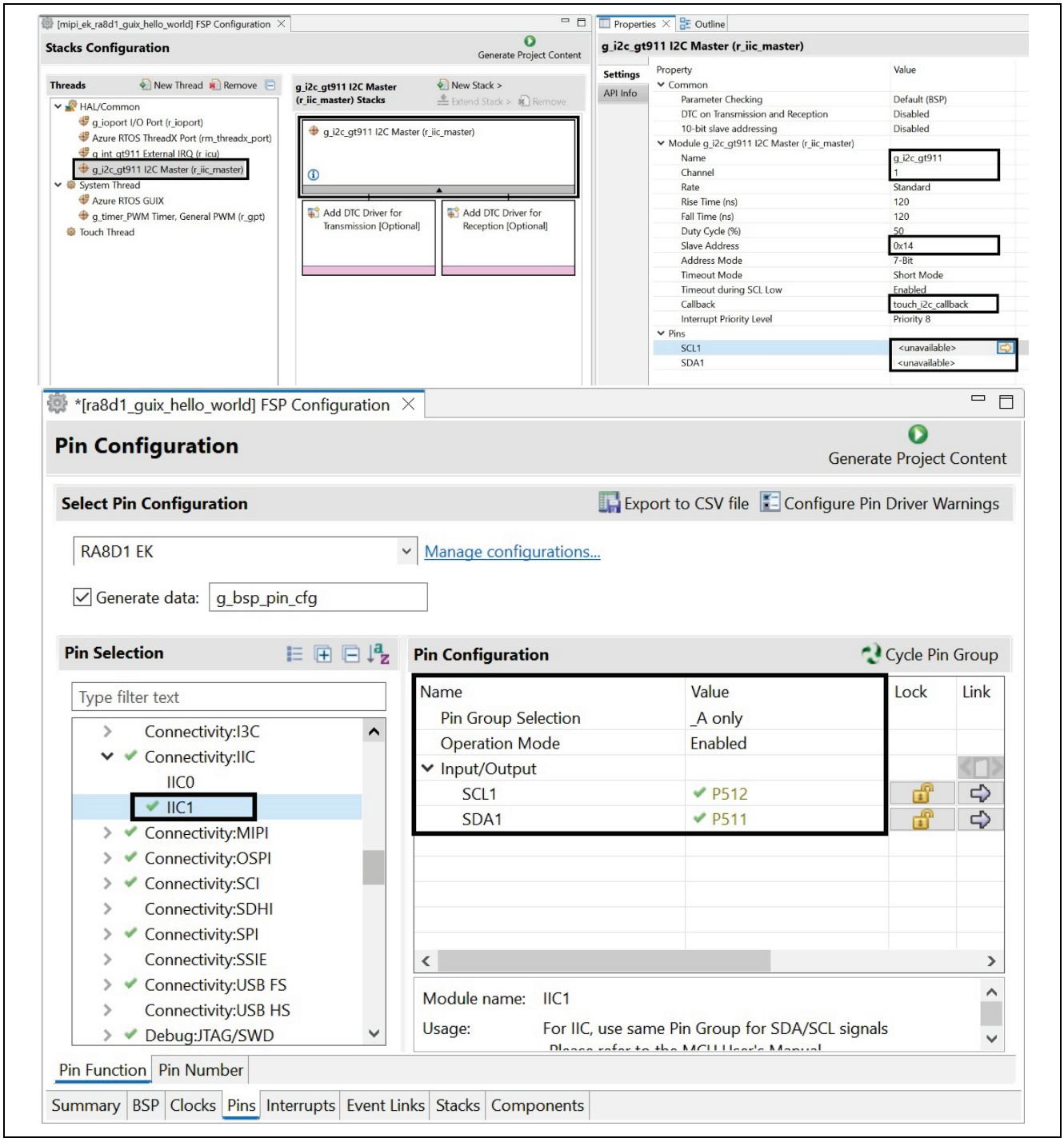


Figure 25. Name and Settings Properties

7. Click **New Object** and add **Semaphore**. This semaphore will indicate when a gt911 touch interrupt occurs.

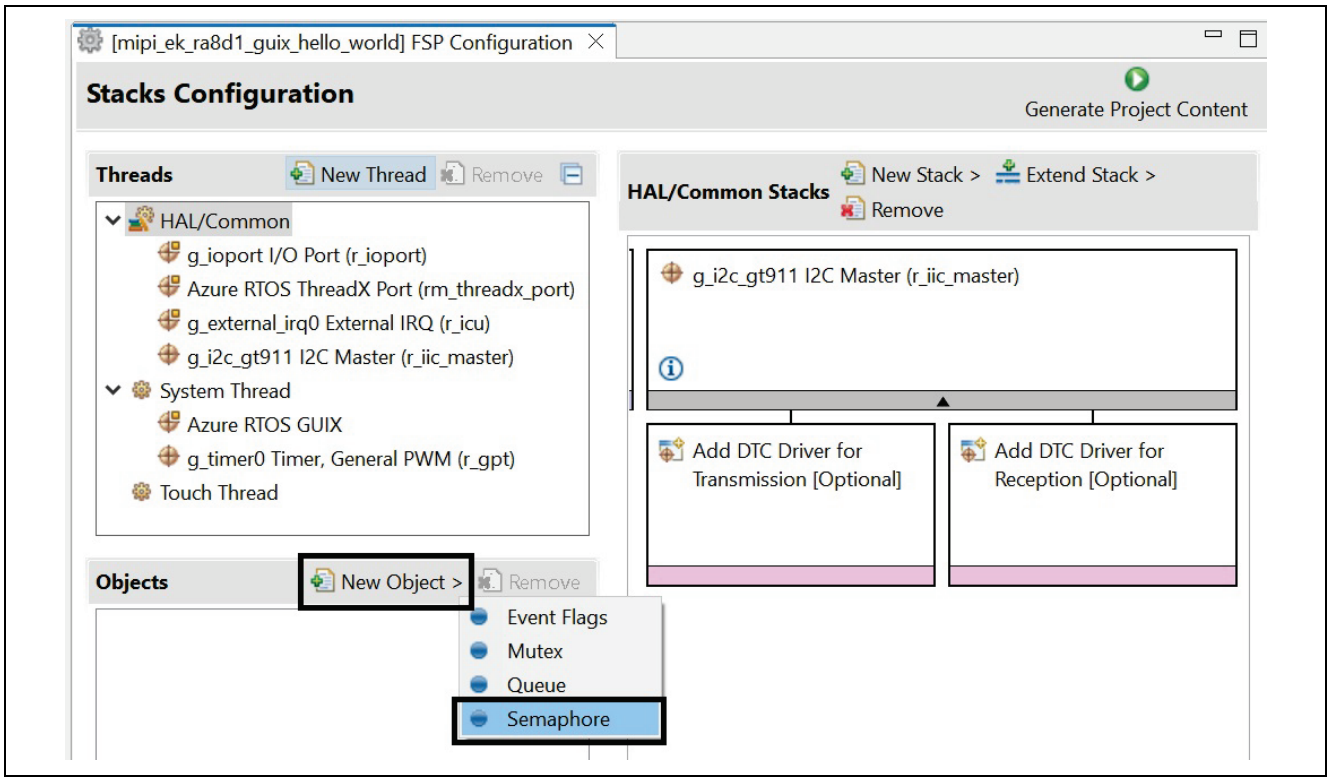


Figure 26. Add New Semaphore

8. Name it **g_semaphore_gt911_irq** and set the following properties.

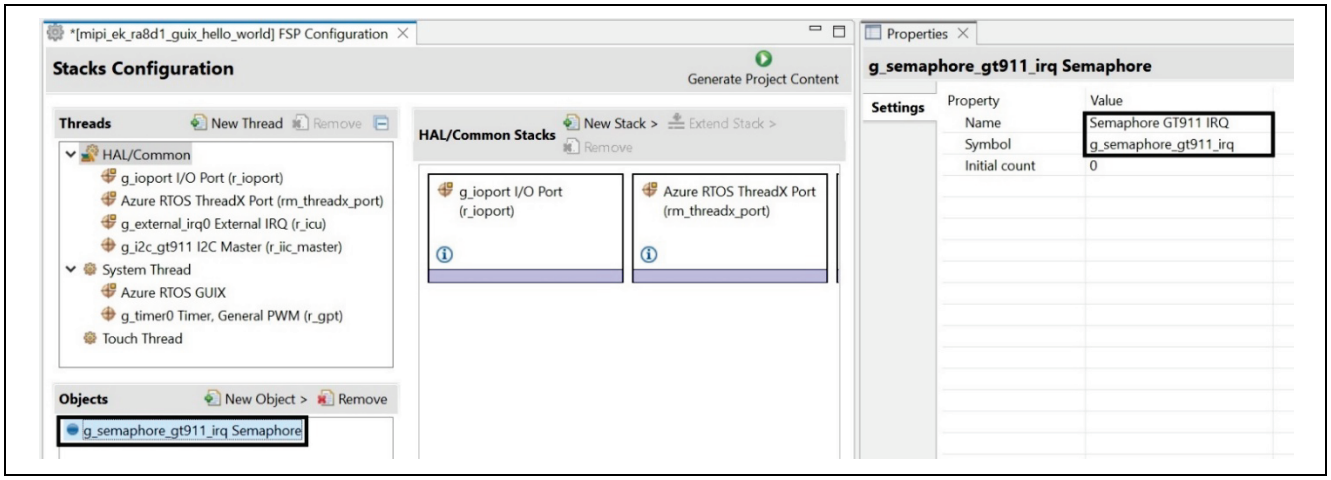


Figure 27. Add and Name Touch Semaphore

9. Click **New Object** and add another **Semaphore**. This semaphore will be used for i2c timing control.

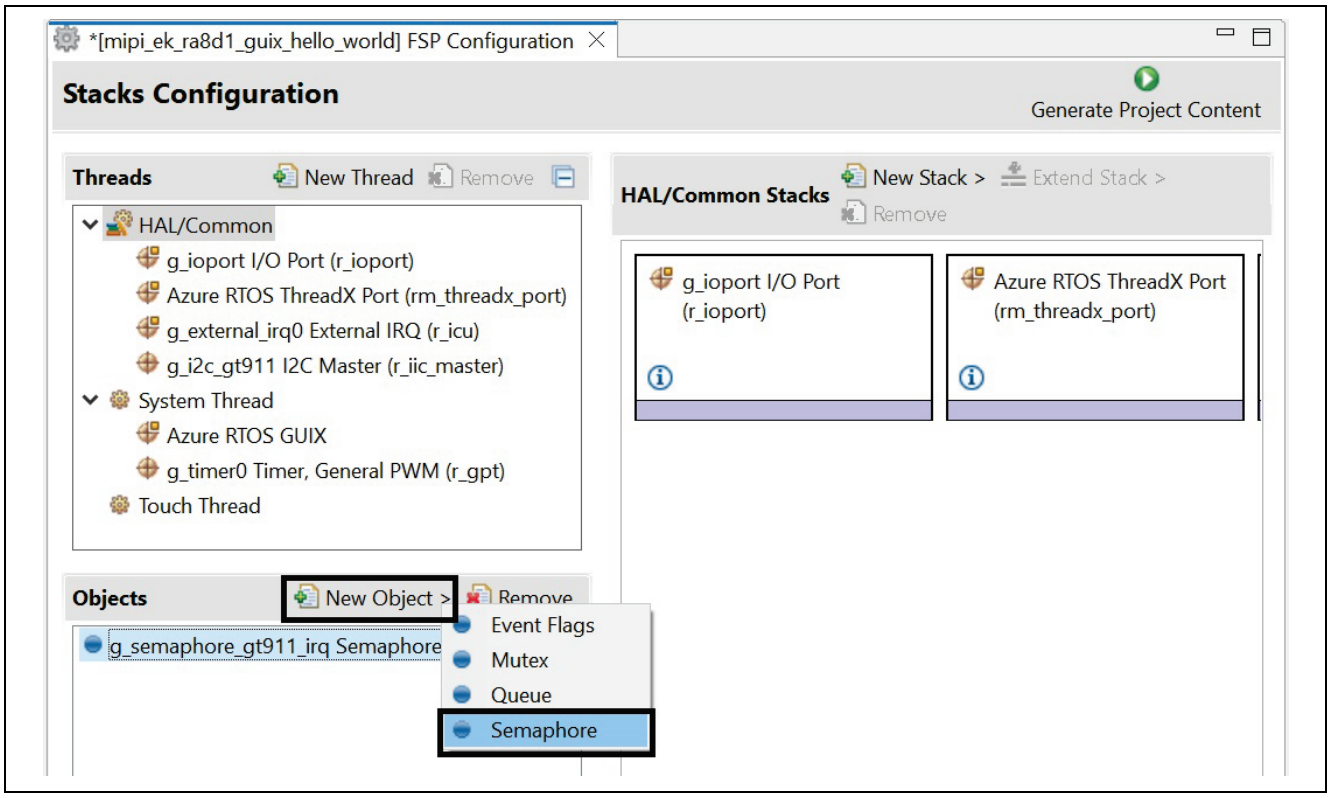


Figure 28. Add Another New Semaphore

10. Name it "**g_semaphore_gt911_i2c**" and set the following properties.

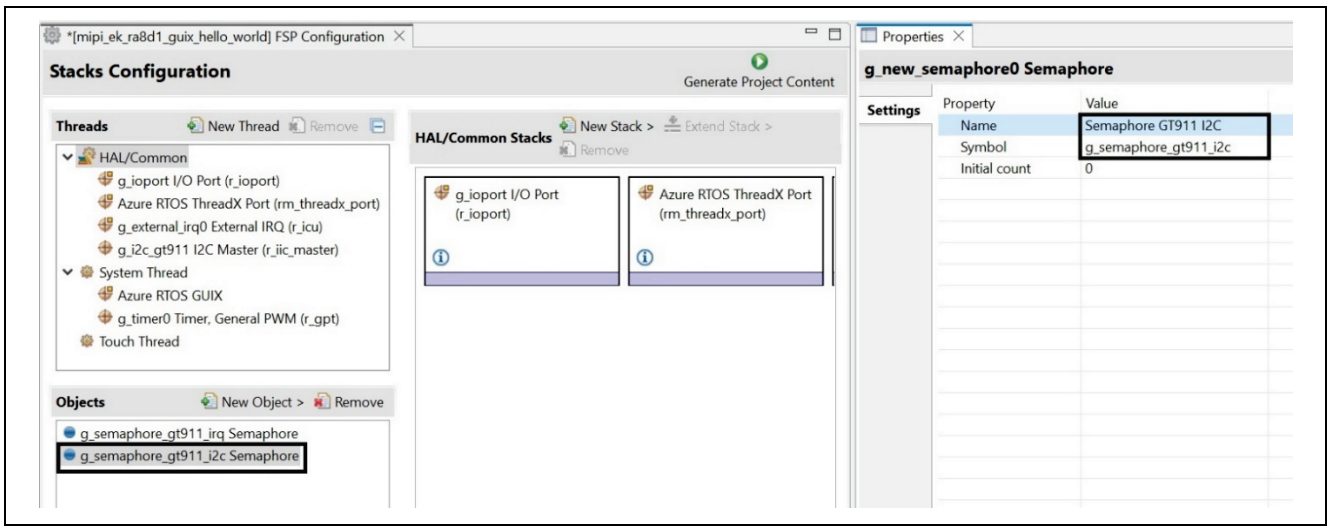


Figure 29. Add and Name I2C Semaphore

11. Click **New Object** and add **Event Flags**. This event flag will be used by the gt911 drivers.

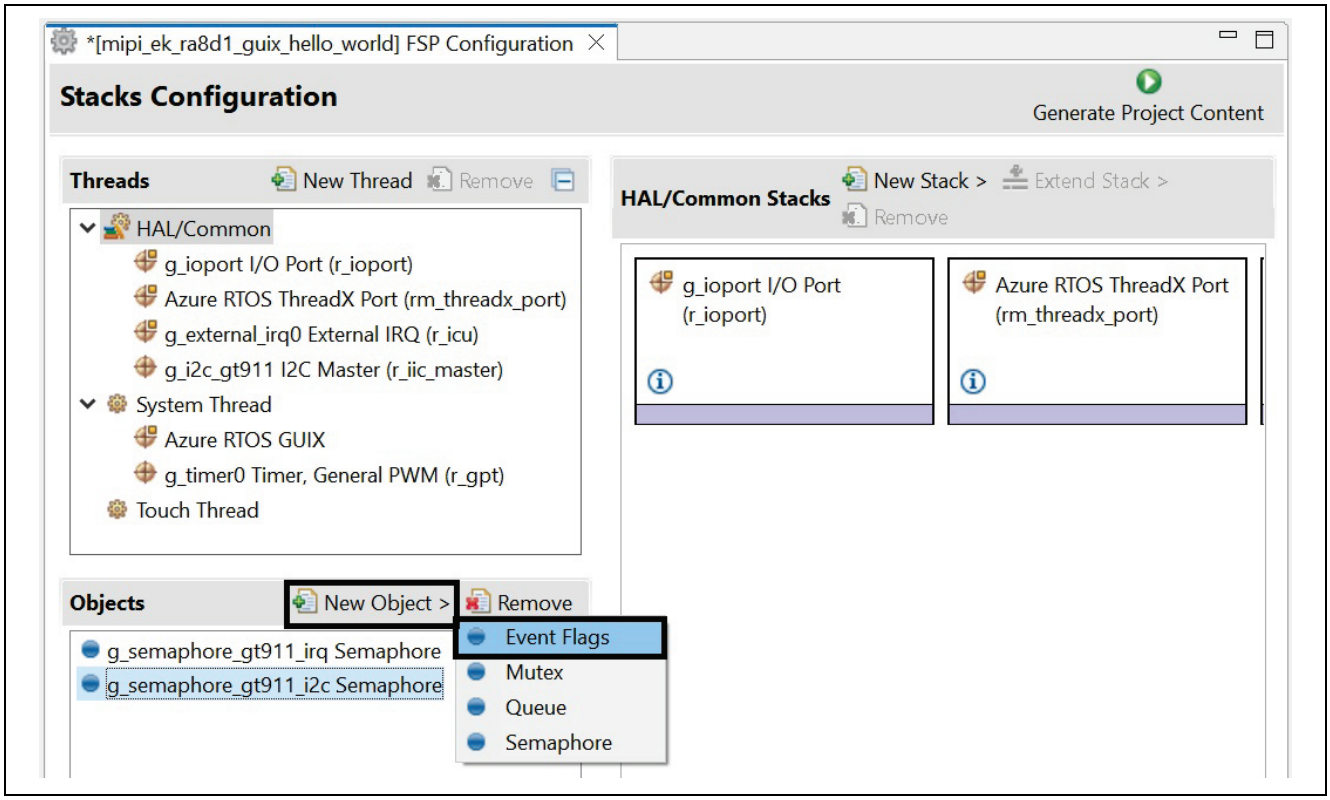


Figure 30. Add Event Flags

12. Name it "**g_event_flags_gt911**" and set the following properties as shown.

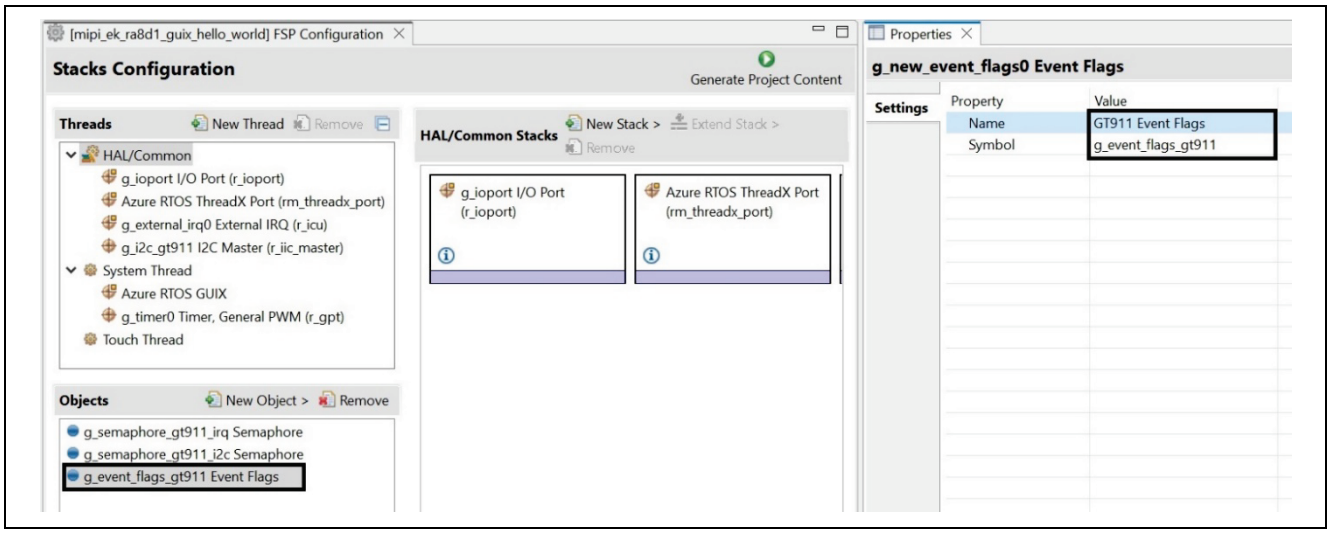


Figure 31. Name Event Flags and Setting Properties

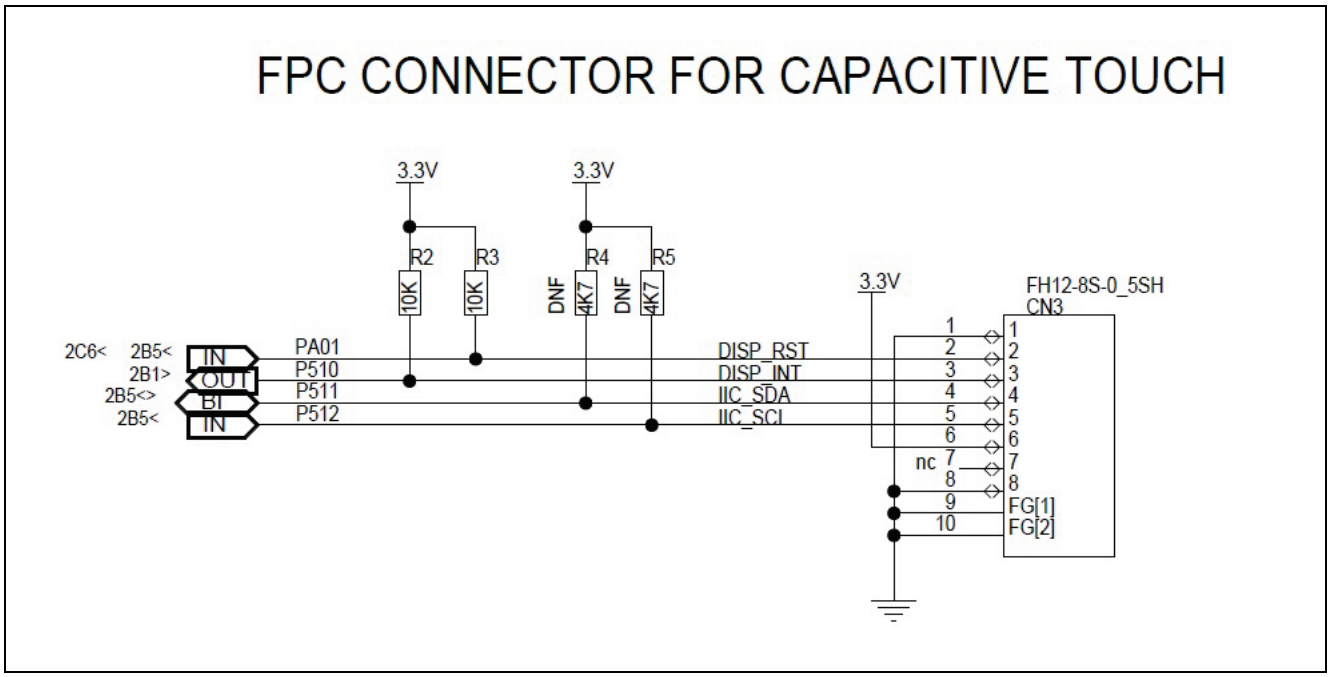


Figure 32. Touch Pins Connected on the MIPI LCD

The pins marked in Figure 32 are used for the touch panel controller on the MIPI LCD board:

- DISP_INT interrupt (P510) is used to trigger touch events.
- I2C channel 1 (P512, P511) is used to read and write data to the touch controller.
- Touch driver folder `touch_gt911` for touch function is inside the provided `Source` folder.
- PA01 is used to reset the MIPI LCD's touch controller.

13. Refer to the `touch_thread_entry.c` file in `Source.zip` for more information. The following code initializes the touch controller and process touch events.

```c
/* New Thread entry function */
void touch_thread_entry(void)
{
    GX_EVENT gxe = {0};
    gxe.gx_event_type = GX_EVENT_PEN_UP;
    uint8_t status = 0;
    /* TODO: add your own code here */


    touch_gt911_ctrl_t gt911_ctrl;

    fsp_err_t err = FSP_SUCCESS;

    uint32_t gt911_err = 0;

    uint16_t gt911_fw_version = 0;

    UINT tx_err = TX_SUCCESS;

#if USE_EVENT_FLAGS == 1
    ULONG actual_flags = 0;
#endif


    err = R_TOUCH_GT911_Validate(&gt911_cfg, &gt911_err);

    if(FSP_SUCCESS != err)
    {
        __BKPT(0);
    }

    memset(&gt911_ctrl, 0, sizeof(touch_gt911_ctrl_t));

    err = R_TOUCH_GT911_Open(&gt911_ctrl, &gt911_cfg);

    if(FSP_SUCCESS != err)
    {
        __BKPT(0);
    }


    err = R_TOUCH_GT911_Reset(&gt911_ctrl);

    if(FSP_SUCCESS != err)
    {
        __BKPT(0);
    }
#if 1
    err = R_TOUCH_GT911_VersionGet(&gt911_ctrl, &gt911_fw_version);

    if(FSP_SUCCESS != err)
    {
        __BKPT(0);
    }
#endif
    while (FSP_SUCCESS == err)
    {

#if USE_EVENT_FLAGS
        tx_err = tx_event_flags_get(&g_event_flags_gt911,
                                    TOUCH_GT911_INT,
                                    TX_OR_CLEAR,
                                    &actual_flags,
                                    TX_WAIT_FOREVER);
#elif USE_SEMAPHORES == 1
        tx_err = tx_semaphore_get(&g_semaphore_gt911_irq, TX_WAIT_FOREVER);
#endif

        if(TX_SUCCESS != tx_err)
        {
            err = FSP_ERR_INVALID_STATE;
            __BKPT(0);
        }
#if USE_EVENT_FLAGS == 1
        if((actual_flags & TOUCH_GT911_INT) != TOUCH_GT911_INT)
        {
            continue;
        }
#endif
        err = R_TOUCH_GT911_StatusGet(&gt911_ctrl, &status, true);

        if(status & GT911_BUFFER_READY)
        {
            gt911_touch_coords.touch_count = status & GT911_MASK_TOUCH_COUNT;
            err = R_TOUCH_GT911_PointsGet(&gt911_ctrl, &gt911_touch_coords);

            if(0U < gt911_touch_coords.touch_count && GX_EVENT_PEN_UP == gxe.gx_event_type)
            {
                /* Send only the TOUCH event of the first finger to the GUIX Model View Controller. */

                translate_touch_coordinates(&gt911_touch_coords.point[0], ORIENTATION_ROTATE_CW);

                gxe.gx_event_payload.gx_event_pointdata.gx_point_x       = gt911_touch_coords.point[0].x;
                gxe.gx_event_payload.gx_event_pointdata.gx_point_y       = gt911_touch_coords.point[0].y;
                gxe.gx_event_type = GX_EVENT_PEN_DOWN;
                gx_system_event_send(&gxe);

            }

            if ( 0U == gt911_touch_coords.touch_count && GX_EVENT_PEN_DOWN == gxe.gx_event_type)
            {
               gxe.gx_event_type = GX_EVENT_PEN_UP;
               gx_system_event_send(&gxe);
            }
            tx_thread_sleep (1);

                    if(FSP_SUCCESS != err)
            {
                __BKPT(0);
            }
        }
    }


    err = R_TOUCH_GT911_Close(&gt911_ctrl);
}
```

Figure 33. Initializes the Touch Controller and Process Touch Events

14. From **Stacks Configuration**, click "**Generate Project Content**" to generate project content.

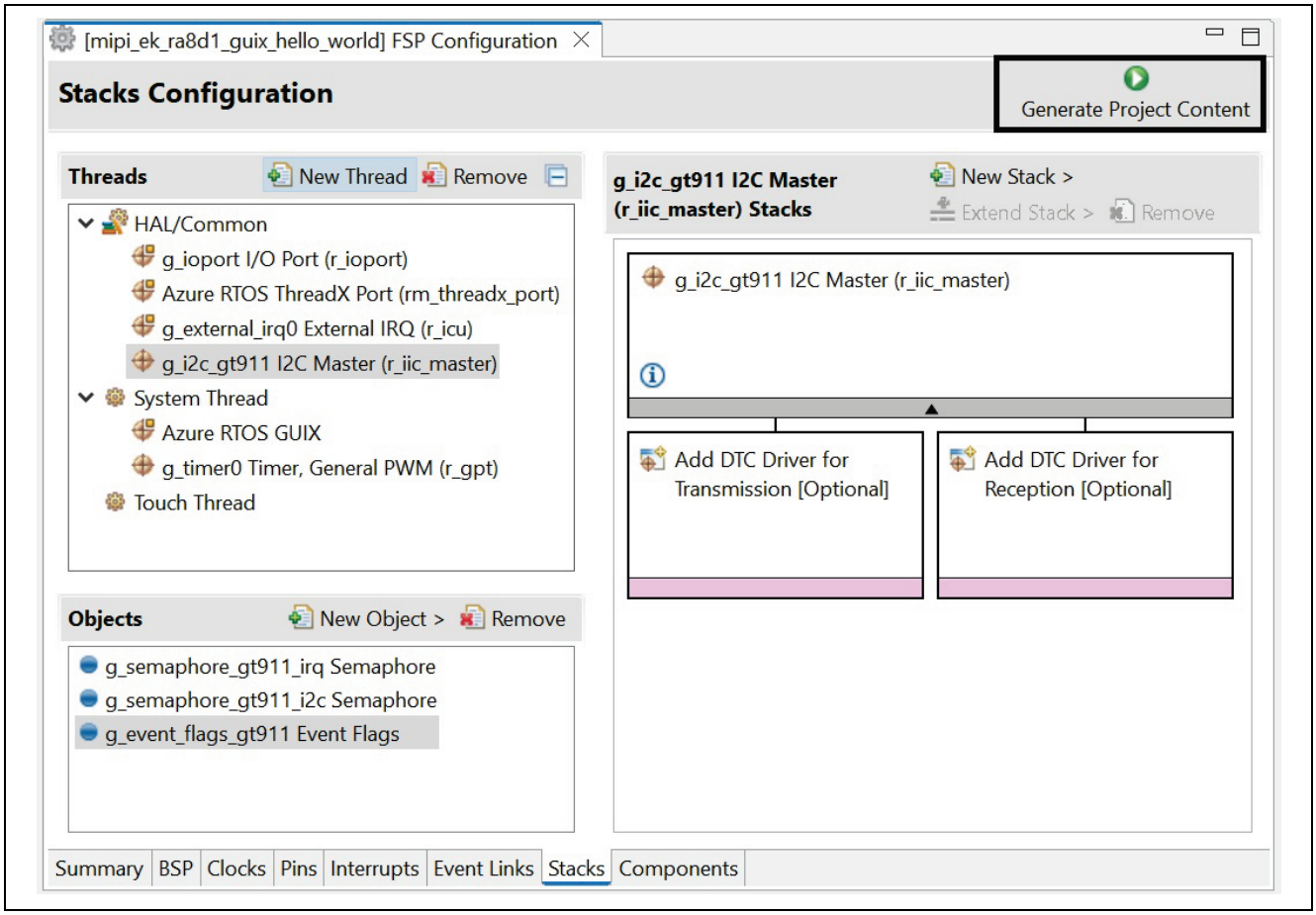


Figure 34. Generate Project Content

15. Unzip and open the provided folder `Source.zip`. Copy 6 files and 2 folders `SEGGER_RTT`, `touch_gt911` and paste into the folder `src` of your project "**mipi_ek_ra8d1_guix_hello_world**". Select "Yes" if you are prompted to overwrite existing files.
    `SEGGER_RTT` folder
    - `touch_gt911 folder`
    - `common_utils.h`
    - `hal_entry.c`
    - `system_thread_entry.c`
    - `system_thread_entry.h`
    - `touch_thread_entry.c`
    - `windows_handler.c`

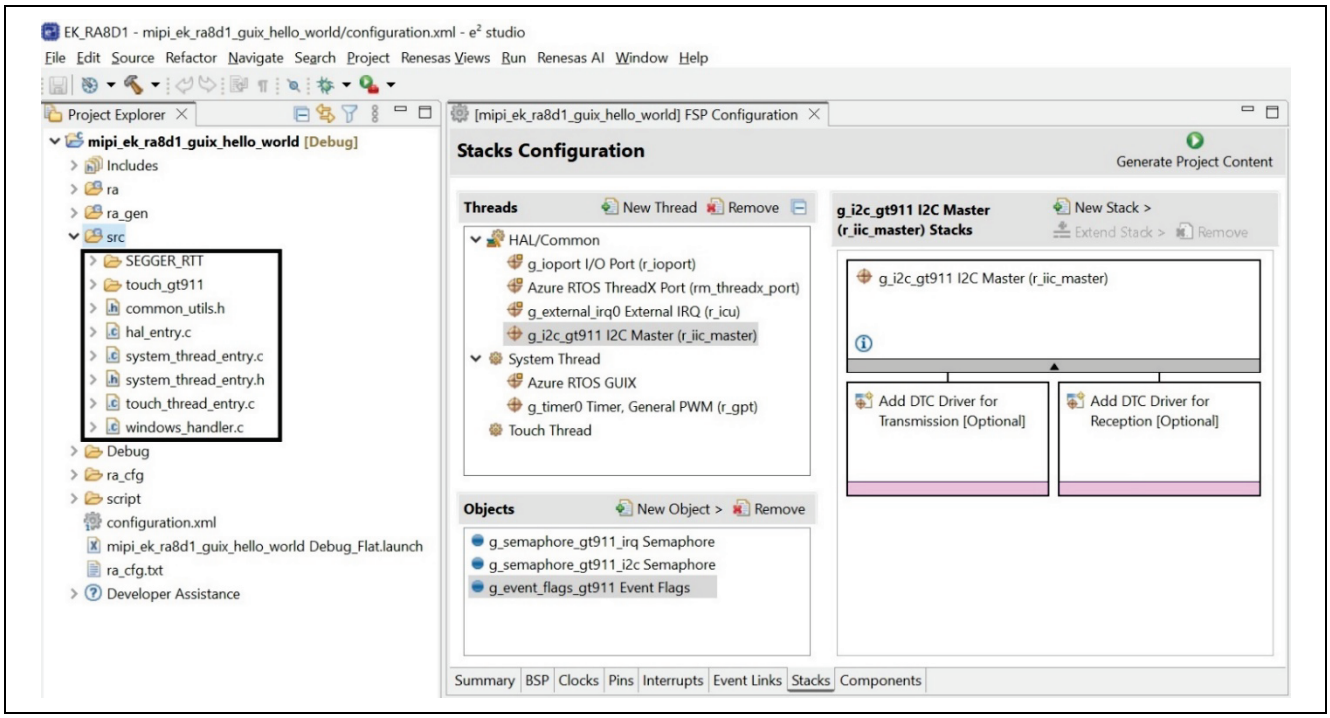


Figure 35. Copy All Folders and Files from Source.zip to the e² studio Project's src Folder

4. Create Folders in the “mpi_ek_ra8d1_guix_hello_world” Project for Azure RTOS GUIX Studio Project

1. Under folder `src`, right click to create a new folder and name it `guix_gen` following the image below, then click **Finish**. This folder will store the files that a GUIX Studio graphics application generates.

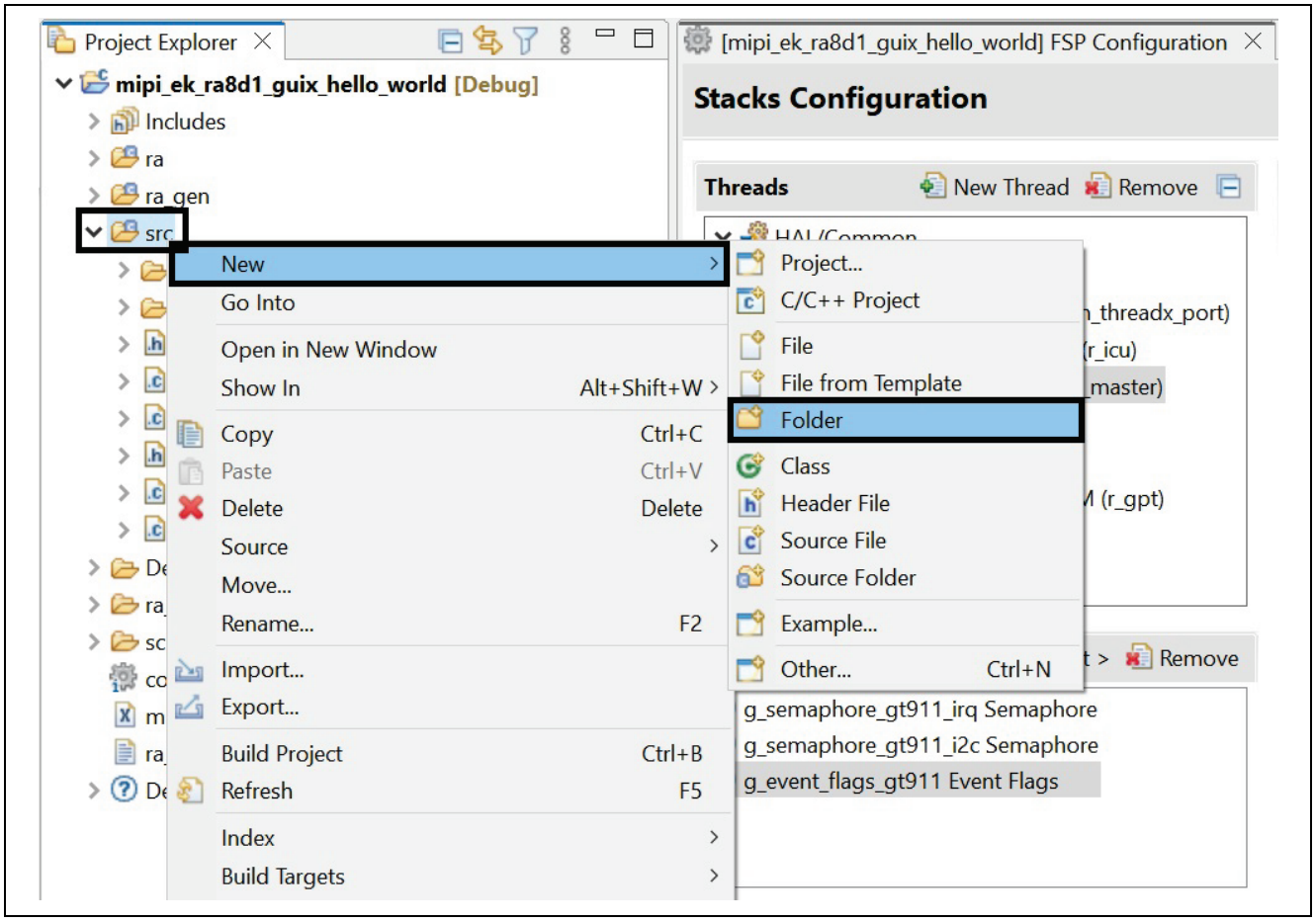


Figure 36. Create a New Folder Named “guix_gen”

2. Create a new folder at the topmost project level and name it `guix_studio` by following the image below. Click **Finish**. This folder will store the GUIX Studio project.

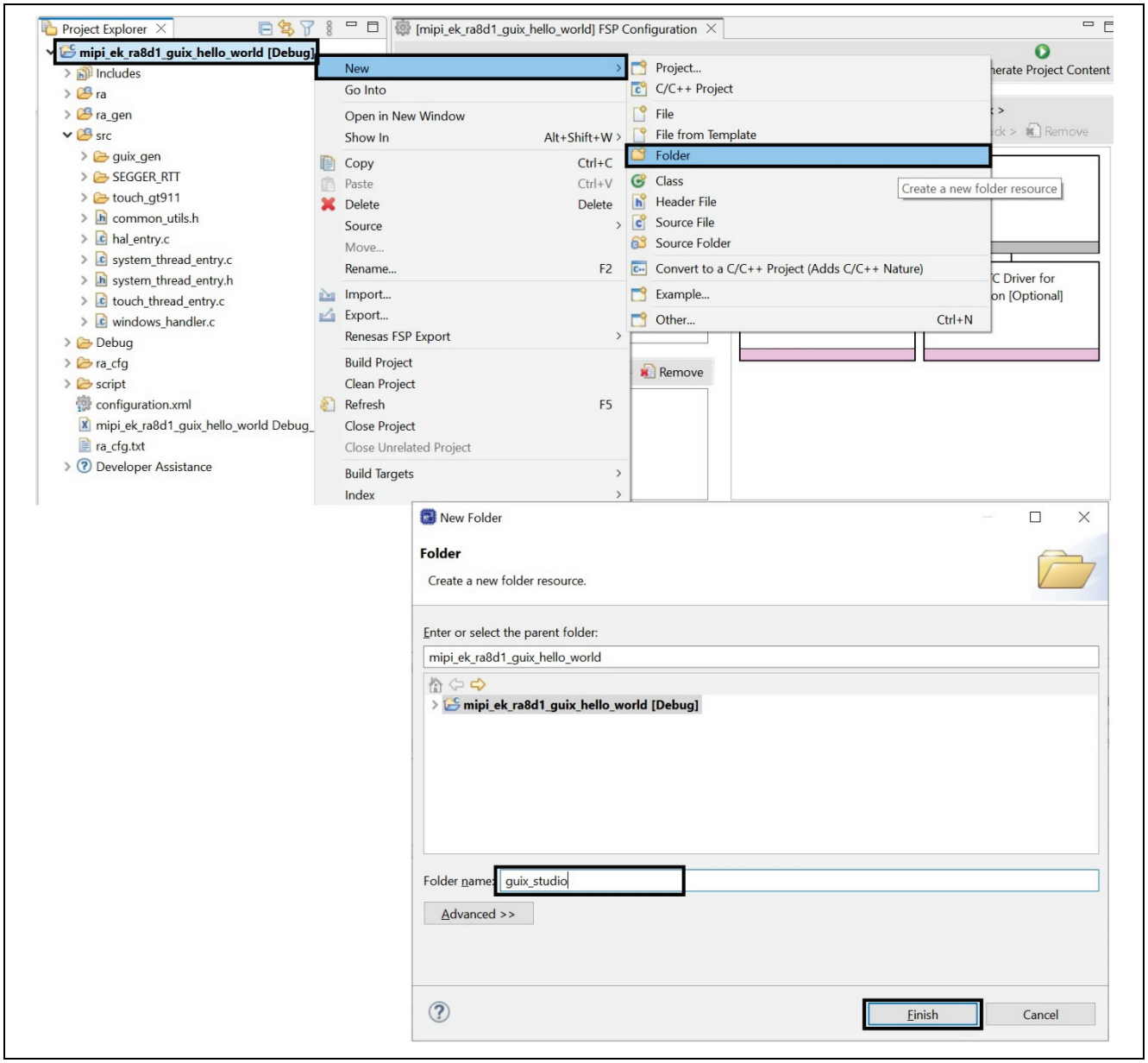


Figure 37. Create New Folder and Name “guix_studio”

3. Under folder `guix_studio`, create a new folder and name `GNU`. Follow the image below, then click **Finish.**

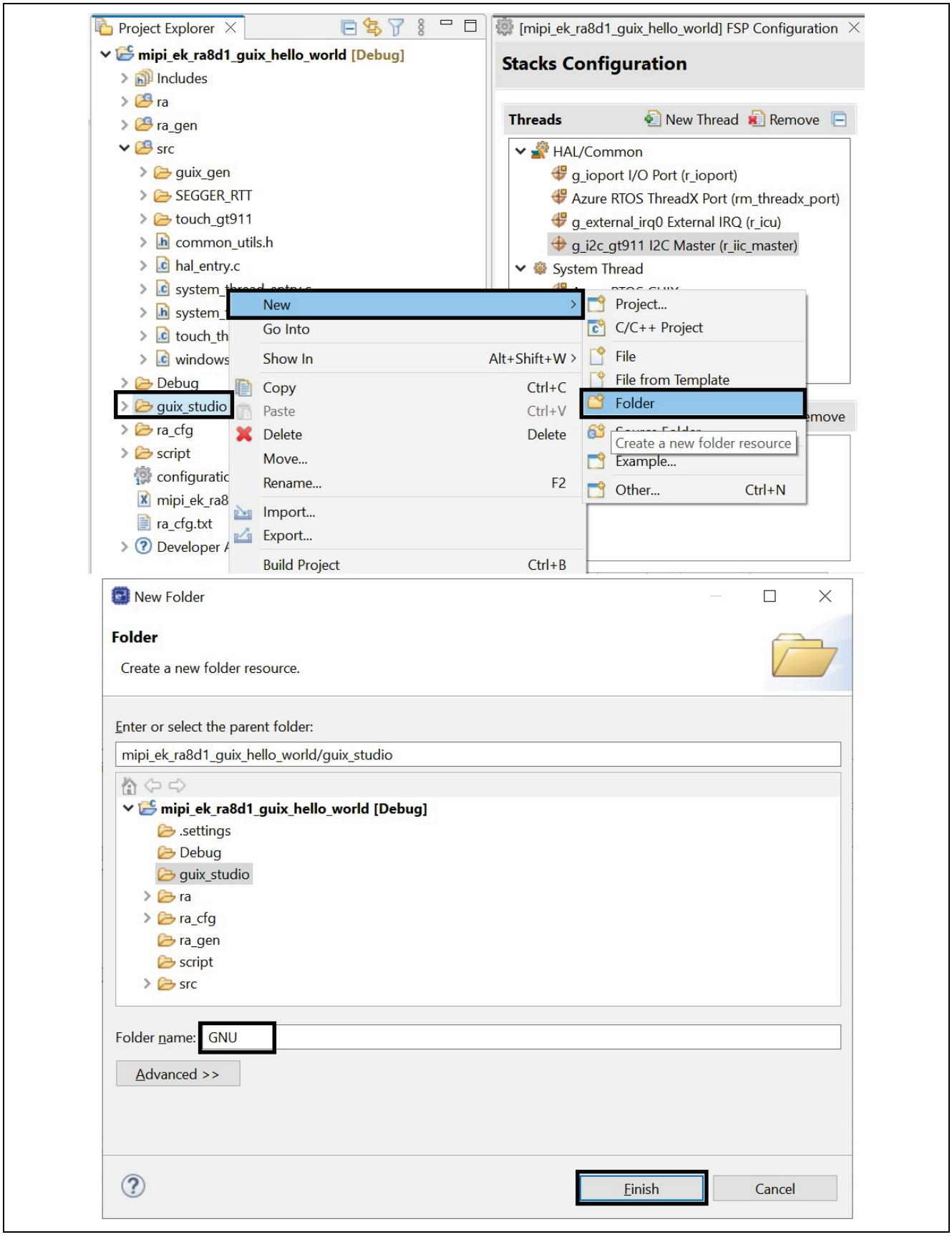


Figure 38. Create New Folder and Name “GNU”

4. After the sub-folder `GNU` is created, the folder structure should look like the image below.

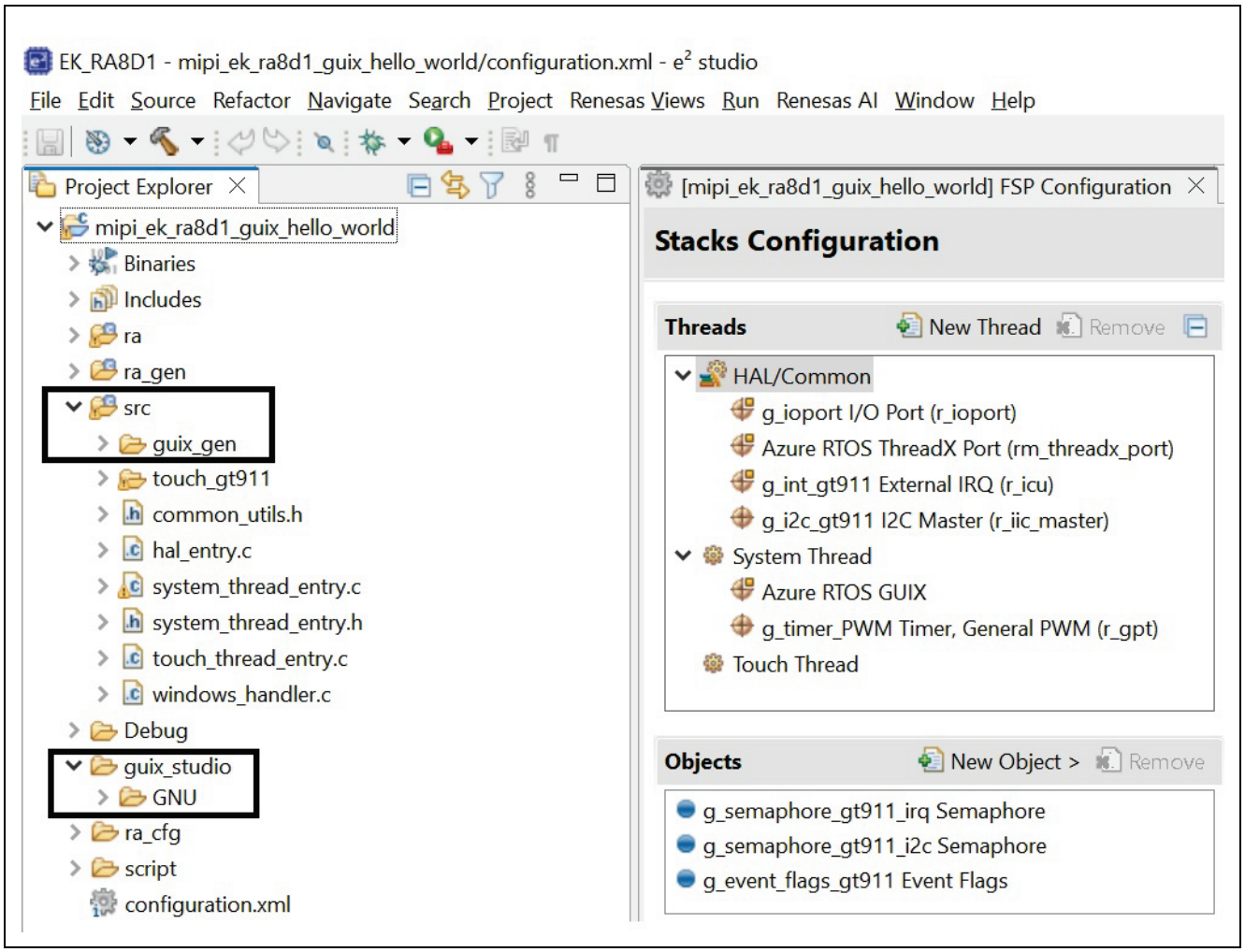


Figure 39. After Created All the Folders and it looks like above

5. Using Azure RTOS GUIX Studio create GUI Windows

1. Open Azure RTOS GUIX Studio v6.3.0.1 or greater.

Figure 40. GUIX Studio Icon

2. The following sub-steps will walk you through creating a new project named Hello World.
   A. Select **Project**. From the drop-down list, select **New Project**.
   B. Project Name: **Hello_World**.
   C. Project Path: Browse to the location of the folder you created in e² studio, the **mipi_ek_ra8d1_guix_hello_world\ guix_studio\GNU** as shown in the image below.
   D. Click the **OK** button and then the **Save** button to confirm your selections.

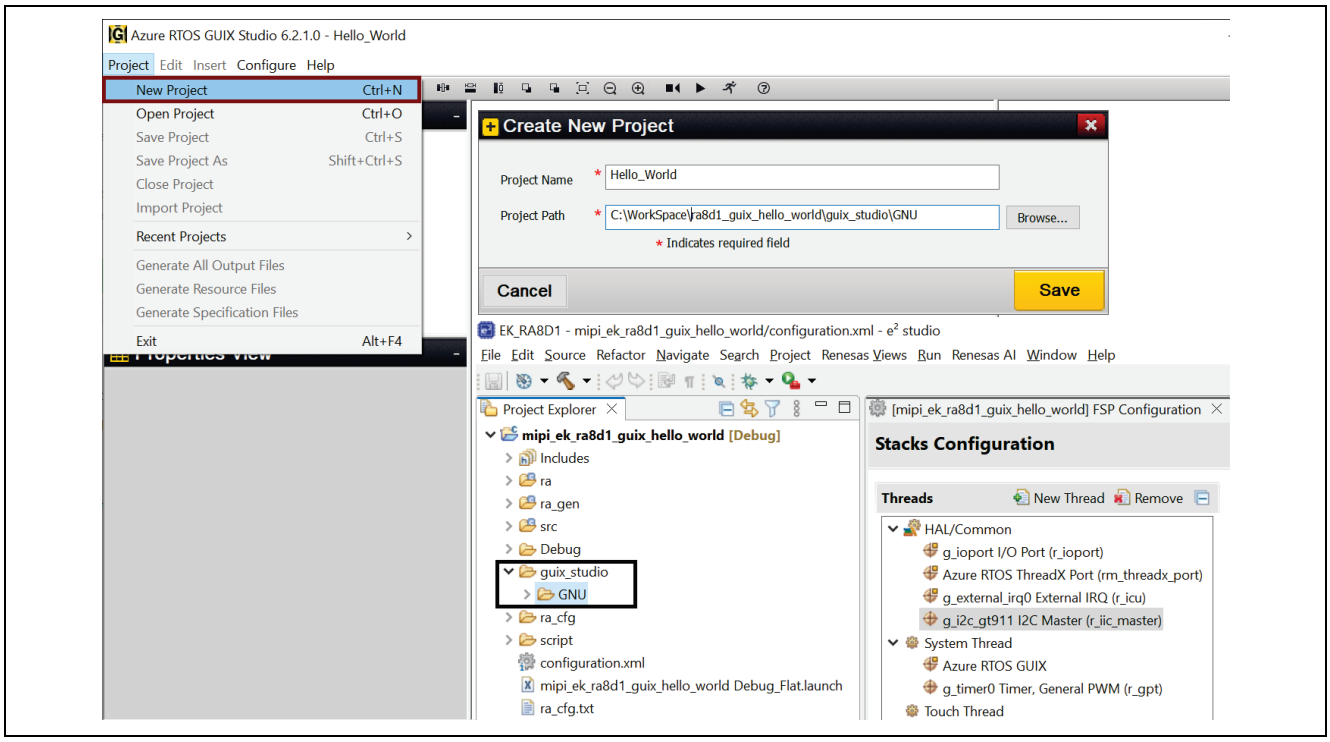


Figure 41. GUIX Studio Creates Hello World Project

3. A new **Configure Project** window will pop up and you need to set all the options as shown in **Figure 42**. Take care to also set the Advanced Settings. Finally, when your project settings match those reflected in the following image, click **Save**.

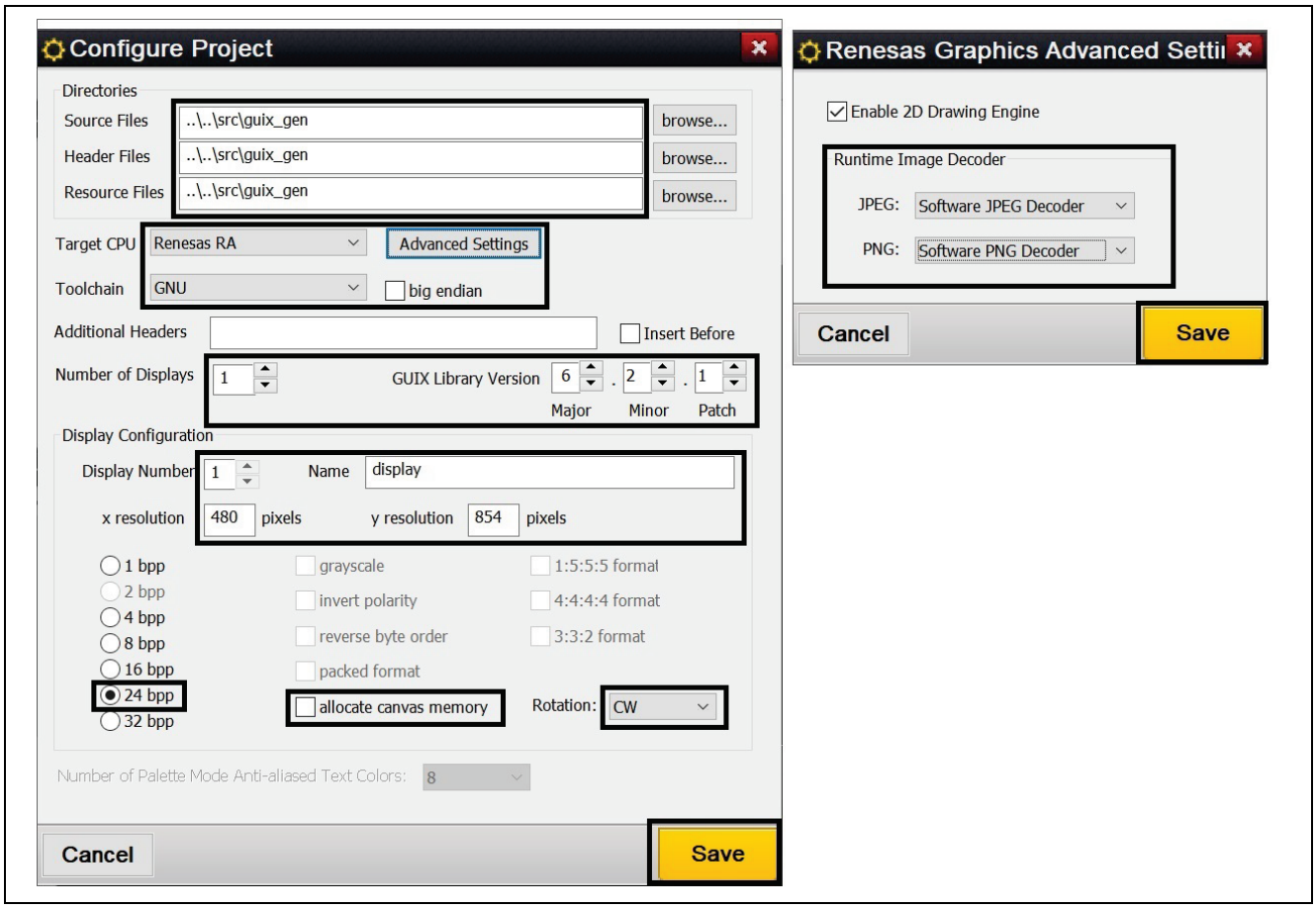


Figure 42. Configuration New Project “Hello World” with Advanced Settings

4. The default view of the New Project “Hello World” will look like the following image:

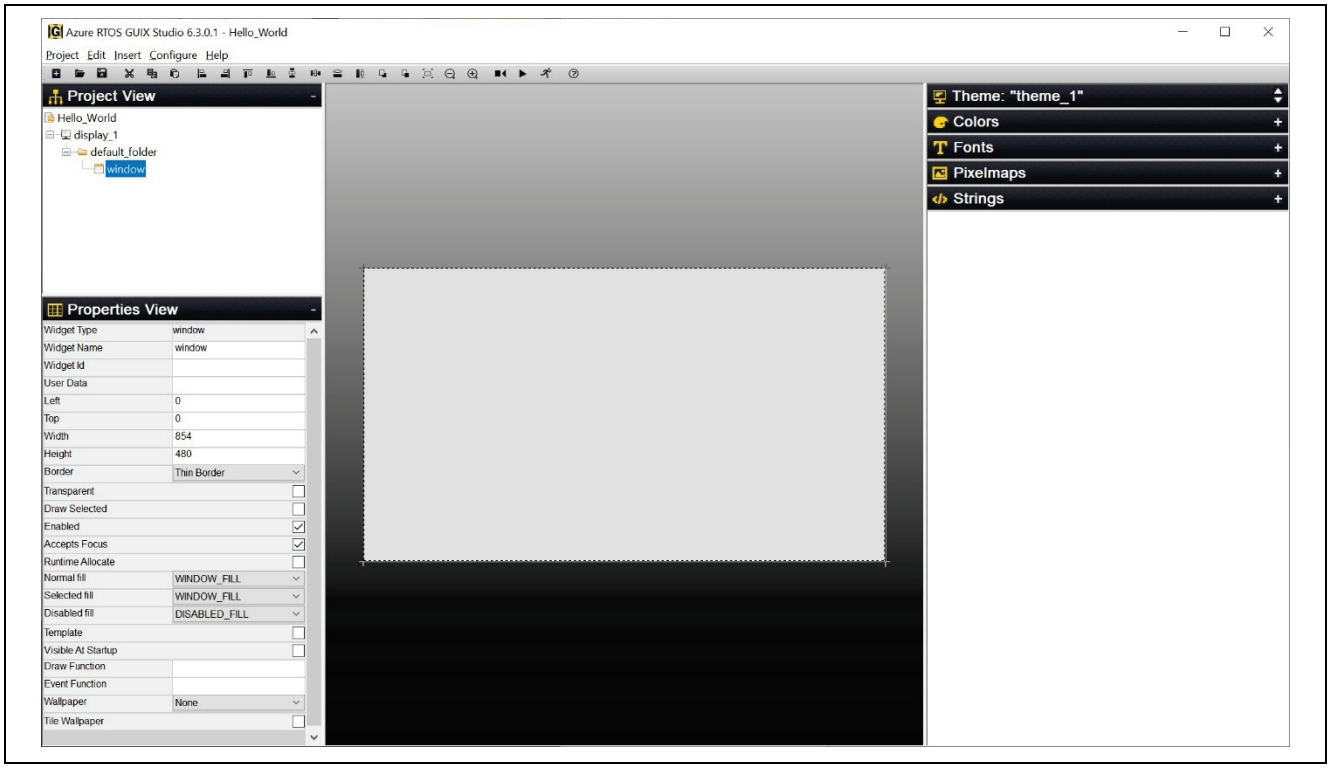


Figure 43. After Settings New Project “Hello World”

5. Select **Window1** in the **Project View,** and in the **Properties View** make sure the settings for Window1 match the following image.

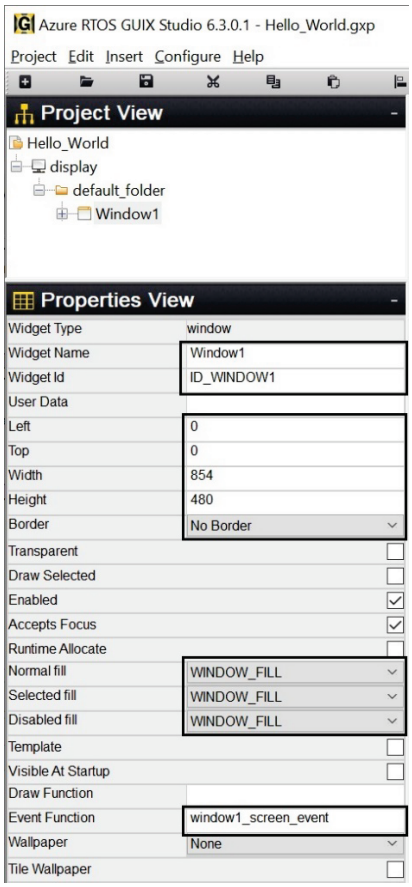


Figure 44. Properties View of Window1

6. To add String ID, click on **Strings**. Follow the images below.

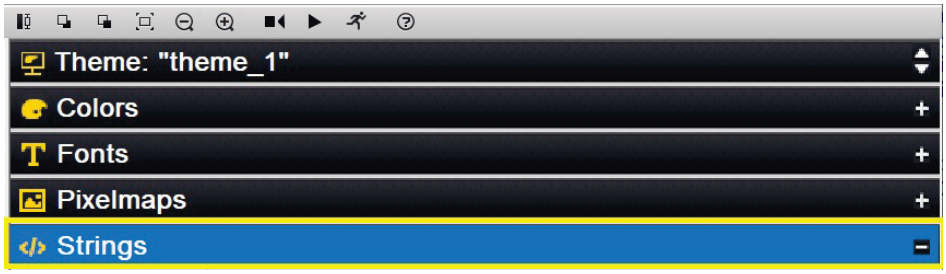


Figure 45. Strings

7. From the **Strings** dropdown, click **+ Add New String**.

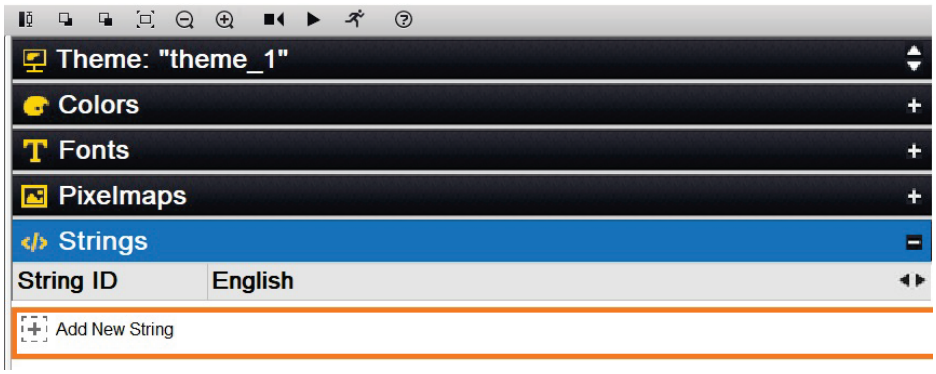


Figure 46. Add New String

8. New **String Table Editor** window will pop up. Click the **Add String** button.

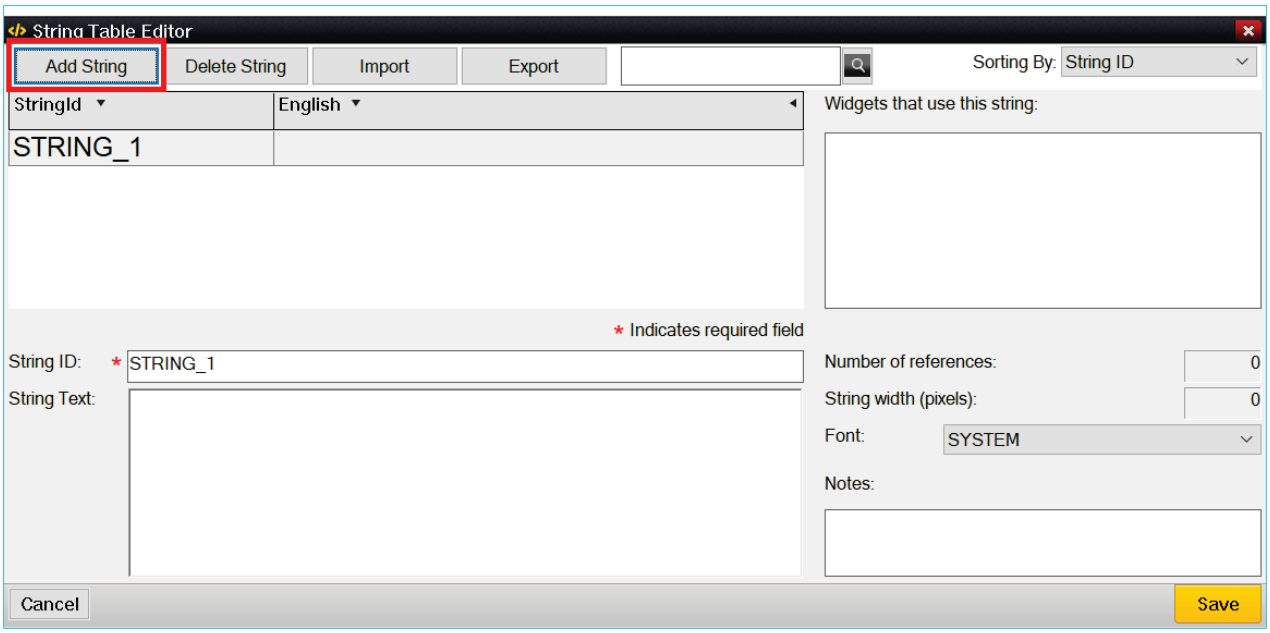


Figure 47. String Table Editor

9. Edit **String ID** and **String Text**.

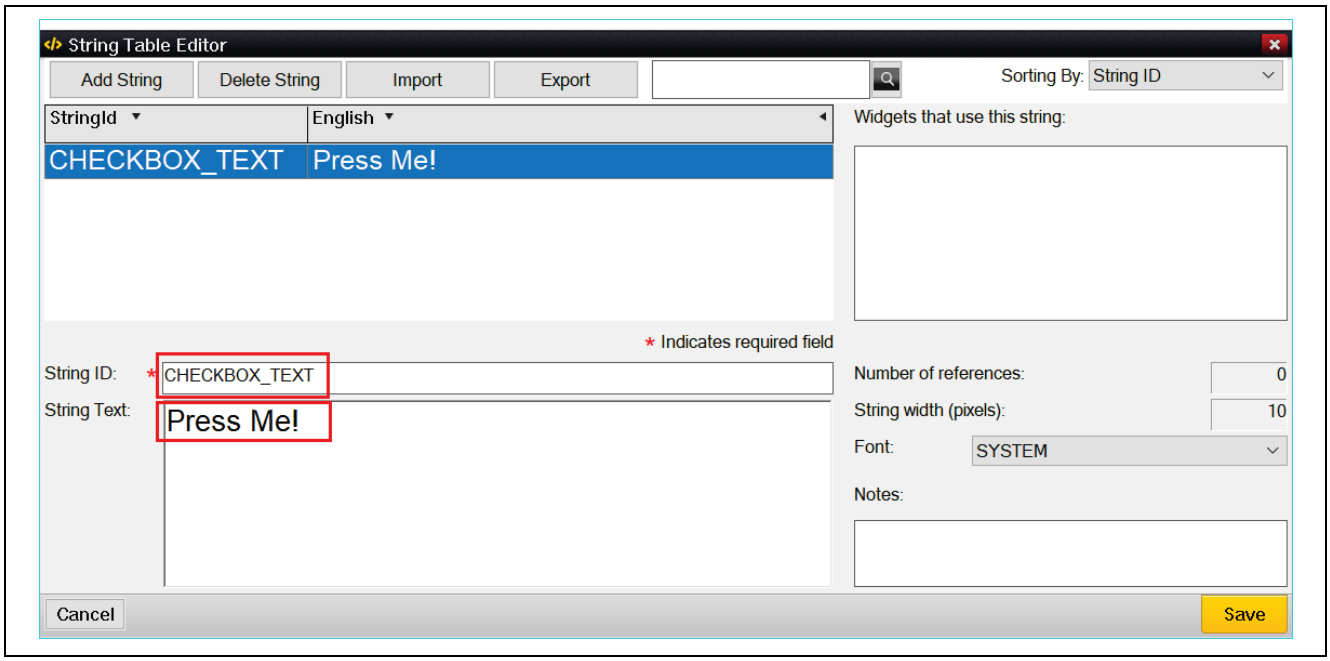


Figure 48. Edit String ID and String Text

10. Continue to click **Add String**, and edit each new entry's corresponding **String ID** and **String Text** until the table appears like **Figure 49**. Then click the **Save** button.

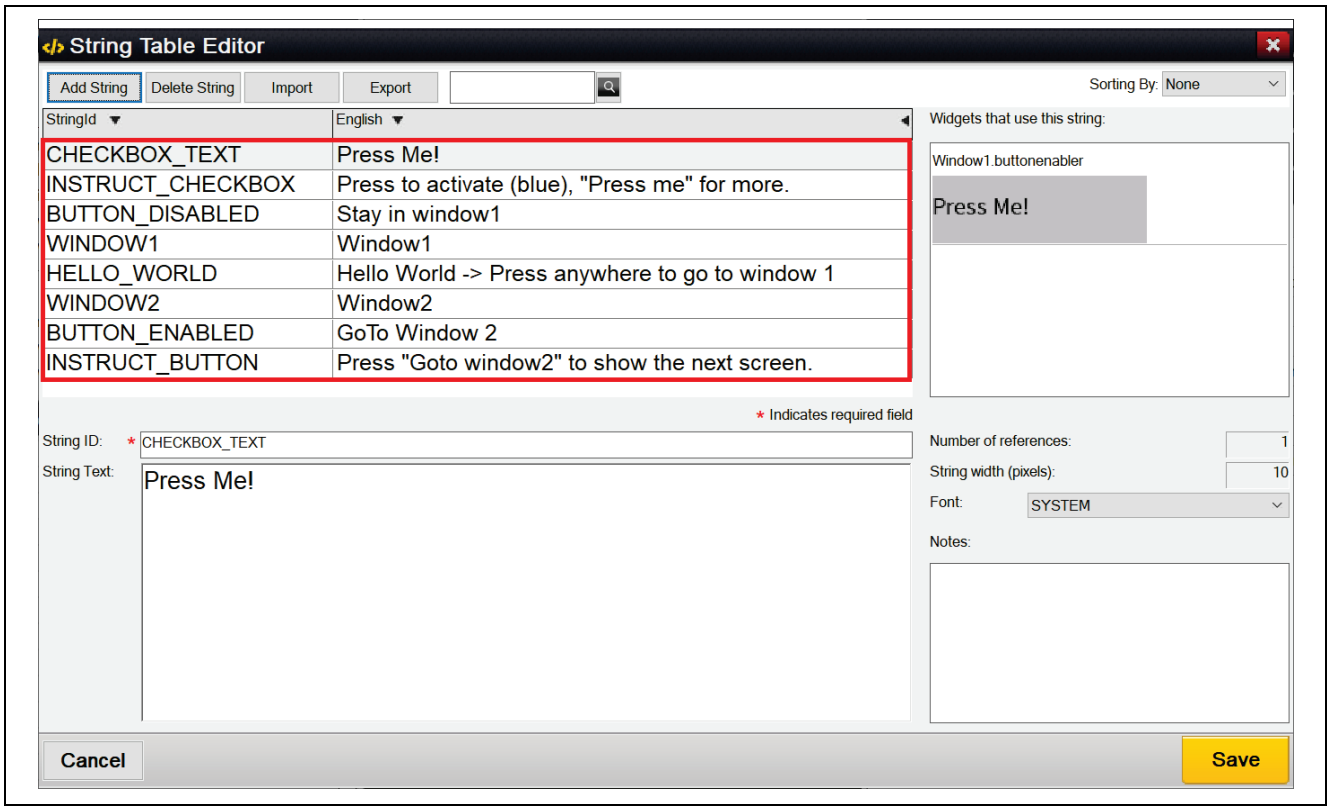


Figure 49. All Strings

11. Right-click on **Window1** in the **Project View** to insert a **Text Button** and follow **Figure 50**.

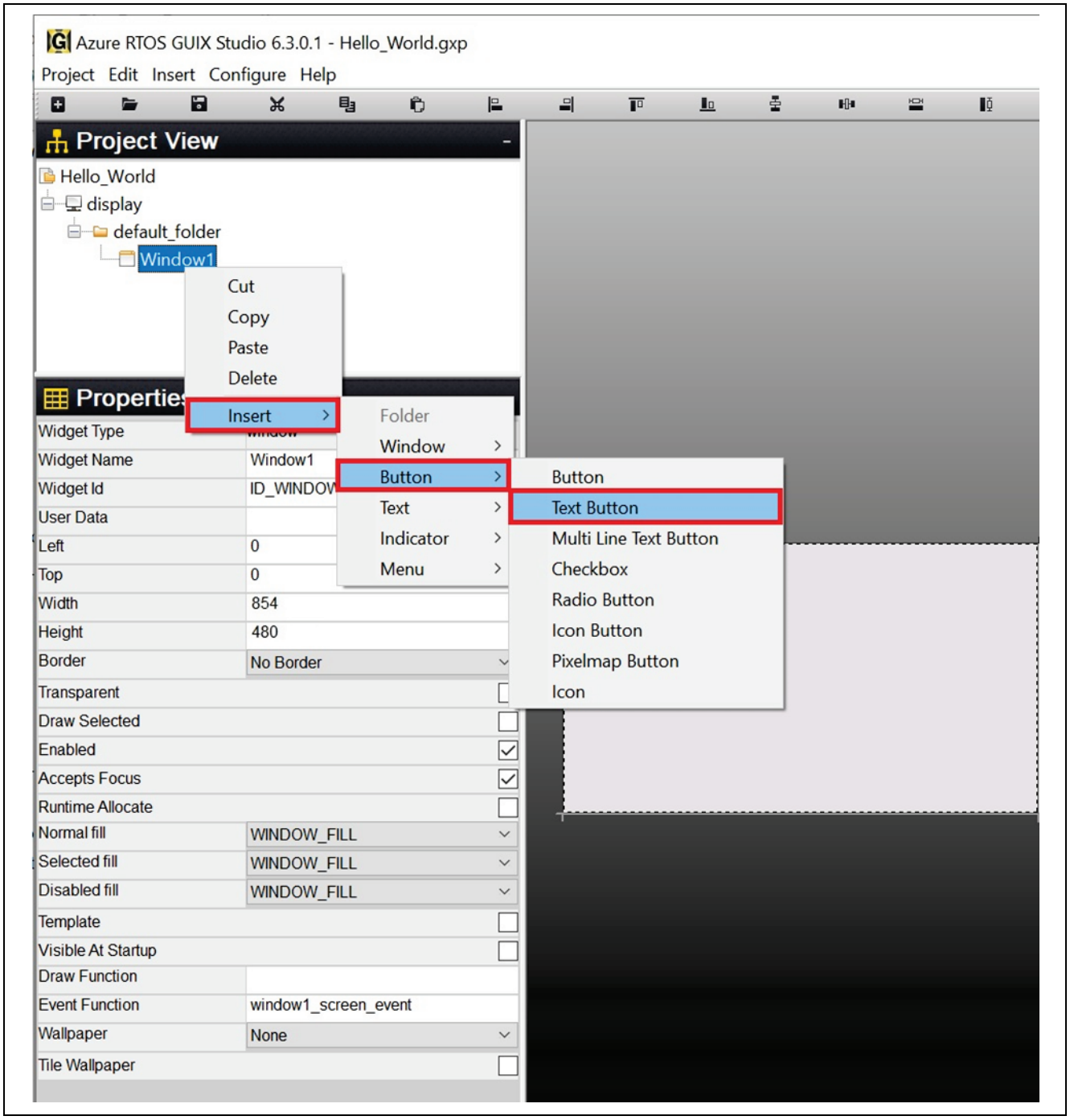


Figure 50. Insert Text Button

12. Set **Properties** View of "**text_button**".

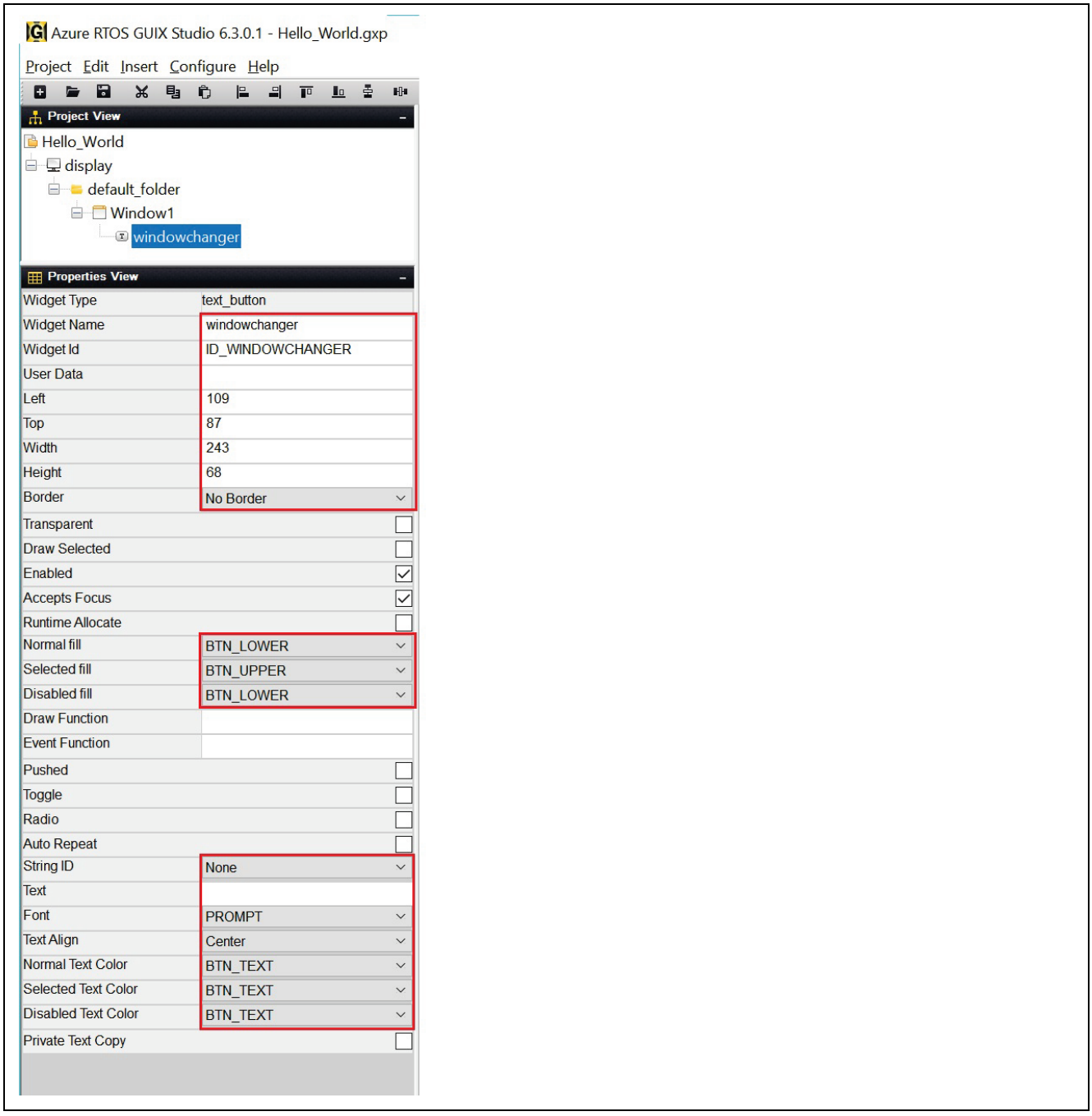


Figure 51. Properties View of text_button

13. Right-click **windowchanger** to insert a **Prompt** and follow **Figure 52**.

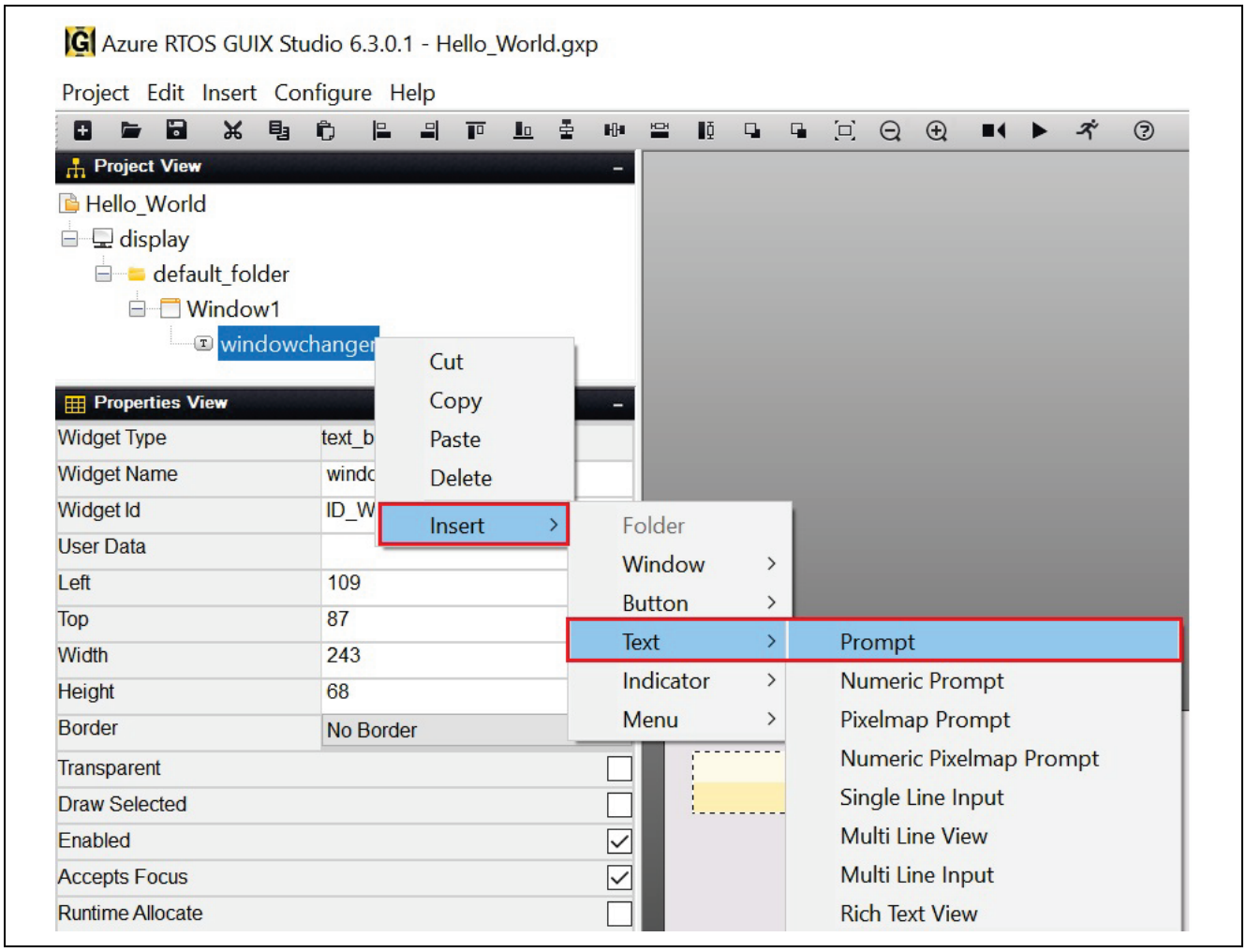


Figure 52. Insert Prompt

14. Set **Properties View** of **prompt**.

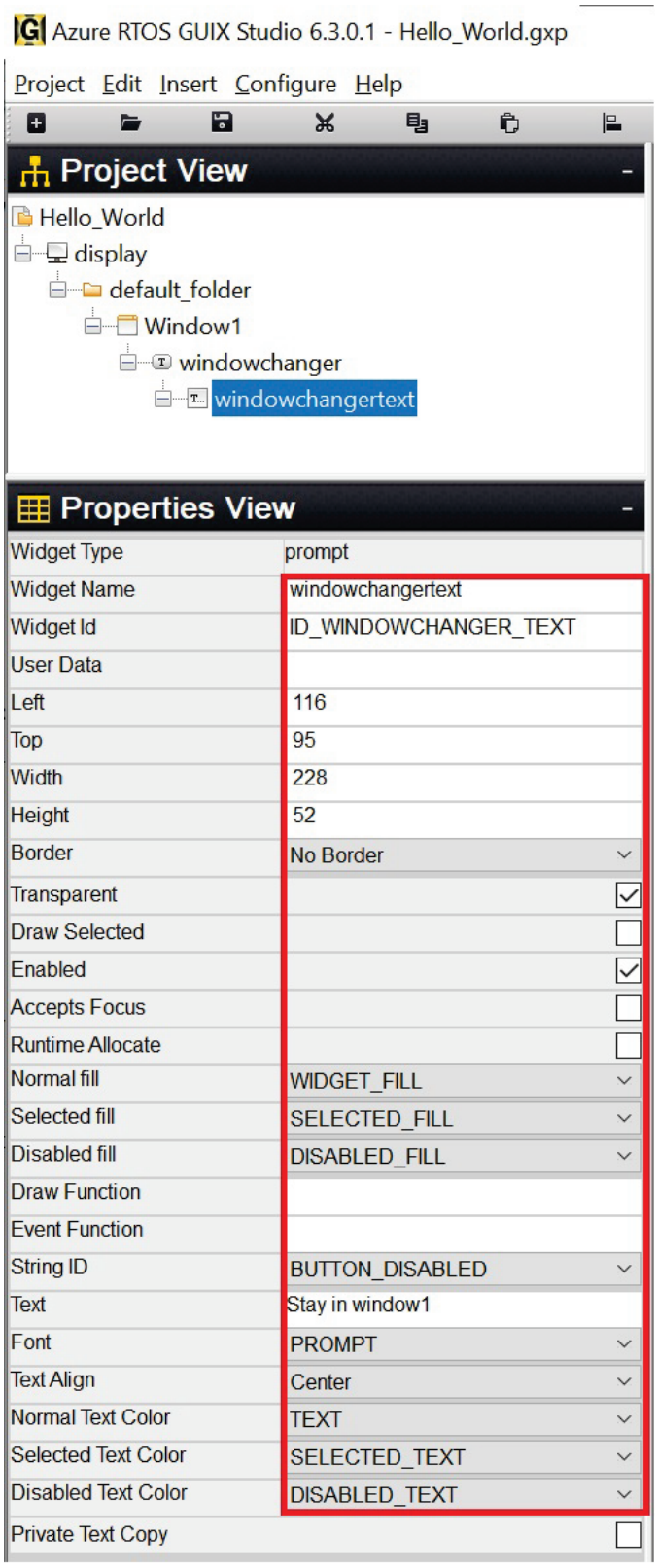


Figure 53. Properties View of Prompt

15. Insert new **Button**. Right-click on **windowchangertext** and follow **Figure 52**.

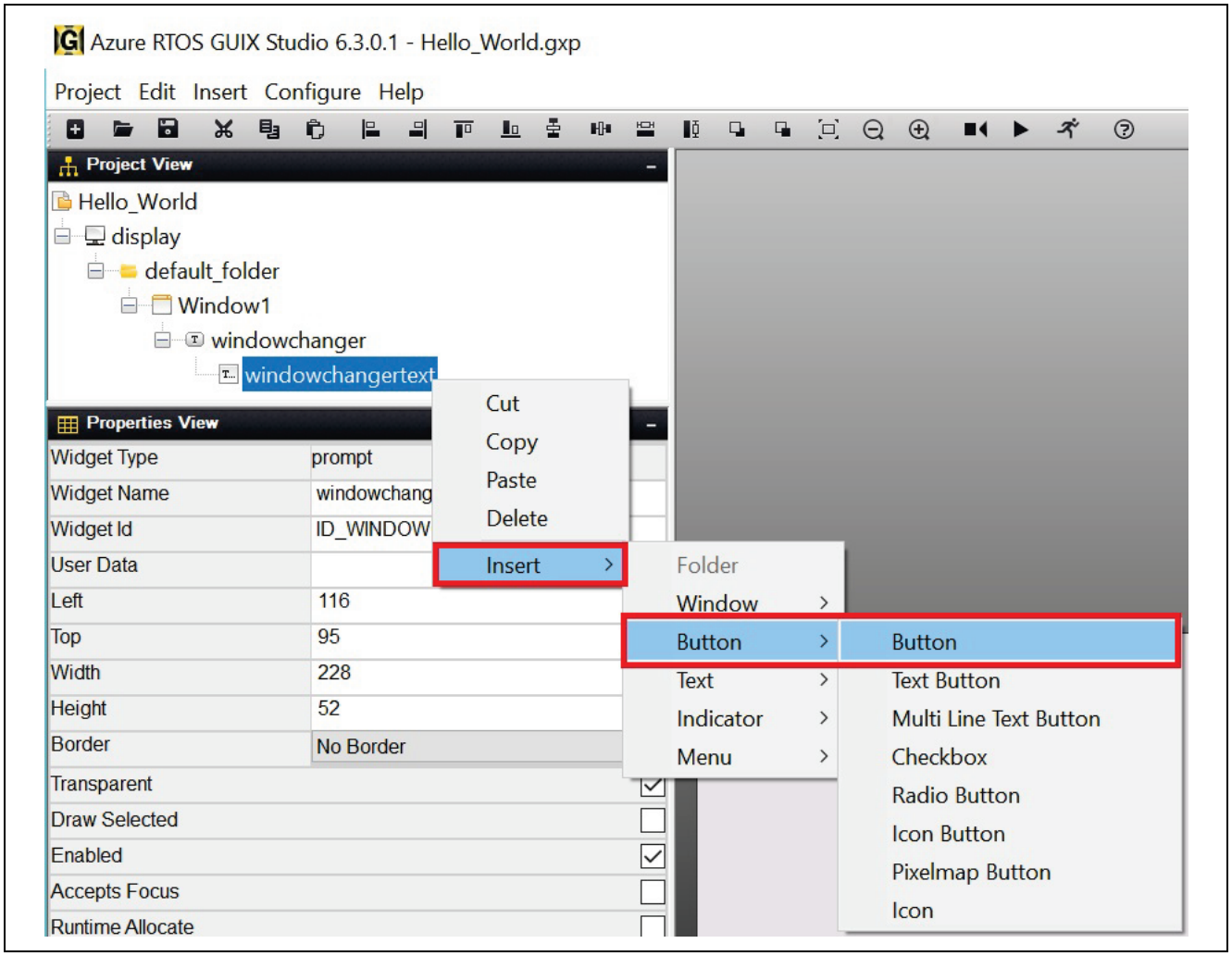


Figure 54. Insert Button

16. Set the **Properties view** of the **button**.

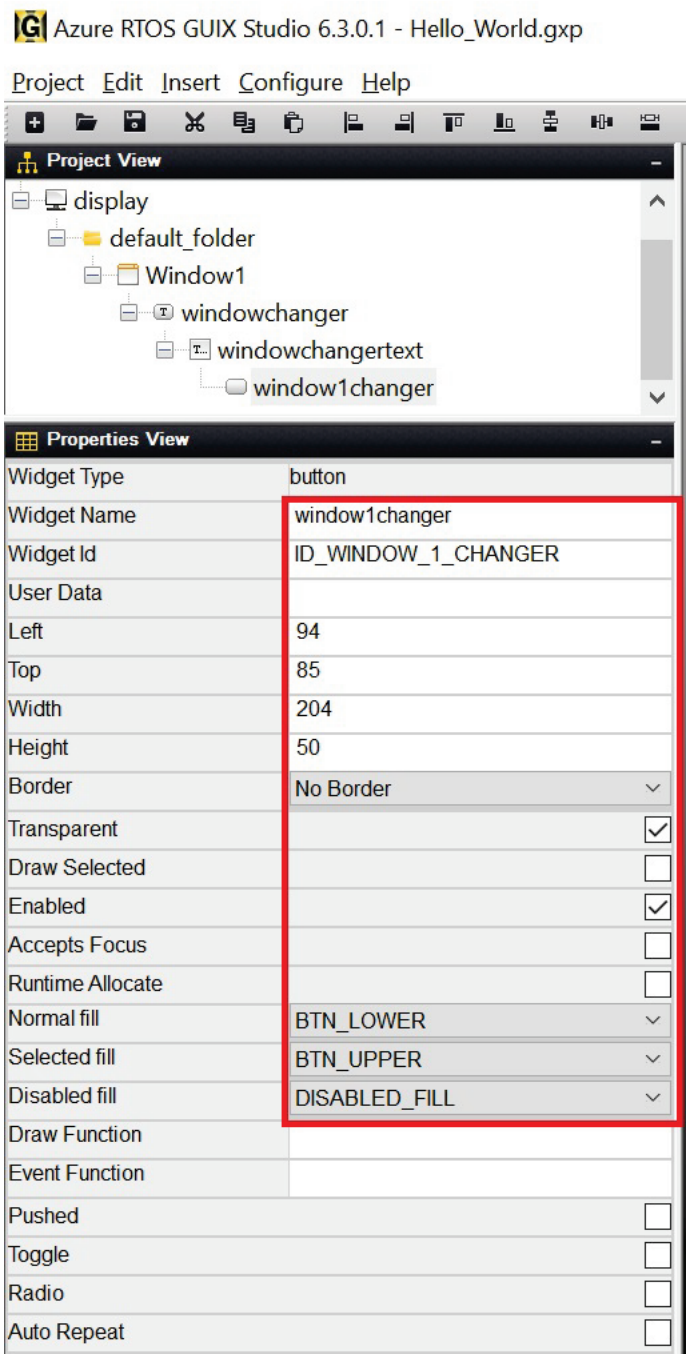


Figure 55. Properties view of the Button

17. Insert **Prompt**. Right click on **Window1** and follow **Figure 56**. Insert two times to get two prompts. **Prompt** and **Prompt1** will show up once you finish.

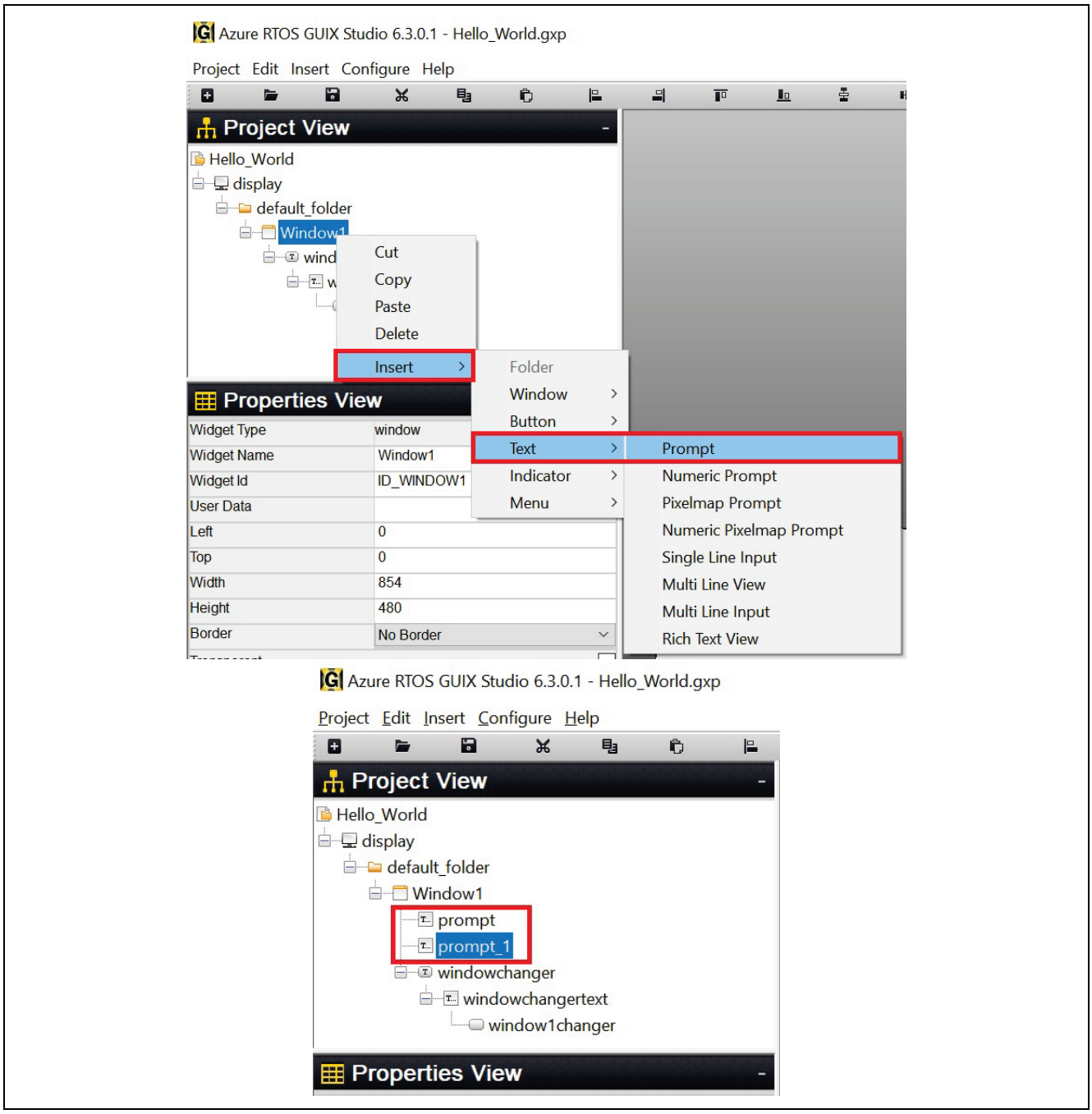


Figure 57. Insert Prompts

18. Set the **Properties View** of **Prompt**.

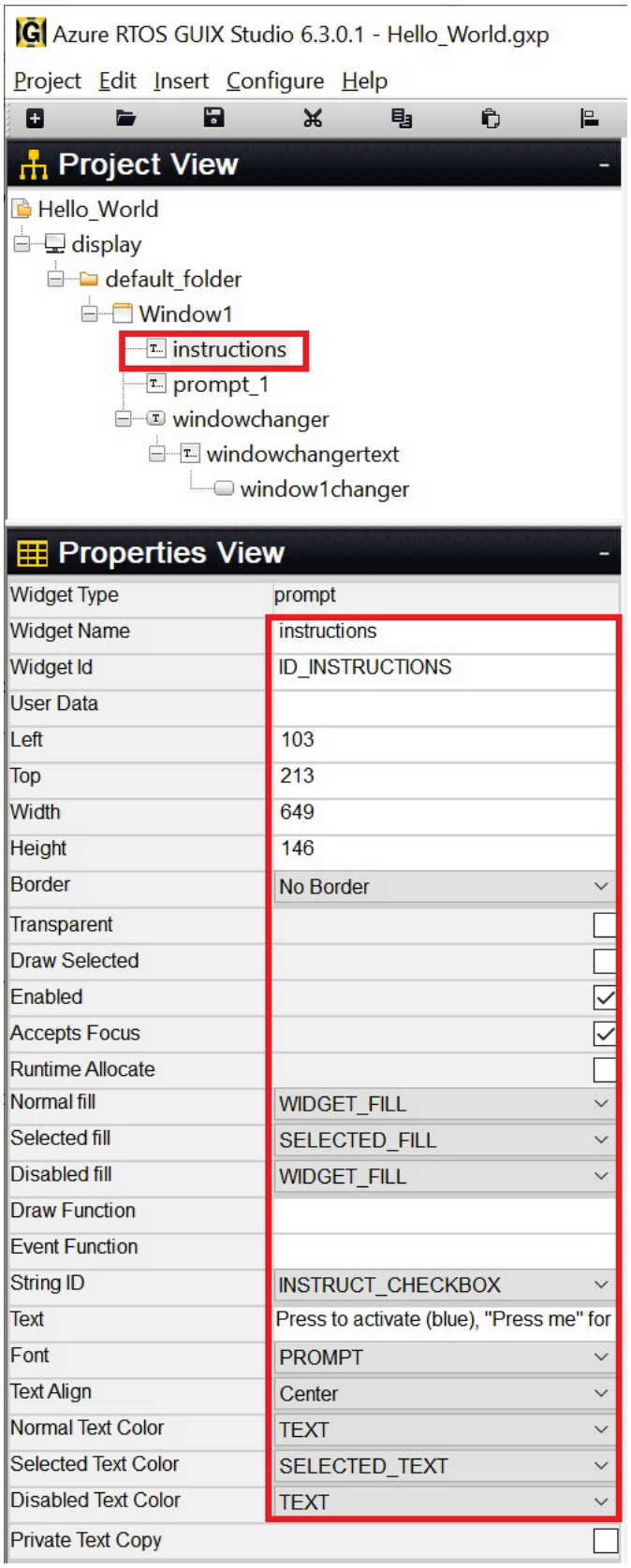


Figure 58. Properties View of Prompt

19. Set the **Properties View** of **Prompt1**.

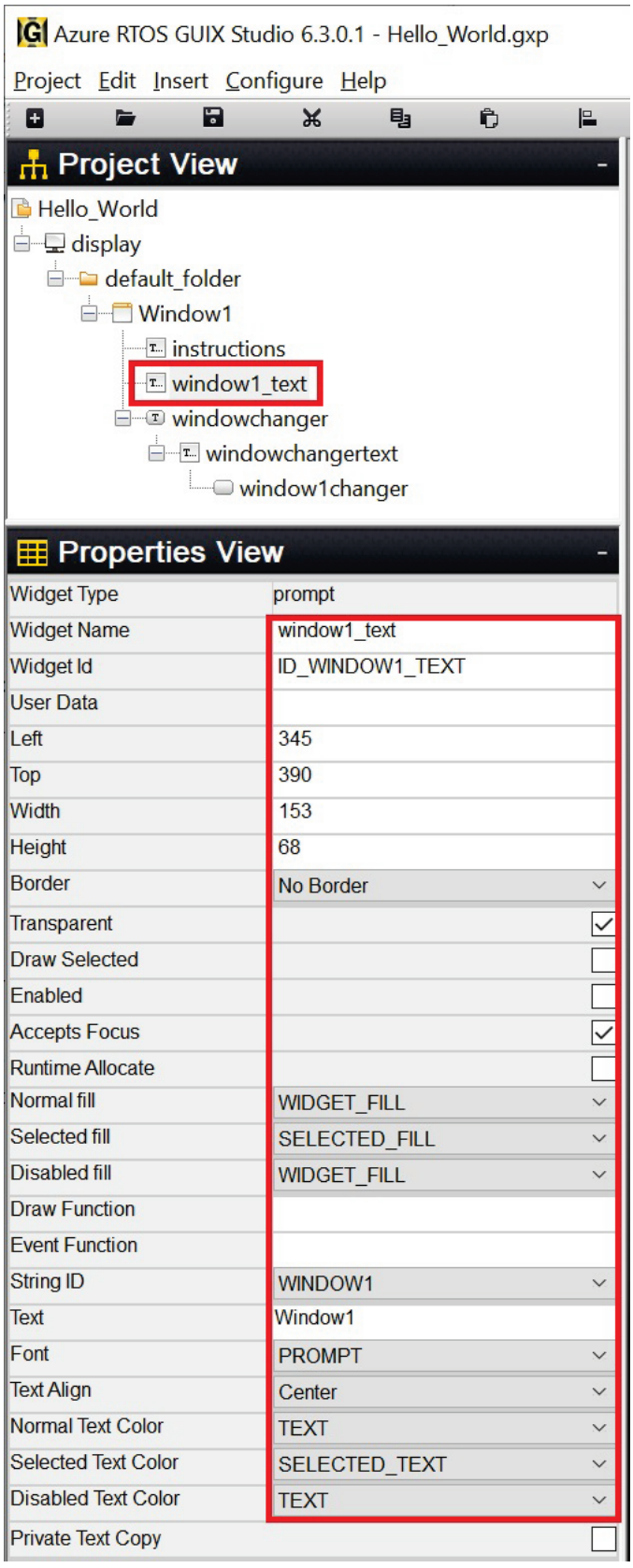


Figure 59. Properties View for Prompt1

20. Click **Insert > Button > Checkbox**. Right click on **Window1** and follow **Figure 59.**

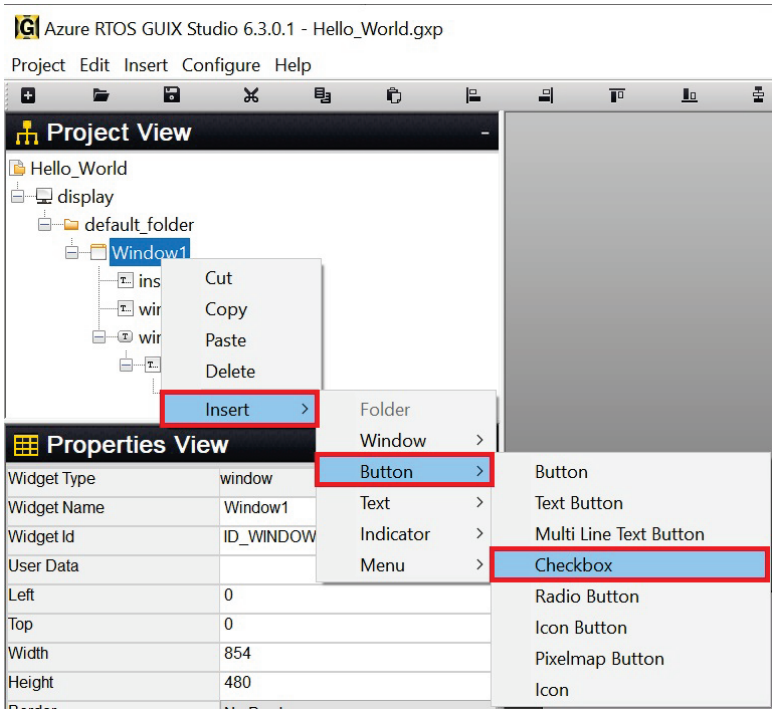


Figure 60. Insert Button Checkbox

21. Setting **Properties View** of **Button Checkbox**.

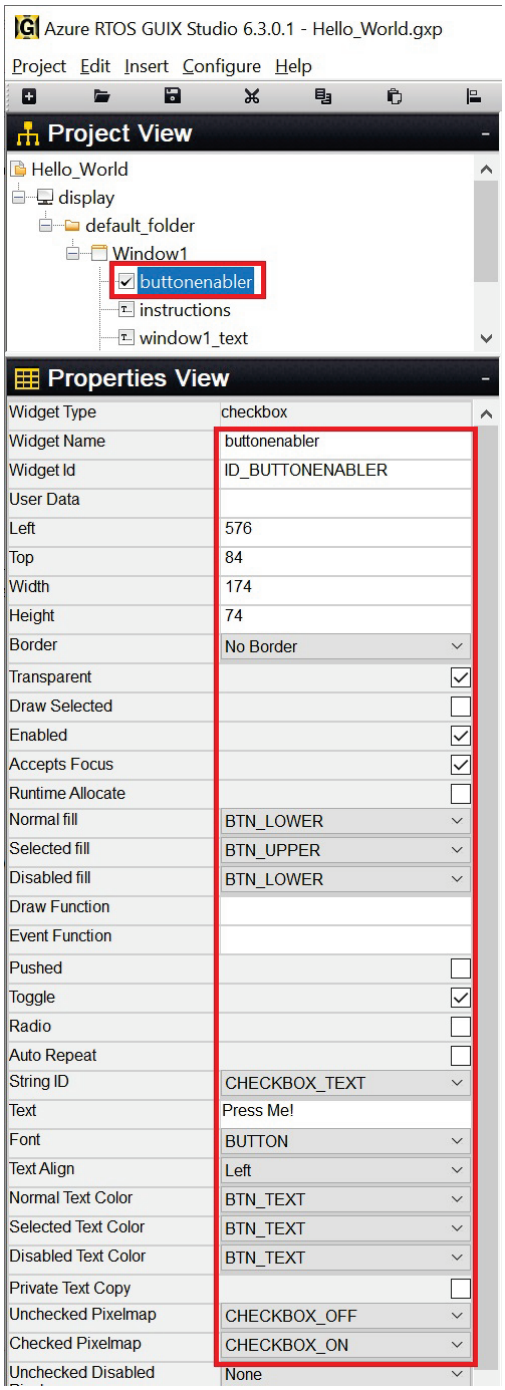


Figure 61. Setting Button Checkbox Properties

22. After you finished creating **Window1**, it should be like the image below.

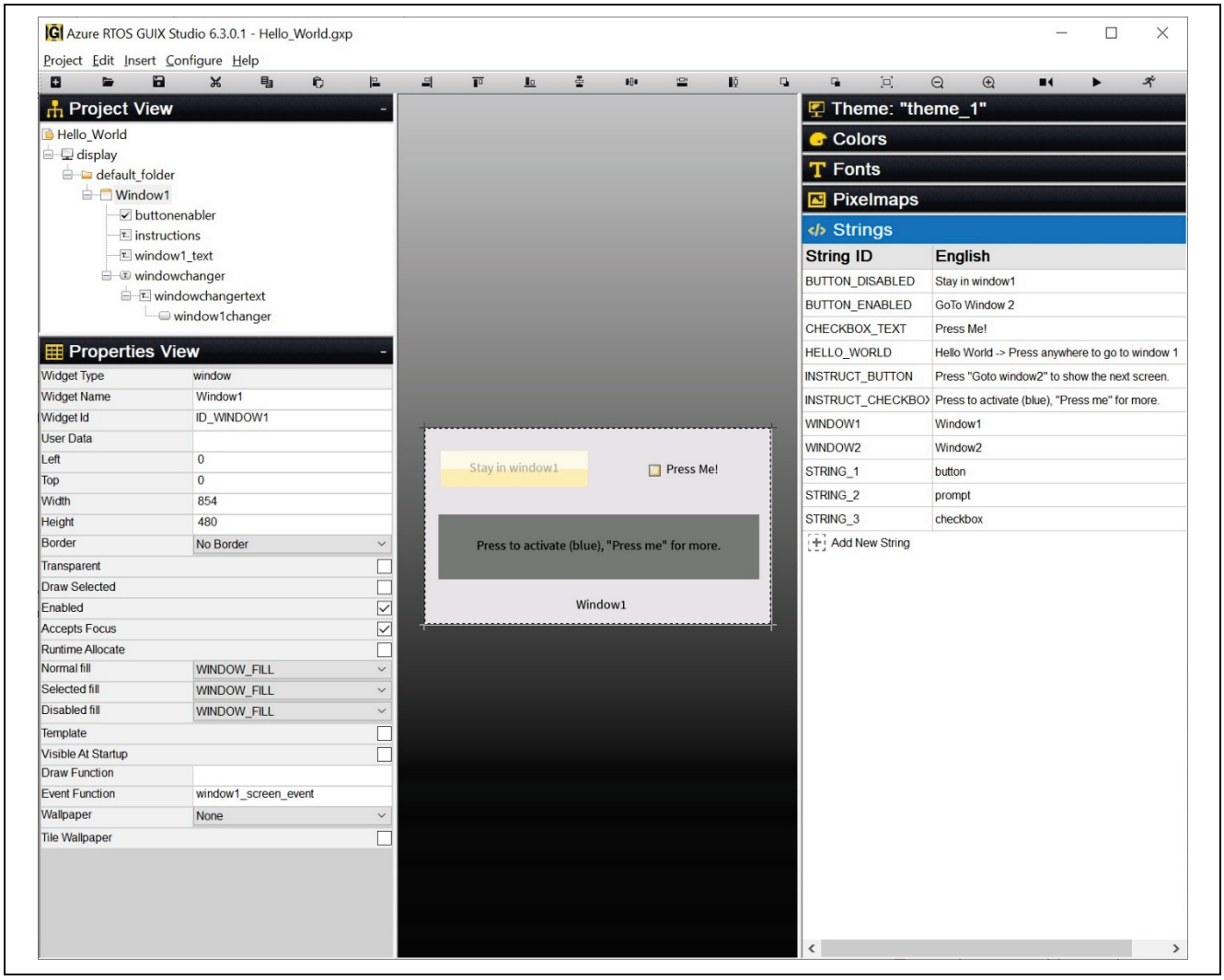


Figure 62. Window1 Created

23. Add another window to the GUIX Studio project**.** Right click on **default_folder** in the **Project View** and click **Insert** > **Window > Window** as shown in **Figure 63**.

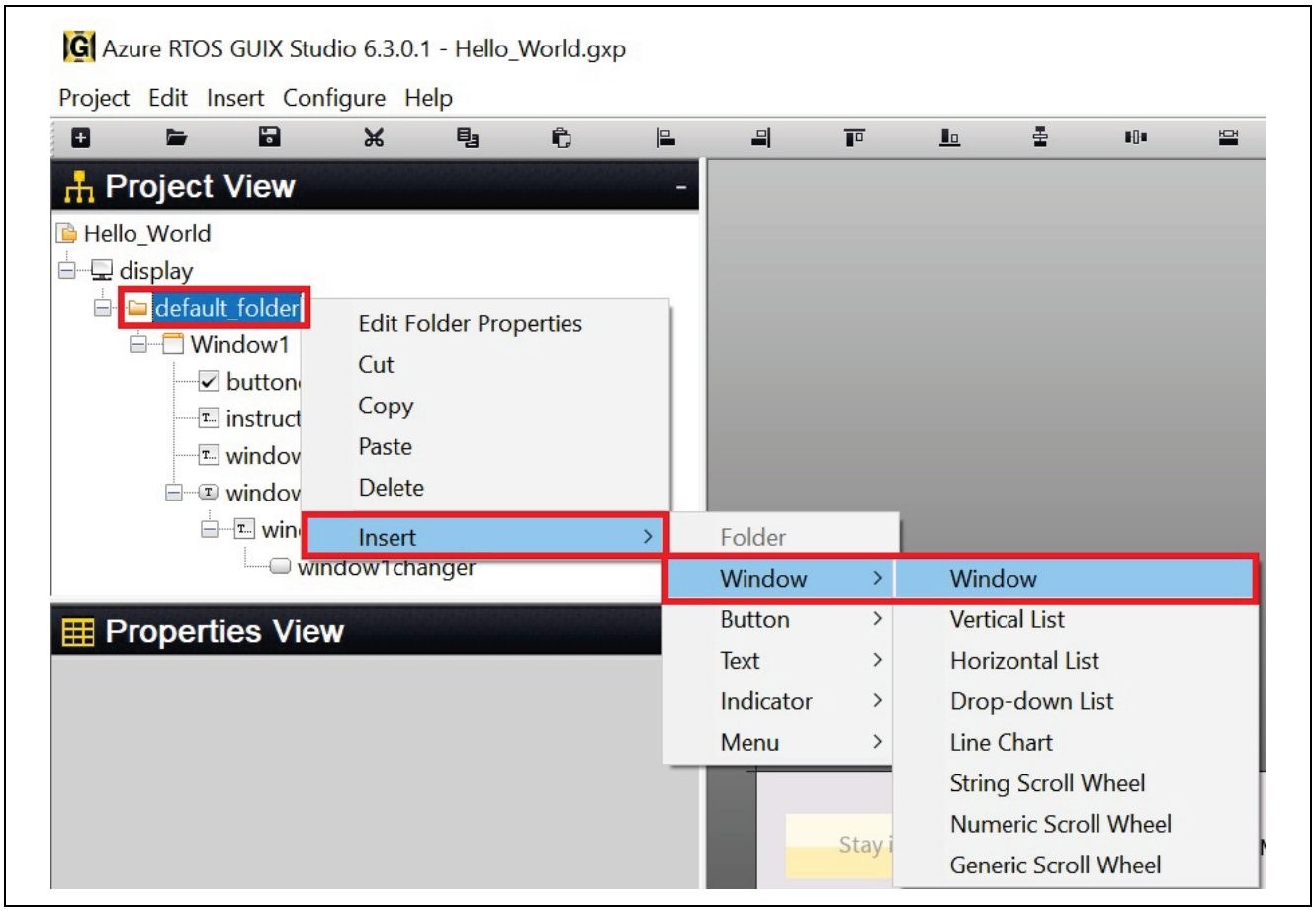


Figure 63. Insert Window2

24. Setting the **Properties View** of **Window2**.

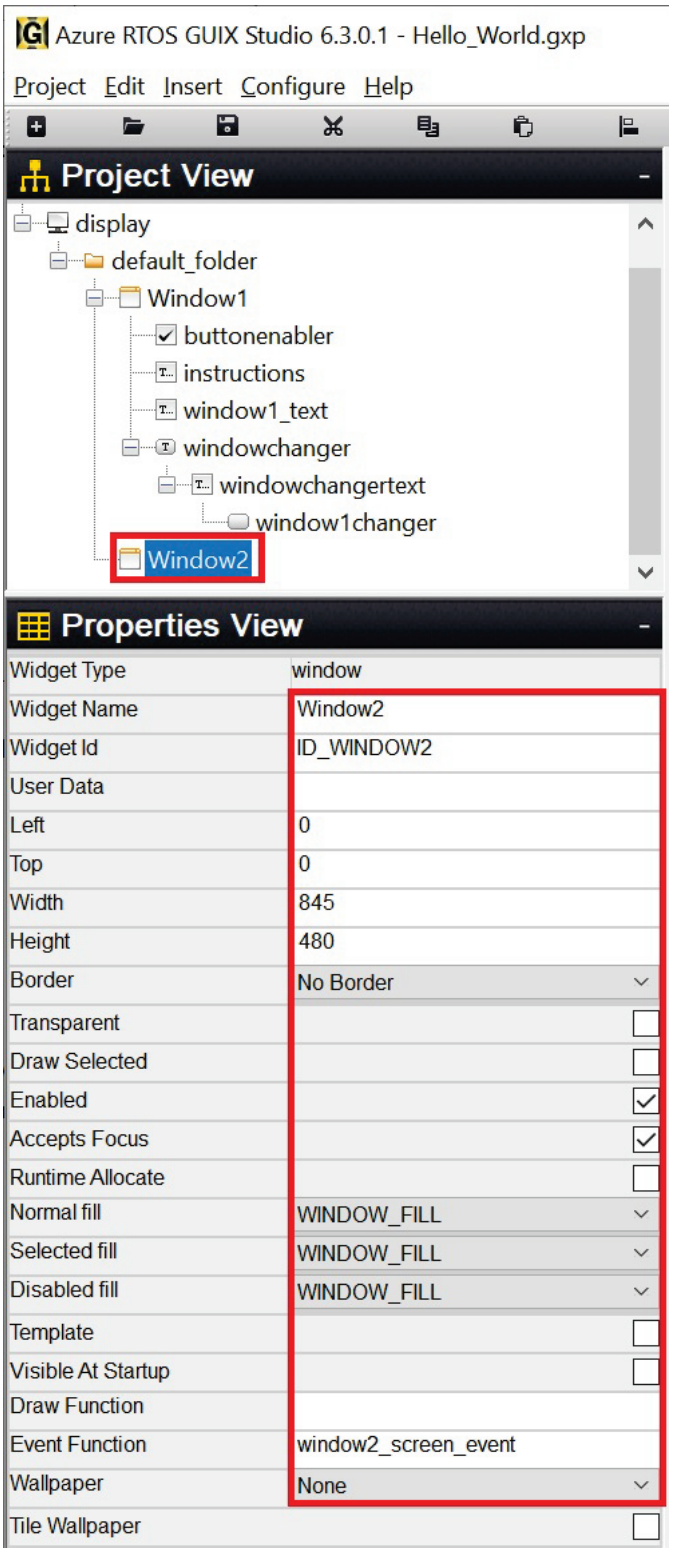


Figure 64. Setting Properties of Window2

25. Insert **Prompt** for Window2. Right click from Window2. Right click on Window2 in the **Project View**, then click **Insert > Text > Prompt** as shown in **Figure 65**.

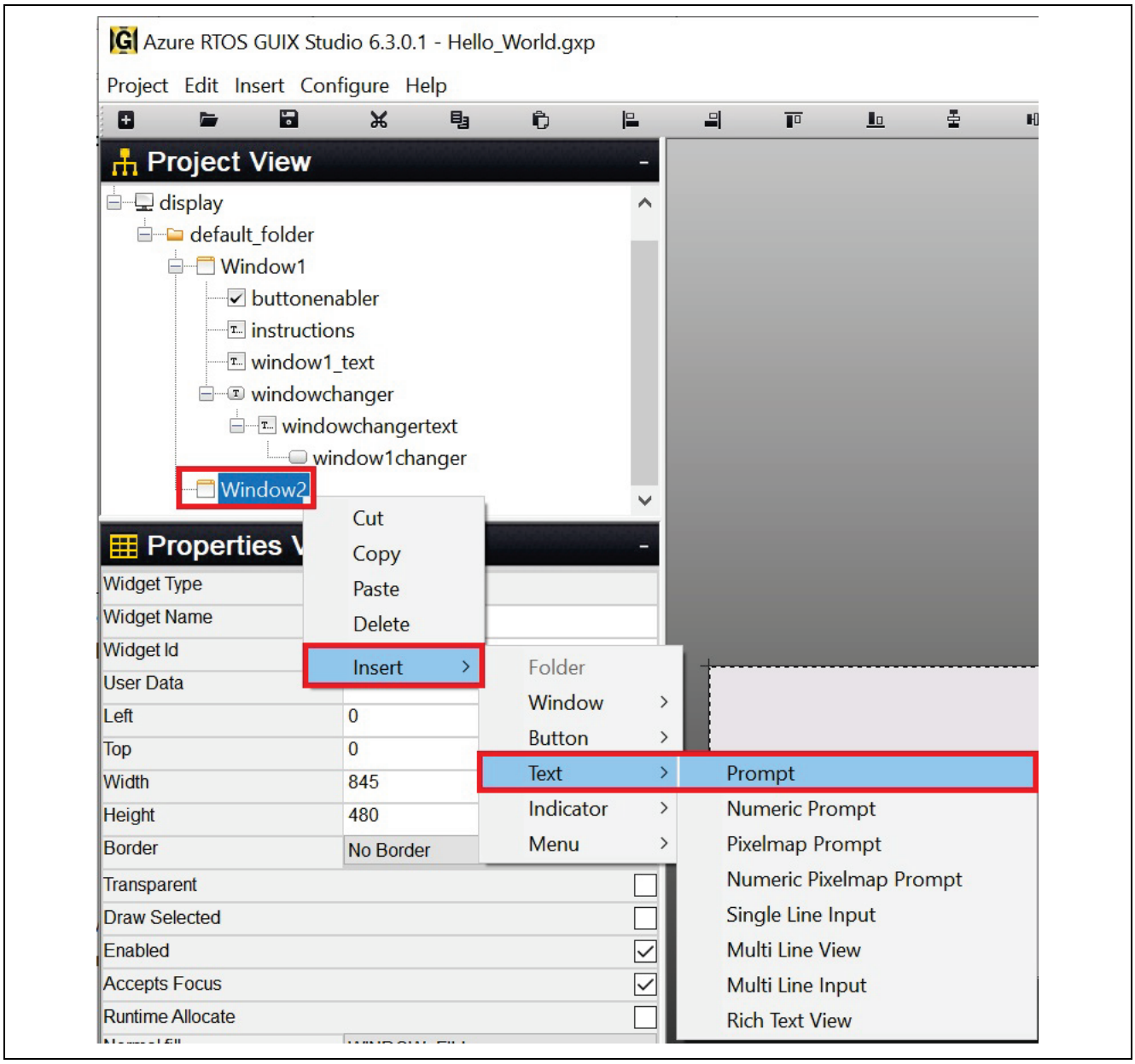


Figure 65. Insert Prompt for Window2

26. Setting **Properties View** of **Prompt**.

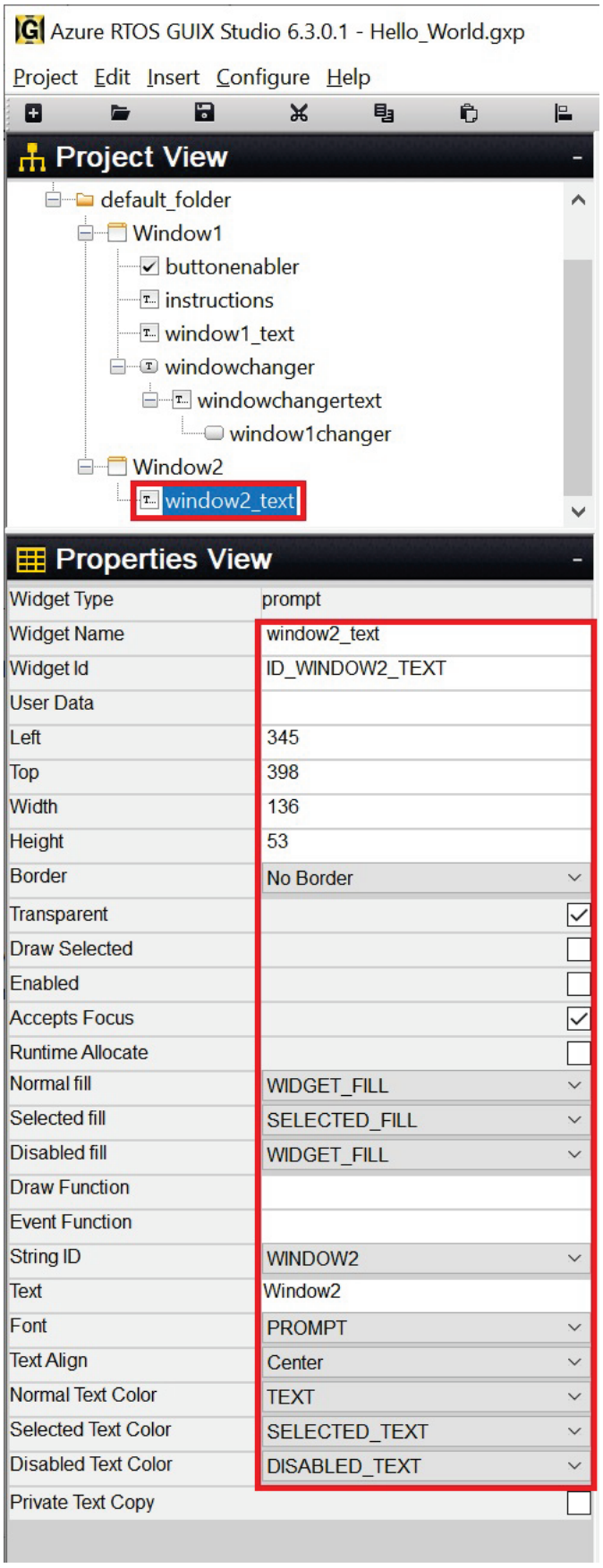


Figure 66. Setting Properties of window2_text

27. Insert **text_button** for window2. Right click on Window2 then click **Insert > Button > Text Button** following **Figure 67**.

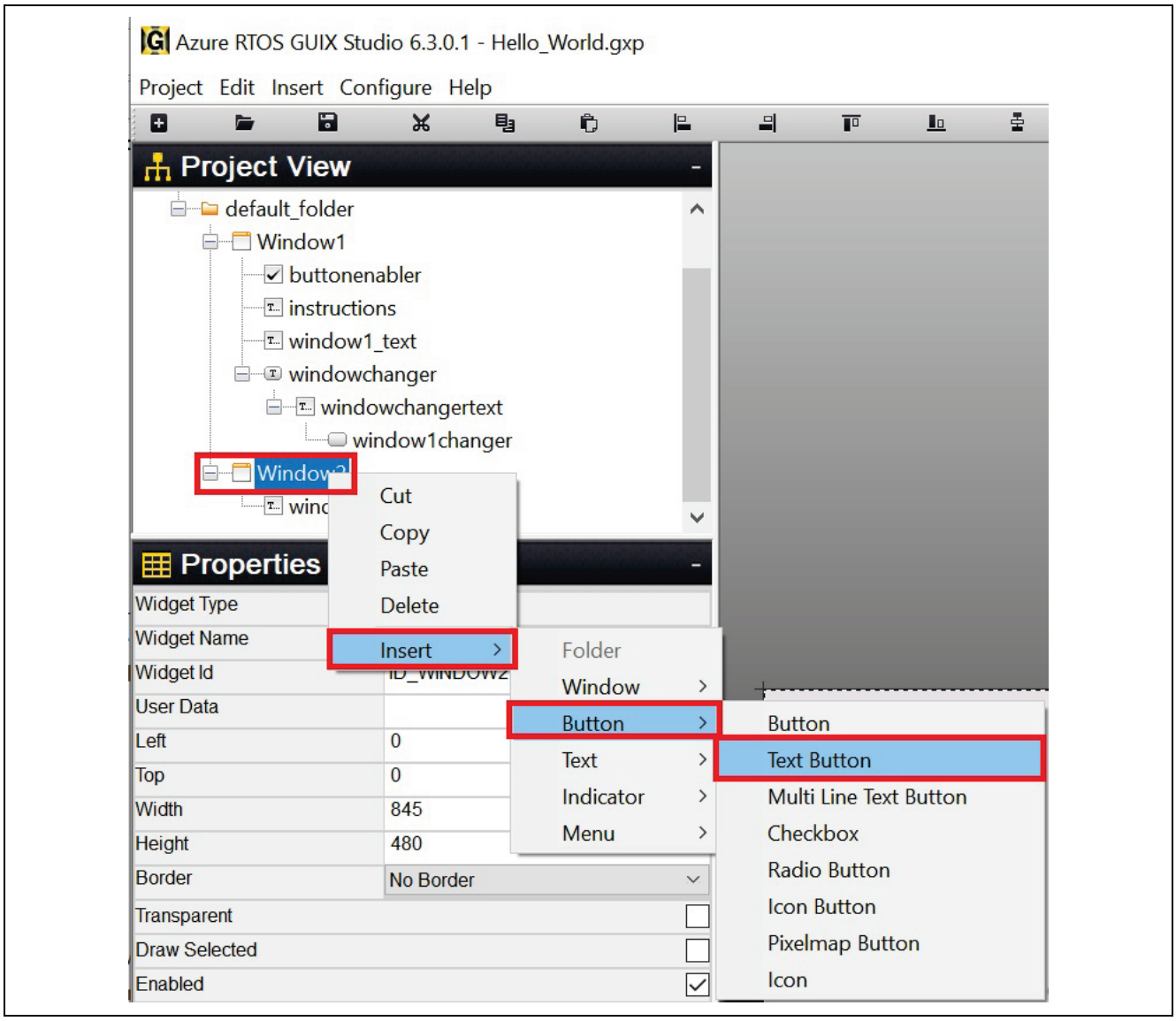


Figure 67. Insert text_button for Window2

28. Set the **Properties View** of **text_button** as shown in **Figure 68**.

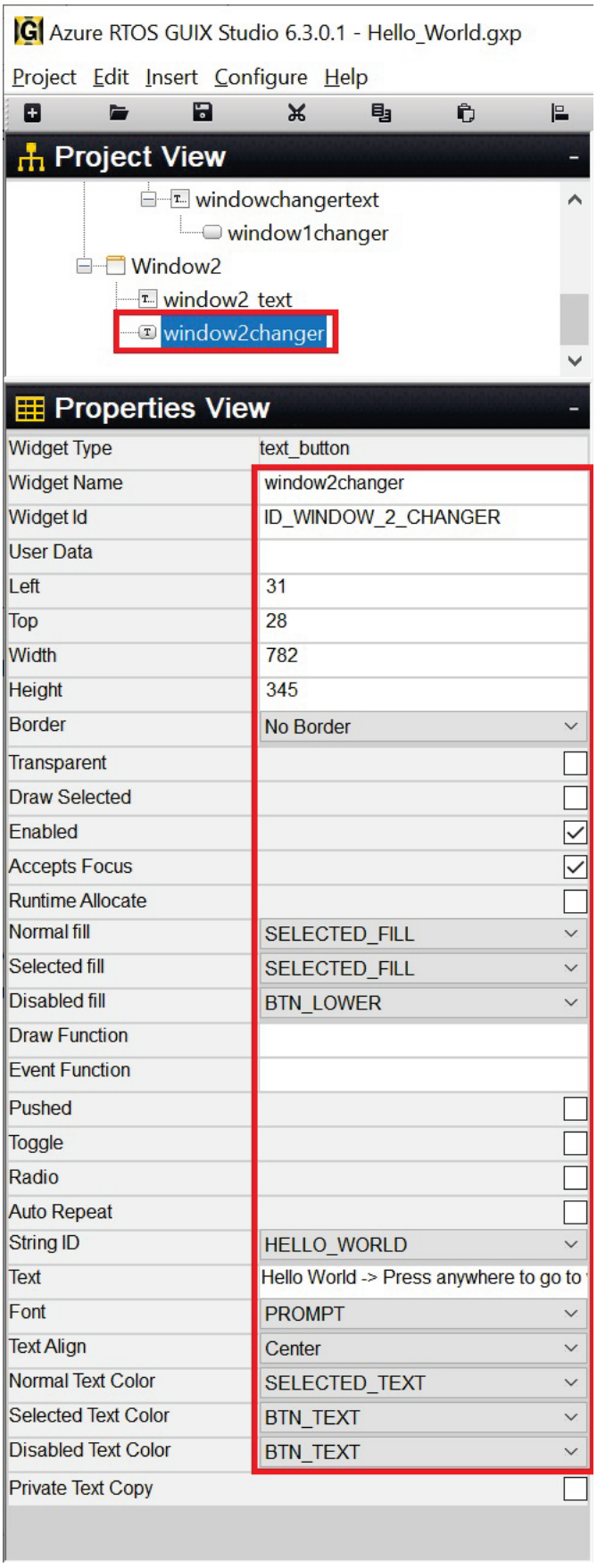


Figure 68. Setting Properties of text_button

29. After completing inserting and configuration, window2 looks like **Figure 69**.

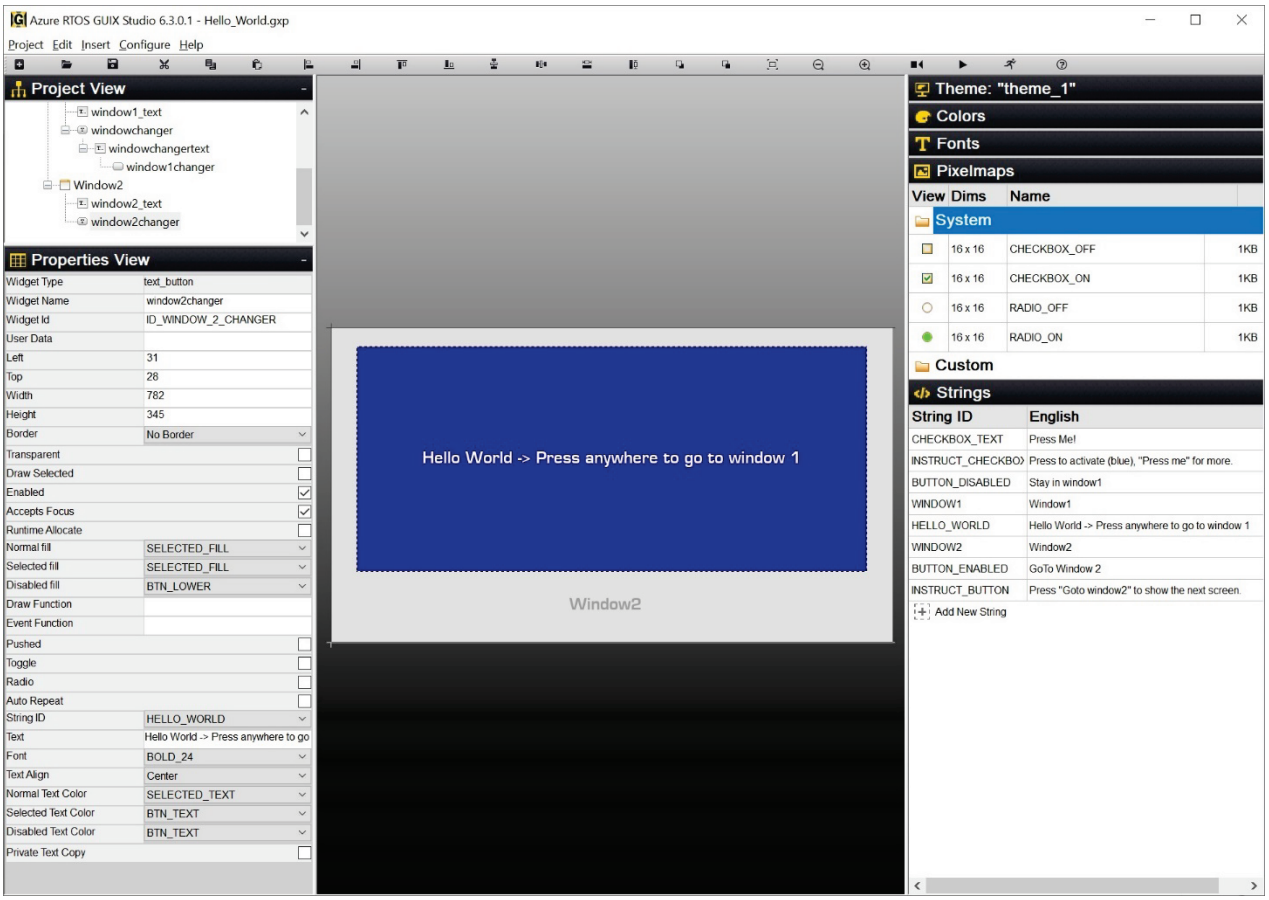


Figure 69. Window2

30. Click on drop-down list **Pixelmaps**, double-click on "**CHECKBOX_OFF**" and a new window will pop up. Uncheck **Compress Output** then click **Save**. Do the same for **CHECKBOX_ON**.

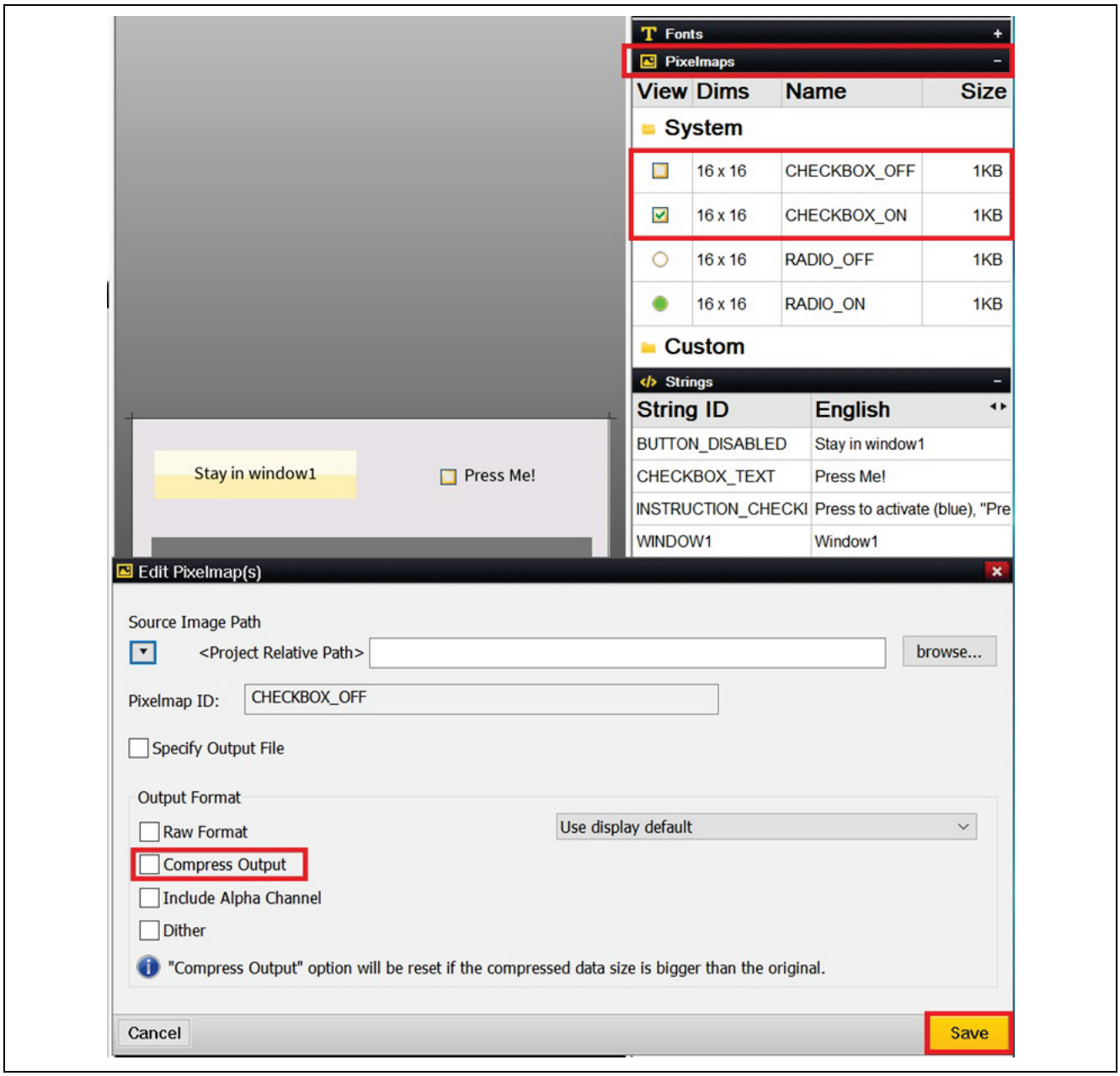


Figure 70. Set Up Pixelmap

31. Now you can click on the **Project** drop down list, **Save Project**, and **Generate All Output Files**. You completed the process of creating and exporting GUIX Hello World into the project.

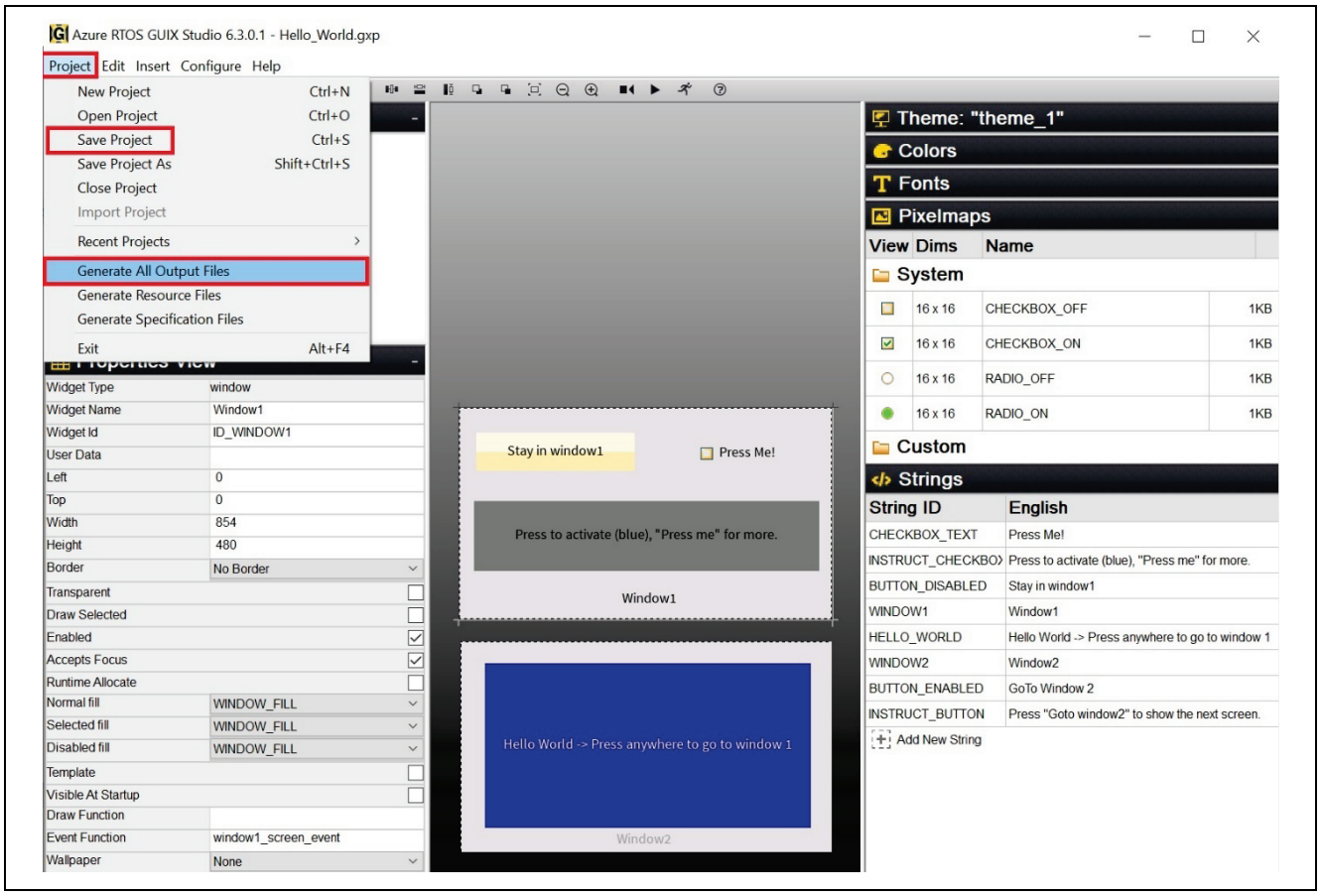


Figure 71. Save and Generate Project

32. Make sure the project is active and click to build the project. It may take a long time to finish building an Azure RTOS/GUIX project on your PC. Project **ra8d1_guix_hello_world** should be built with no errors or warnings.

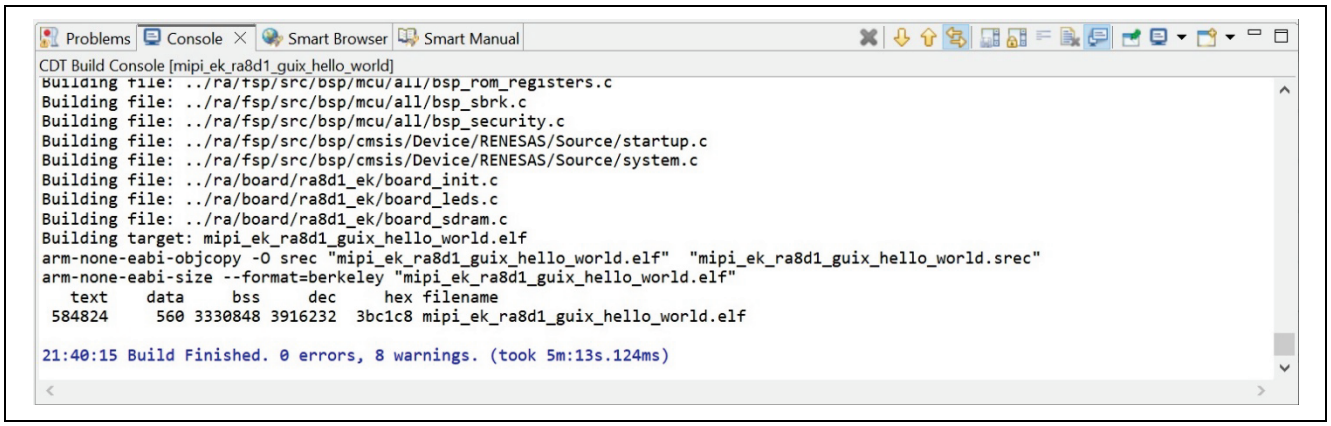


Figure 72. Built The Code

33. Using the Micro USB cable, connect to J10 on EK-RA8D1 board and the other end to your PC. Download and run the “**mipi_ek_ra8d1_guix_hello_world**” project.

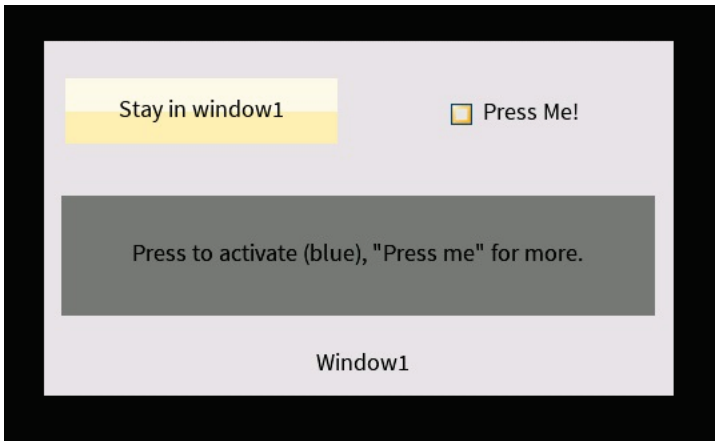


Figure 73. Window1 Display

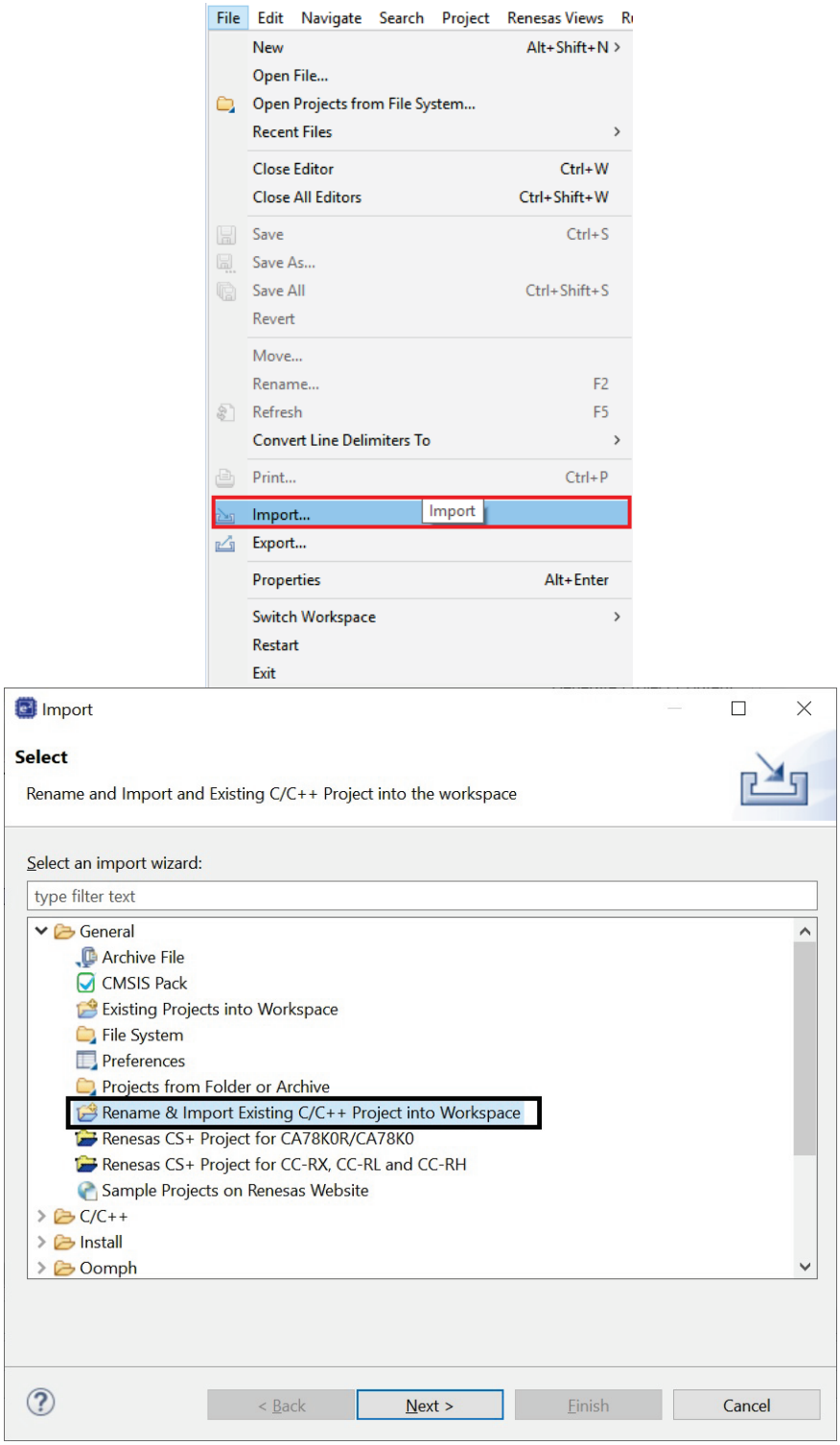
6. Overview of Fully Functional Project

6.1 Overview

In this section, you will import and run the complete “**mipi_ek_ra8d1_guix_hello_world**” project. You can enable or disable the check box function. The text on the button, which is “Stay in window1” or “Go to Window 2”, will be updated. Once you press the button, the screen will change from window1 to window2. Follow the text message on the screen; you can change from window2 back to window1.

6.2 Procedural Steps

1. You can try the provided project “**mipi_ek_ra8d1_guix_hello_world**” for the full function application. Use the **Rename & Import Existing C/C++ Project into Workspace** feature of the **Import** menu to do so since you already had a project with the same in the workspace.


File Edit Navigate Search Project Renesas Views
New Alt+Shift+N
Open File...
Open Projects from File System...
Recent Files
Close Editor Ctrl+W
Close All Editors Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Import...
Import
Export...
Properties Alt+Enter
Switch Workspace
Restart
Exit
Import
Select
Rename and Import and Existing C/C++ Project into the workspace
Select an import wizard:
type filter text
General
Archive File
CMSIS Pack
Existing Projects into Workspace
File System
Preferences
Projects from Folder or Archive
Rename & Import Existing C/C++ Project into Workspace
Renesas CS+ Project for CA78K0R/CA78K0
Renesas CS+ Project for CC-RX, CC-RL and CC-RH
Sample Projects on Renesas Website
C/C++
Install
Oomph
< Back
Next >
Finish
Cancel

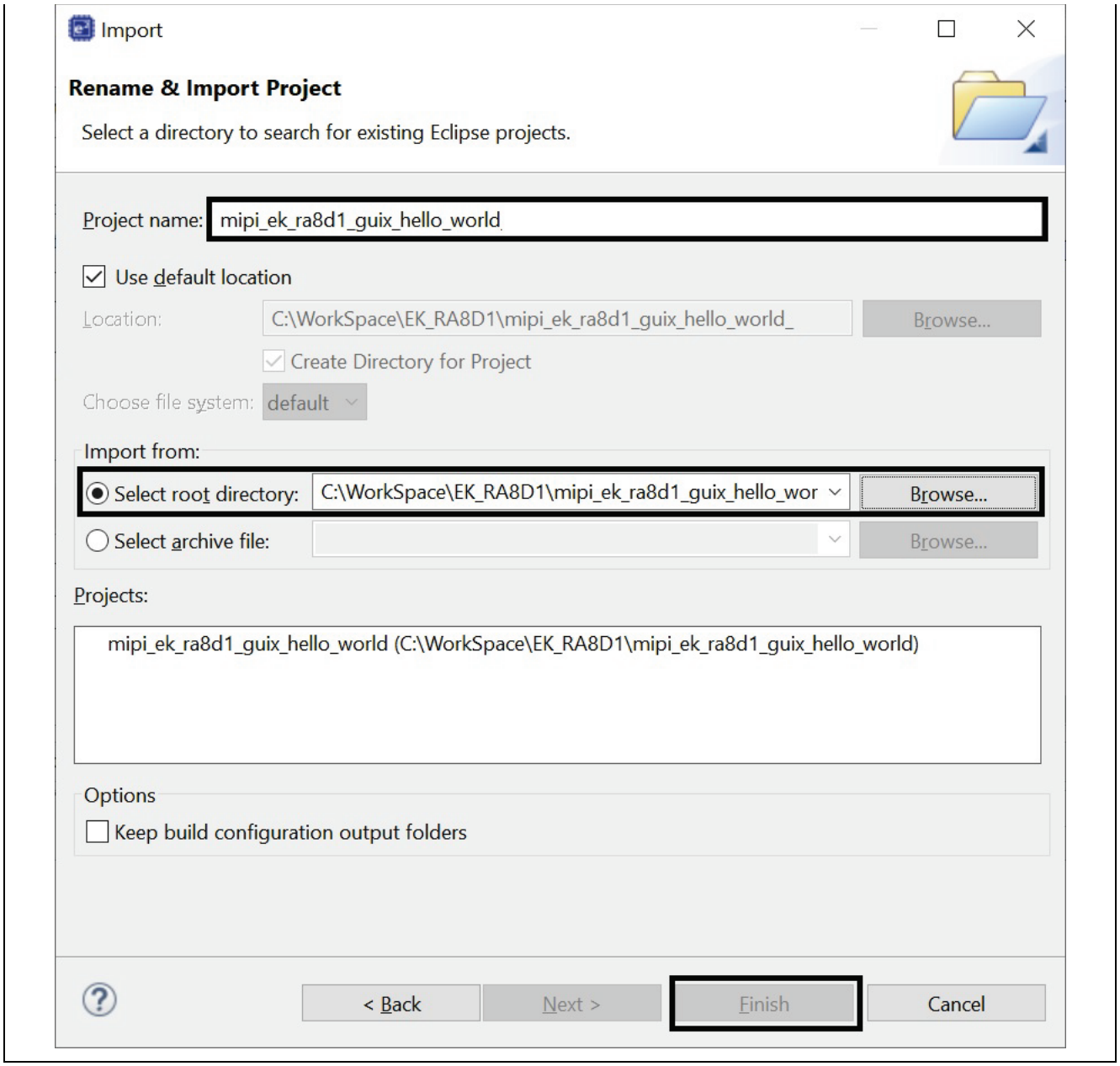


Figure 74. Import Existing Project

7. Website and Support

Visit the following URLs to learn about key elements of the RA family, download components and related documentation, and get support:

RA Product Information	renesas.com/ra
RA Product Support Forum	renesas.com/ra/forum
RA Flexible Software Package	renesas.com/FSP
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Feb.09.24	—	Initial release

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

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

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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