ルネサス RA ファミリ

EK-RA8D1 GUIX「Hello World」

はじめに

このアプリケーションノートでは、Azure RTOS GUIX Studioを使用してEK-RA8D1キット向けにシンプルな2画面のGUIを作成するプロセスについて説明します。このアプリケーションは、ルネサスのフレキシブルソフトウェアパッケージ（FSP）を使用して、ユーザが簡単に新規アプリケーションを作成、設定できる方法を説明しています。
ルネサスフレキシブルソフトウェアパッケージには、Azure RTOS ThreadX®リアルタイムオペレーティングシステム、Azure RTOS GUIXライブラリ、ハードウェアドライバが1つの堅牢なソフトウェアパッケージとして統合されています。この強力な一連のツールは、複雑な組み込みアプリケーションを迅速に開発するための統合フレームワークを提供します。

本アプリケーションは、FSP を使用し、e^2 studio で開発しています。

必要なリソース

開発ツールとソフトウェア

- e^2 studio IDE バージョン:2023-10 (23.10.0)
- ルネサスフレキシブルソフトウェアパッケージ（FSP）v5.1.0
- Azure RTOS GUIX Studio V6.2.1.0

ハードウェア

- ルネサス EK-RA8D1 キット（RA8D1 MCU グループ）
  - 静電容量式タッチパネル 40 ピン接続を備えた ER-TFT043-3
  - EK-RA6M3G キットの LCD を使用することを推奨します。
- ルネサス EK-RA8D1 の SW1 スイッチの設定。
  - GLCDC スイッチ#6 を ON 設定、SDRAM スイッチ#7 を ON 設定します。
- Renesas-app-lcd-conv_v1_b_mfg のリンクからご注文ください：https://oshpark.com/shared_projects/pzfp0mCD
  - LCD コンバーターボードを注文するには、Actions ボタンをクリックしてください。
  - ピン接続については、３章のステップ 11 の図 29 を参照してください。

参考マニュアル

- RA フレキシブルソフトウェアパッケージ ドキュメントリリース v5.0.0
- Azure RTOS GUIX および GUIX Studio v6.2.1.0
- ルネサス RA8D1 グループ ユーザーズマニュアル ハードウェア編 Rev.1.1.0
- EK-RA8D1-v1.0 回路図

提供ソフトウェアファイル

- Source.zip フォルダには、touch_ft5x06 フォルダと 4 つの*.c ファイルが含まれています。
- hal_entry.c, system_thread_entry.c, touch_thread_entry.c, windows_handler.c

目的

このドキュメントでは、e^2 studioでのAzure RTOS GUIXタッチスクリーンインターフェースのHello Worldアプリケーションのセットアップと、FSPに含まれるドライバとライブラリを設定する方法を説明します。これらにより、GLCDコントローラ、タッチスクリーンドライバ、およびセマフォをセットアップして、アプリケーションタスクと通信することができます。また、Azure RTOS GUIX Studioエディタを使用してシンプルなGUIインターフェイスの作成方法も説明しています。さらに、このアプリケーションノートで、プロジェクトのセットアップと基本的なデバッグ操作についても説明します。実行中は、アプリケーションがタッチスクリーン操作に応答し、グラフィカルユーザインターフェース(GUI)を表示します。

対象読者

対象読者は、GUIアプリケーションを設計したいユーザです。

注： もしユーザが弊社で提供のプロジェクト **ra8d1_guix_hello_world** をすぐに実行したい場合は、6 章を参照してください。

目次

1. ツールのダウンロードとインストール

1.1 概要

ここでは、e² studio v2023-10 /FSP v5.1.0とAzure RTOS GUIX studio v6.2.1.0をインストールします。

1.2 手順

1. すでに FSP v5.1.0 以降がインストールされている e² studio v2023-10 がある場合は、この手順をスキップできます。それ以外の場合、https://www.renesas.com/jp/ja/software-tool/flexible-software-package-fsp からダウンロードできます。

   e² studio および FSP の詳細なインストール手順については、ルネサスウェブサイト https://www.renesas.com/fsp で入手できます。e² studio のリリース ノートを参照して、e² studio バージョンが選択した FSP バージョンをサポートしていることを確認します。インストーラのスタートバージョンには、RA MCU のすべての機能が含まれています。

2. このリンクから Azure RTOS GUIX Studio v6.2.1.0 以上をダウンロードできます。

   次の手順により web ブラウザにウィンドウ表示されます。
   注: Azure RTOS GUIX studio をインストールするには、PC に Microsoft Store がインストールされ、動作している必要があります。

3. **Install** ボタンをクリックすると、新規ウィンドウがポップアップします。

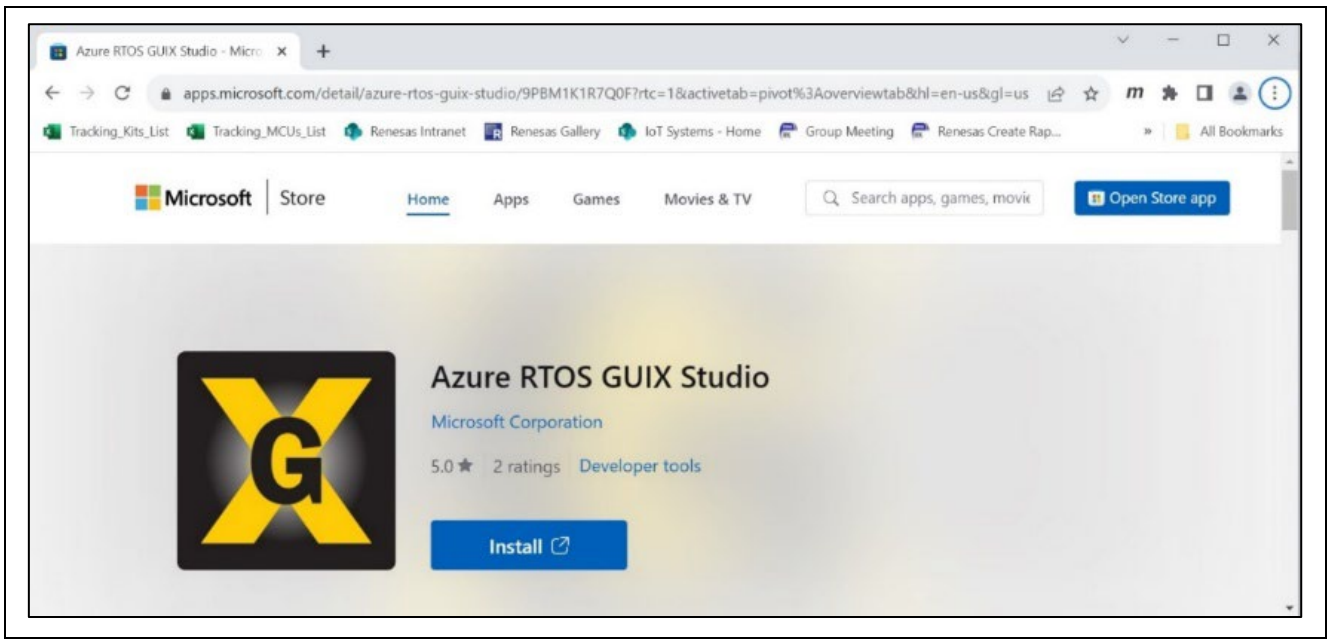


図 1.　Azure RTOS GUIX Studio の入手

4. Azure RTOS GUIX Studio のインストールを続行するには、**Open Microsoft Store** をクリックします。

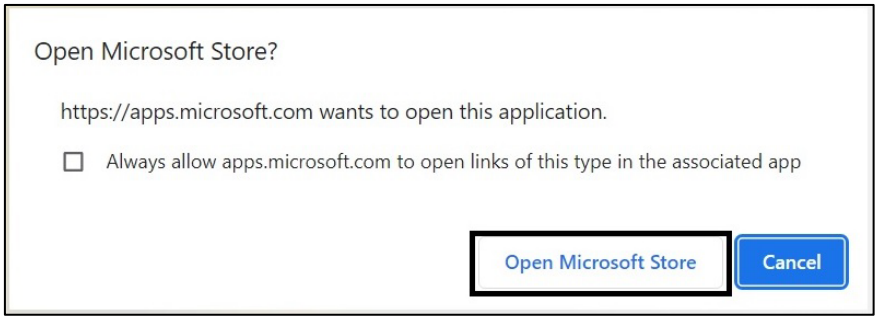


図 2. Microsoft Store を開く

5. **Open** をクリックして、**Azure RTOS GUIX Studio** アプリを開きます。

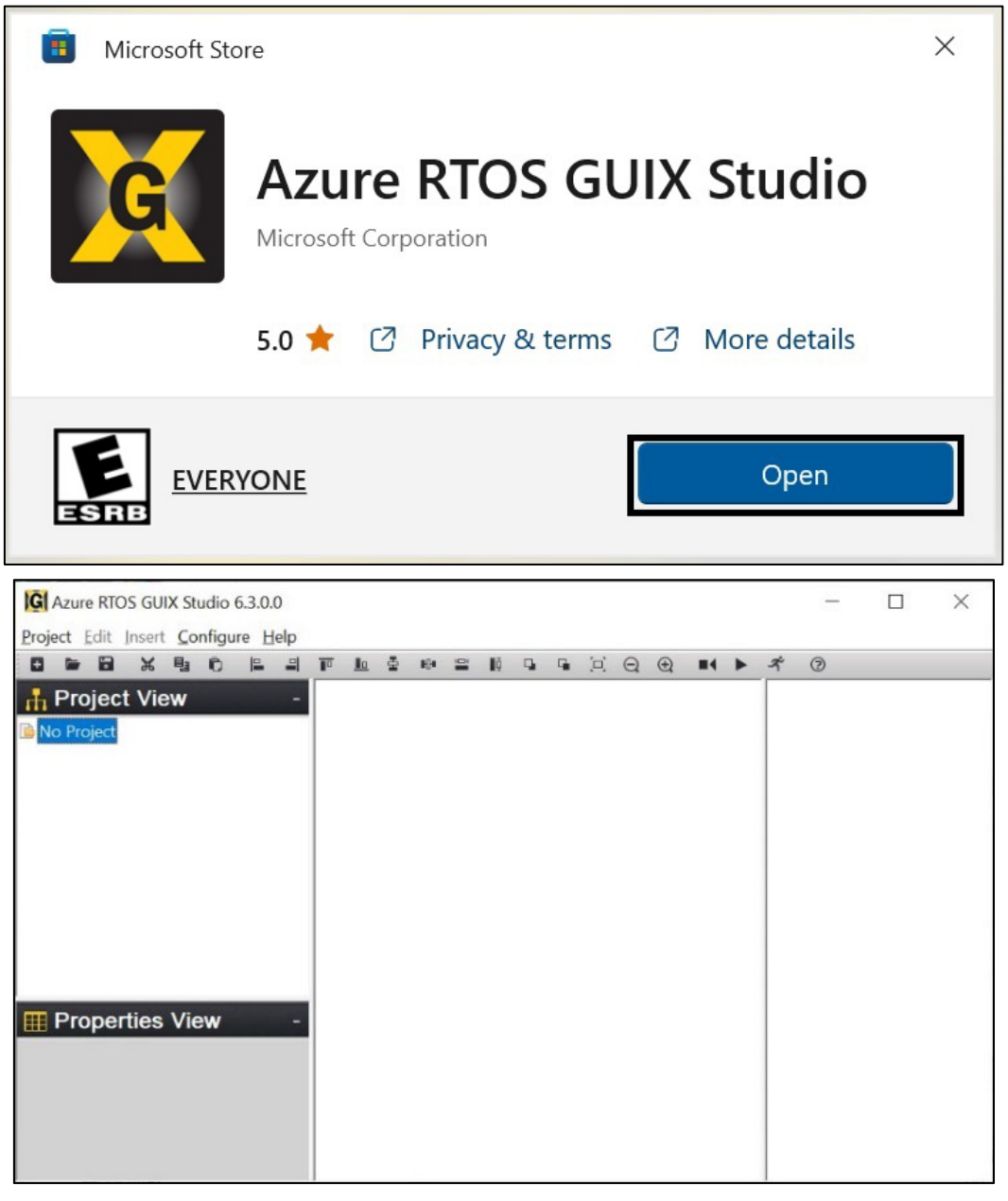


図 3. Azure RTOS GUIX Studio の起動

2. アプリケーションプロジェクトの作成とバックライトの有効化

2.1 概要

この章では、新たなプロジェクトを作成し、Azure RTOS GUIX studio プロジェクトと統合する手順を示します。

2.2 手順

1. 新規に Renesas RA C/C++ project を作成します。Project name を ra8d1_guix_hello_world にします。

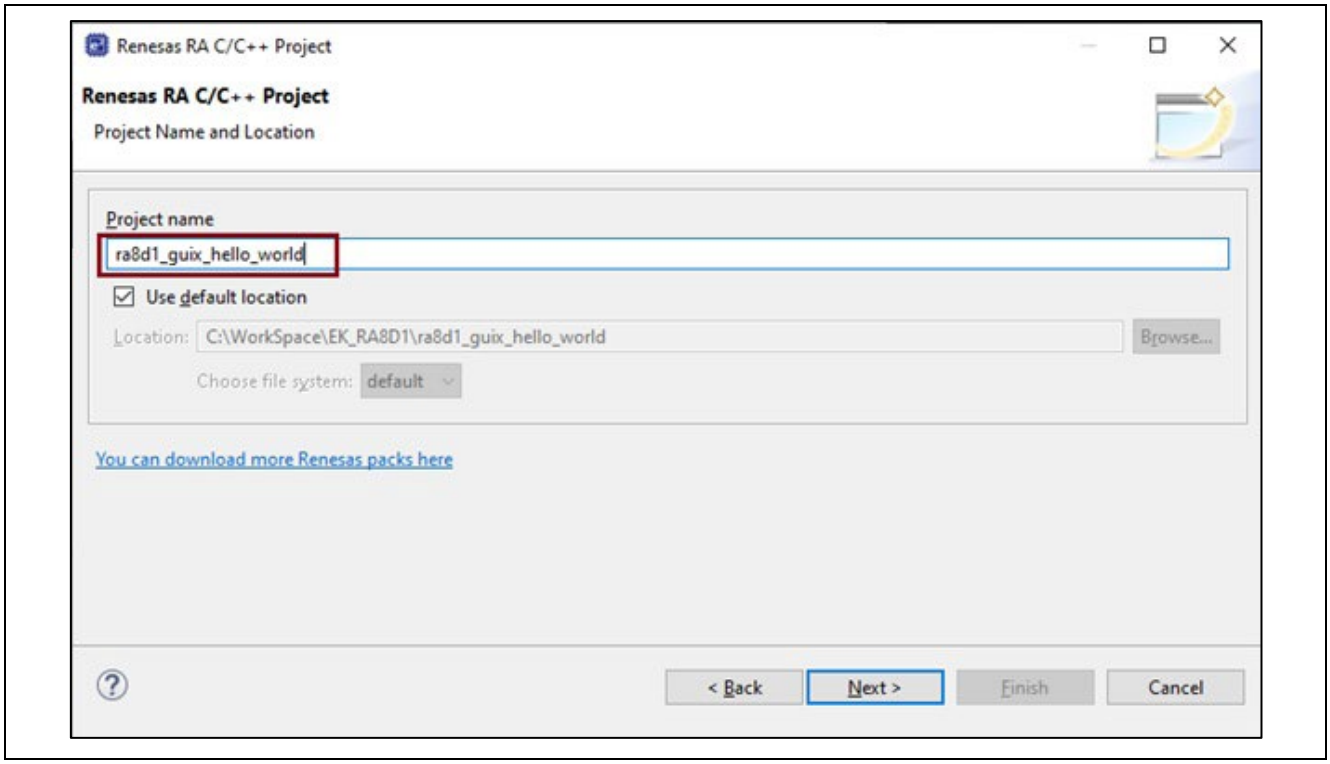


図 4. 新規プロジェクトの作成

2. ボードを選択して、EK-RA8D1 に設定します。

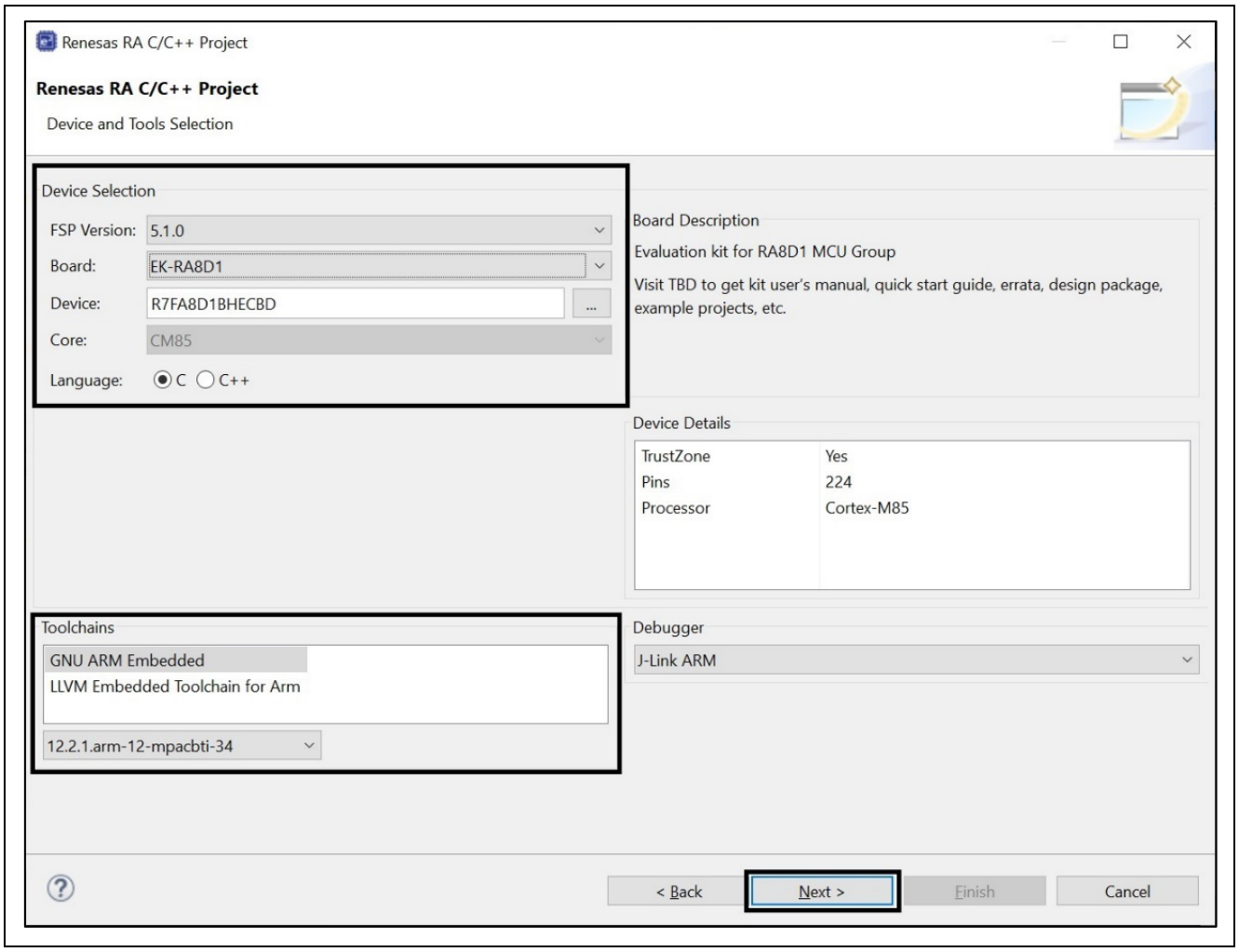


図 5. 評価ボードの選択・設定

3. Project Type Selection では Flat(Non-TrustZone)を選択。Build Artifact and RTOS Selection では、Executable と Azure RTOS ThreadX (v6.2.1+FSP.5.1.0)を選択します。

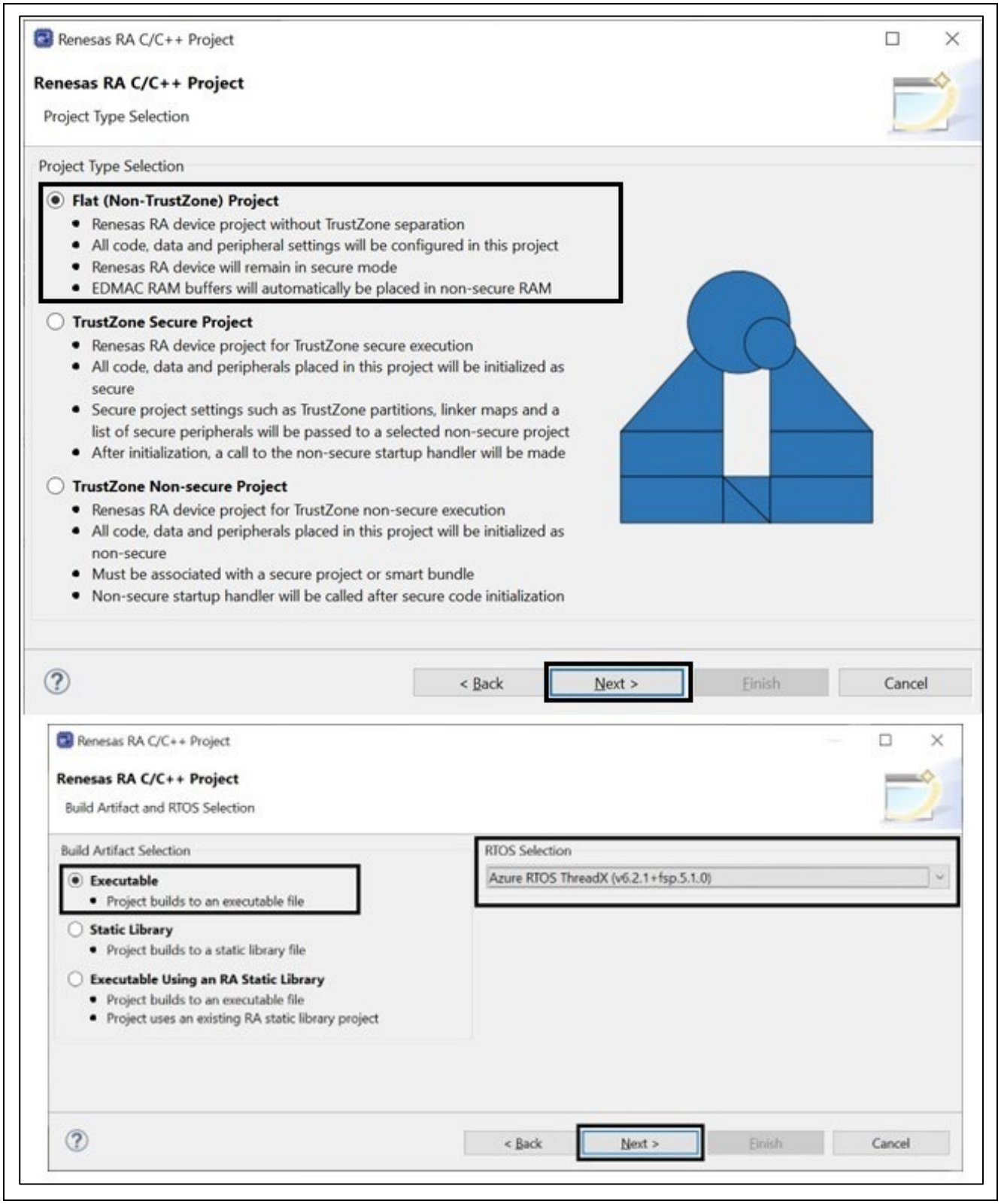


図 6. Azure RTOS ThreadX の選択

4. FreeRTOS-Minimal テンプレートを使用します。

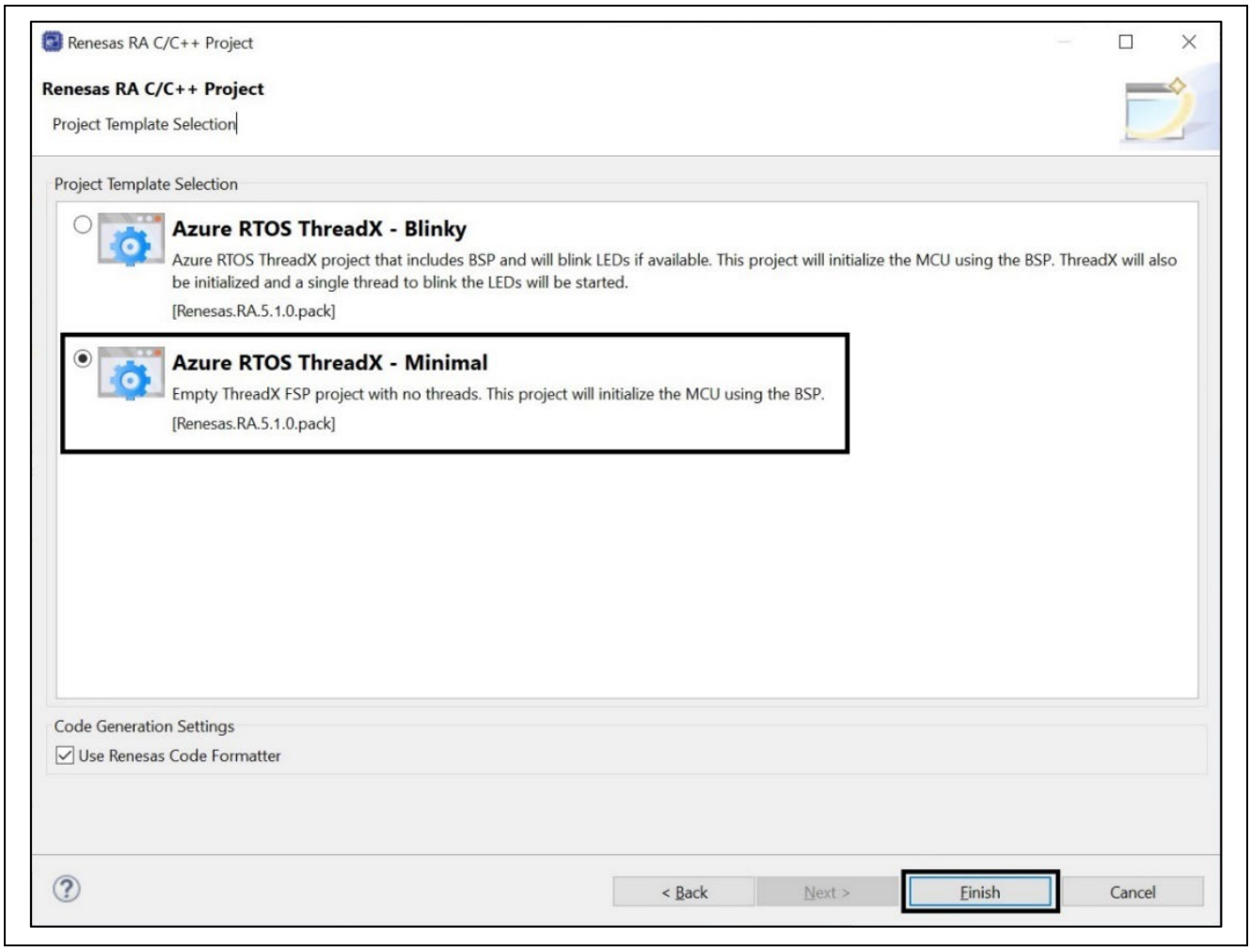


図 7. Azure RTOS ThreadX Minimal を選択後 Finish

5. プロジェクト設定を開き、BSP タブに移動します。Heap size(bytes)を 0x2000 に変更します。

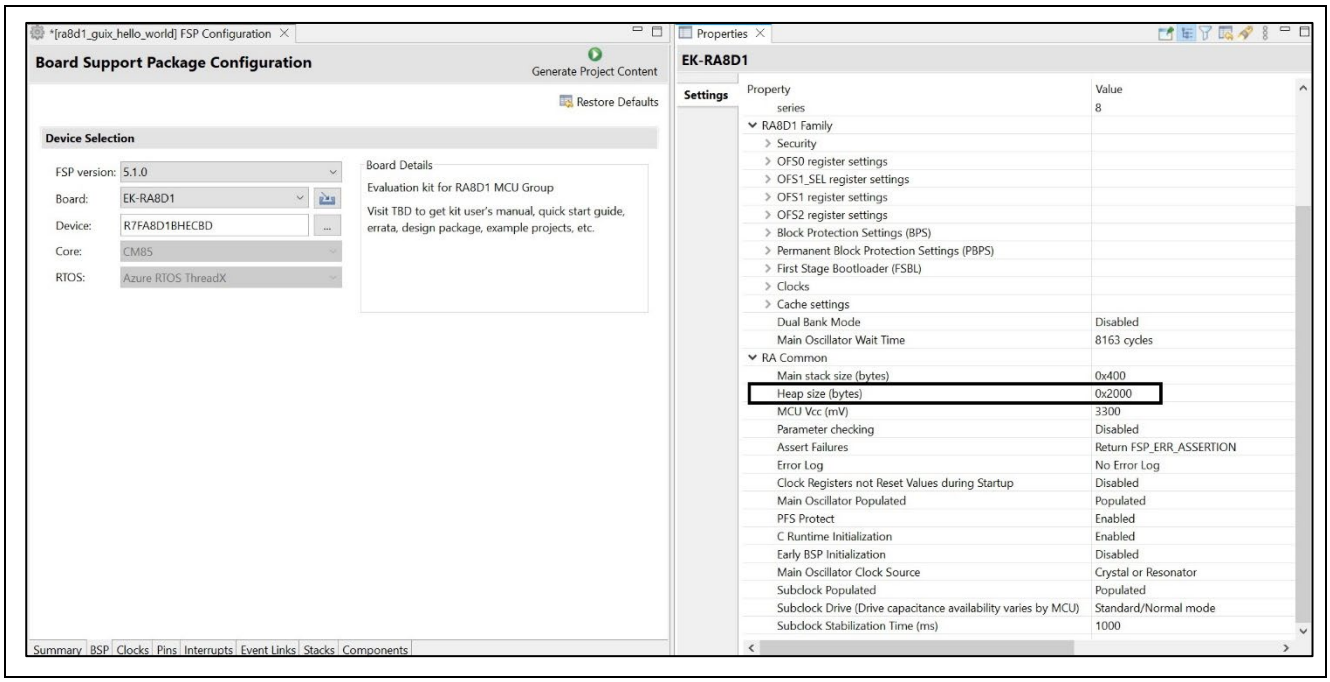


図 8. Heap Size の変更

6. Clocks タブをクリックし、LCD にクロックを設定します。

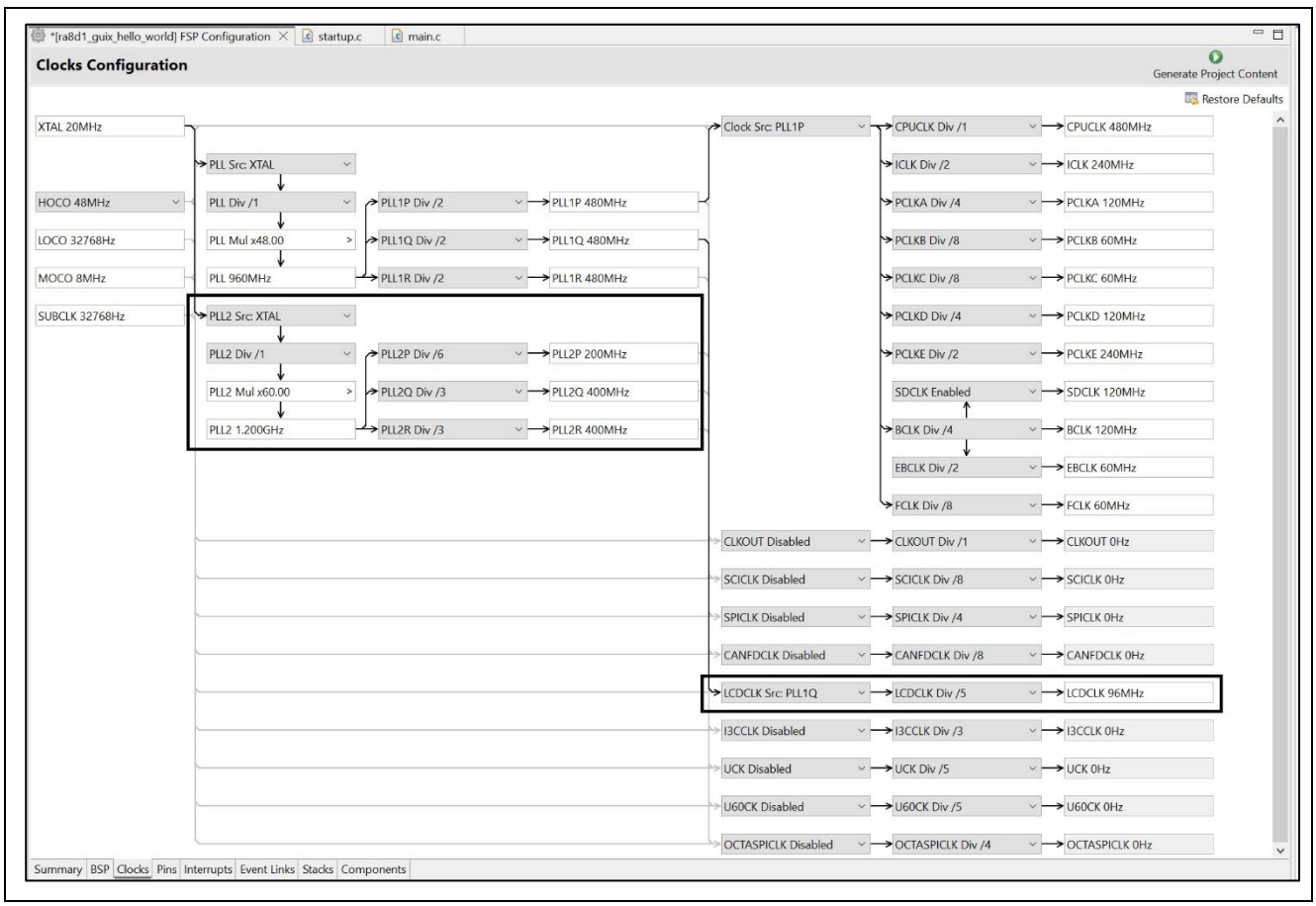


図 9. クロックの設定

7. **New Thread** を追加し、以下の設定で **System Thread** と名前を付けます。

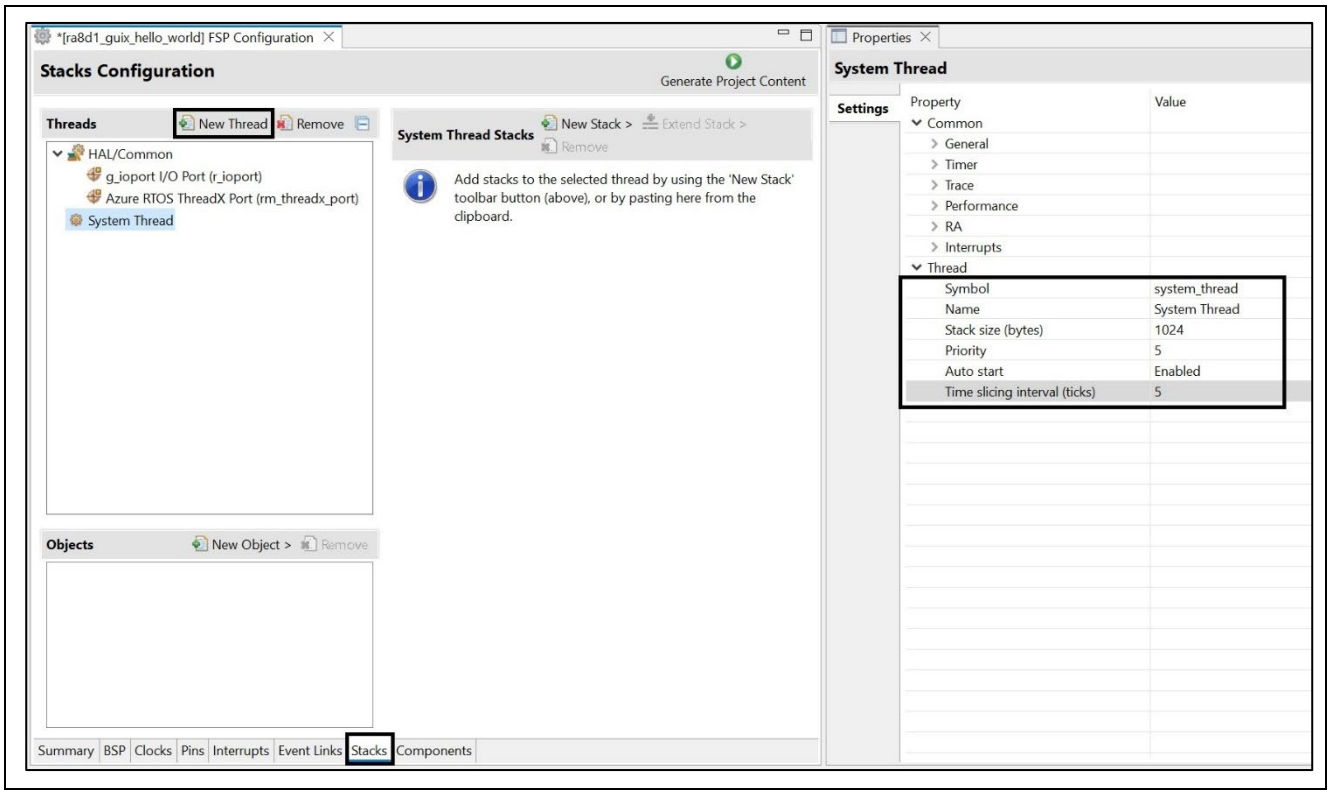


図 10. システムスレッドの追加

8. **New Stack** をクリックし、**Azure RTOS GUIX** をシステムスレッドに追加します。

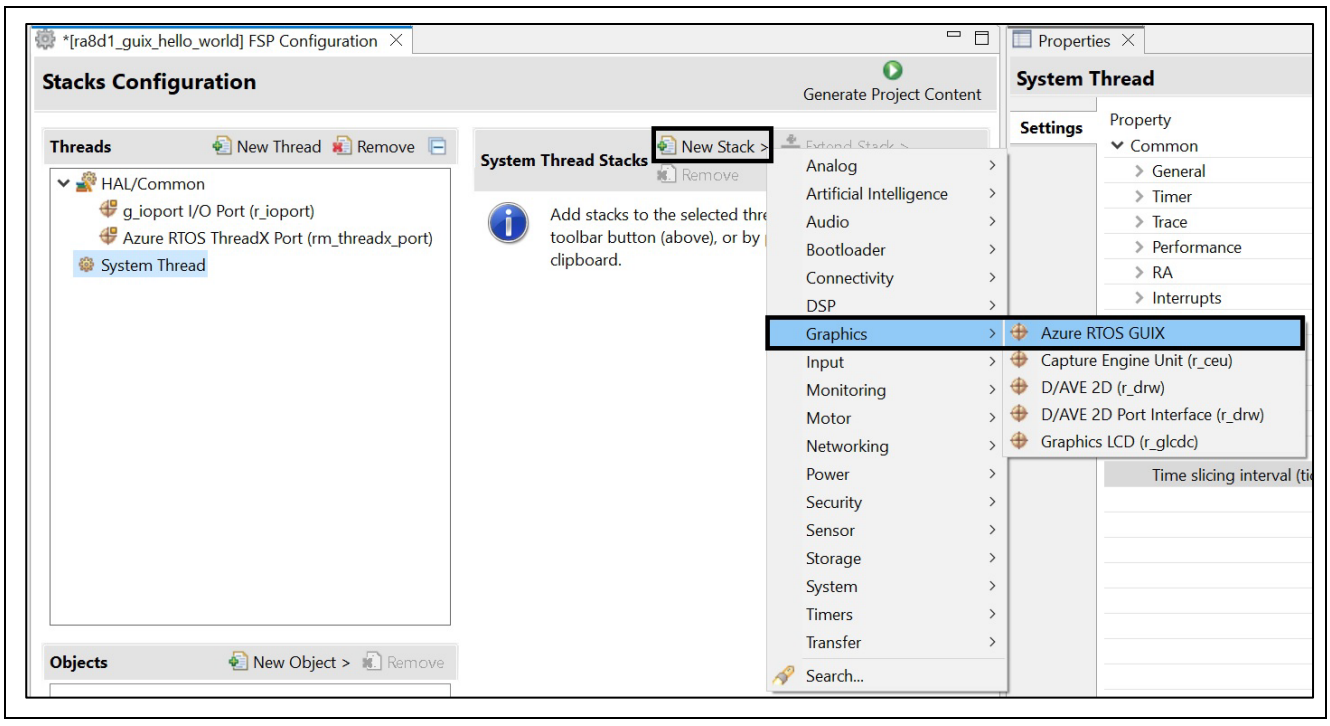


図 11. Azure RTOS GUIX の追加

9. Azure RTOS GUIX のプロパティ設定

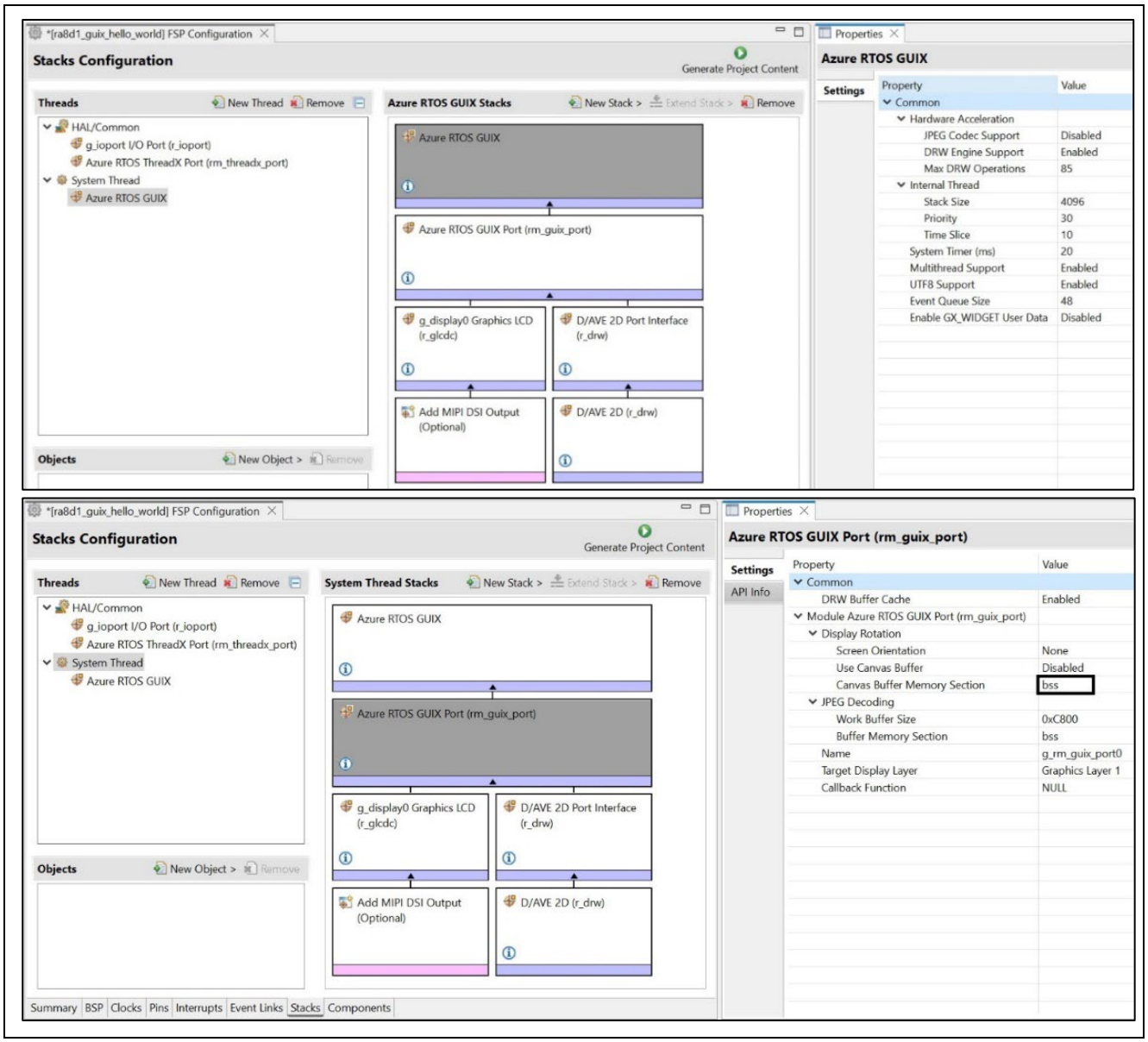


図 12. ハードウェア プロパティ設定と確認

10. グラフィックス LCD のプロパティ設定

注: LCD のプロパティを設定します。

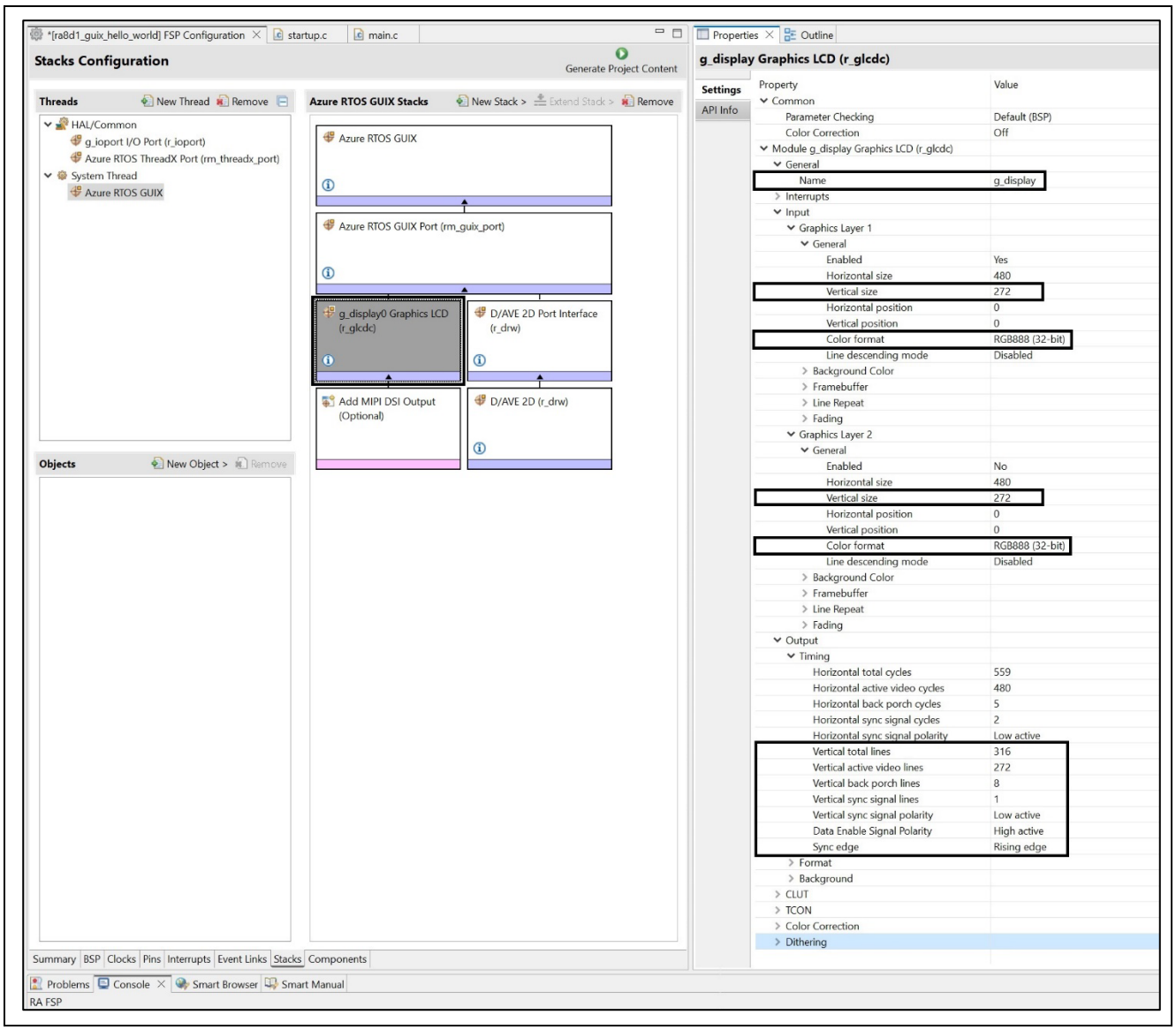


図 13. グラフィック LCD のプロパティ設定

11. **New Stack** をクリックし、PWM タイマ追加します。

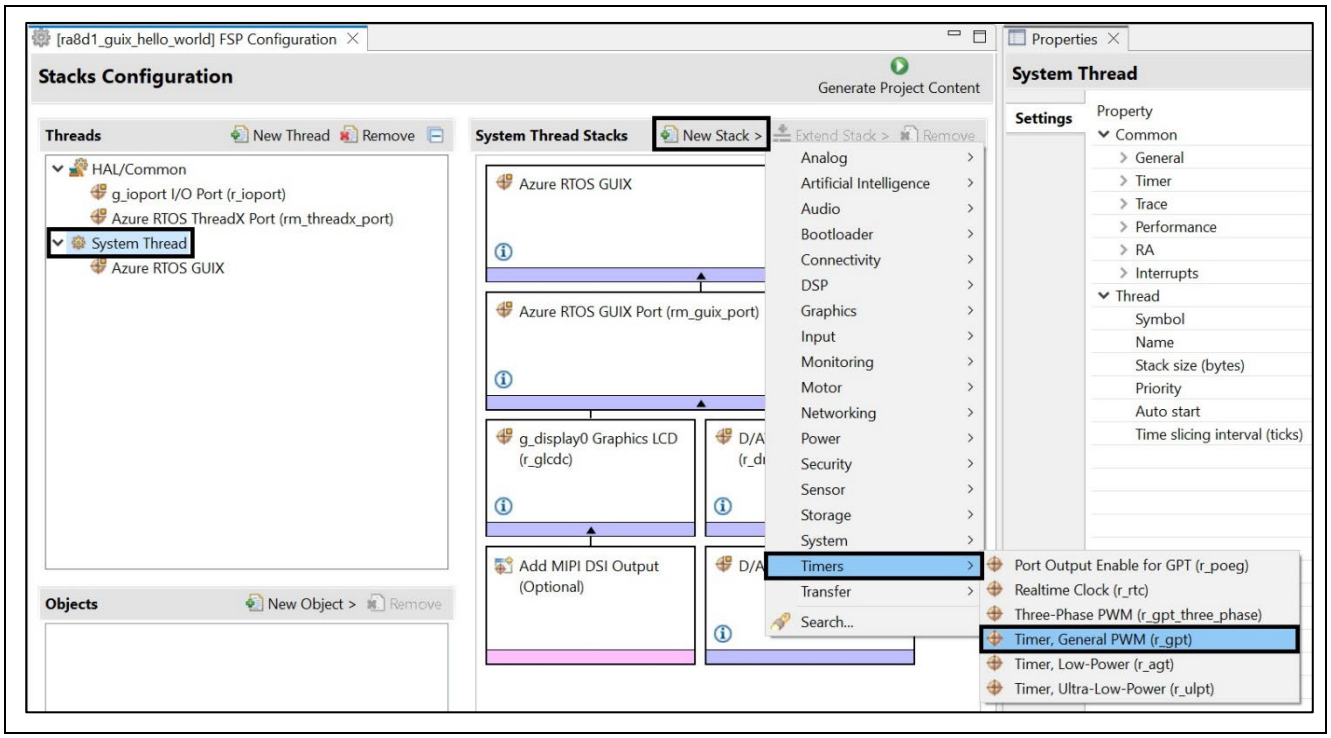


図 14. タイマの追加

12. タイマモジュールのプロパティを設定します。  をクリックして、P404 を LCD パネルの DISP_BLEN 信号に設定します。

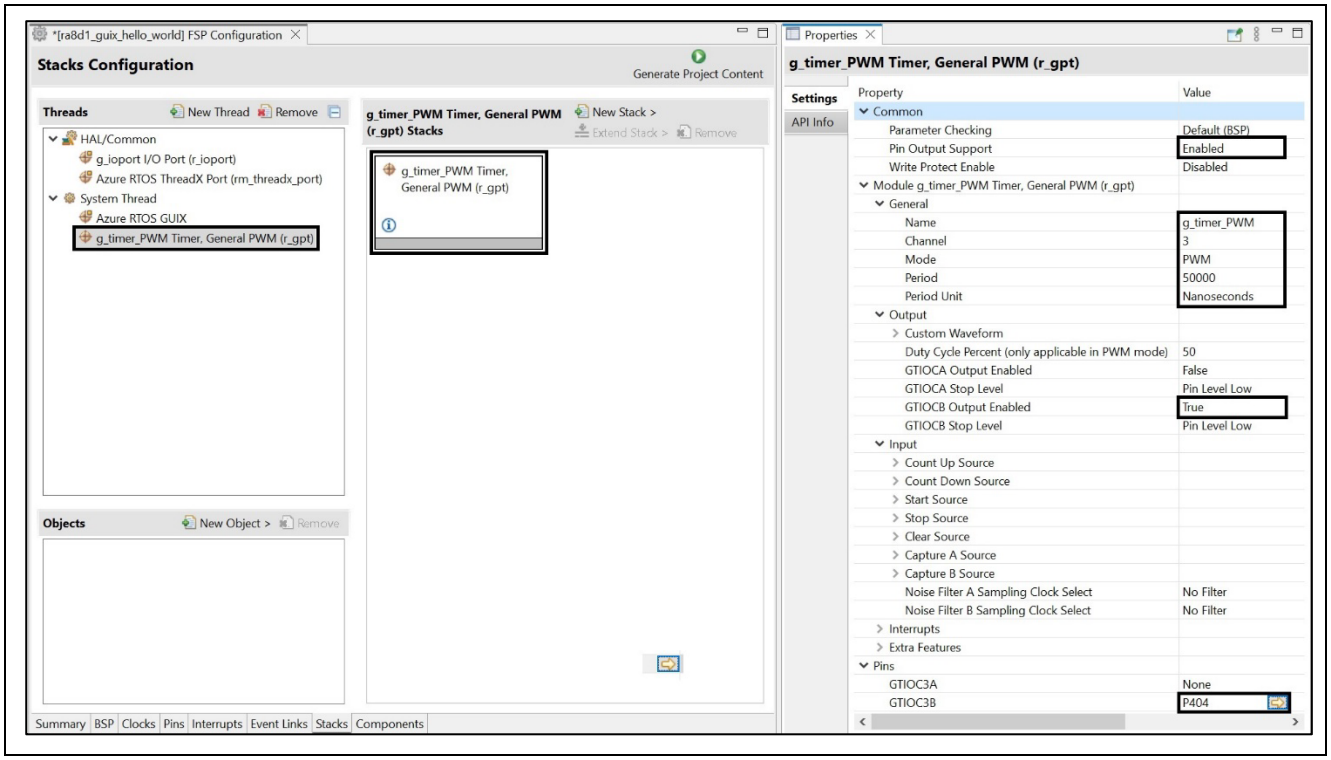
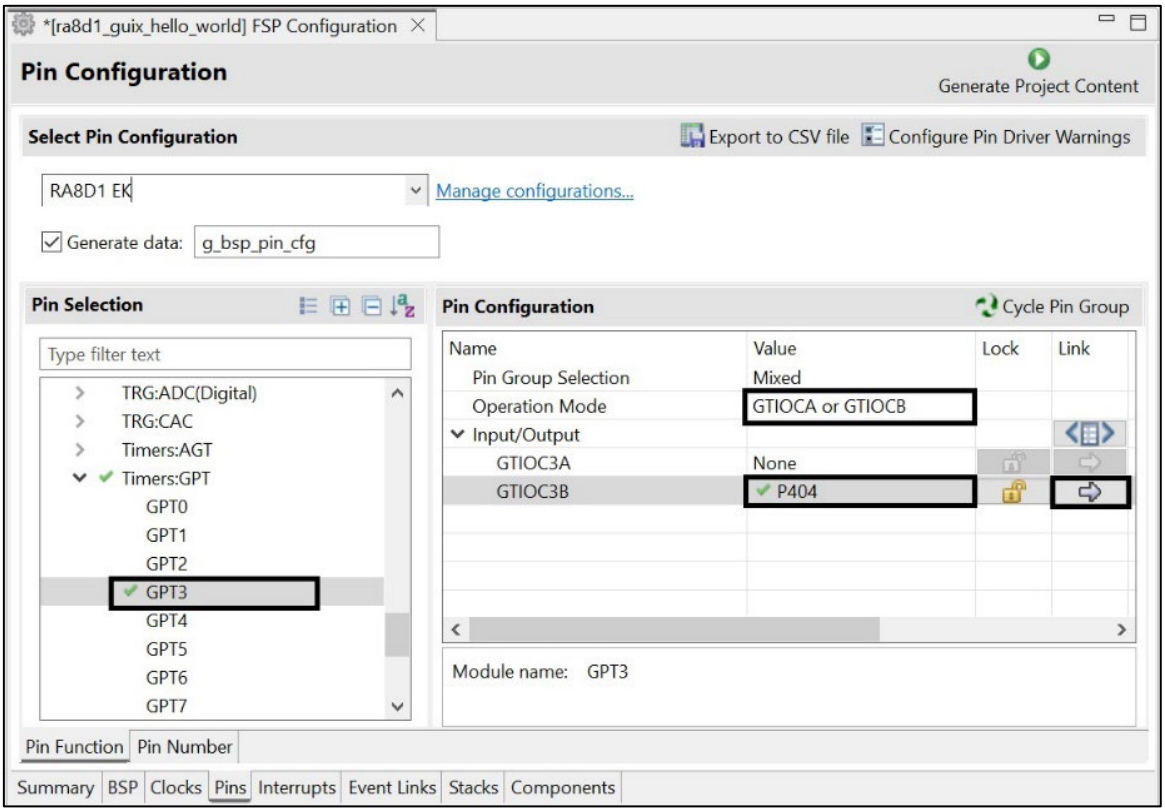


図 15. タイマのプロパティと DISP_BLEN P404 の設定

13. LCD パネルの **DISP_BLEN** 信号を **P404** に割り当てます

注：GTIOC3B ピンが<unavailable>と表示された場合は、まず手順 13 を実行し次に手順 12 を実行する必要があります。



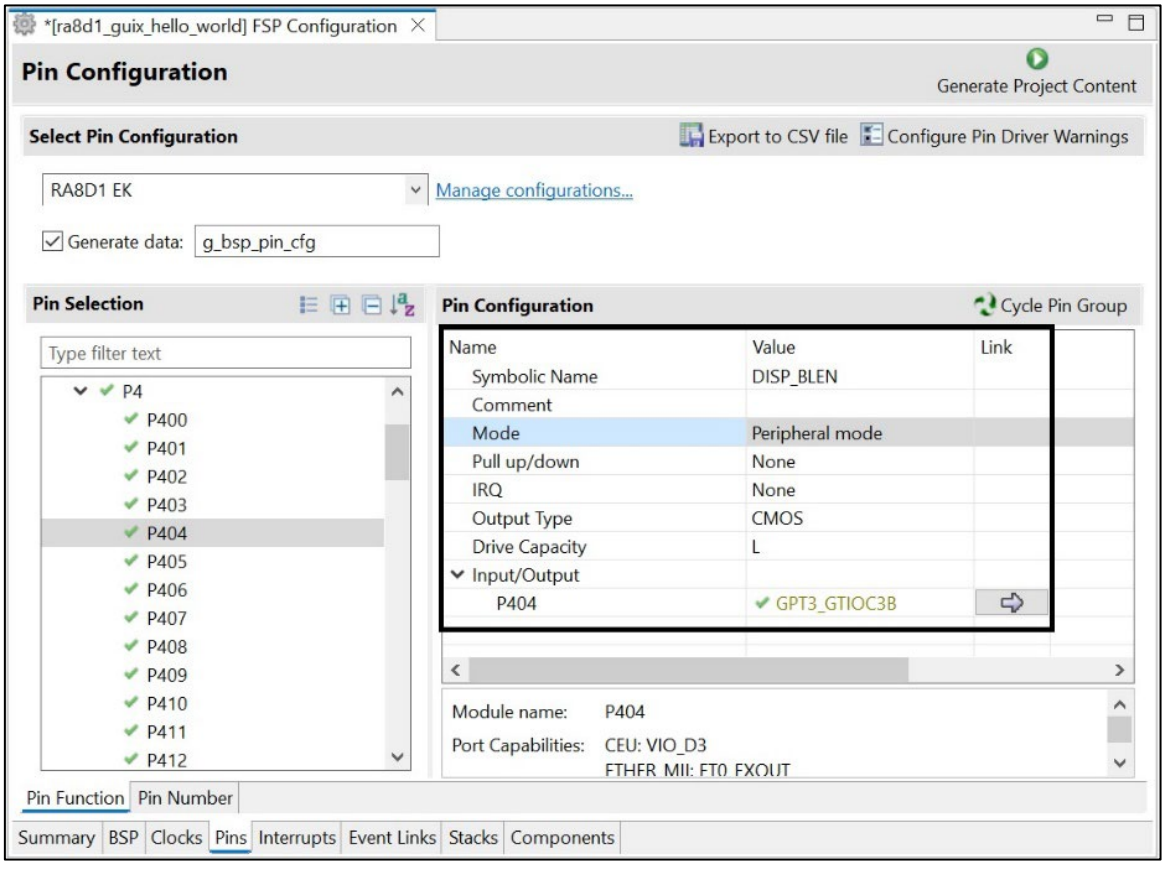


図 16. P404 DISP_BLEN の設定

14. LCD パネルの LCD_BLEN ピンは、ボード RA8D1 の DISP_BLEN P404 に接続されます。(図 17)

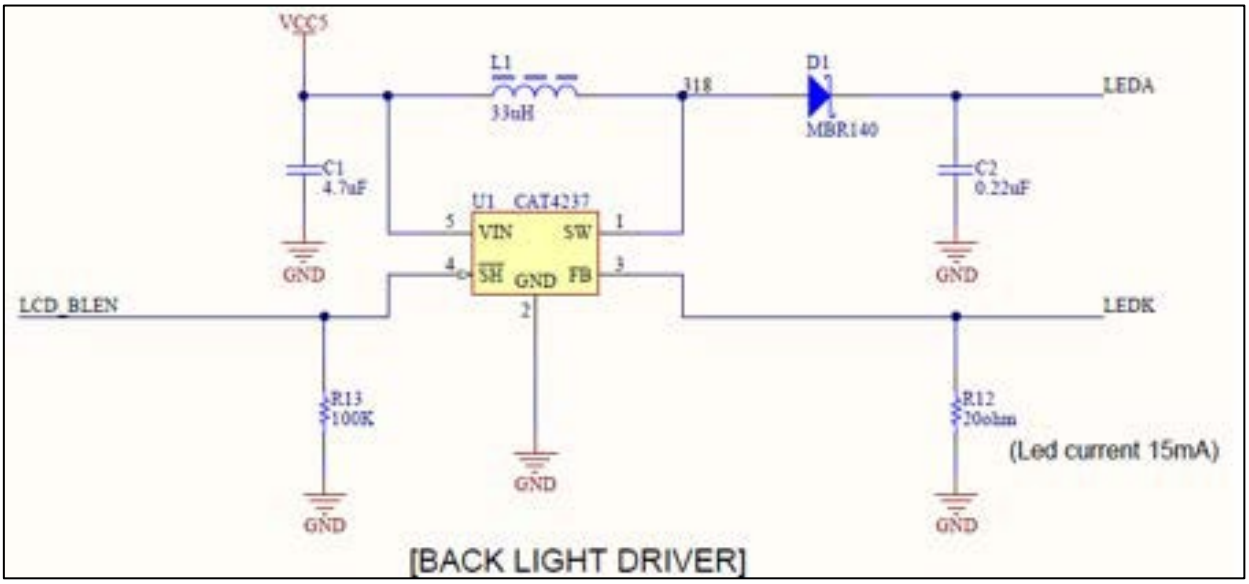


図 17. LCD_BLEN と DISP_BLEN P404 の接続

15. 詳細については、Source.zip の system_thread_entry.c ファイルを参照してください。
以下の関数は PWM 出力を制御します。

```
 * @brief This function is setting up GPT/PWM timer
static fsp_err_t gpt_timer_PWM_setup(void)
{
    fsp_err_t err = FSP_SUCCESS;
    /* Open GPT */
    err = R_GPT_Open(&g_timer_PWM_ctrl, &g_timer_PWM_cfg);
    if(FSP_SUCCESS != err)
    {
        return err;
    }
    /* Enable GPT Timer */
    err = R_GPT_Enable(&g_timer_PWM_ctrl);
    /* Handle error */
    if (FSP_SUCCESS != err)
    {
        return err;
    }
    /* Start GPT timer */
    err = R_GPT_Start(&g_timer_PWM_ctrl);;
    if(FSP_SUCCESS != err)
    {
        return err;
    }

    return err;
}
```

図 18. gpt_timer_PWM

3. タッチ機能ドライバの追加と設定

1. **New Thread** をクリックし、以下の設定で **Touch Thread** と名前を付けます。

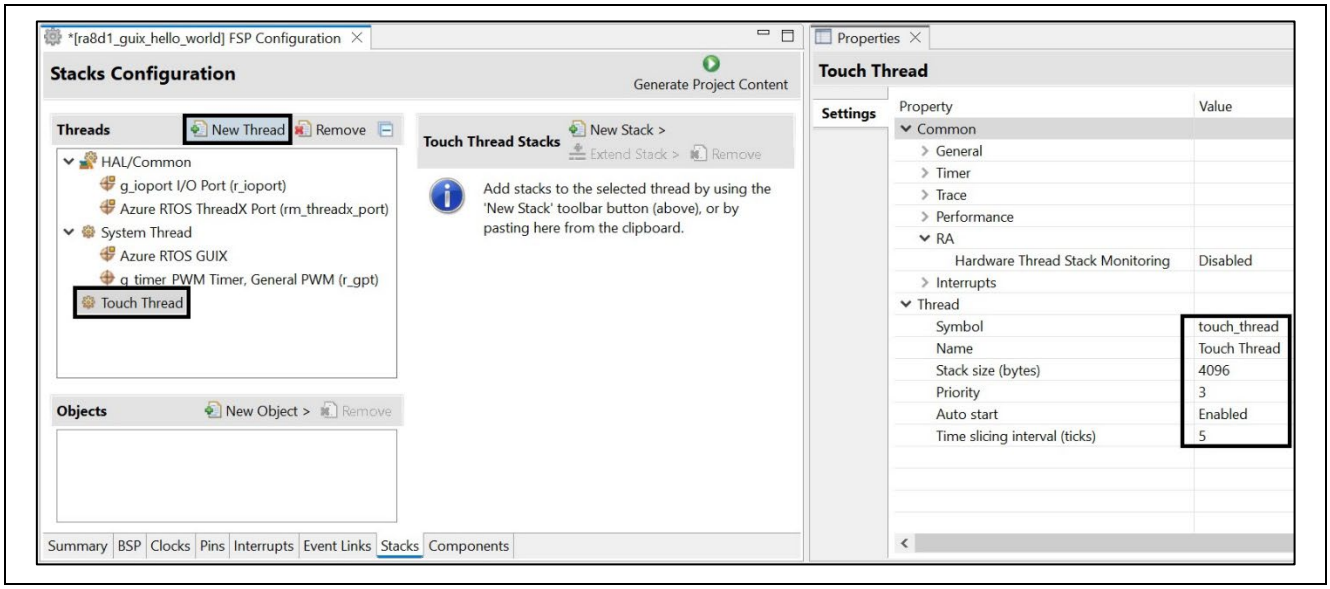


図 19. タッチスレッドプロパティの追加と設定

2. **New Stack** をクリックし、**External IRQ** モジュールを追加します。

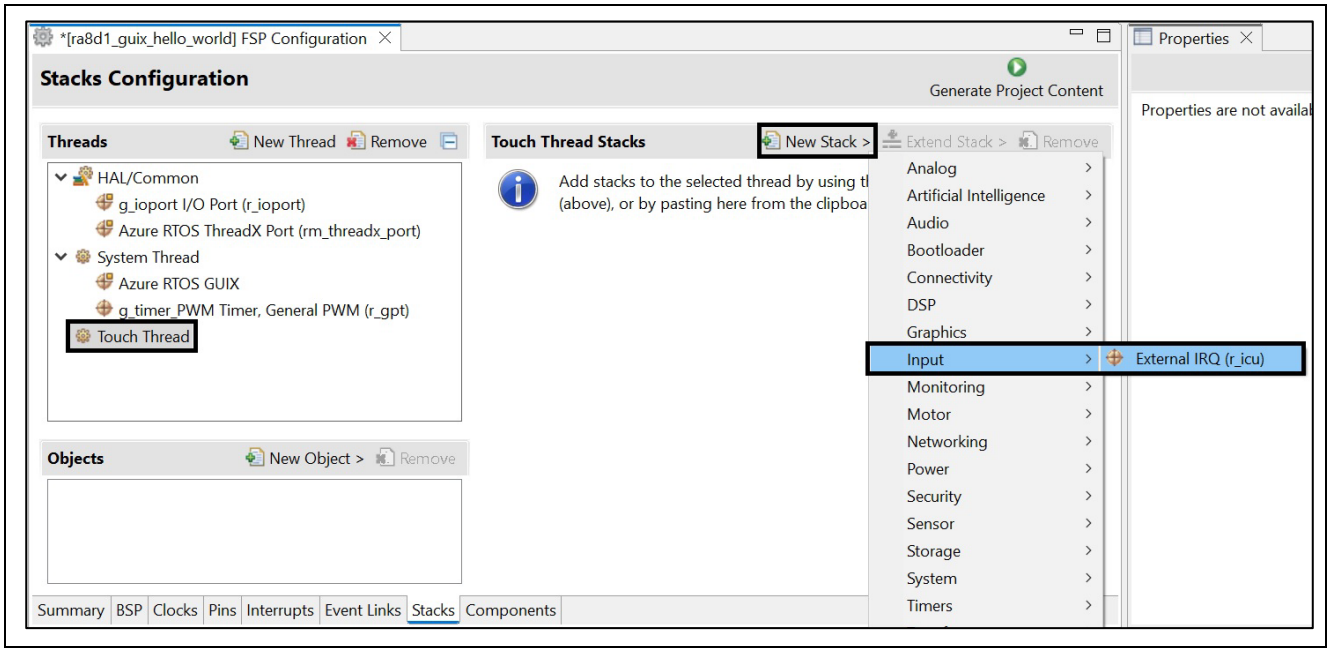


図 20. External IRQ の追加

3. g_touch_irq と名前を付け、External IRQ プロパティを設定します。

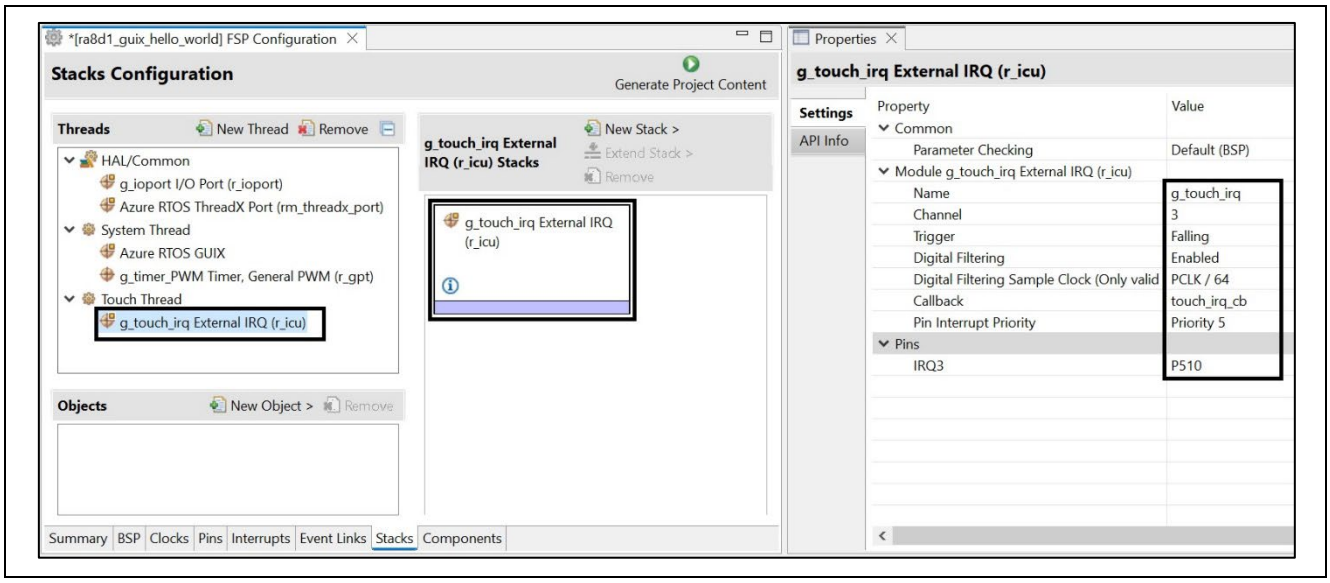


図 21. External IRQ のプロパティ設定

4. DISP_INT 信号を P510 に割り当てます。

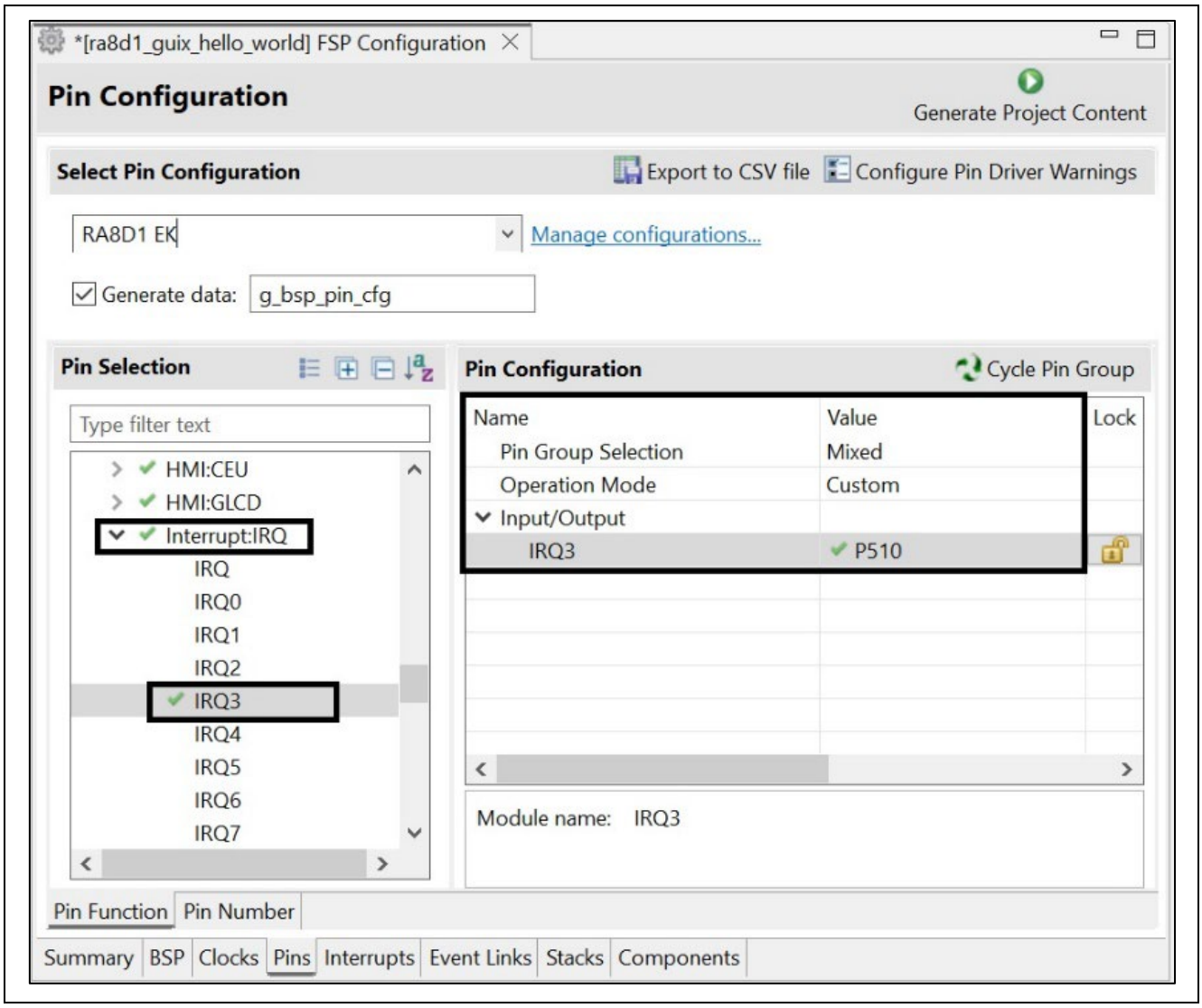


図 22. IRQ3 を P510 に割り当てる

5. **New Stack** をクリックし、**I2C Master** を追加します。

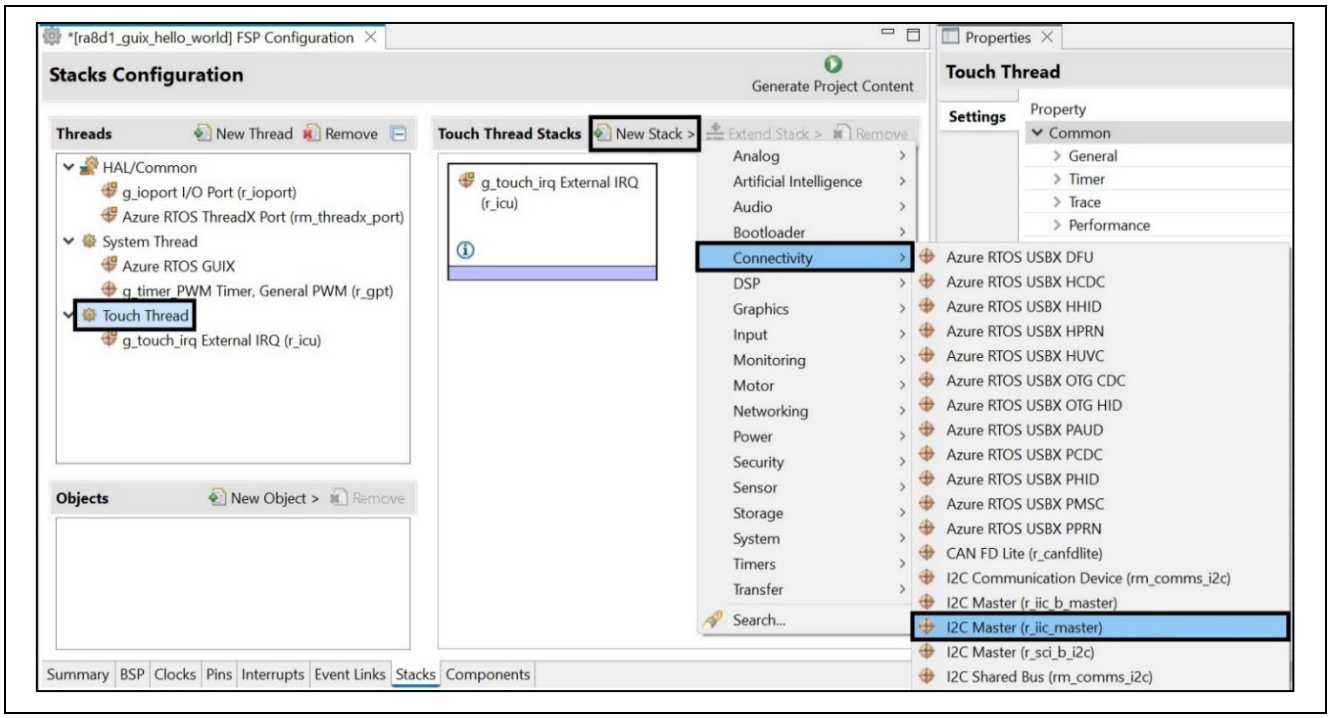


図 23. タッチ I2C マスターを追加

6. **I2C Master** に **g_i2c_touch** と名前を付け、プロパティを設定します。

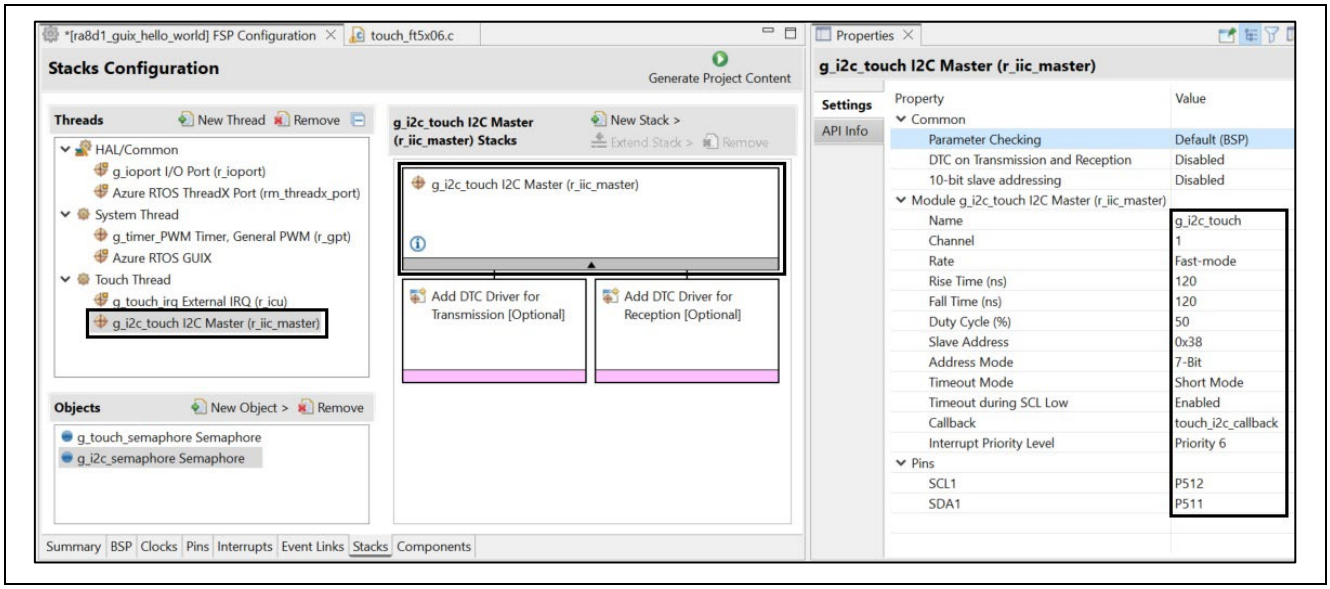


図 24. 名前とプロパティ設定

7. **New Object** をクリックし、セマフォを追加します。

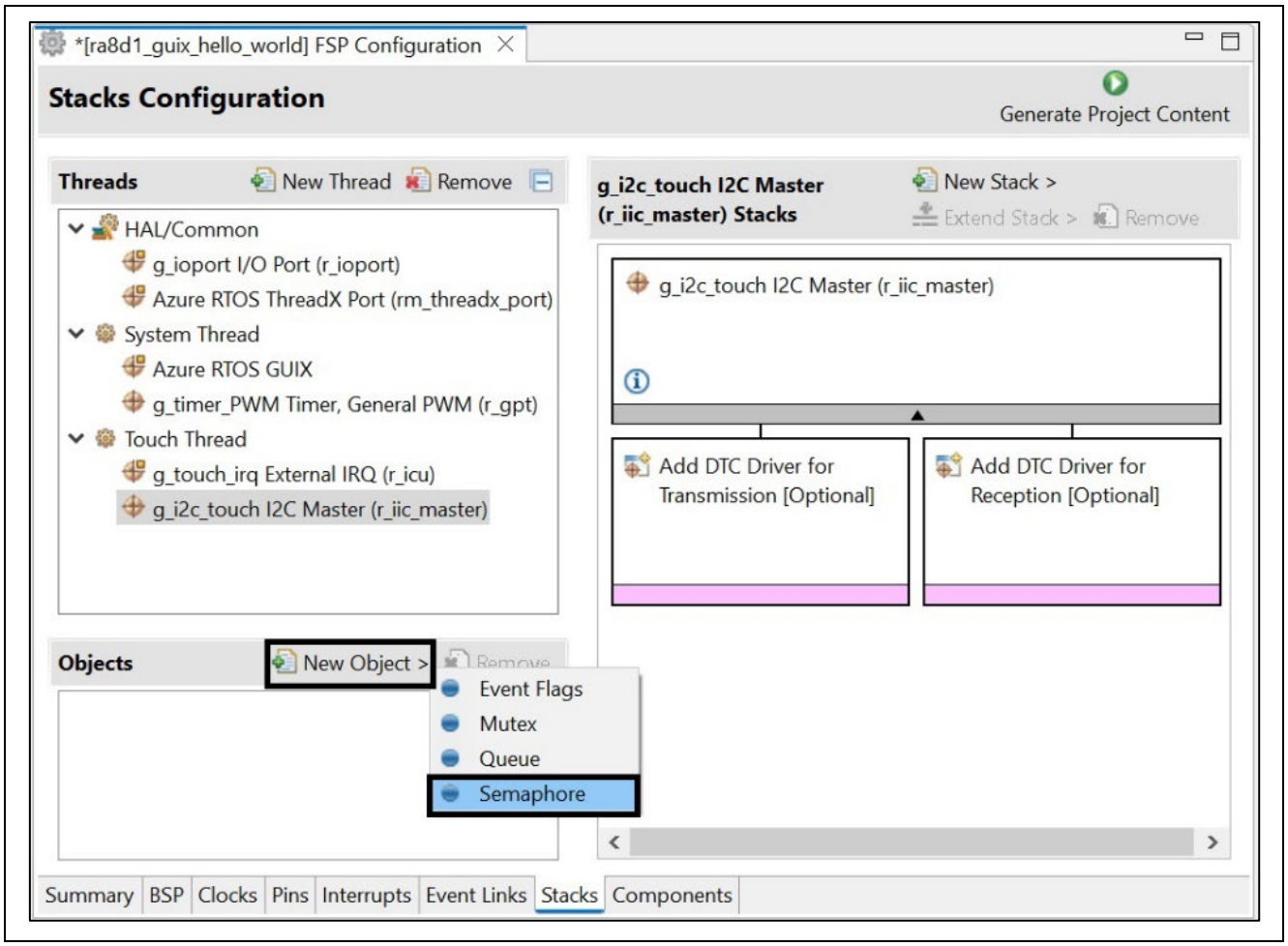


図 25. 新規セマフォの追加

8. **Touch Semaphore** と名前を付け、プロパティを設定します。

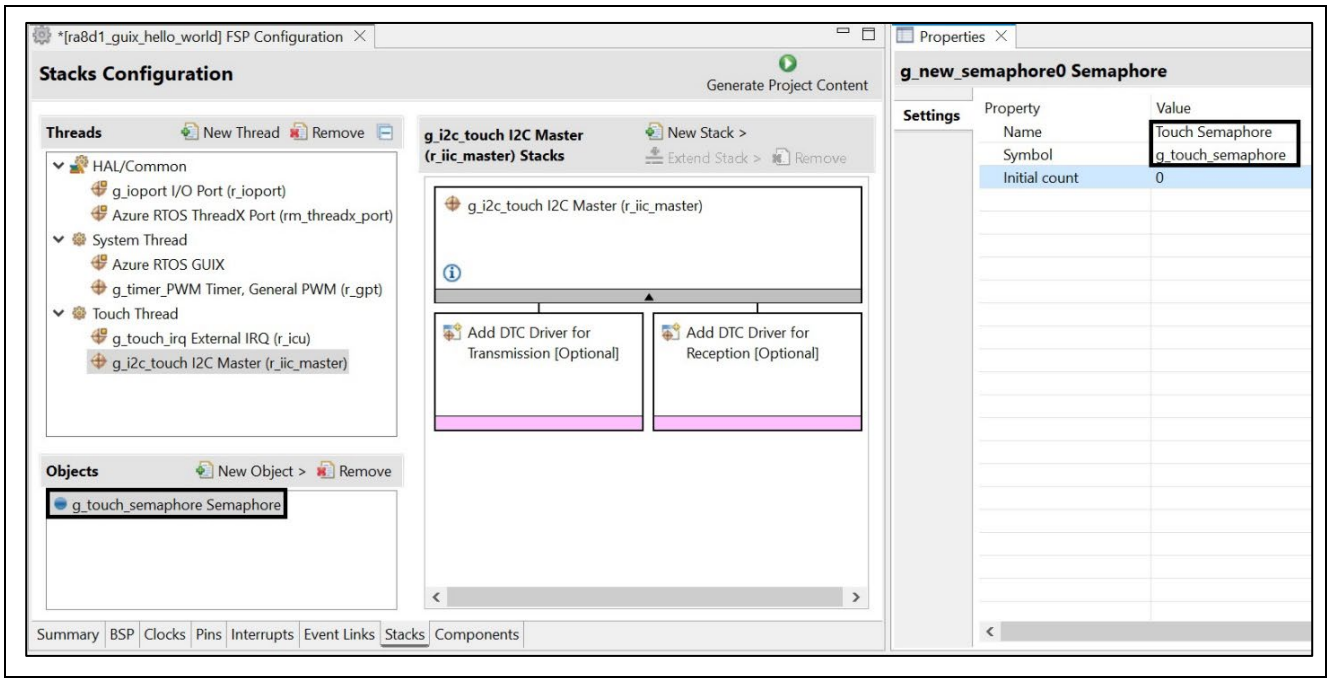


図 26. タッチ セマフォの追加と名前設定

9. **New Object** をクリックして別のセマフォを追加します。

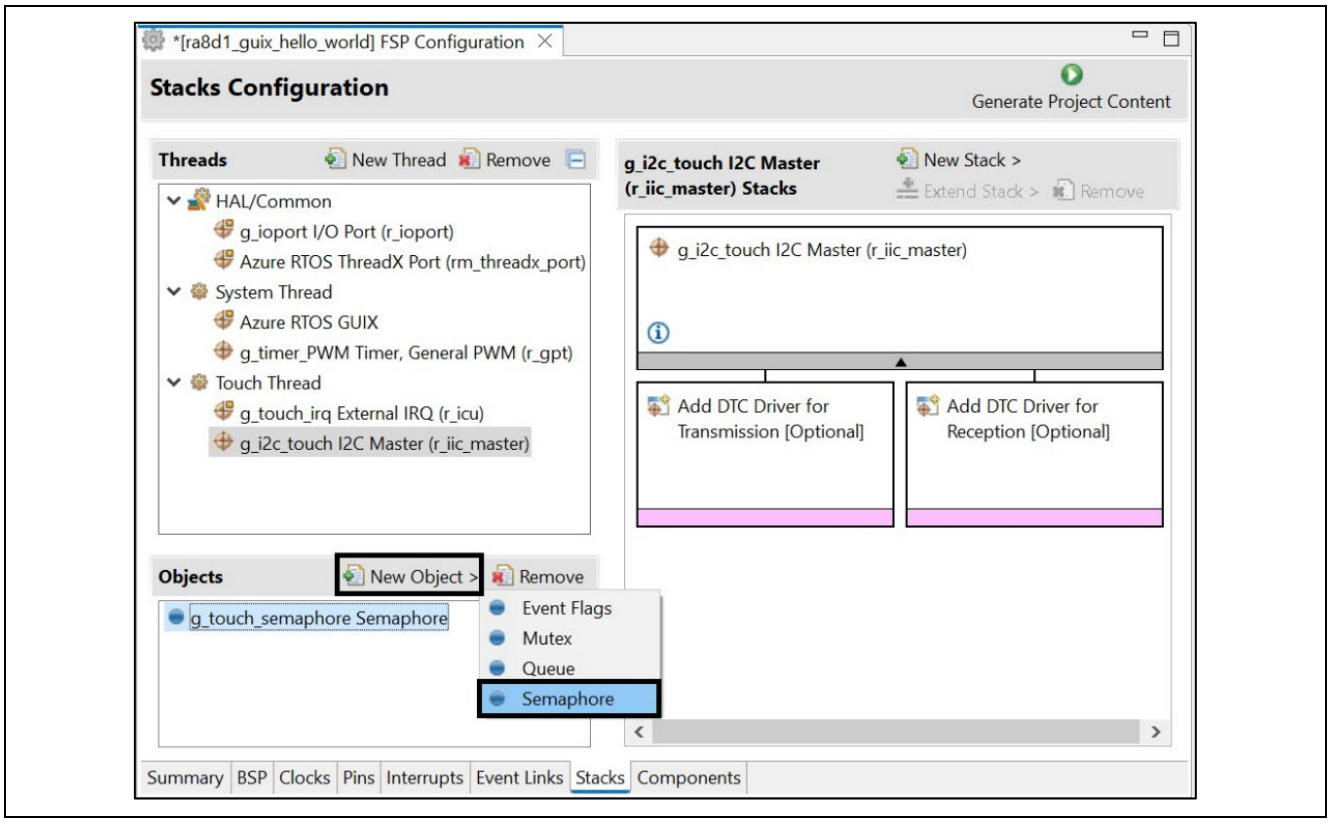


図 27. 別の新規セマフォの追加

10. I2C Semaphore と名前を付け、プロパティを設定します。

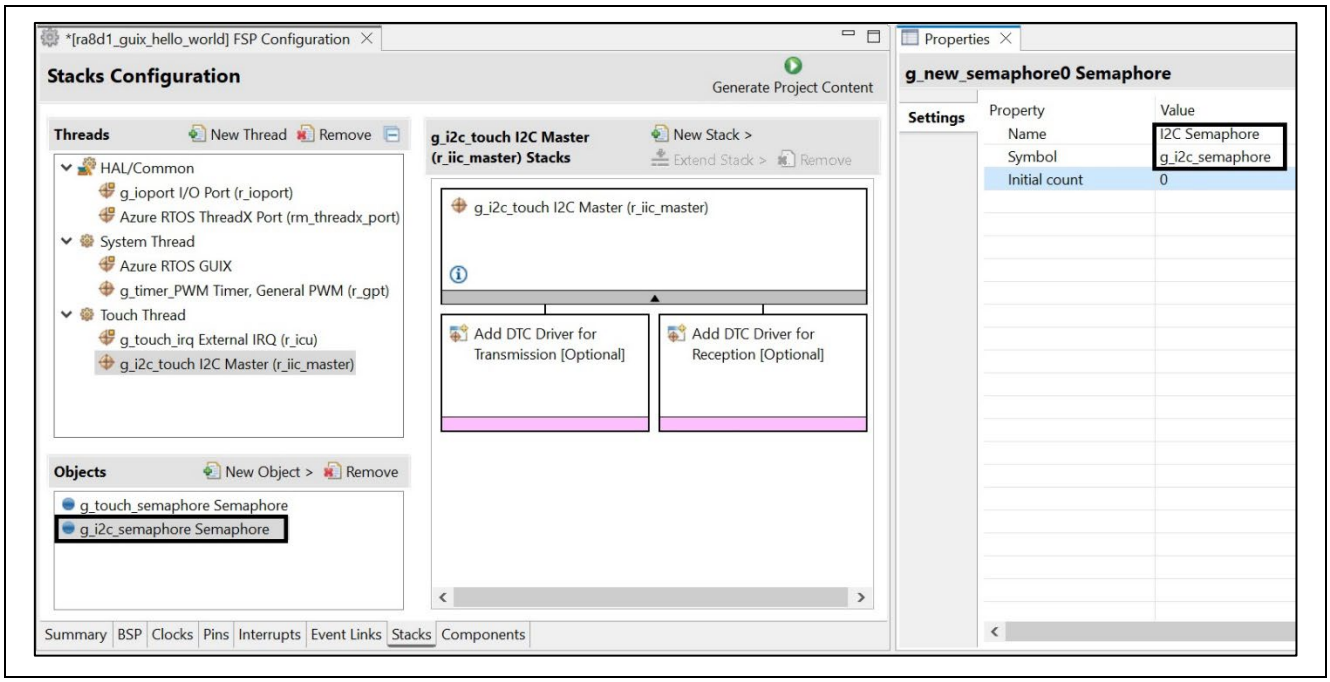


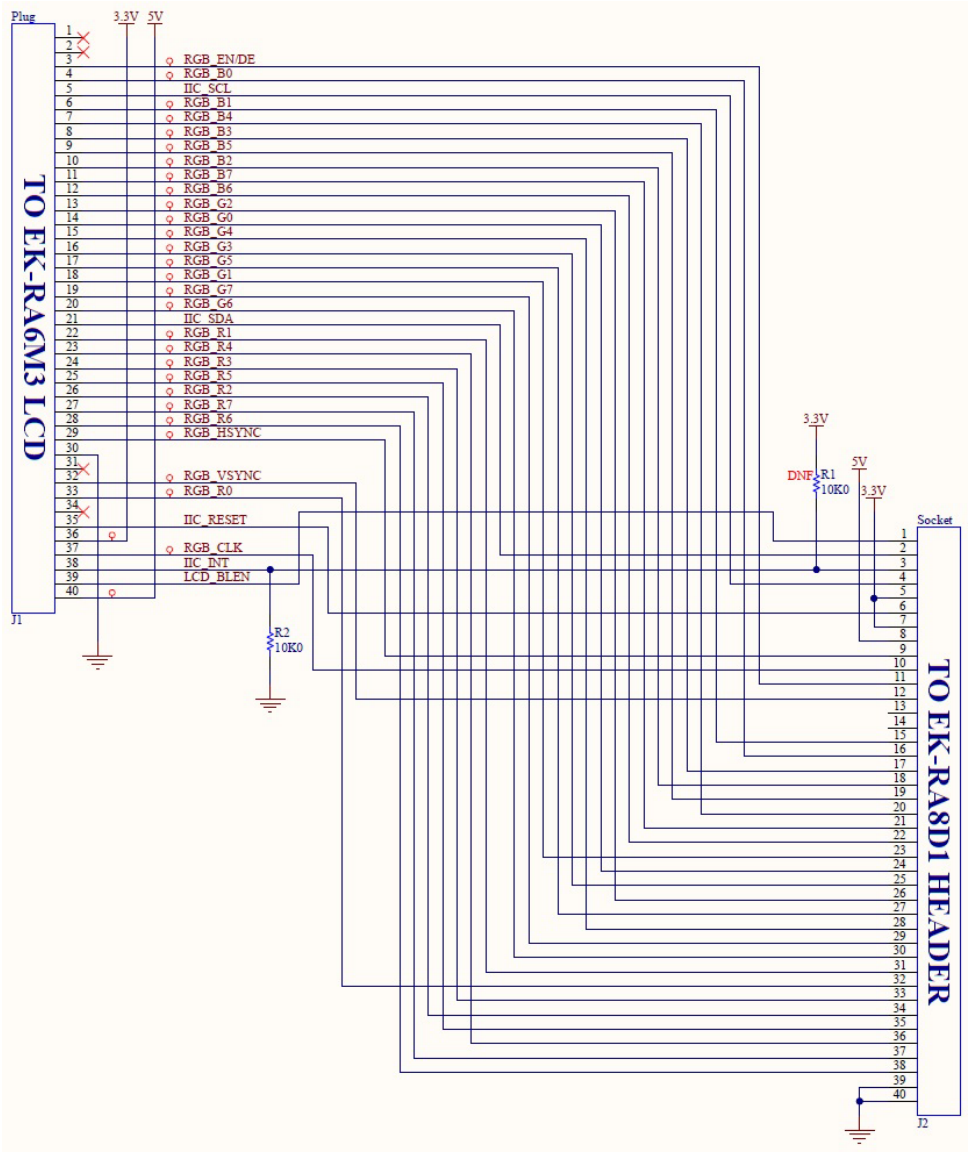
図 28. I2C セマフォの追加と名前設定

11. RA6M3G の LCD ピンコネクタ NC1 を EK-RA8D1 のピンヘッダ J57 に以下のように配線します。

RA6M3 LCD ボードの変更が必要			RA8D1 ボード上のピンコネクタ		
LCD ボード上のピン#	LCD ピン名			RA8D1 ボード上のピン#	RA8D1 ピン名
コネクタ CN1 の変更が必要		NC1			
1	OPEN	1			
2	OPEN	2			
3	DE = TCON2	3	➔	11	LCDC_TCON2
4	RGB_B0	4	➔	16	LCDC_DATA00
5	SCL	5	➔	4	IIC_SCL
6	RGB_B1	6	➔	15	LCDC_DATA01
7	RGB_B4	7	➔	20	LCDC_DATA04
8	RGB_B3	8	➔	17	LCDC_DATA03
9	RGB_B5	9	➔	19	LCDC_DATA05
10	RGB_B2	10	➔	18	LCDC_DATA02
11	RGB_B7	11	➔	21	LCDC_DATA07
12	RGB_B6	12	➔	22	LCDC_DATA06
13	RGB_G2	13	➔	26	LCDC_DATA10
14	RGB_G0	14	➔	24	LCDC_DATA08
15	RGB_G4	15	➔	28	LCDC_DATA12
16	RGB_G3	16	➔	25	LCDC_DATA11
17	RGB_G5	17	➔	27	LCDC_DATA13
18	RGB_G1	18	➔	23	LCDC_DATA09
19	RGB_G7	19	➔	29	LCDC_DATA15
20	RGB_G6	20	➔	30	LCDC_DATA14
21	IIC_SDA	21	➔	2	IIC_SDA
22	RGB_R1	22	➔	31	LCDC_DATA17
23	RGB_R4	23	➔	36	LCDC_DATA20
24	RGB_R3	24	➔	33	LCDC_DATA19
25	RGB_R5	25	➔	35	LCDC_DATA21
26	RGB_R2	26	➔	34	LCDC_DATA18
27	RGB_R7	27	➔	37	LCDC_DATA23
28	RGB_R6	28	➔	38	LCDC_DATA22
29	TCON0 = HSYNC	29	➔	9	LCDC_TCON0
30	GND	30	➔	39, 40	GND
31	OPEN	31			
32	TCON4 = VSYNC	32	➔	12	LCDC_TCON1
33	RGB_R0	33	➔	32	LCDC_DATA16
34	OPEN	34			
35	IIC_RST	35	➔	6	DISP_RST
36	=+V3.3	36	➔	5, 7	=+V3.3
37	RGB_CLK	37	➔	10	CLK
38	IIC_INT	38	➔	3	DISP_INT
39	LCD_BLEN	39	➔	1	DISP_BLEN
40	=+V5.0	40	➔	8	=+V5.0

タッチ機能に関する重要な注意事項: 接続ボードの R1 に 10KΩの抵抗を追加する必要があります。

(40 ピンボードは LCD と J57 ピンコネクタの間に接続)



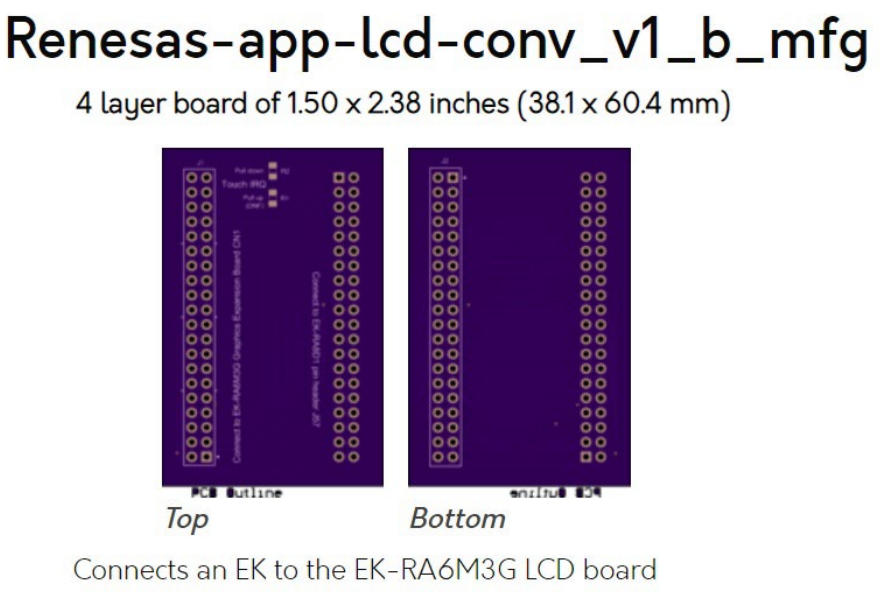


図 29. EK-RA8D1 ボードの LCD と J57 コネクタ間の回路図とボード接続

12. LCD モジュール ピン接続

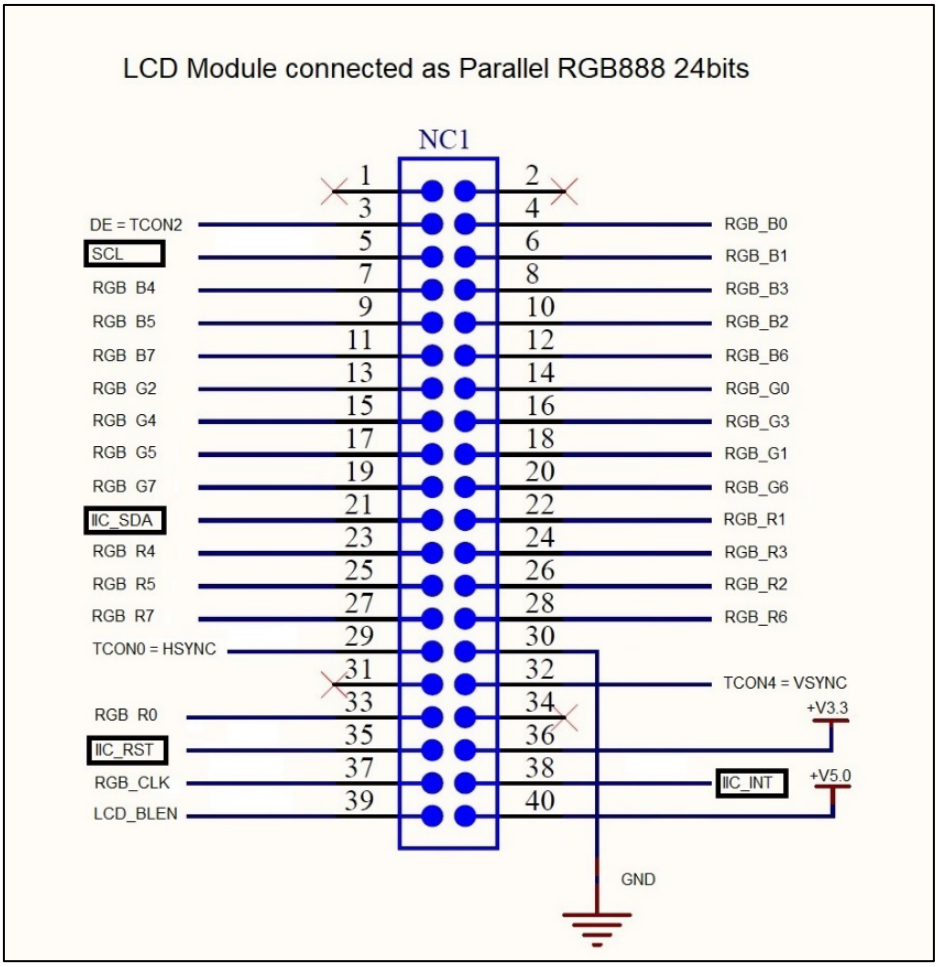


図 30. LCD モジュールの RGB888 24 ビット パラレル接続

上記の黒い四角で示されたピンは、LCD ボード上のタッチパネルコントローラに使用されます。

- DISP_INT 割り込み(P510)はタッチイベントのトリガに使用されます。
- I2C チャンネル 1 (P512、P511) はタッチコントローラへのデータの読み書きに使用されます。
- タッチ機能用のタッチドライバフォルダ touch_ft5x06 は提供のソースフォルダ内にあります。
- PA01 は、LCD のタッチコントローラをリセットするために使用されます。

13. 注:　詳細については、Source.zip の touch_thread_entry.c ファイルを参照してください。
次のコードは、タッチコントローラを初期化し、タッチイベントを処理します。

```
/* Touch Thread entry function */
void touch_thread_entry(void)
{
    GX_EVENT gxe = {0};
    UINT status  = 0;
    /*
     * Initialize Touch controller:
     *  FT5X06 driver
     *  IRQ interrupt
     */
    ft5x06_init(&g_i2c_touch, &g_i2c_semaphore, BSP_IO_PORT_10_PIN_01);
    /* Enable touch IRQ */
    R_ICU_ExternalIrqOpen(g_touch_irq.p_ctrl, g_touch_irq.p_cfg);
    R_ICU_ExternalIrqEnable(g_touch_irq.p_ctrl);

    while (1)
    {
        /* Wait for IRQ from touch controller. */
        status = tx_semaphore_get(&g_touch_semaphore, TX_WAIT_FOREVER);
        if(TX_SUCCESS != status)
        {
            APP_ERR_TRAP(FSP_ERR_ASSERTION);
        }

        /* Get touch data from the FT5X06 */
        ft5x06_payload_get(&touch_data);

        if(0U < touch_data.num_points)
        {
            /* Send only the TOUCH event of the first finger to the GUIX Model View Controller. */
            if( TOUCH_EVENT_DOWN == touch_data.point[0].event)
            {
                gxe.gx_event_payload.gx_event_pointdata.gx_point_x       = touch_data.point[0].x;
                gxe.gx_event_payload.gx_event_pointdata.gx_point_y       = touch_data.point[0].y;
                gxe.gx_event_type = GX_EVENT_PEN_DOWN;
                gx_system_event_send(&gxe);

            }
            else if (TOUCH_EVENT_UP == touch_data.point[0].event && GX_EVENT_PEN_DOWN == gxe.gx_event_type)
            {
                gxe.gx_event_payload.gx_event_pointdata.gx_point_x       = touch_data.point[0].x;
                gxe.gx_event_payload.gx_event_pointdata.gx_point_y       = touch_data.point[0].y;
                gxe.gx_event_type = GX_EVENT_PEN_UP;
                gx_system_event_send(&gxe);
            }
```

図 31.　タッチコントローラの初期化とタッチイベント処理

14. Stacks Configuration から **Generate Project Content** をクリックしてプロジェクトコンテンツを生成します。

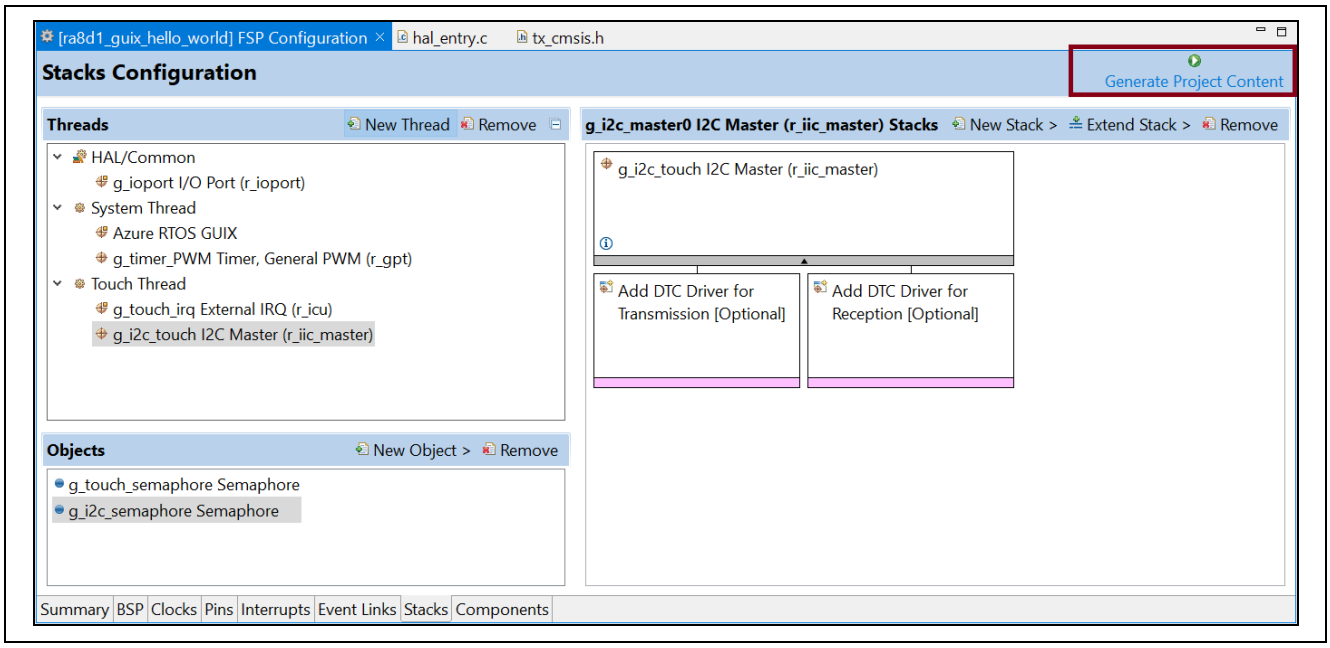


図 32.　プロジェクトコンテンツの生成

15. 提供のフォルダ `Source.zip` を解凍します。touch_ft5x06 フォルダと 4 つの`*.c` ファイルをコピーし、プロジェクト **ra8d1_guix_hello_world** のフォルダ `src` に貼り付けます。
    - Touch_ft5x06 フォルダ
    - hal_entry.c
    - system_thread_entry.c
    - touch_thread_entry.c
    - windows_handler.c

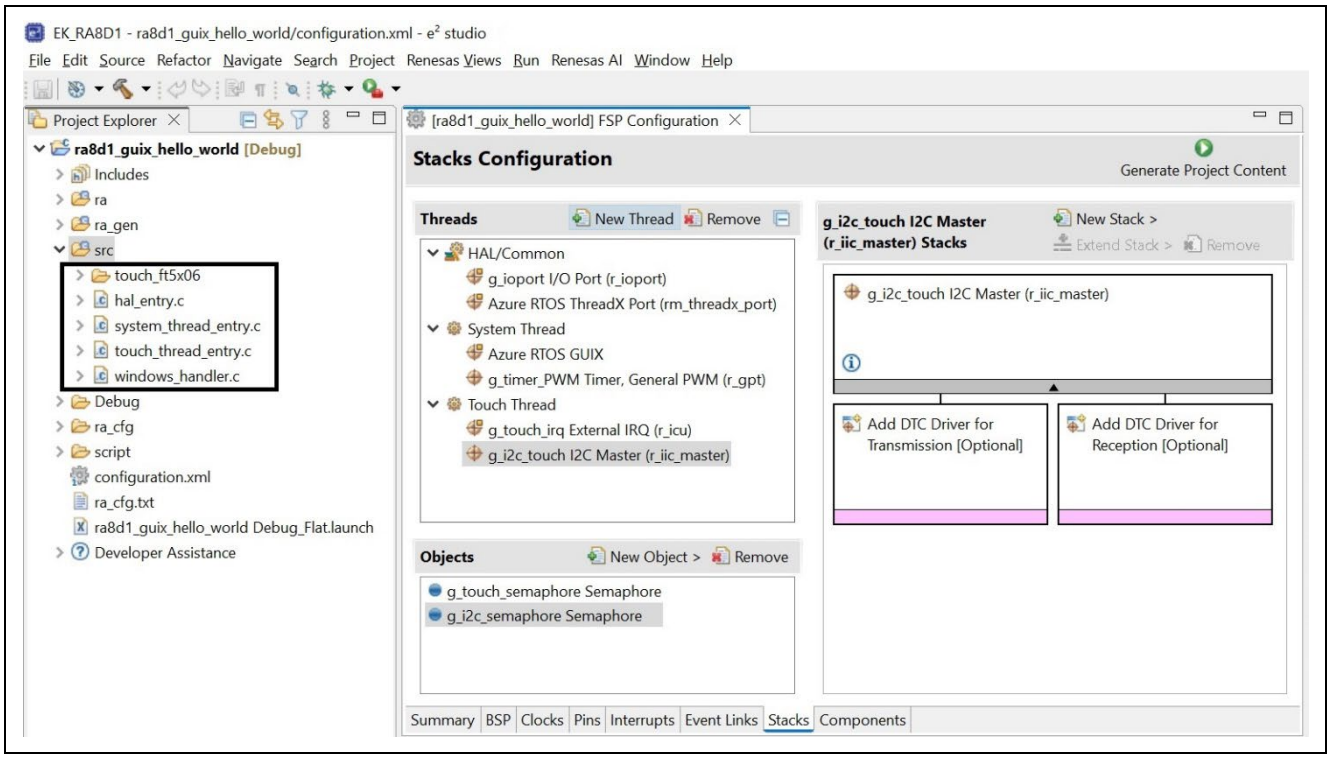
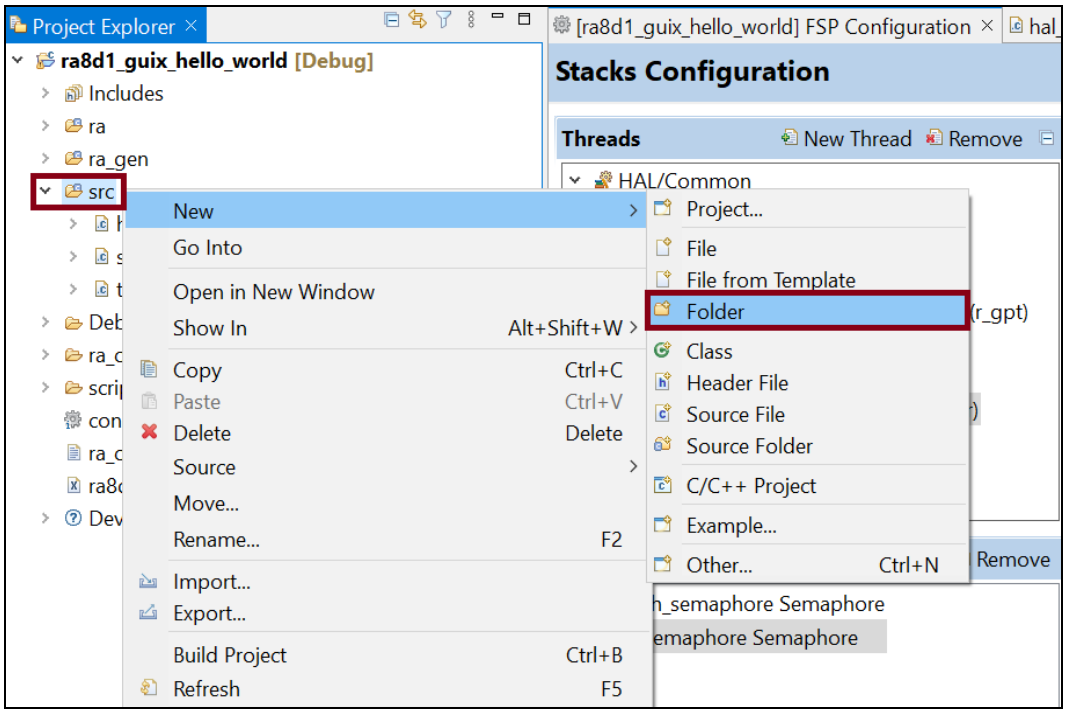


図 33.　フォルダとファイルのコピー

4. Azure RTOS GUIX Studio プロジェクトの Hello_World GUIX_EK_RA8D1 プロジェクトでのフォルダ作成

1. フォルダ src の下に新規フォルダを作成し、guix_gen と名前を付けます。
以下の画像に従って、Finish をクリックします。



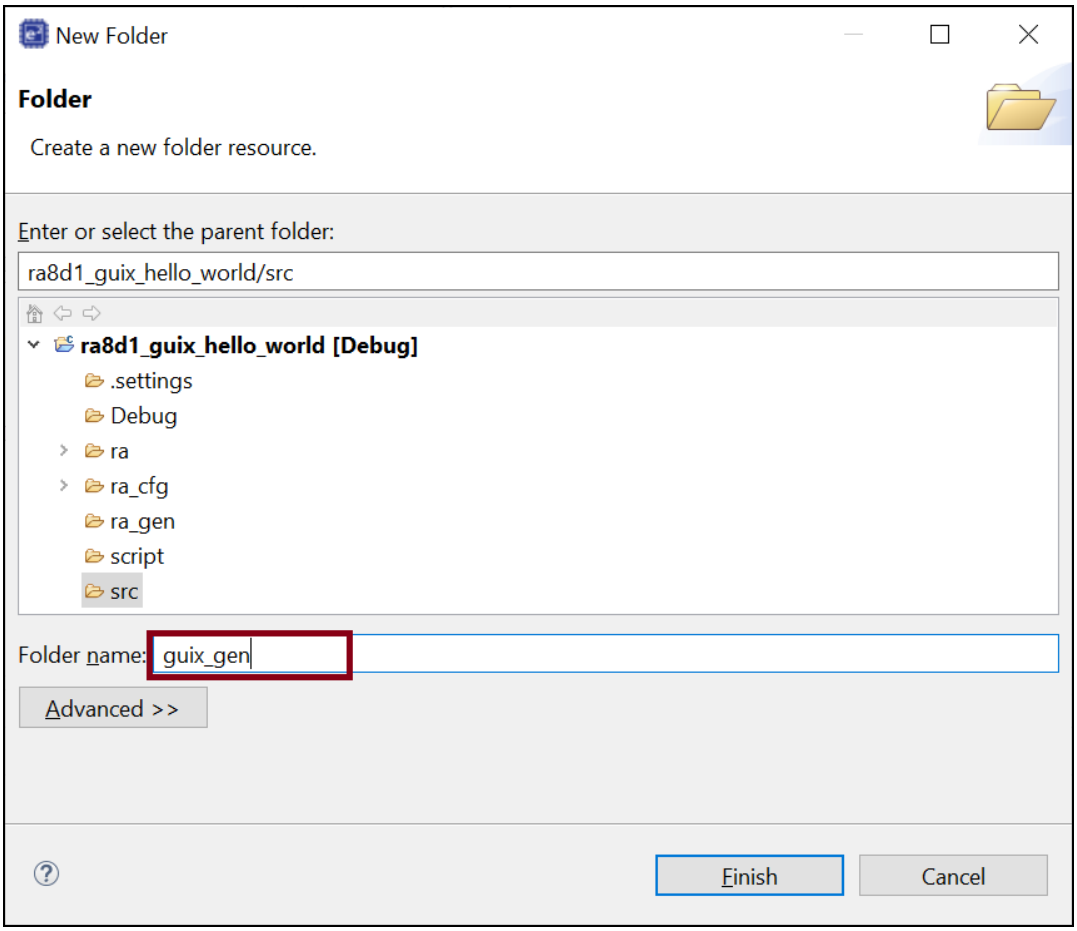


図 34. フォルダ guix_gen の作成

2. フォルダ src の下に新規フォルダを作成し、guix_studio と名前を付けます。
以下の画像に従って、Finish をクリックします。

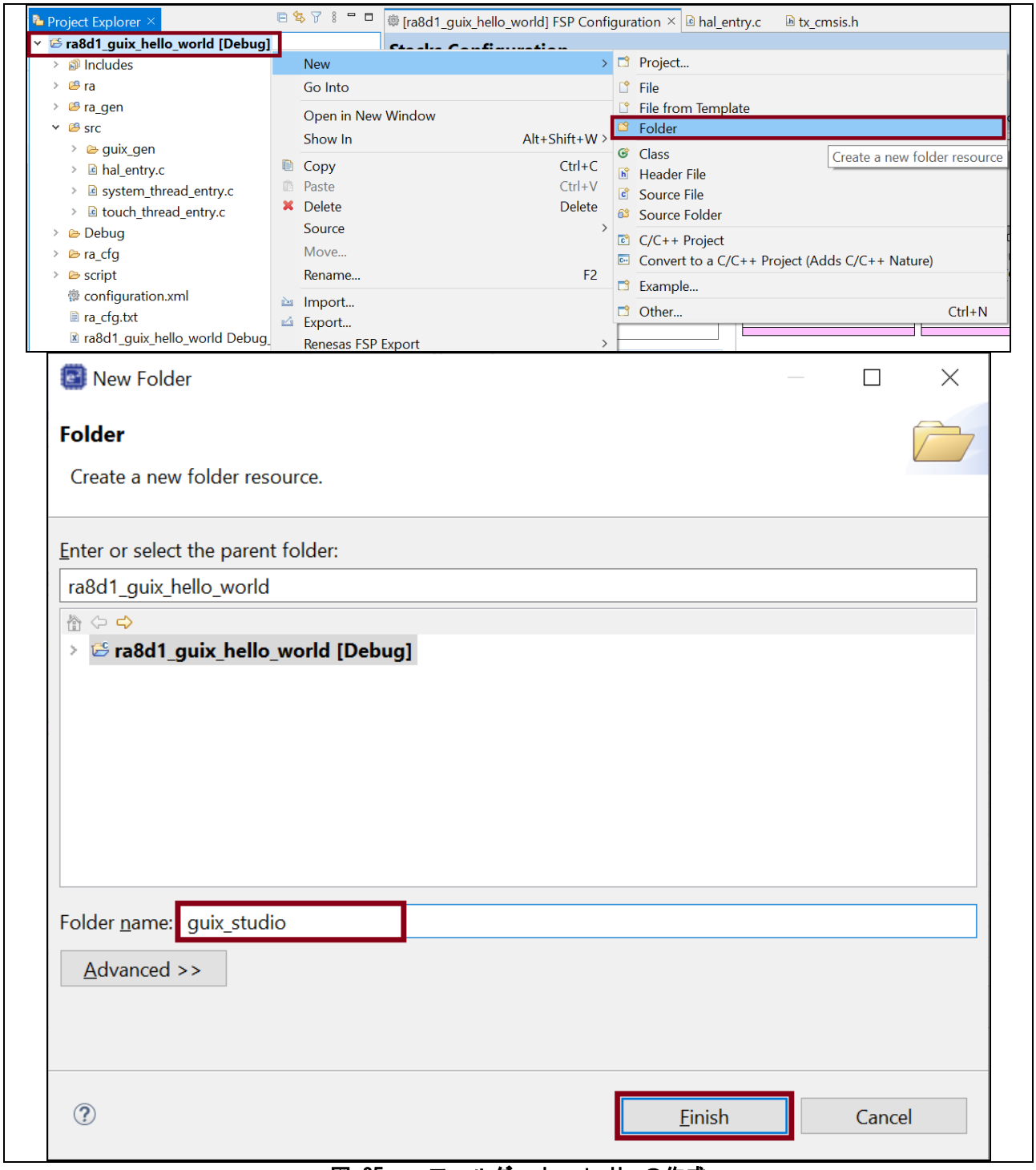


図 35. フォルダ guix_studio の作成

2. 作成した guix_studio フォルダの下に新規フォルダを作成し、GNU と名前を付けます。
以下の画像に従って Finish をクリックします。

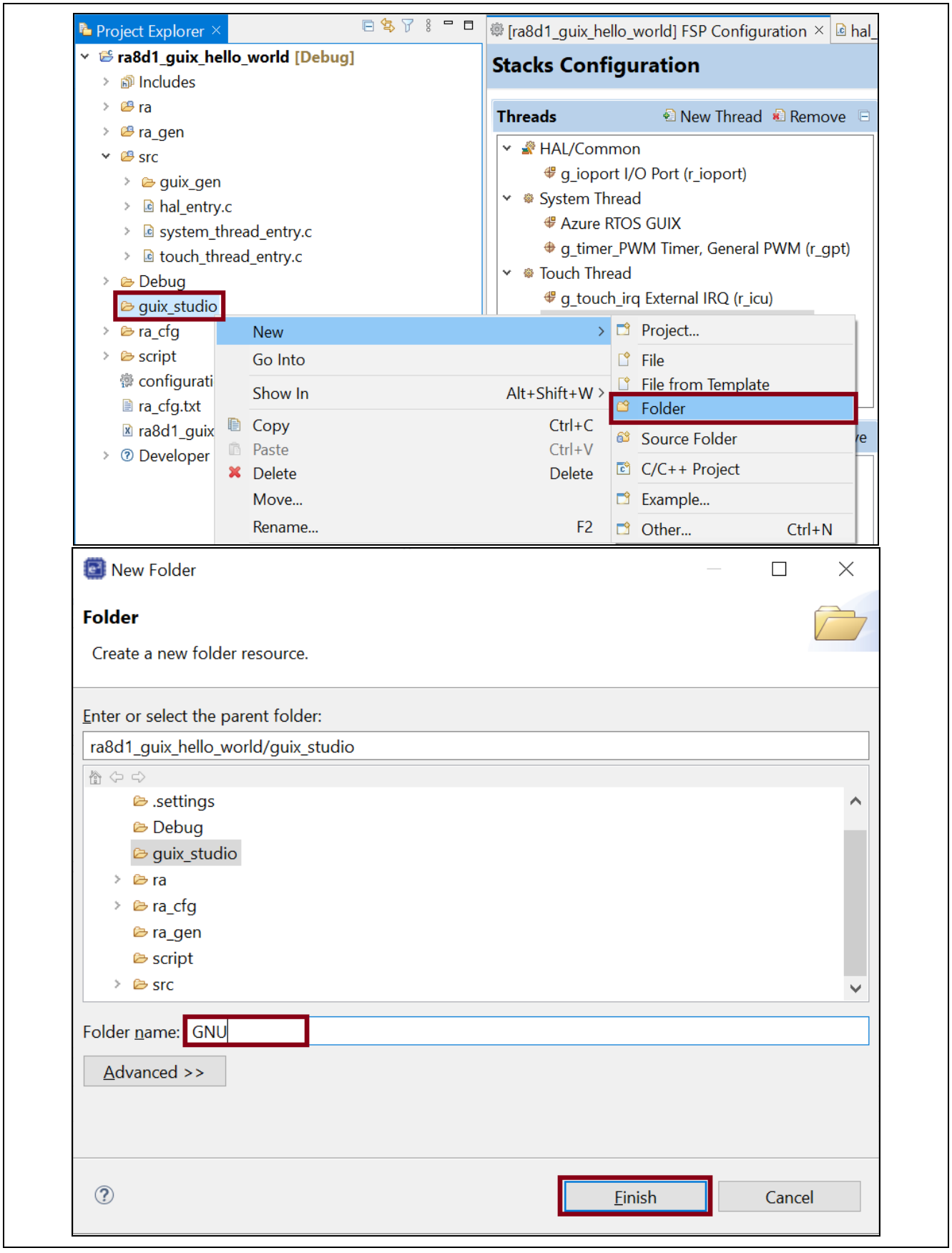


図 36. フォルダ GNU の作成

4. サブフォルダ **GNU** が作成されると、フォルダ構造は以下の画像のようになります。

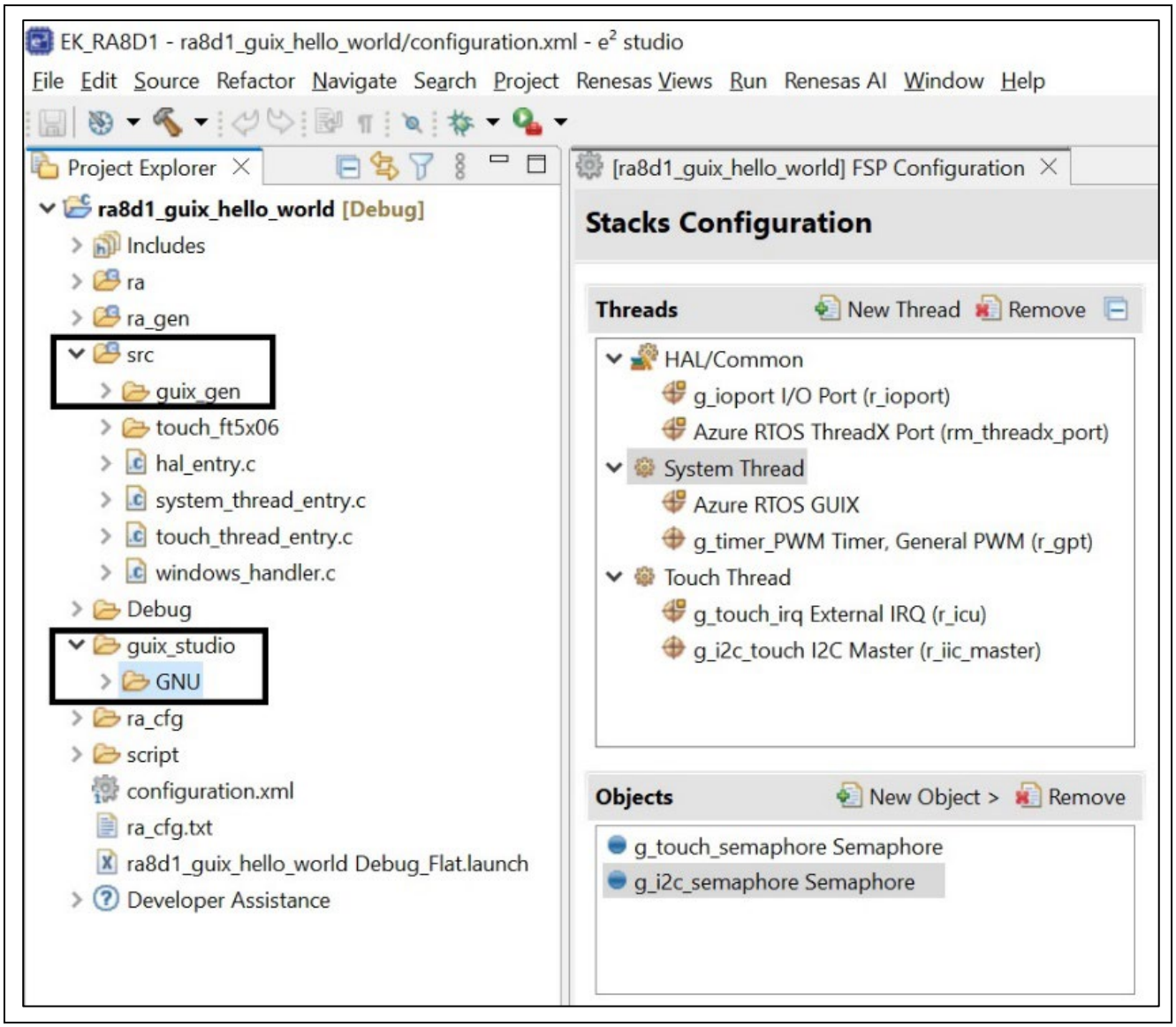


図 37. 新規フォルダのプロジェクト

5. Azure RTOS GUIX Studio を使用して GUI ウィンドウを作成する

1. Azure RTOS GUIX Studio v6.2.1.0 以降を開きます。

図 38. GUIX Studio アイコン

2. **新規プロジェクト**を作成し、**Hello World** という名前を付けます。
   A. **Project** を選択します。ドロップダウンリストから **New Project** を選択します。
   B. プロジェクト名:**Hello_World**
   C. プロジェクトパス:**ra8d1_guix_hello_world**>で作成したフォルダの場所を参照します。
   D. **guix_studio\GNU** を以下の画像のように指定します。
   E. **OK** ボタンを押し、**Save** ボタンを押して選択を確定します。

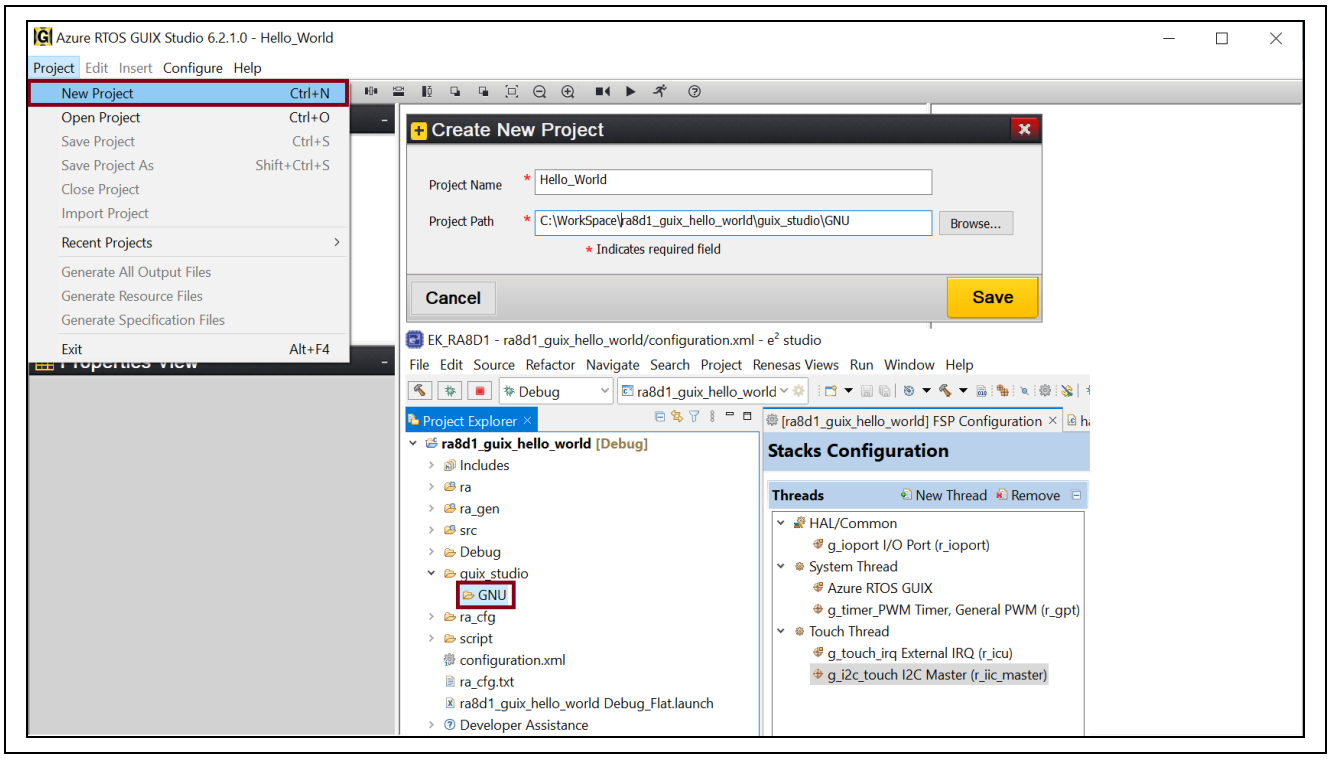


図 39. GUIX Studio で Hello World プロジェクト作成

3. 新規 Configure Project ウィンドウがポップアップ表示され、ユーザは図 40 に示すように、詳細設定を含むすべてのオプションを設定する必要があります。最後に、Save をクリックします。

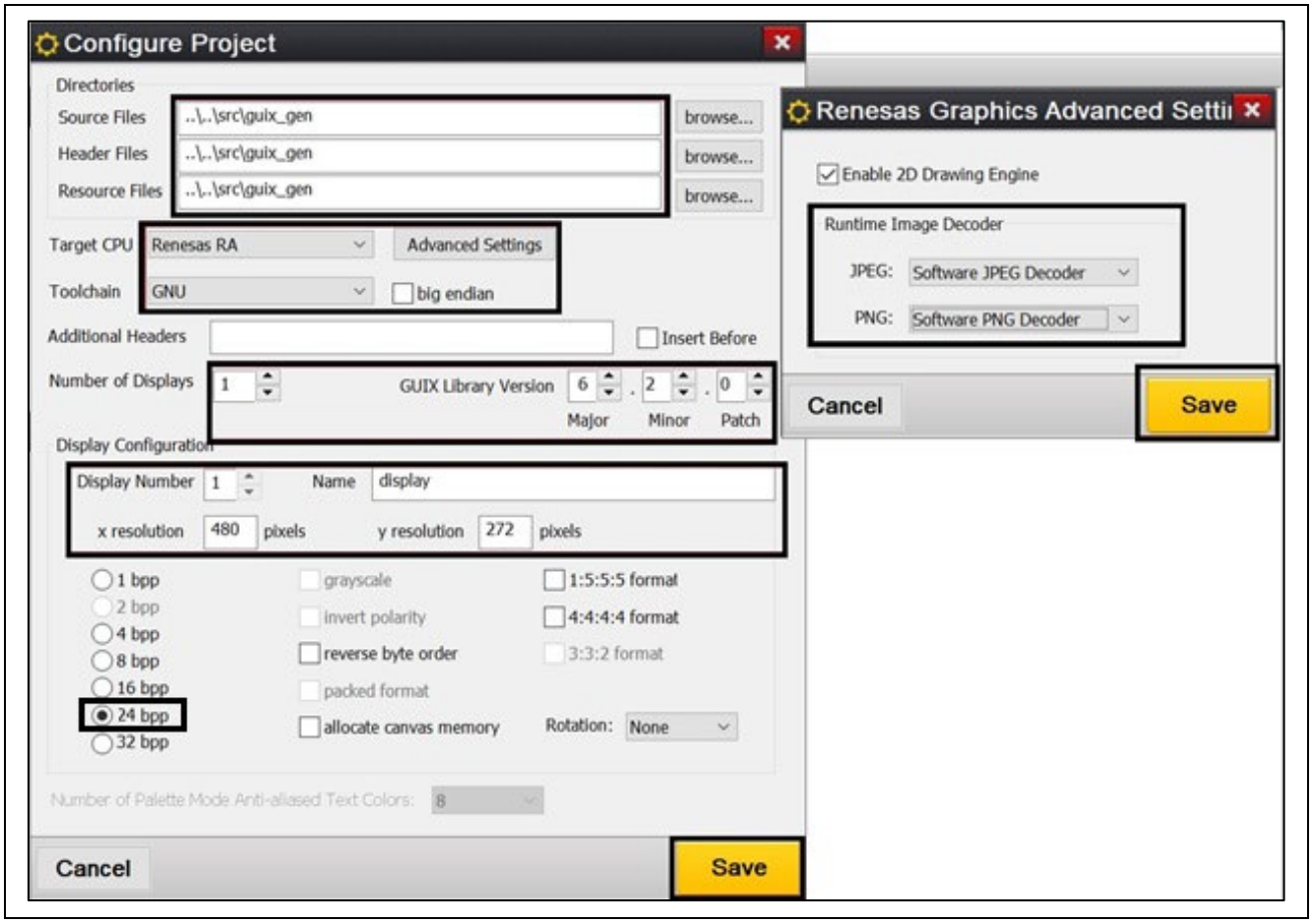


図 40. 新規プロジェクト Hello World 詳細設定

4. 新規プロジェクト Hello World を開始すると図 41 のようになります。

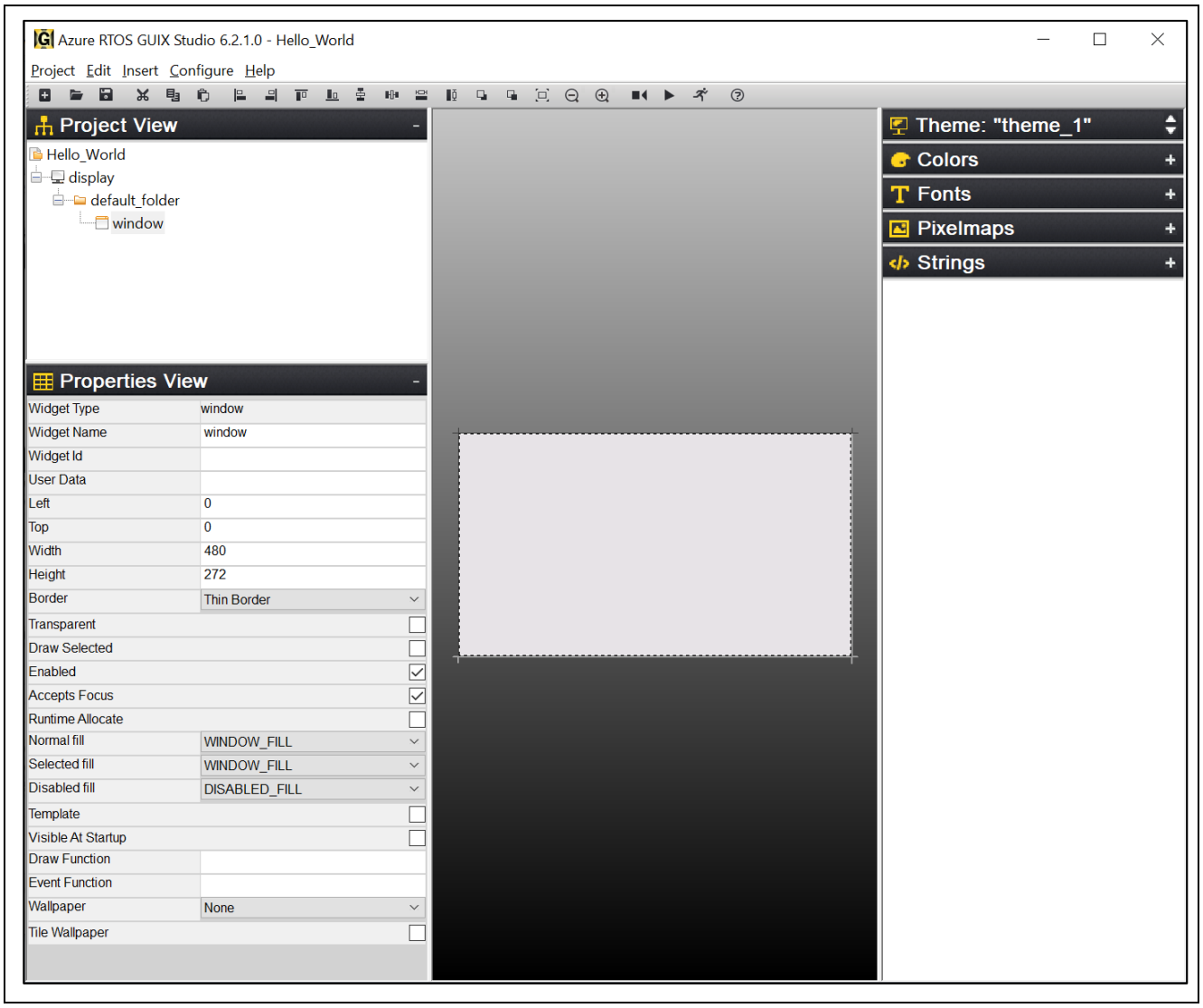


図 41. 新規プロジェクト Hello World 設定後

5. Window1 のプロパティの設定をします。

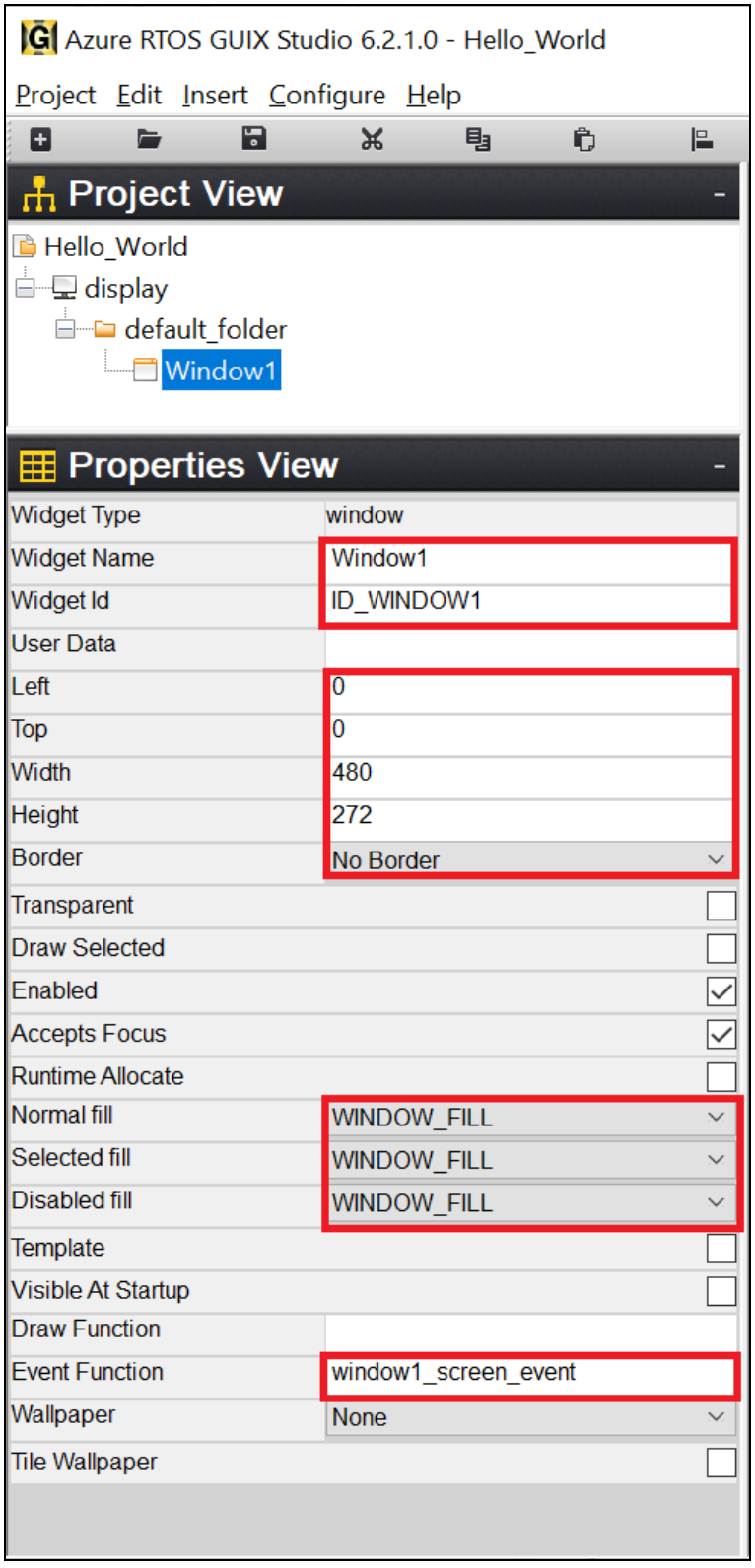


図 42.　Window1 のプロパティ設定

6. String ID を追加するには、**Strings** をクリックします。以下の画像に従ってください。

図 43. String ID の追加手順

7. **String** ドロップダウンから、**+ Add New String** クリックします。

図 44. 新規 String の追加

8. 新規 String Table Editor ウィンドウがポップアップします。Add String ボタンをクリックします。

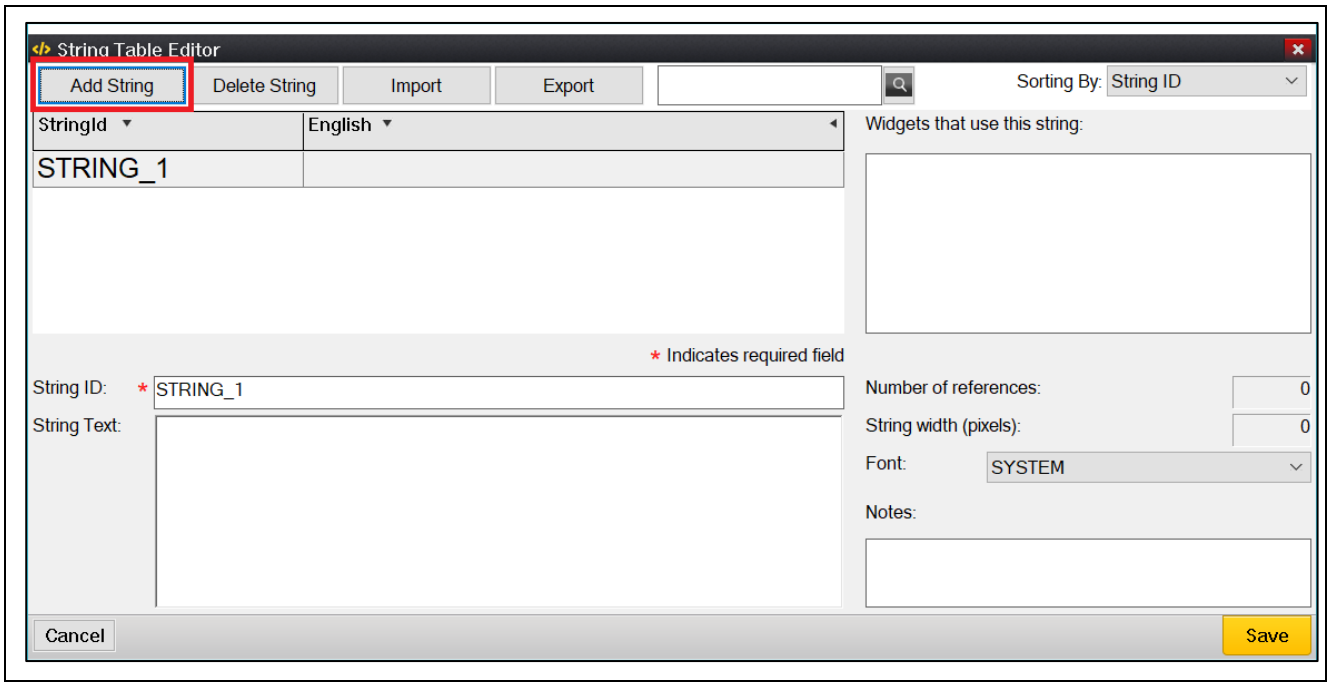


図 45. String Table Editor

9. String ID と String Text を編集します。

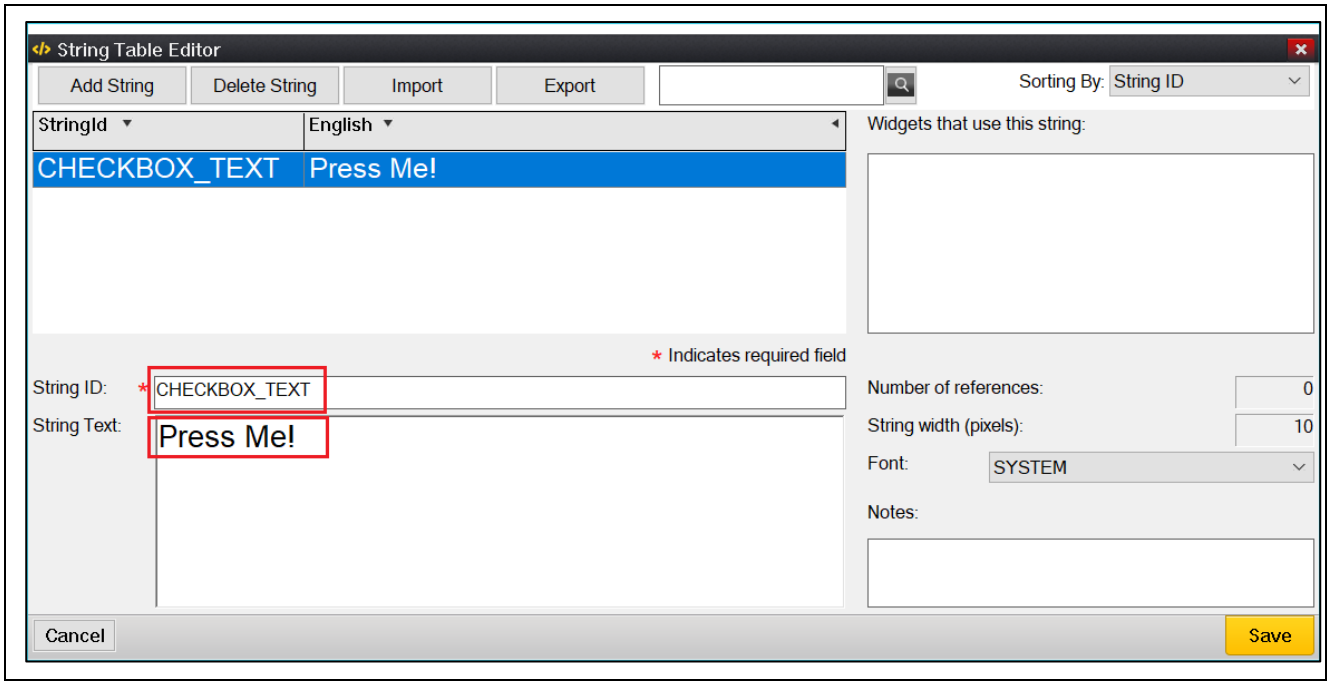


図 46. String ID と String Text の編集

10. 続けて **Add String** をクリックし、表が**図 47** のように表示されるまで **String ID** と **String Text** を編集します。次に、**Save** ボタンをクリックします。

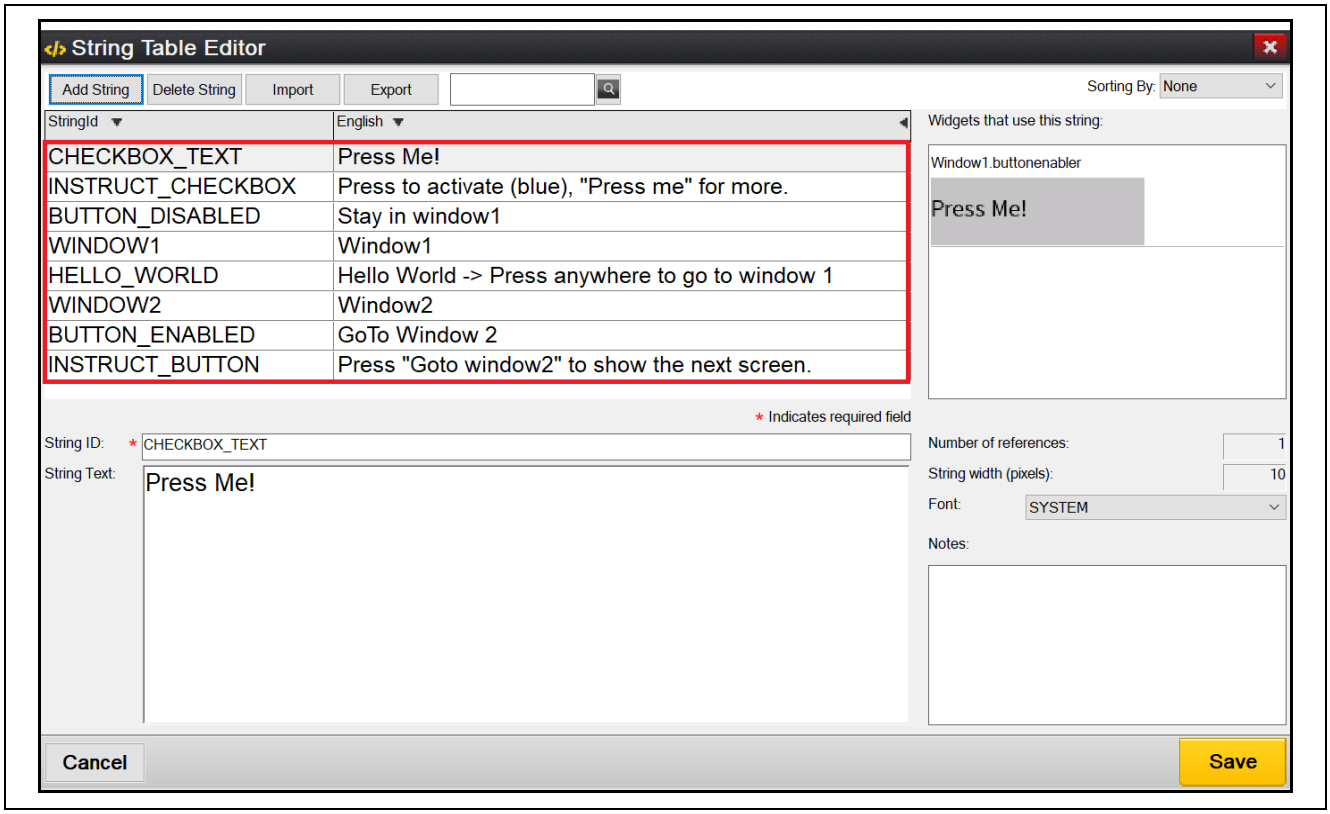


図 47.　String ID と String Text

11. 図 48 のように、**Window1** を右クリックし **Text Button** を挿入します。

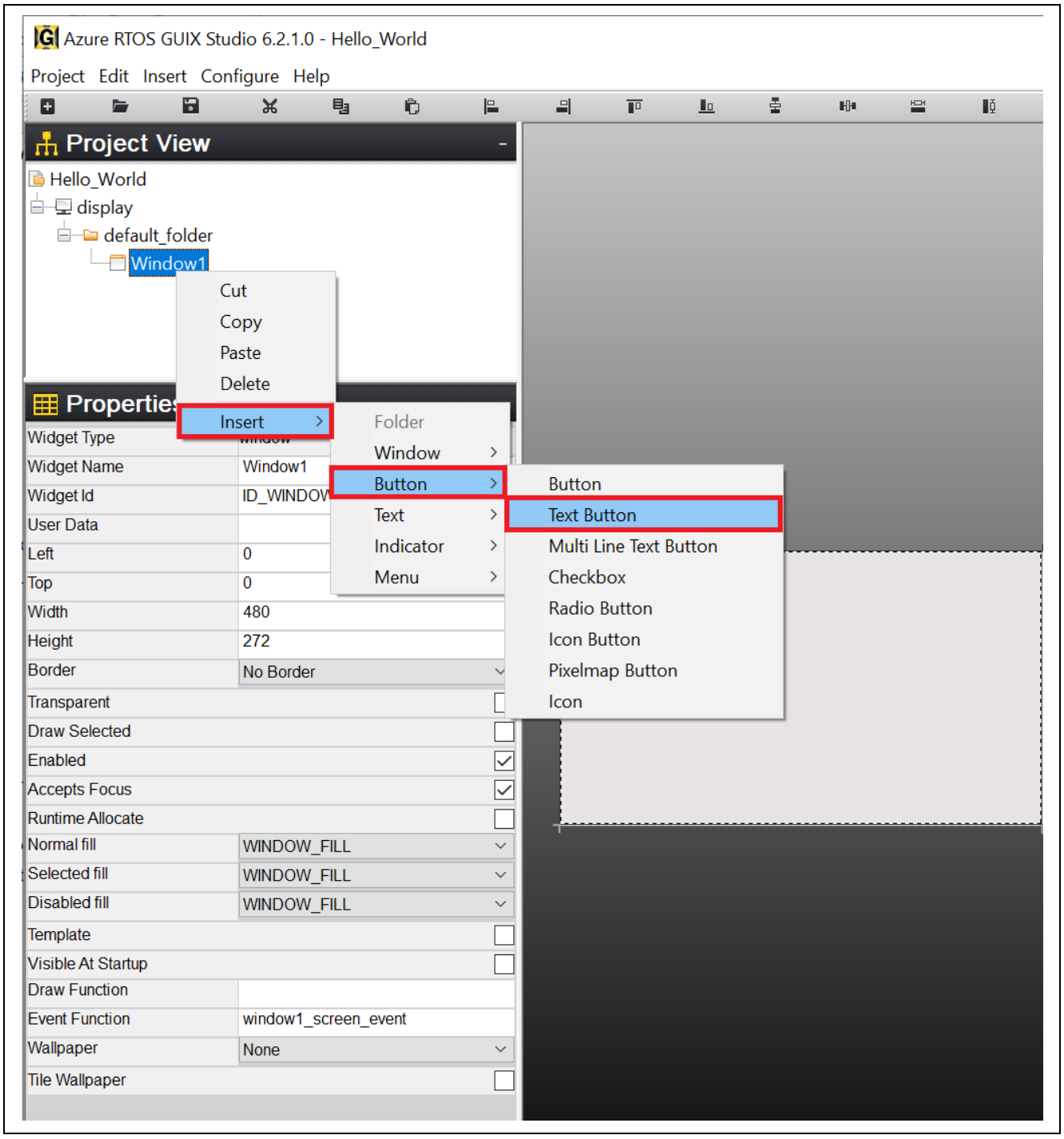


図 48. Text Button の挿入

12. text_button のプロパティを設定します。

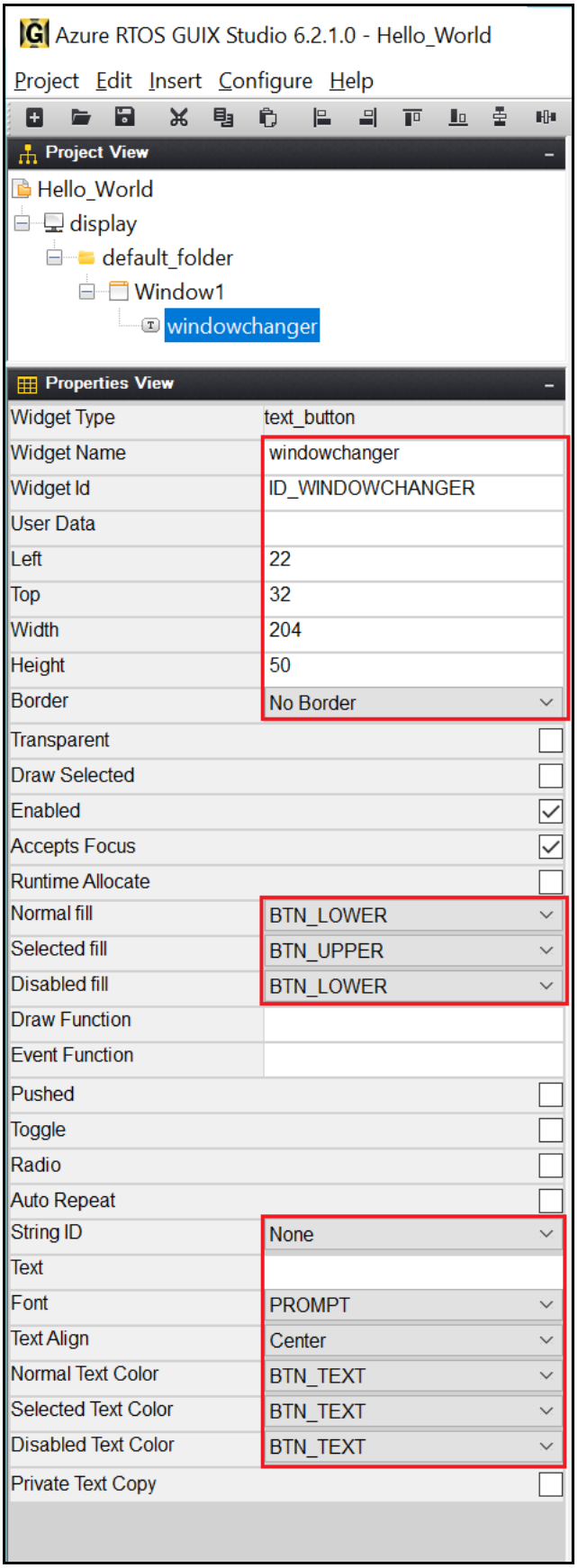


図 49. text_button のプロパティの設定

13. 図 50 のように、**windowchanger** を右クリックして **Prompt** を挿入します。

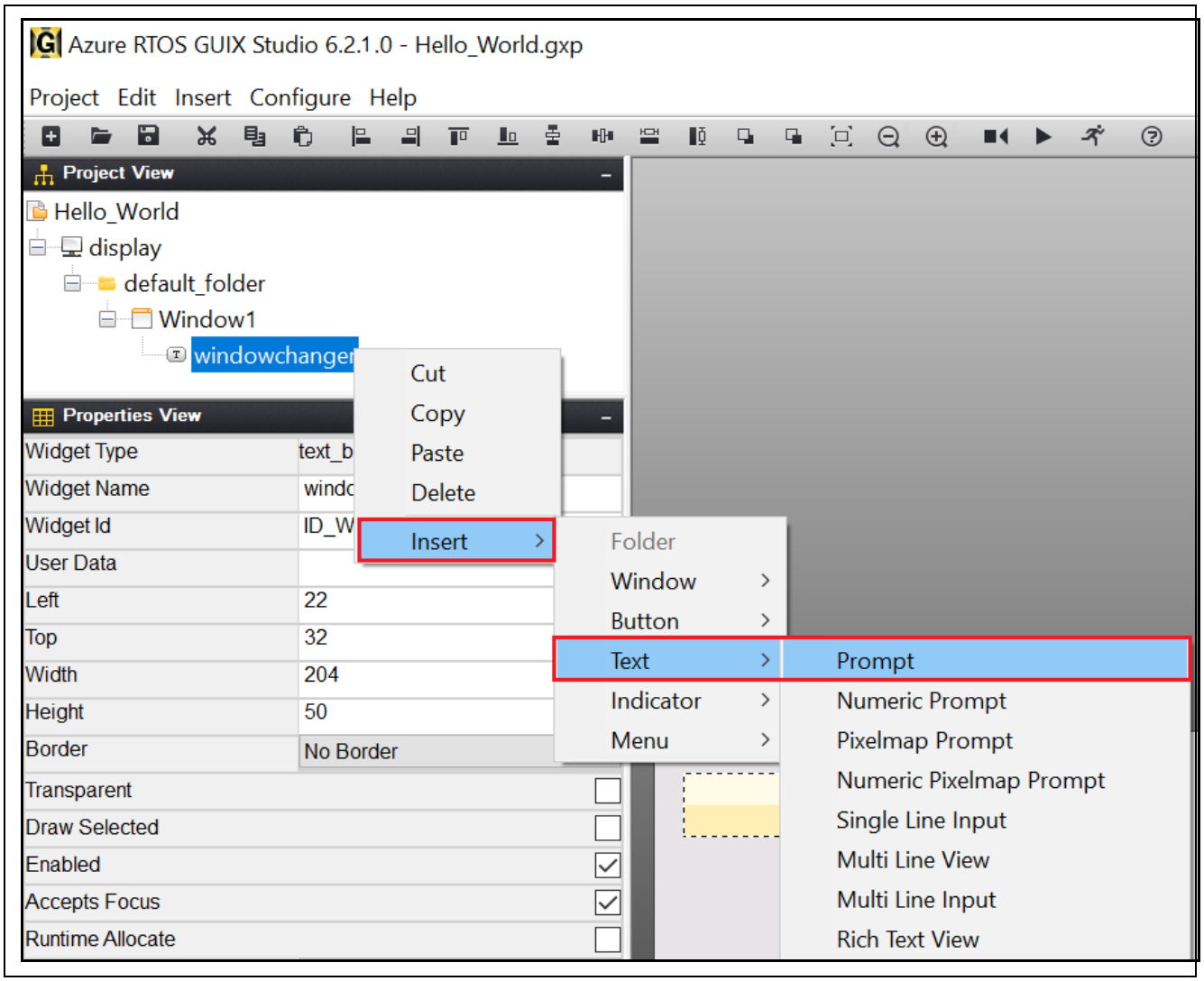


図 50. Prompt の挿入

14. Prompt のプロパティを設定します。

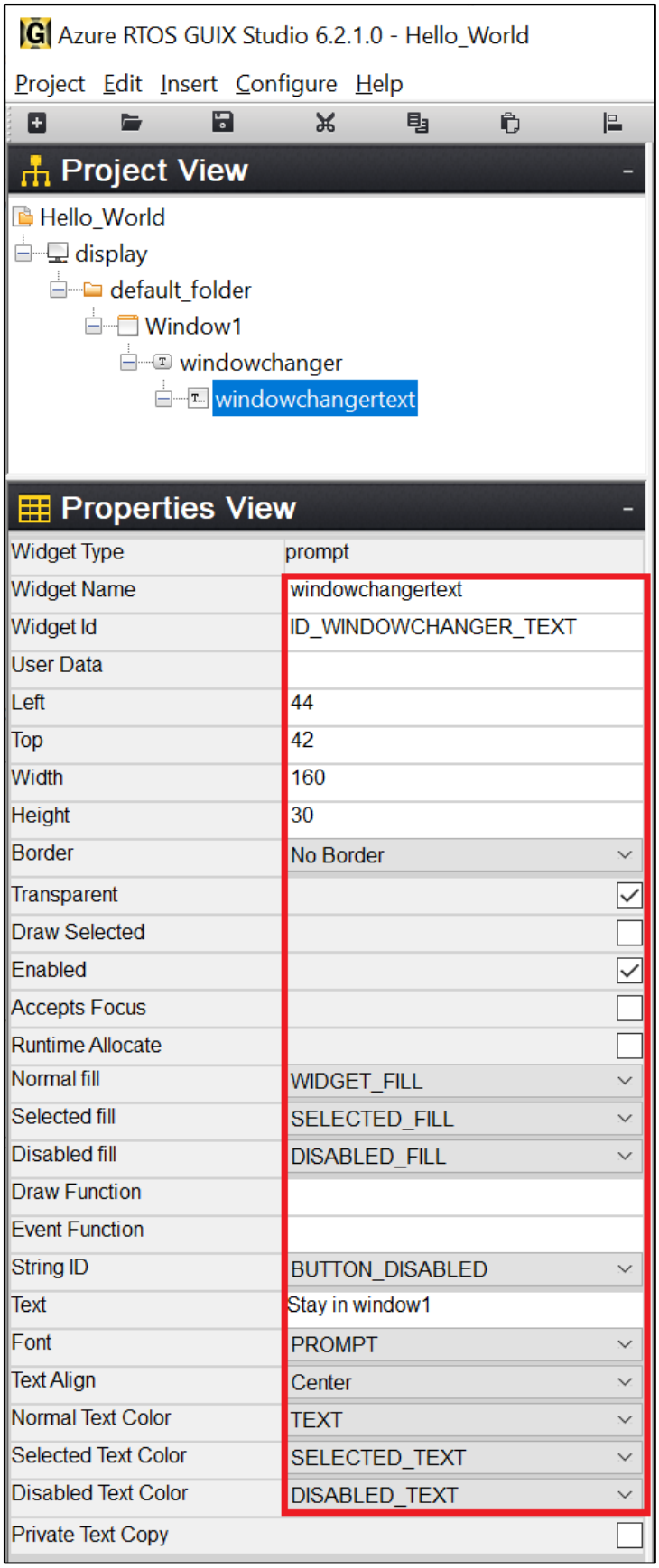


図 51. Prompt のプロパティの設定

15. 図 52 のように、**windowchangertext** を右クリックして新規 **Button** を挿入します。

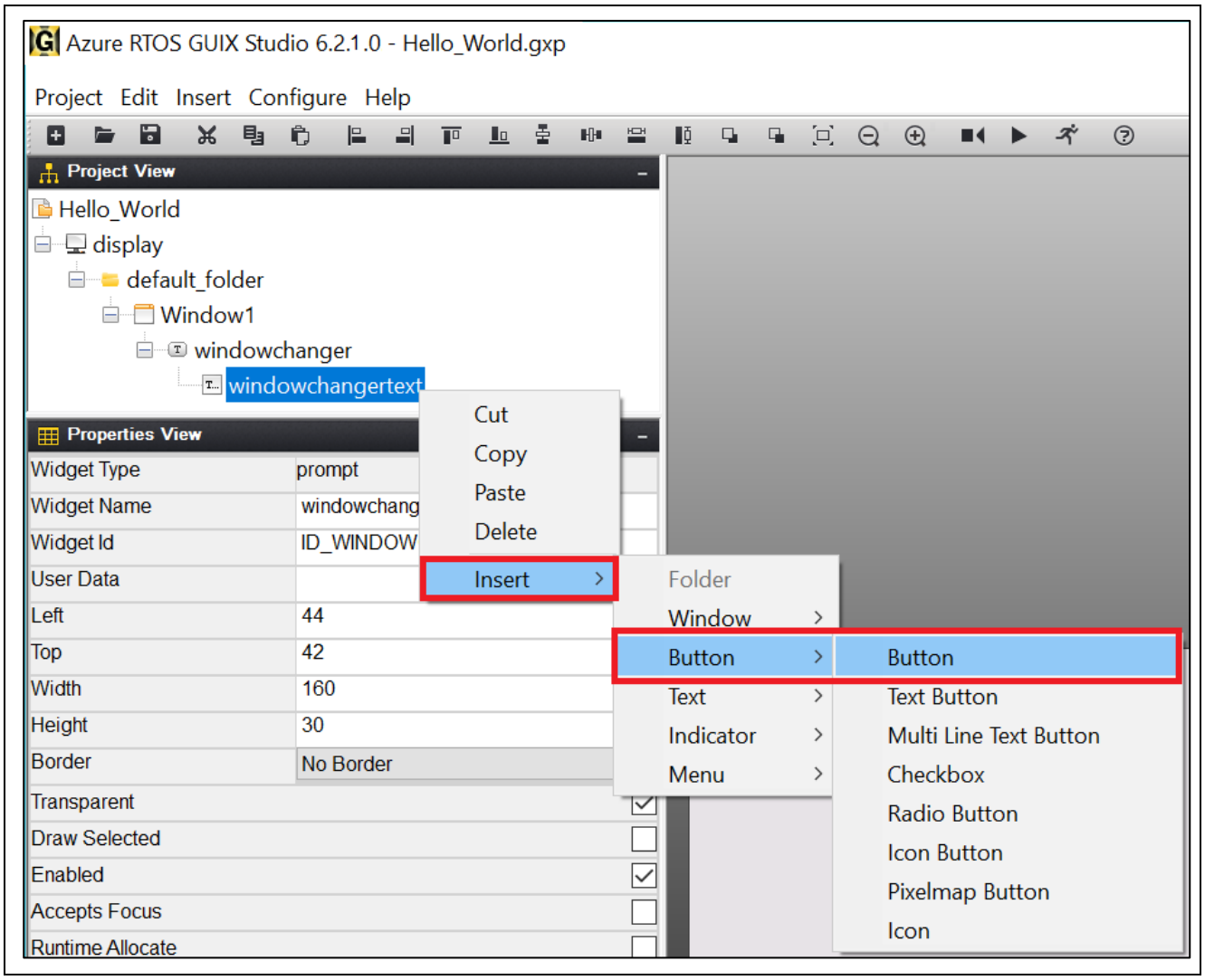


図 52. Button の挿入

16. button のプロパティを設定します。

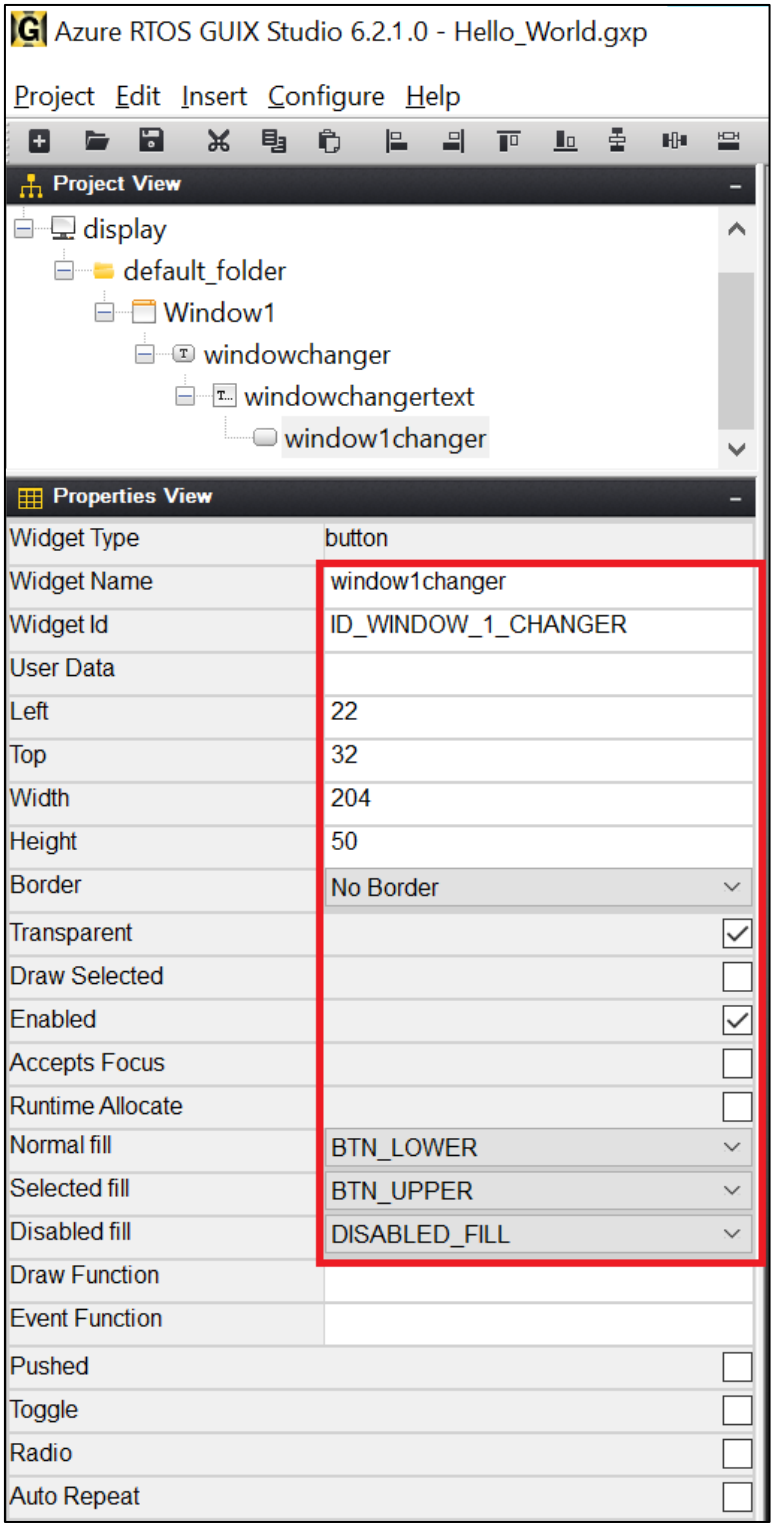
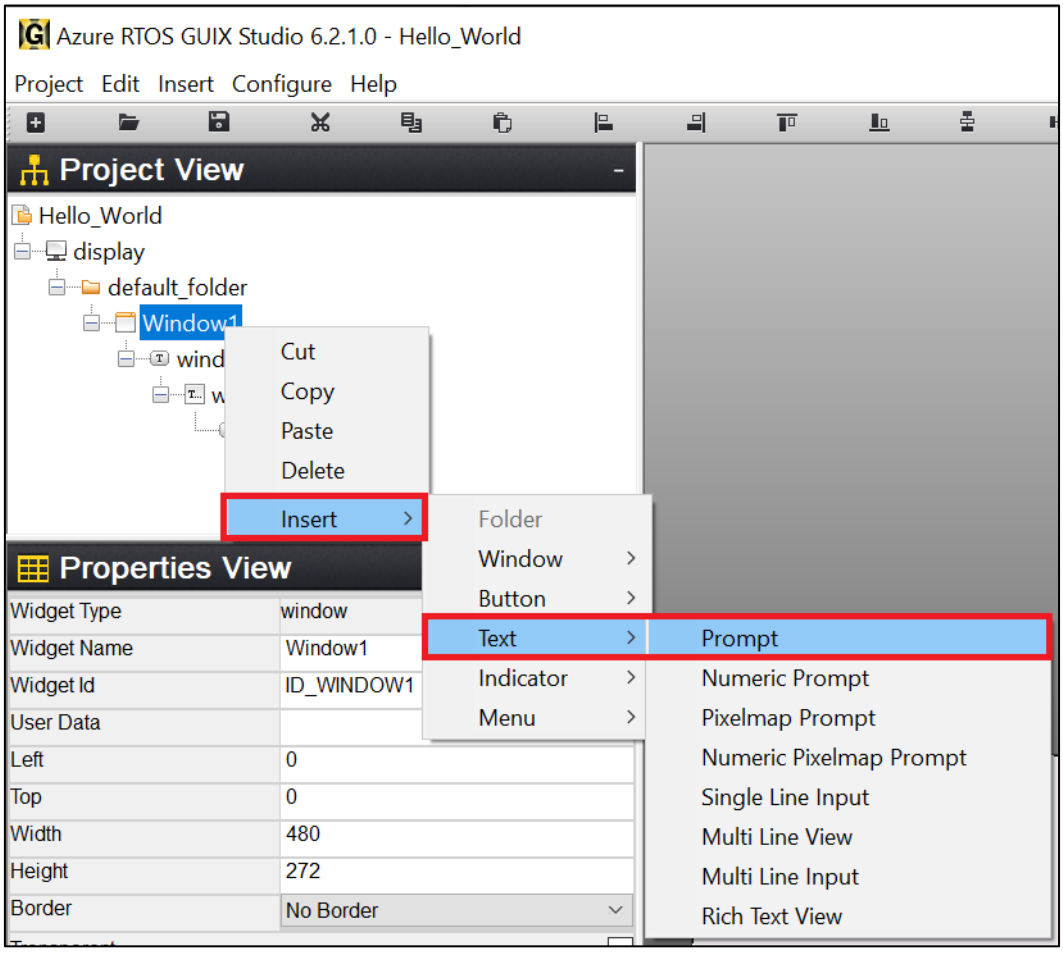


図 53. button のプロパティ設定

17. 図 54 のように、Window1 を右クリックし、Prompt を挿入します。2 回挿入を行うと、Prompt、Prompt.1 のように２つの Prompt が表示されます。



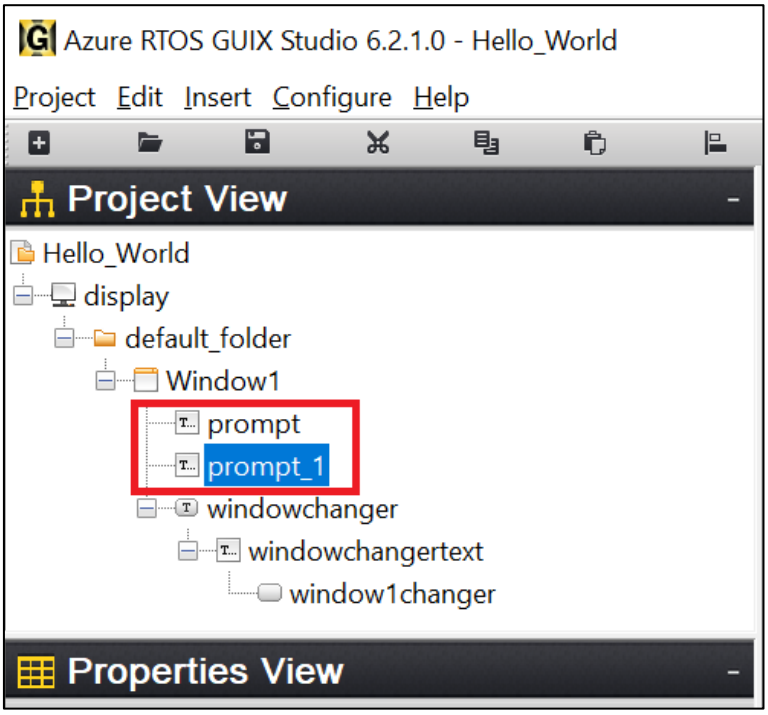


図 54. Prompt の挿入

18. Prompt のプロパティを設定します。

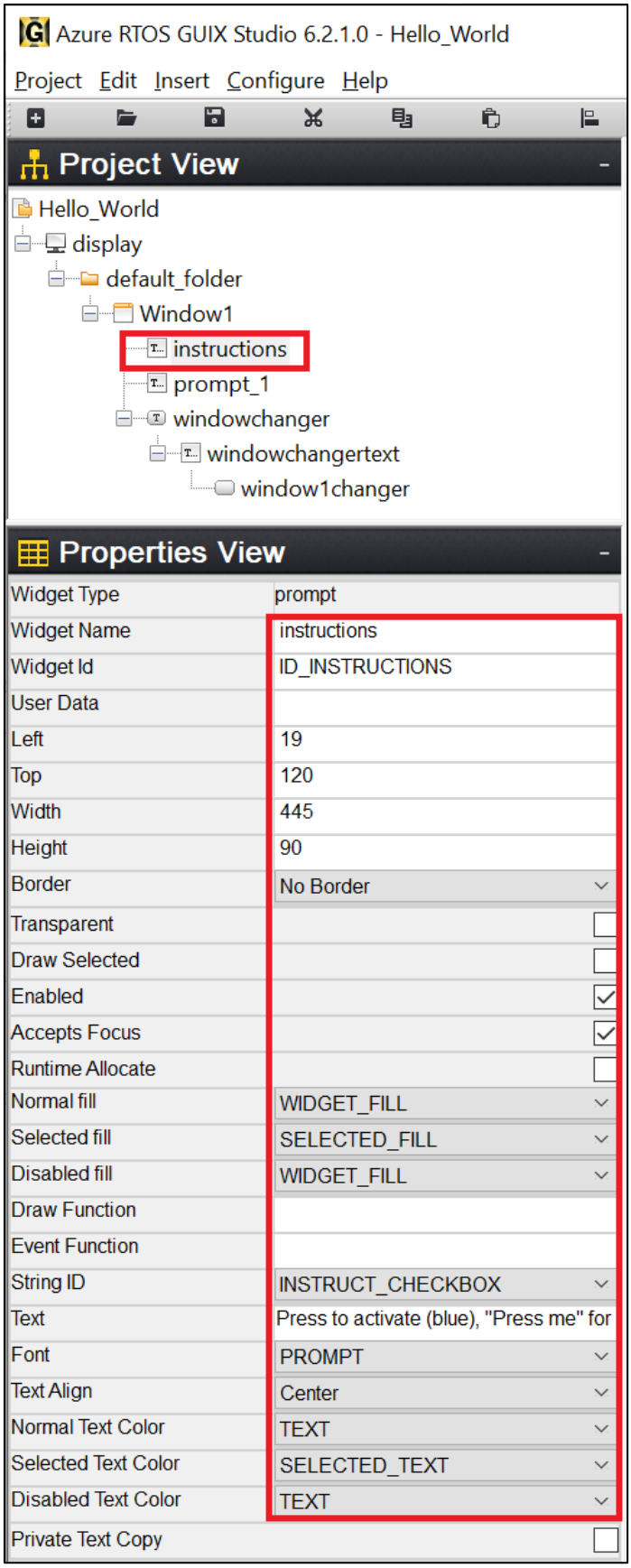


図 55. Prompt のプロパティ設定

19. Prompt1 のプロパティを設定します。

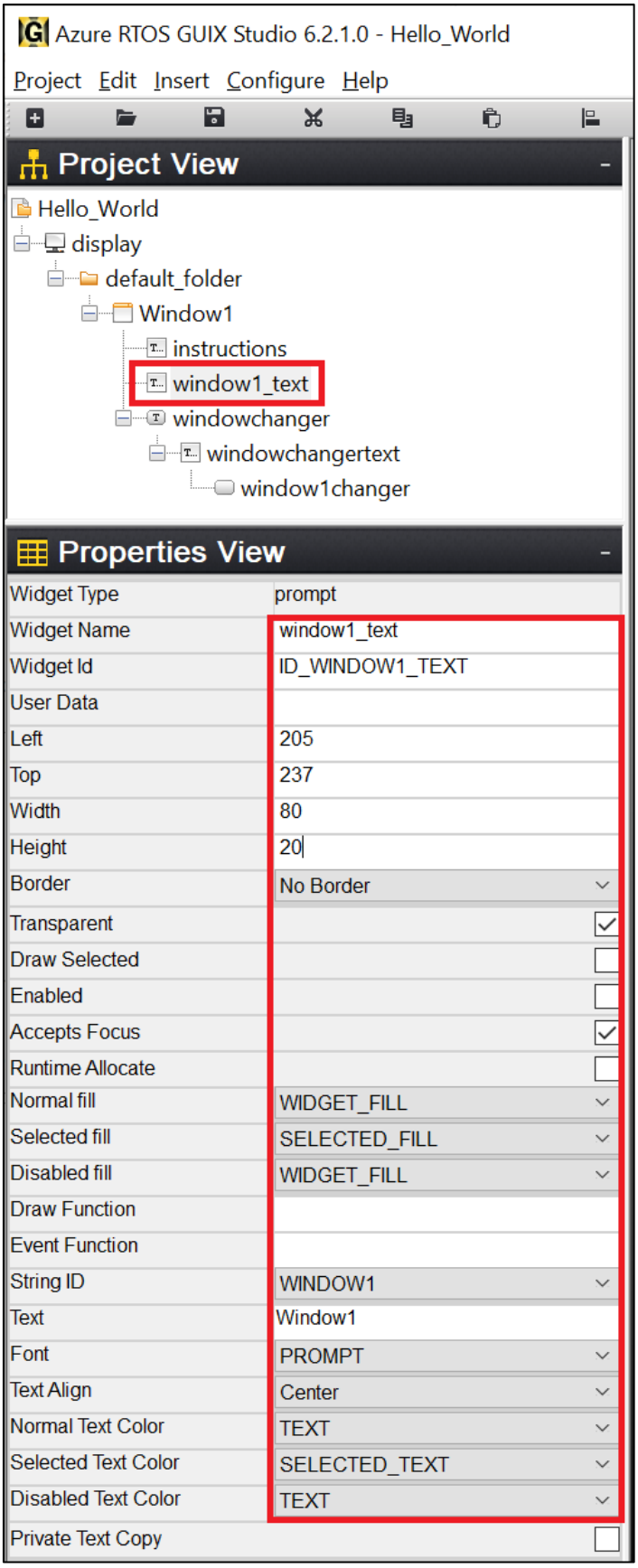


図 56. Prompt1 のプロパティ設定

20. 図 57 のように、**Window1** を右クリックし、**Checkbox** を挿入します。

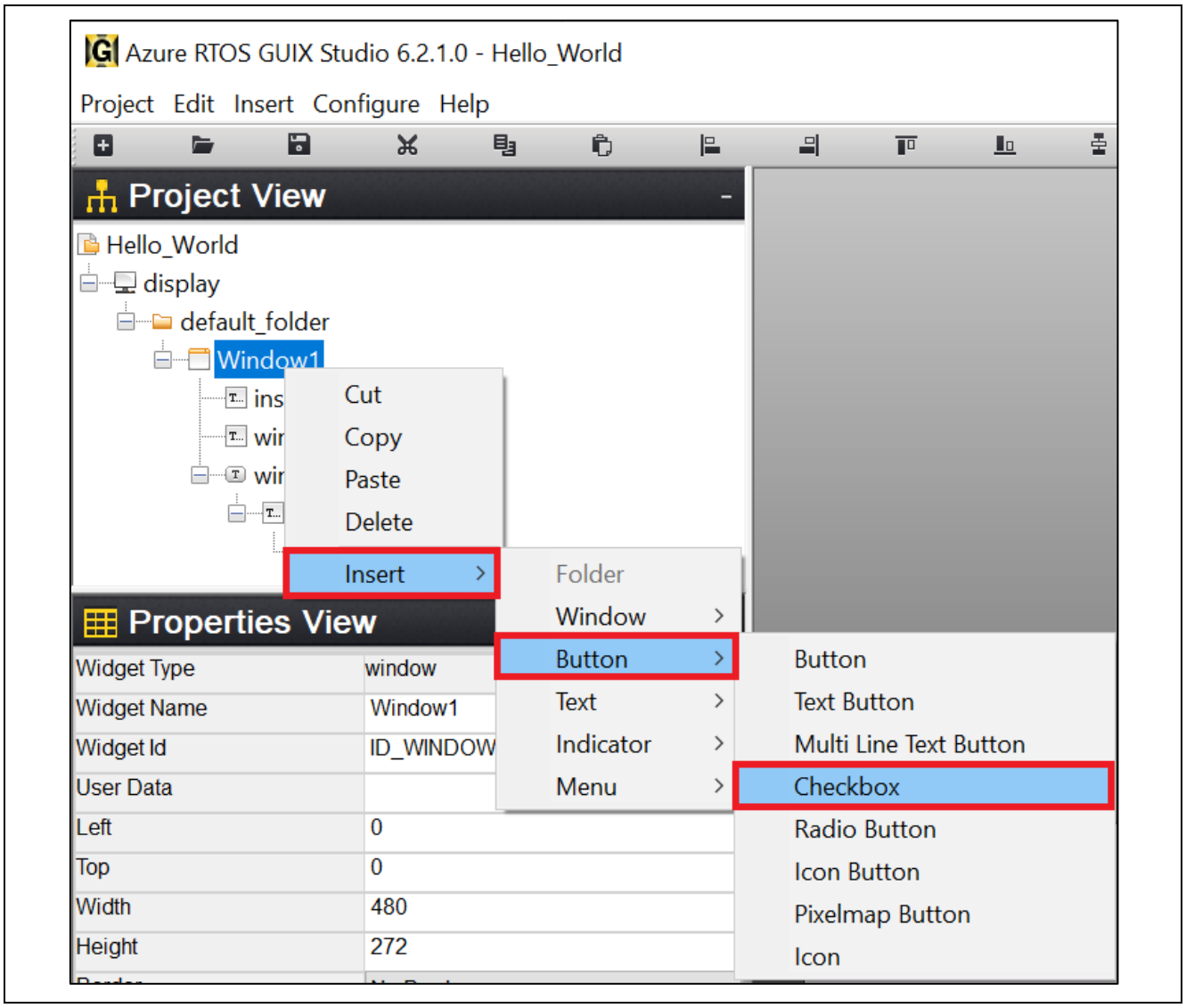


図 57. Button Checkbox の挿入

21 Checkbox のプロパティを設定します。

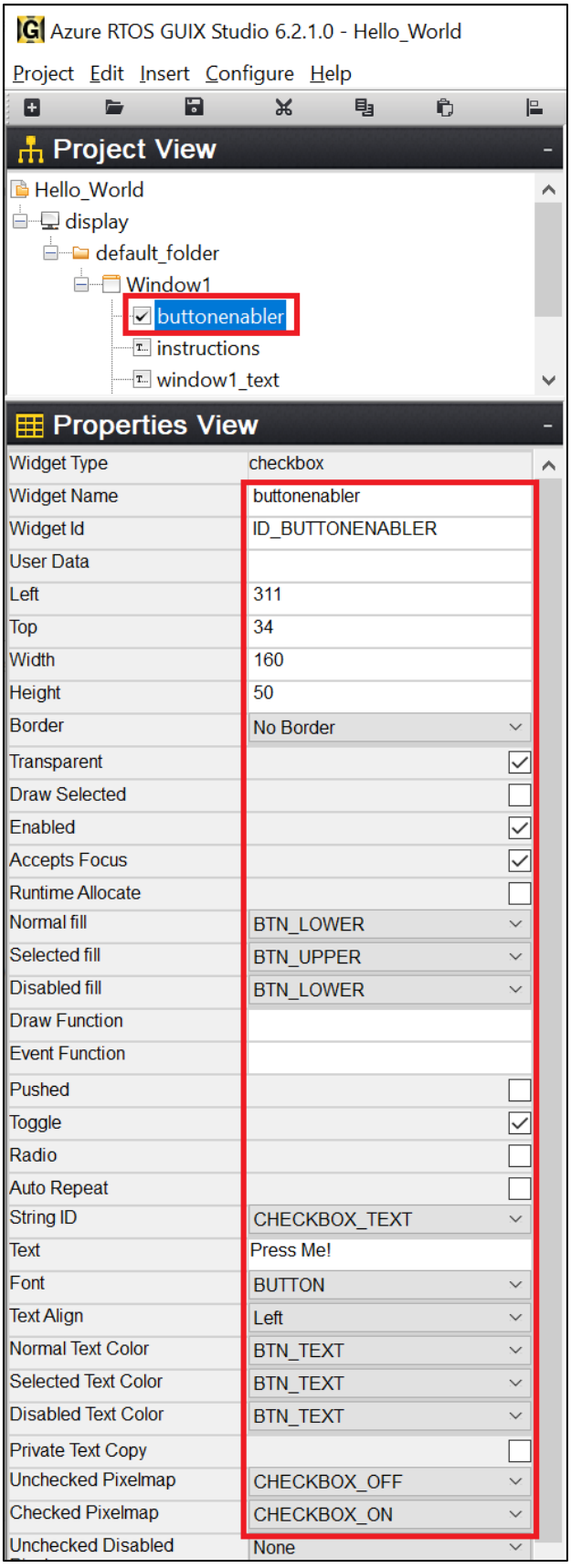


図 58. Checkbox のプロパティ設定

22. Window1 の作成が完了すると、以下の画像のようになります。

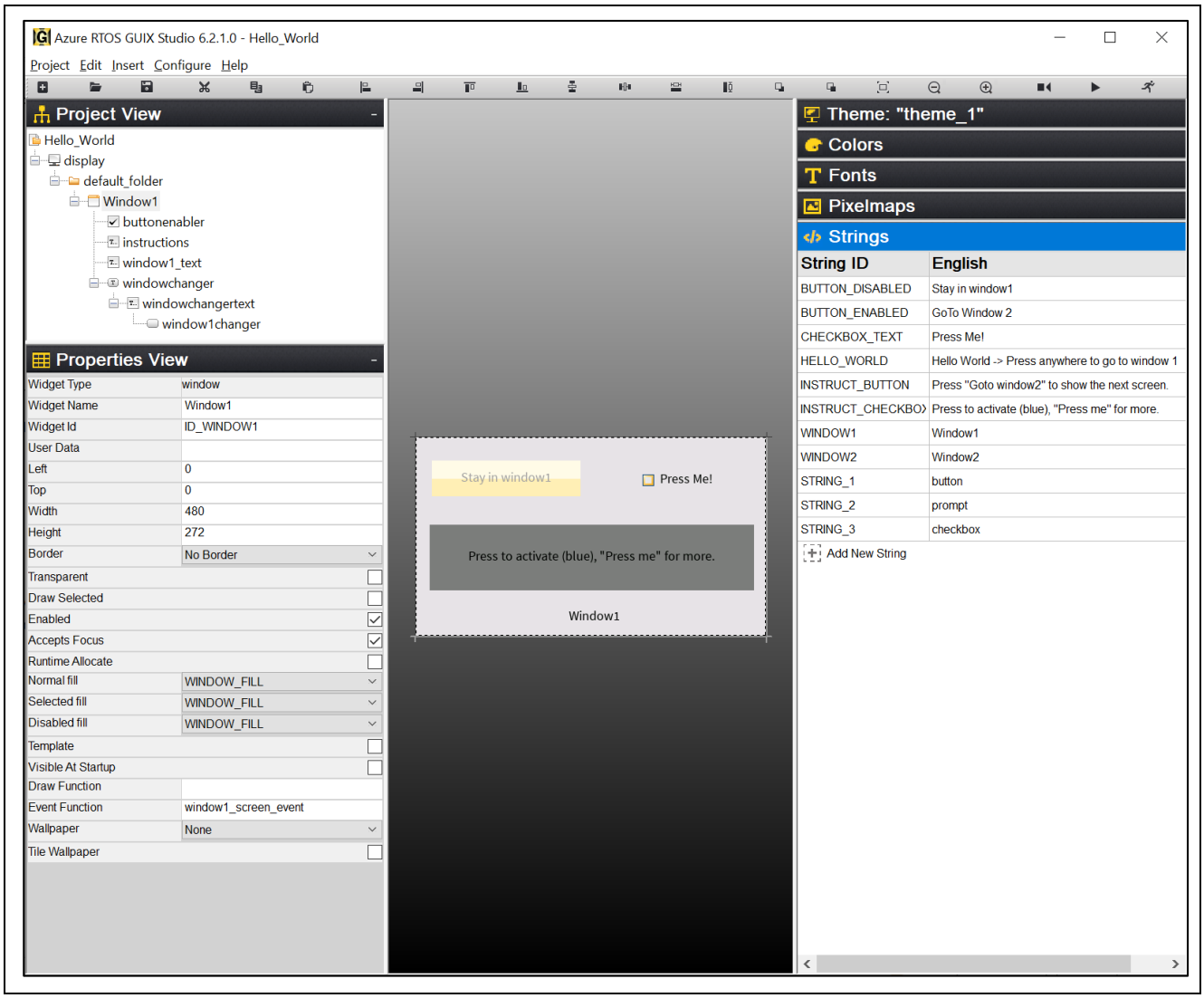


図 59. 作成した Window1

23. 図 60 のように、default_folder を右クリックし、Window2 を挿入します。（Insert > Window > Window）

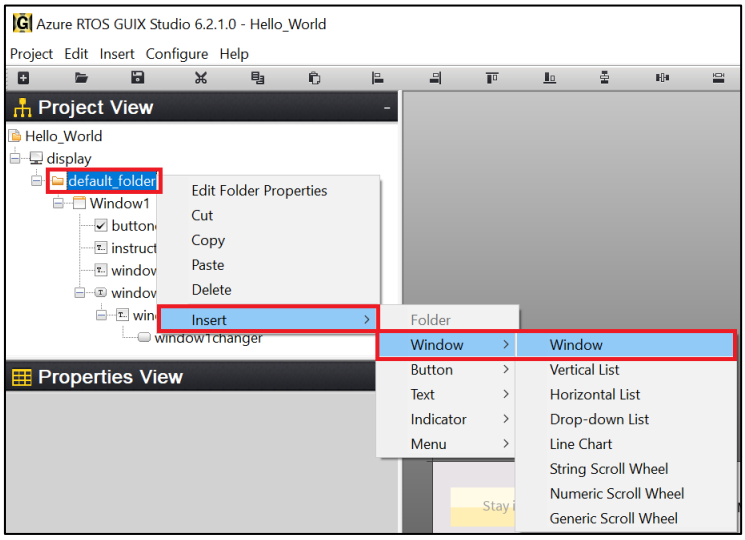


図 60. Window2 の挿入

24. Window2 のプロパティを設定します。

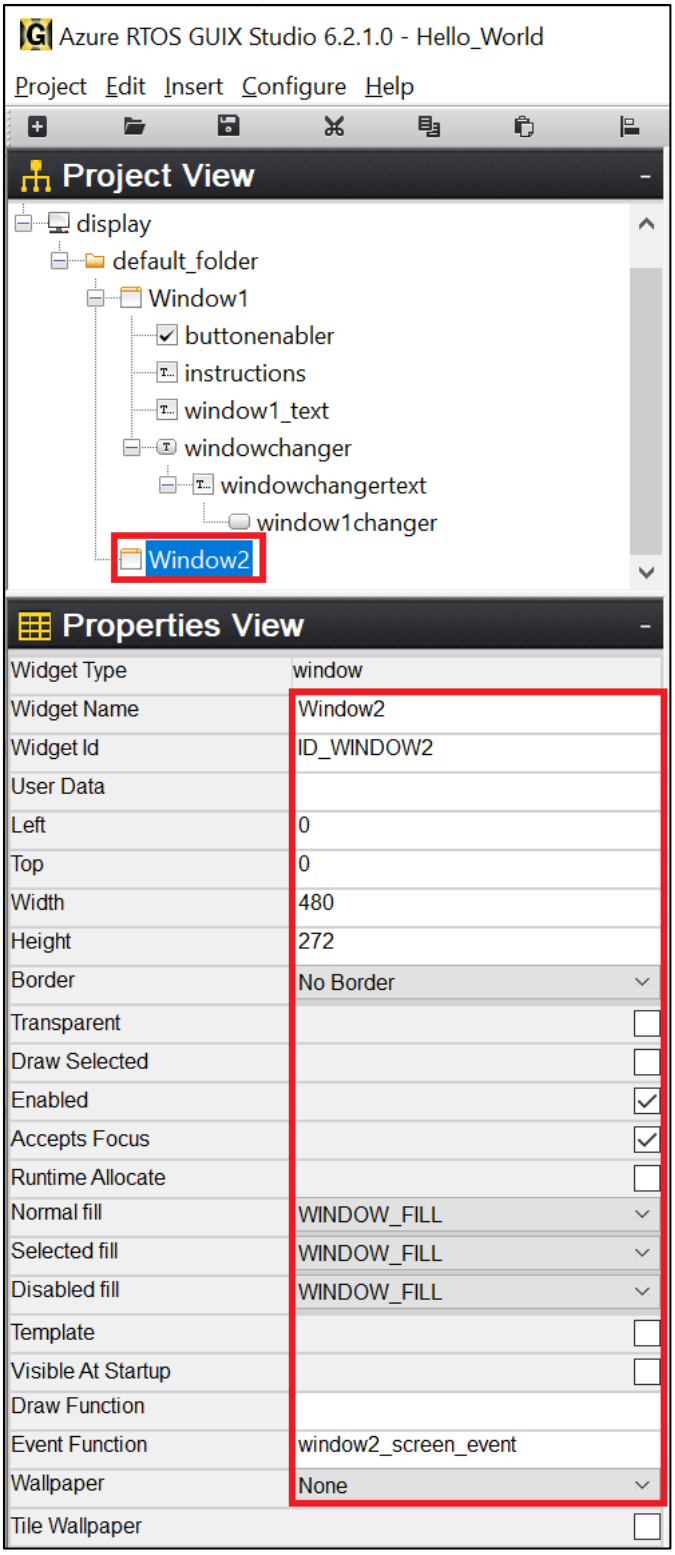


図 61. Window2 のプロパティ設定

25. 図 62 のように、**Window2** を右クリックし、**Prompt** を挿入します。（**Insert** > **Text** > **Prompt**）

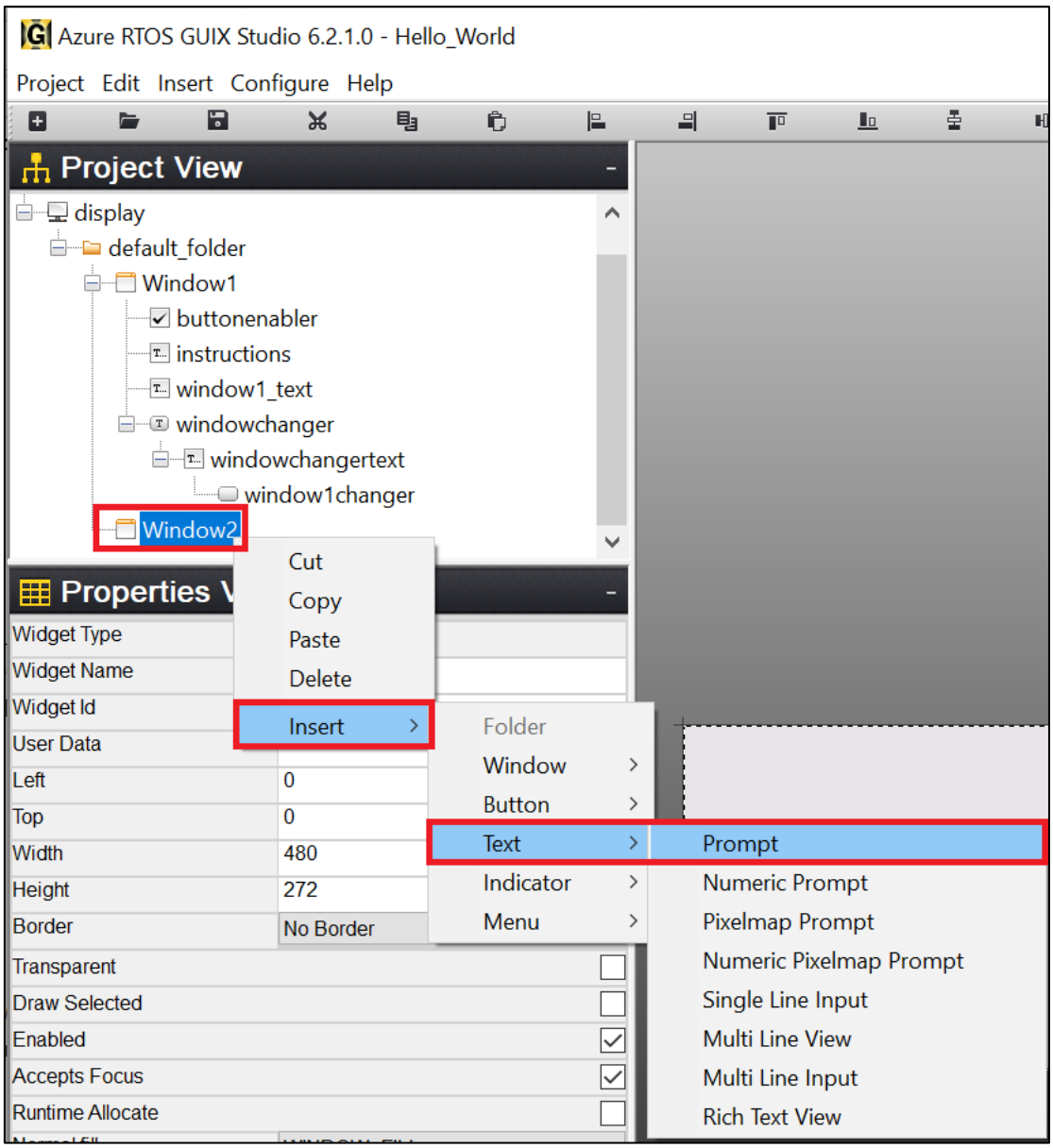


図 62. Window2 へ Prompt を挿入

26. Prompt のプロパティを設定します。

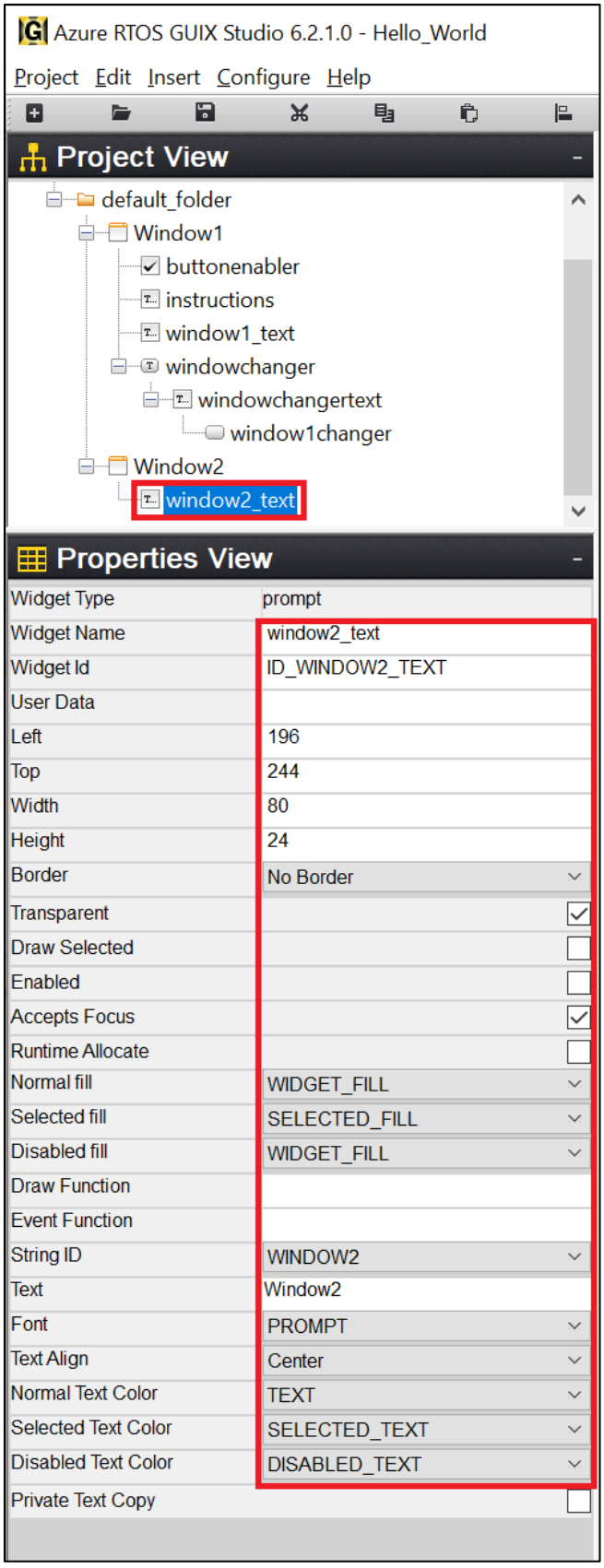


図 63.　Windows2_text のプロパティ設定

27. 図 64 のように、Window2 を右クリックし、text_button を挿入します。（Insert > Button > Text Button）

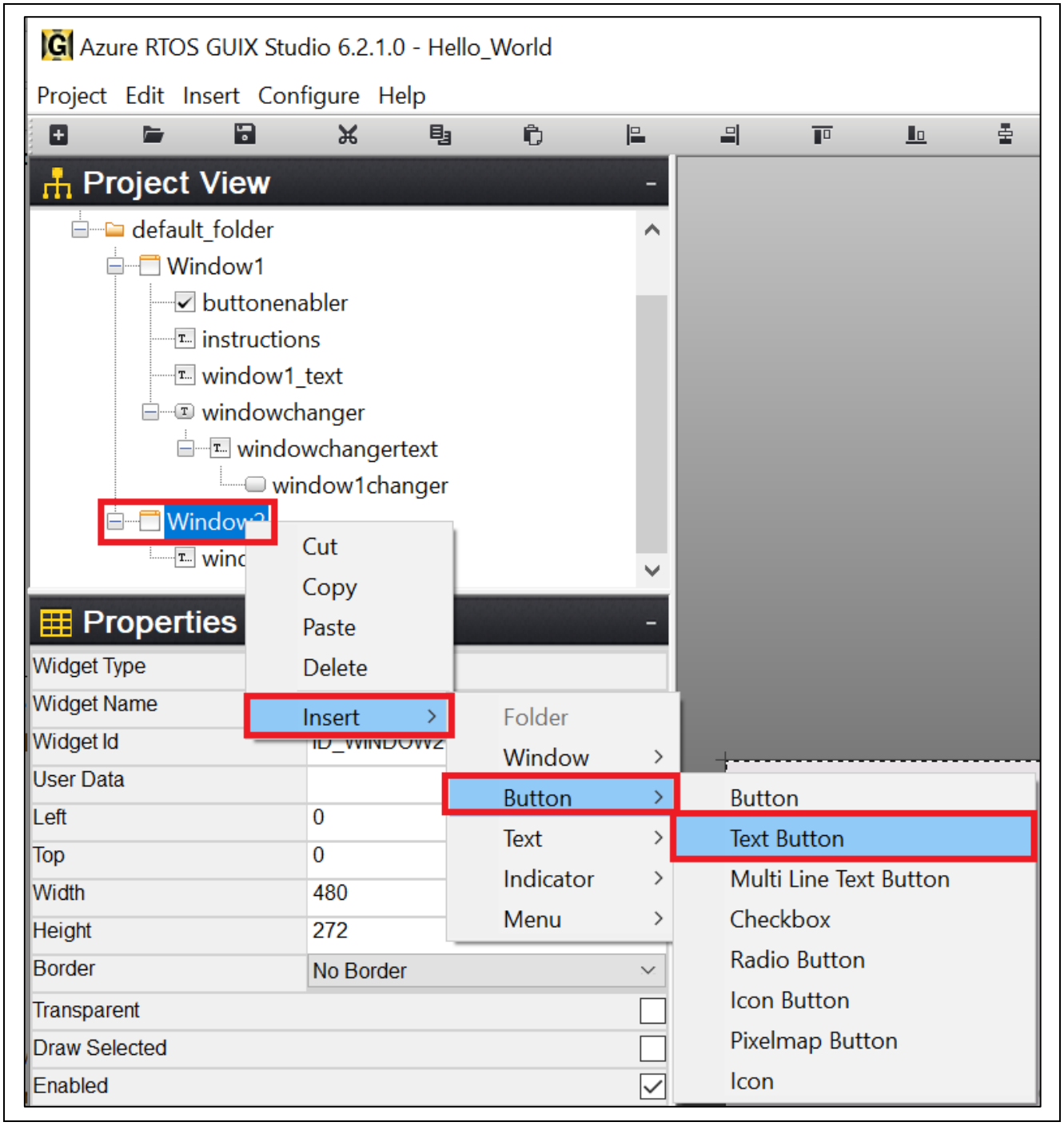


図 64. Window2 の Text Button の挿入

28. Window2 の text_button のプロパティの設定をします。

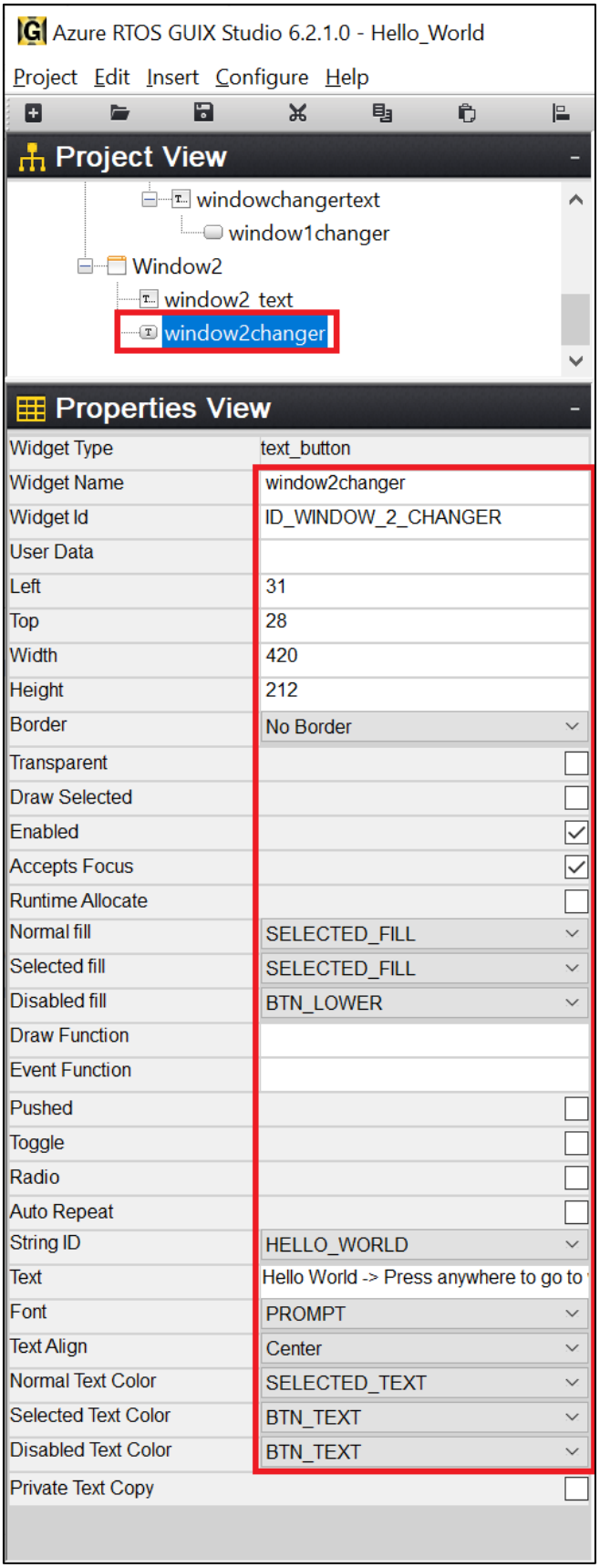


図 65. Window2のtext_buttonのプロパティ設定

29. 設定終了後の Window2 を図 66 に示します。

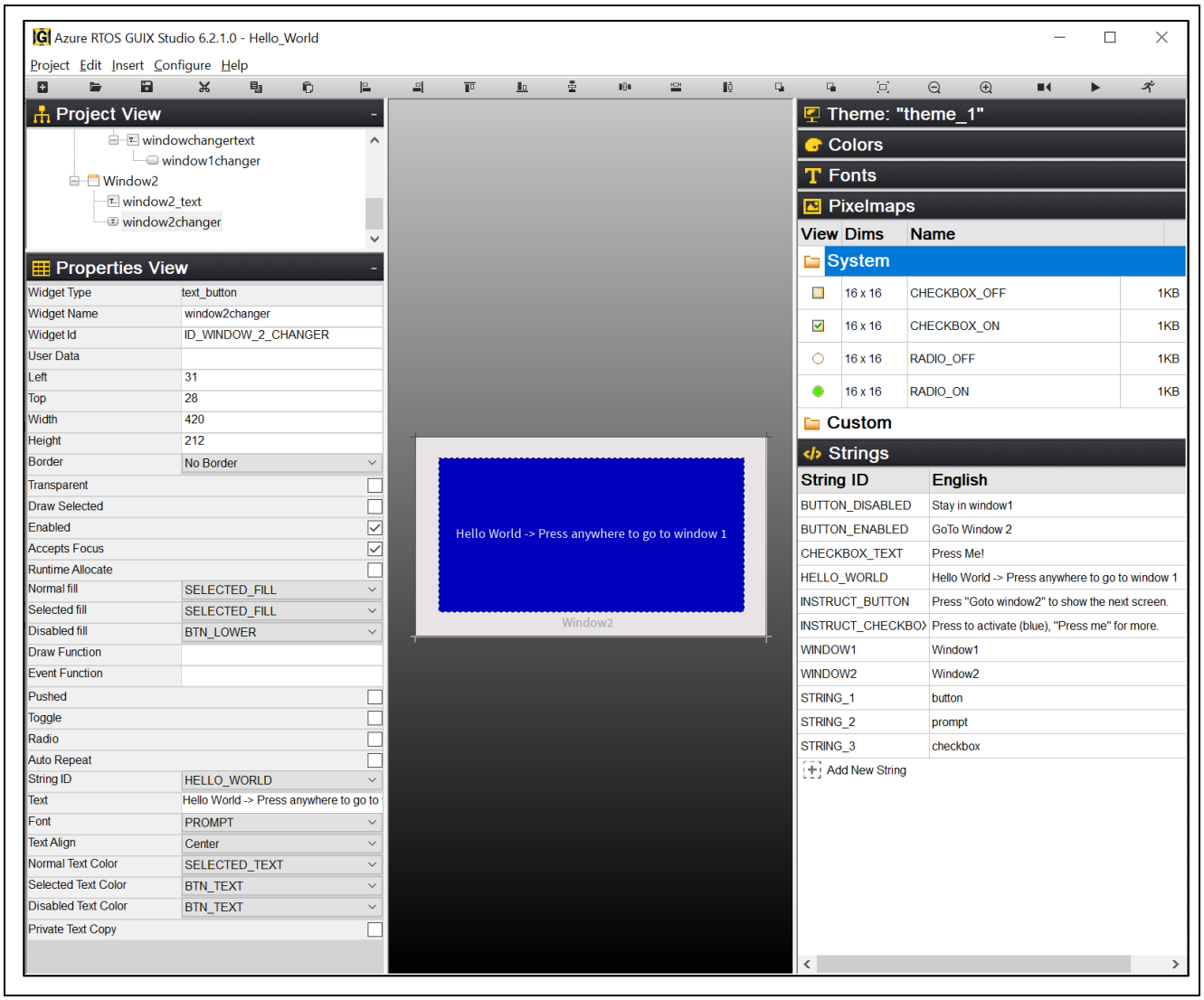


図 66. Window2

30. ドロップダウンリストの Pixelmaps をクリックし、CHECKBOX_OFF をダブルクリックすると新規ウィンドウがポップアップ表示されます。

Compress Outputのチェックを外して、Saveをクリックします。CHECKBOX_ONの場合も同様の操作を行います。

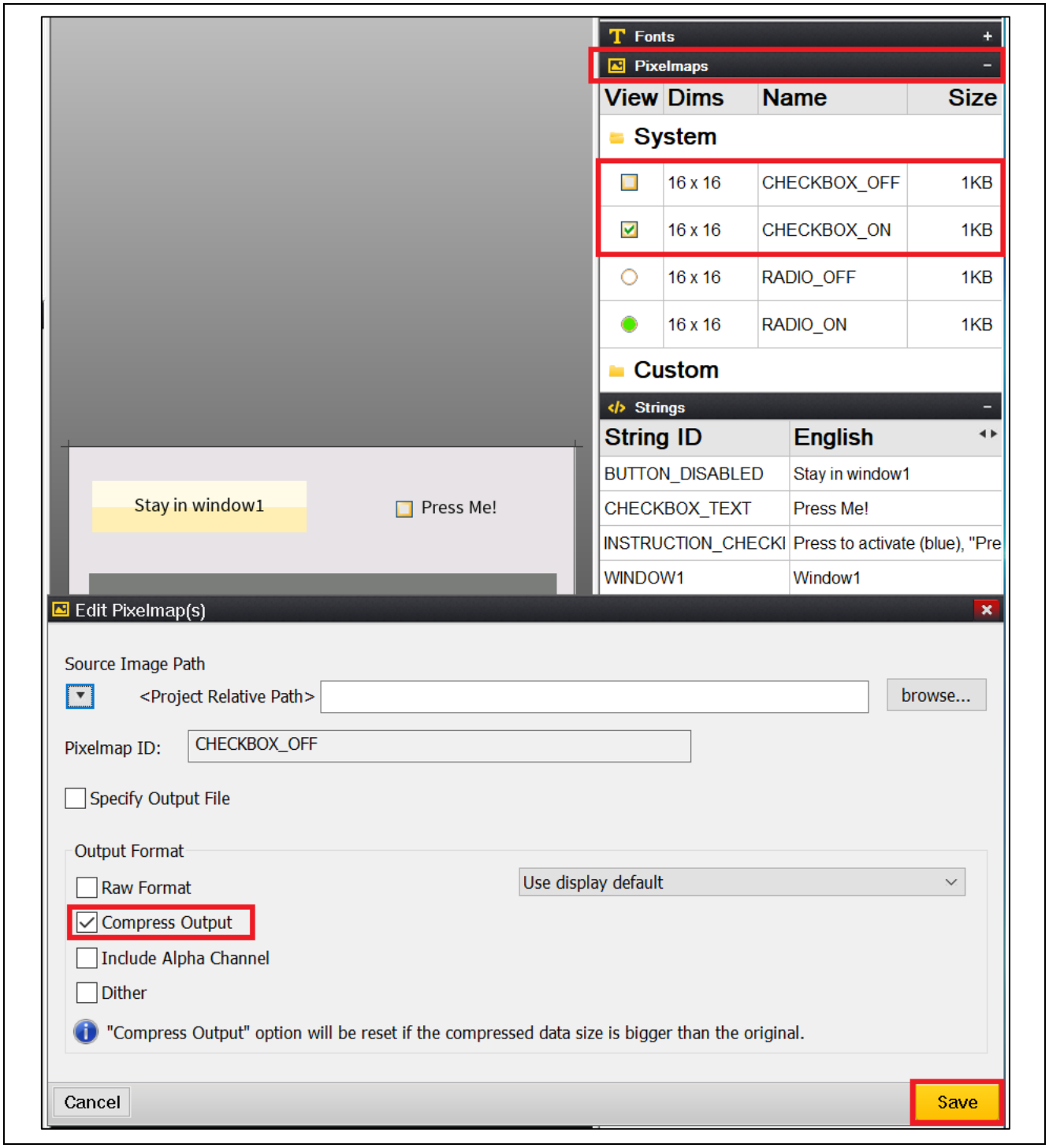


図 67. Pixelmap の設定

31. ここで、Project のドロップダウンリストをクリックし、**Save Project**、および **Generate All Output Files** をクリックします。**GUIX Hello World** を作成してプロジェクトにエクスポートするプロセスが完了しました。

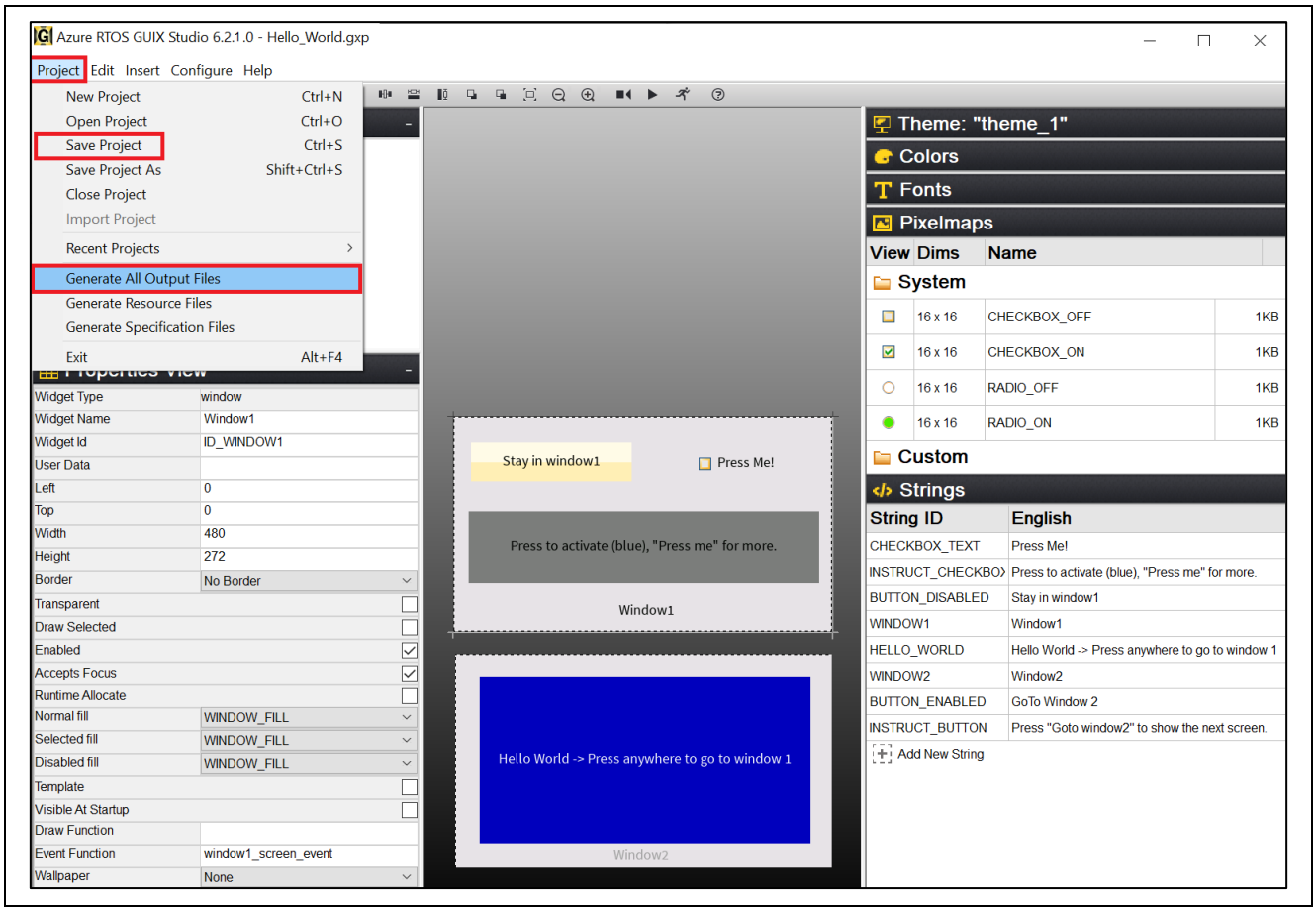


図 68. プロジェクトの保存と生成

32. プロジェクトがアクティブであることを確認し、クリックしてプロジェクトをビルドします。
PC で Azure RTOS/GUIX プロジェクトのビルドが完了するまでに時間がかかる場合があります。
プロジェクト **ra8d1_guix_hello_world** は、エラーや警告なしでビルドする必要があります。

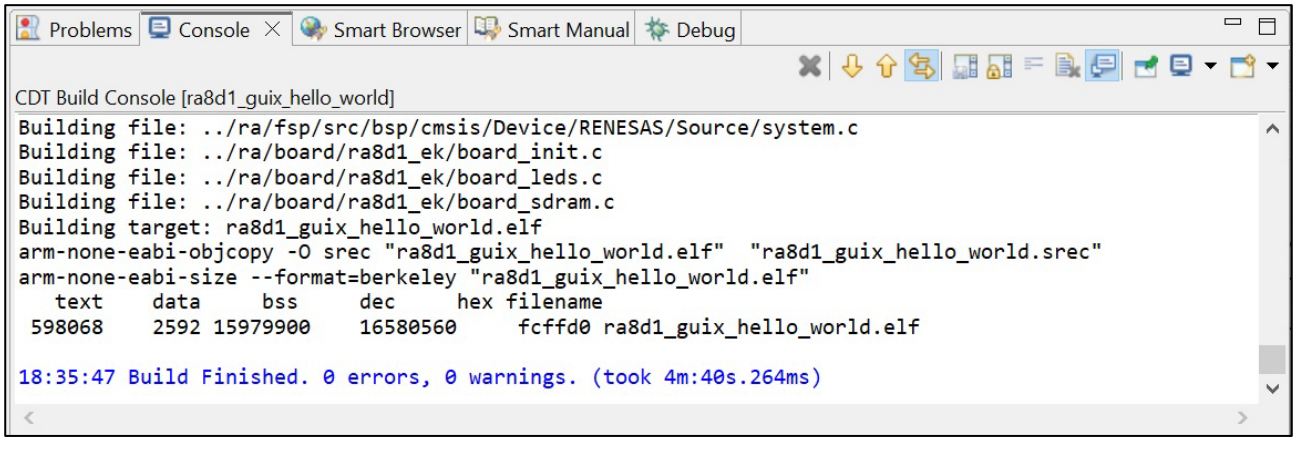


図 69.コードのビルド

33. マイクロ USB ケーブルを EK-RA8D1 ボードの J10 に接続し、もう一方を PC に接続します。
ra8d1_guix_hello_worldプロジェクトをダウンロードして実行します。

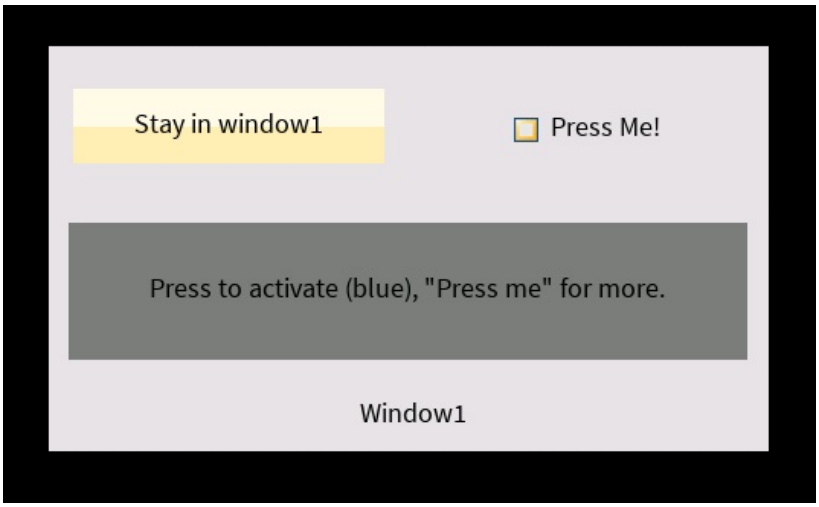


図 70. Window1 の表示

6. 既存プロジェクトの概要

6.1 概要

この章では、弊社で「作成済のプロジェクト」ra8d1_guix_hello_worldのインポートの実行方法を説明します。
本プロジェクトでは
-チェックボックス機能を有効または無効に設定可能。
-ボタンのStay in window1またはGo to Window2のテキストが更新。
-ボタンを押すと、画面がウィンドウ1からウィンドウ2に切り替え可能。
-画面のテキストメッセージに従って、ウィンドウ2からウィンドウ1に戻ることが可能。

手順は、図 71を参照してください。

6.2 手順

1. 提供されているプロジェクト ra8d1_guix_hello_world でフル機能のアプリケーションを試すことができます。すでに同じ名前のプロジェクトがワークスペースにあるため、Import メニューの Rename & Import Existing C/C++ Project into Workspace 機能を使用してください。

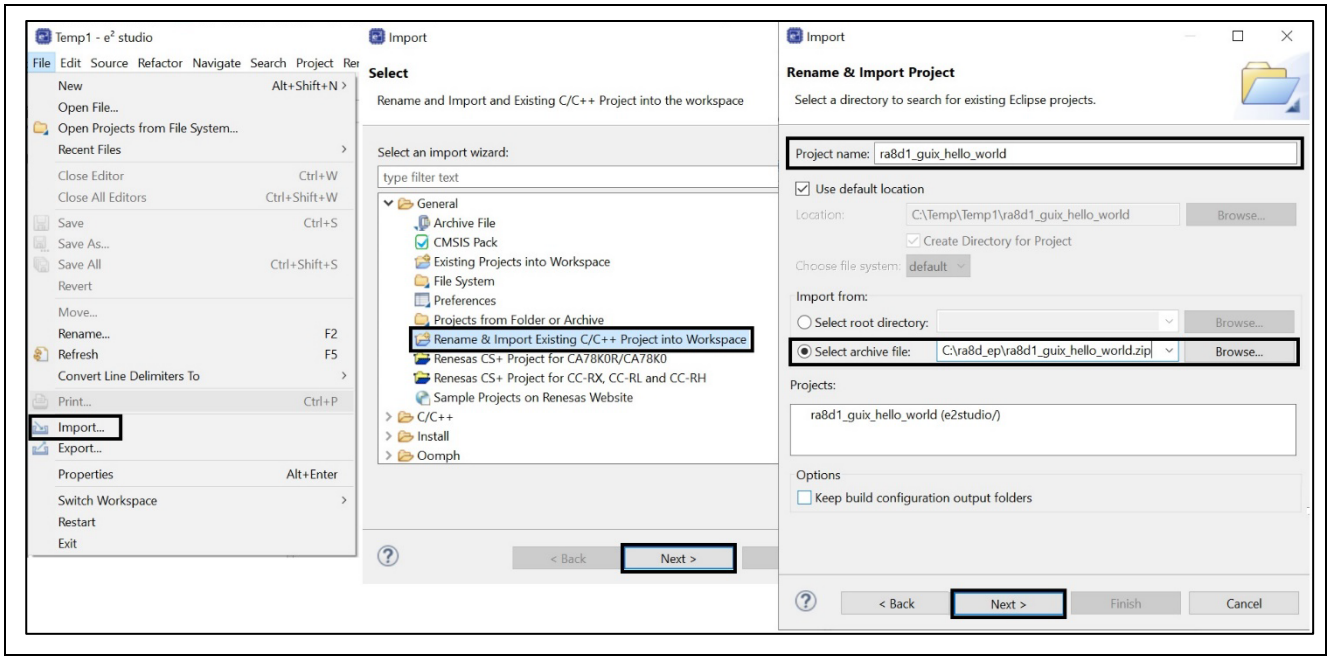


図 71. 既存プロジェクトのインポート

7. ウェブサイトとサポート

RA ファミリの主要な要素についての学習、コンポーネントと関連ドキュメントをダウンロード、サポートを受けるなどは、以下の URL にアクセスしてください。

RA 製品情報	www.renesas.com/ra
RA 製品サポートフォーラム	www.renesas.com/ra/forum
RA フレキシブルソフトウェアパッケージ	www.renesas.com/FSP
ルネサスサポート	www.renesas.com/support

改訂履歴

Rev.	日付	説明	
		ページ	概要
1.00	2024 年 6 月 25 日	–	初版バージョン

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する使用上の注意事項について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策

   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に電源オフ時における入力信号についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

   未使用端子は、未使用端子の処理に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を標準水準および高品質水準に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（脆弱性問題といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の半導体デバイスの使用上の一般的な注意事項等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、外国為替及び外国貿易法その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注1. 本資料において使用されている当社とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注2. 本資料において使用されている当社製品とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1 2024.10)

本社

豊洲フォレシア、豊洲3-2-24、
135-0061 東京都江東区豊洲3-2-24 豊洲フォレシア TOYOSU FORESIA, 3-2-24
www.renesas.com

お問い合わせ先

製品、技術、ドキュメントの最新版、最寄りの営業所などに関する詳しい情報は、こちらをご覧ください。
www.renesas.com/contact/

商標