ルネサス RA ファミリ

EK-RA8D1 MIPI LCD ディスプレイ向け 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。

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)以上
- 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 以上
- Azure RTOS GUIX Studio V6.3.0.1 以上

ハードウェア

- ルネサス EK-RA8D1 キット（RA8D1 MCU グループ）
- ルネサス EK-RA8D1 の SW1 スイッチの設定
  - GLCDC スイッチ#6 を ON に設定、SDRAM スイッチ#7 を ON に設定し、その他のスイッチはすべて OFF に設定します。

参考

- 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 v5.1.0
- Azure RTOS GUIX および GUIX Studio v6.3.0.1
- ルネサス RA8D1 グループ ユーザーズマニュアル 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_gt911 および SEGGER_RTT という 2 つのフォルダと、4 つの.c ファイルおよび 2 つの.h 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, system_thread_entry.h, common_utils.h

目的

このドキュメントでは、e² 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、MIPI ディスプレイコントローラ、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。

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 v2023-10.0 と FSP v5.1.0 と、Azure RTOS GUIX studio v6.3.0.1 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 を参照してください。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が表示されます。
   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。

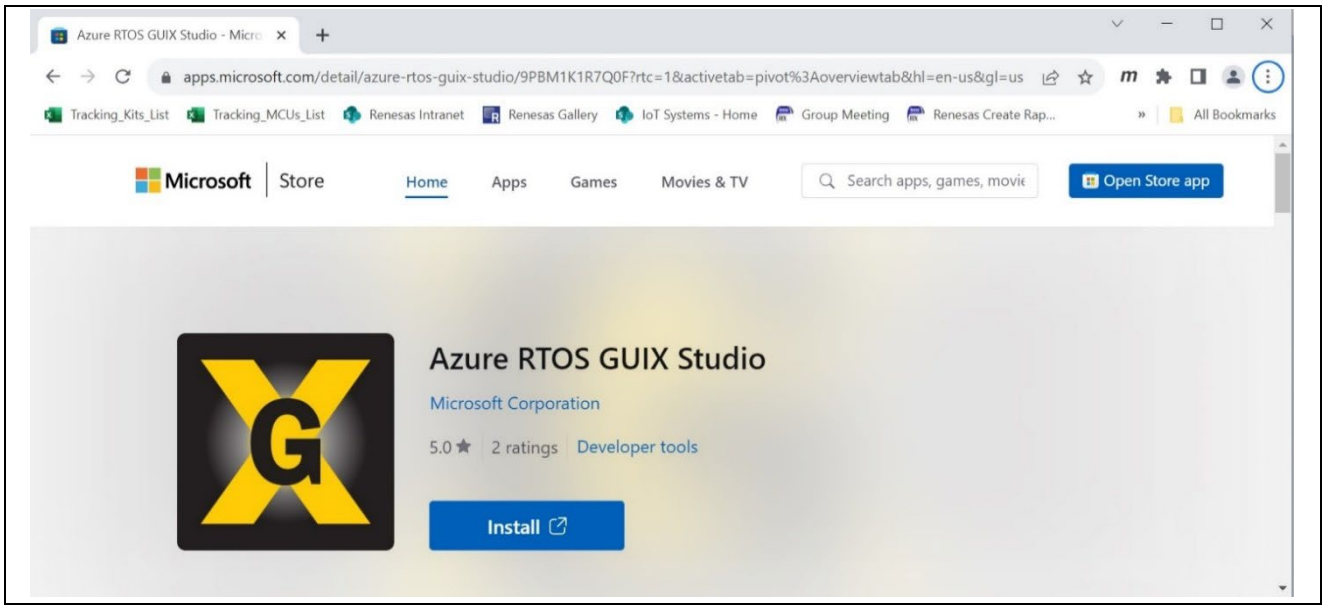


図 1. Azure RTOS GUIX Studio の入手

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。

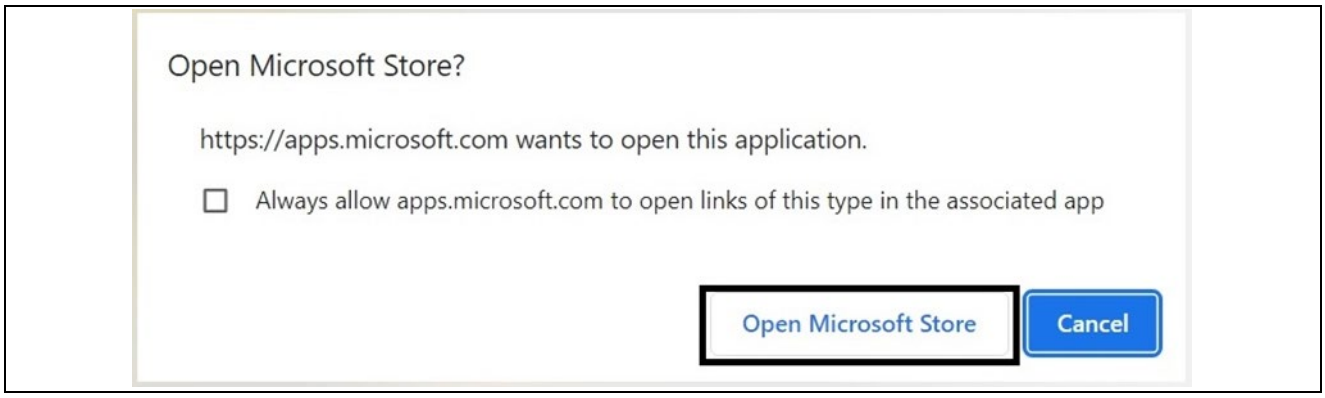


図 2.　Microsoft Store を開く

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

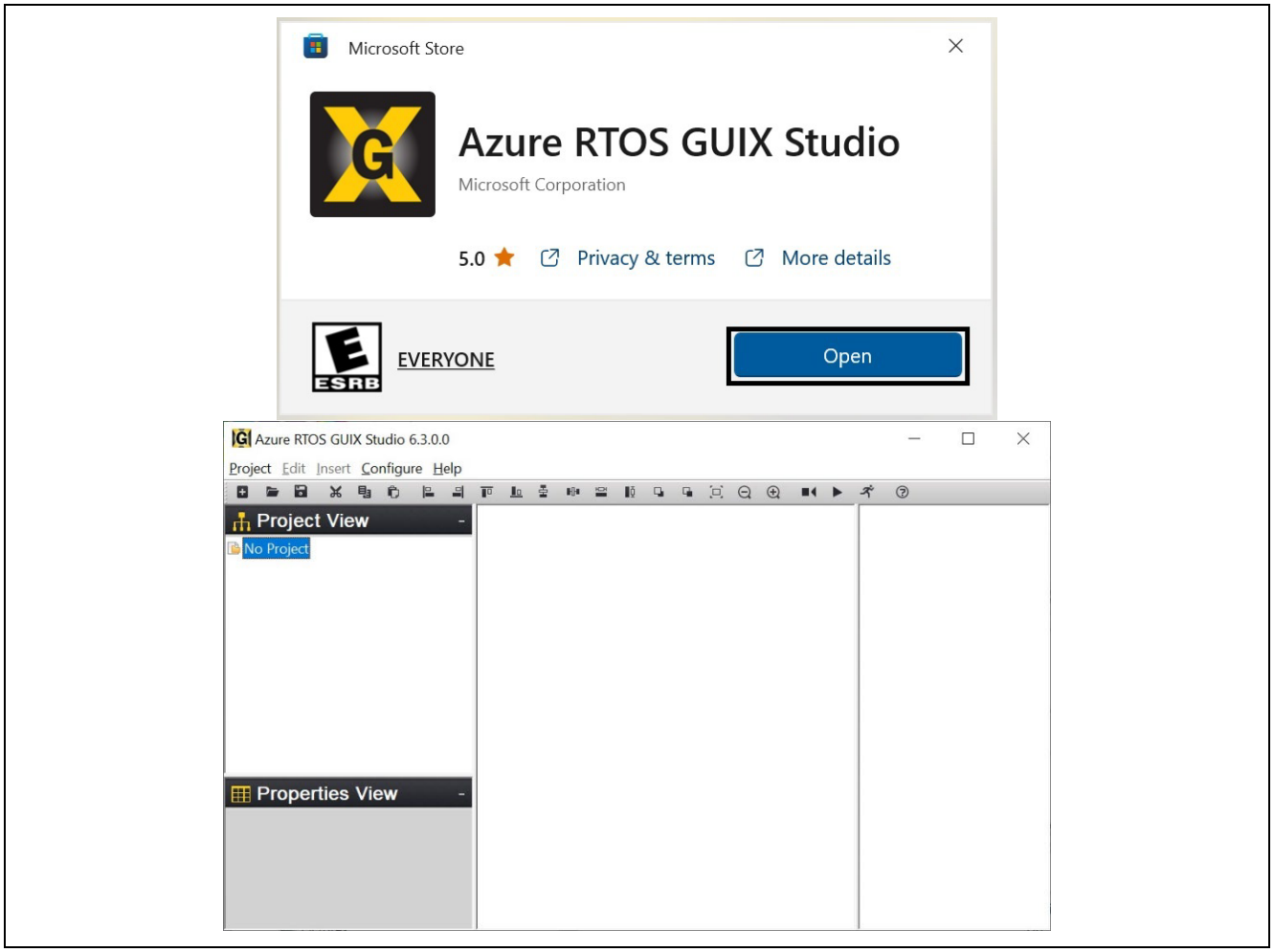


図 3.　Azure RTOS GUIX Studio の起動

2. アプリケーションノート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 **mipi_ek_ra8d1_guix_hello_world** にします。

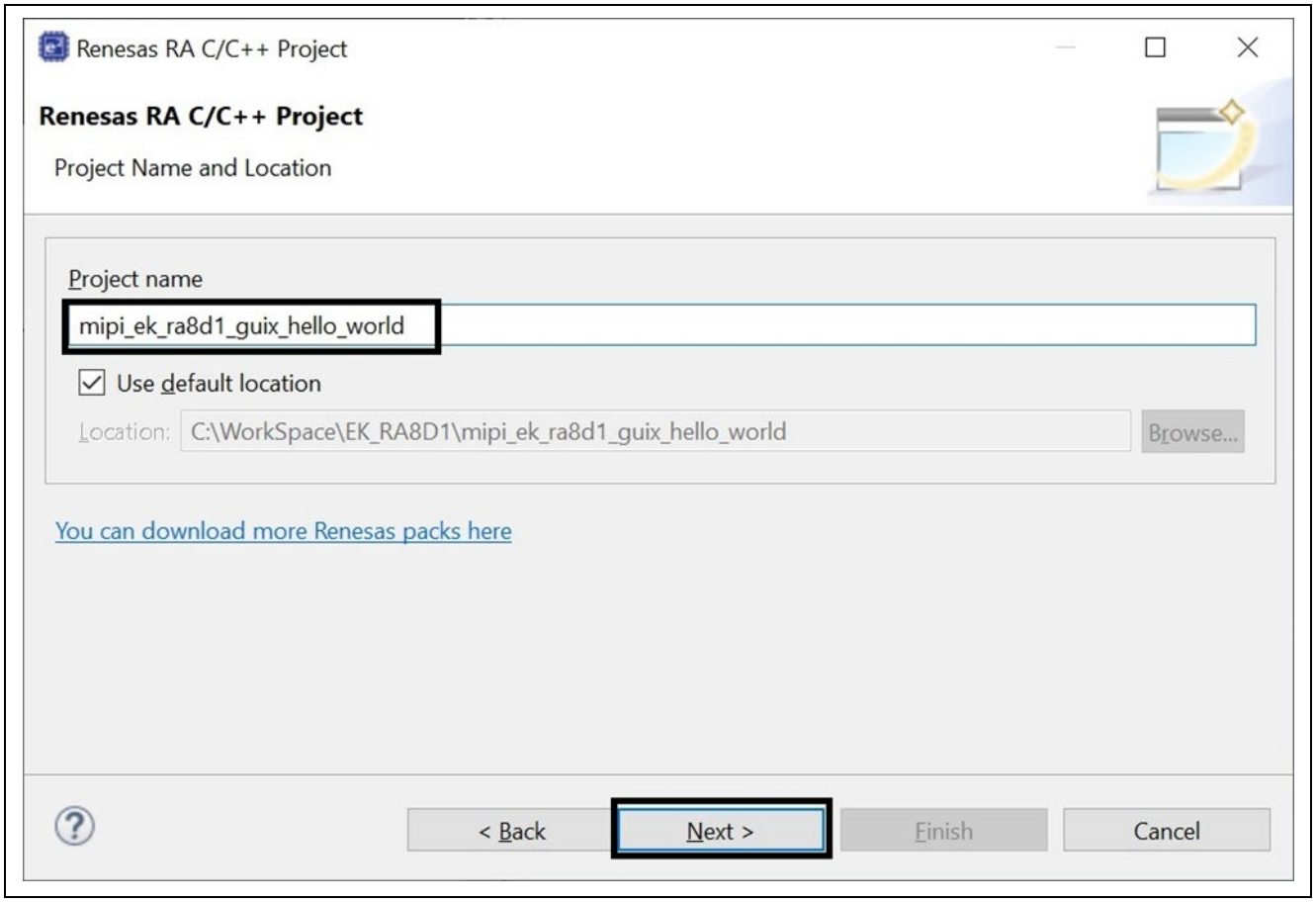


図 4. 新規プロジェクトの作成

2. ボード選択で、EK-RA8D1 を選択します。

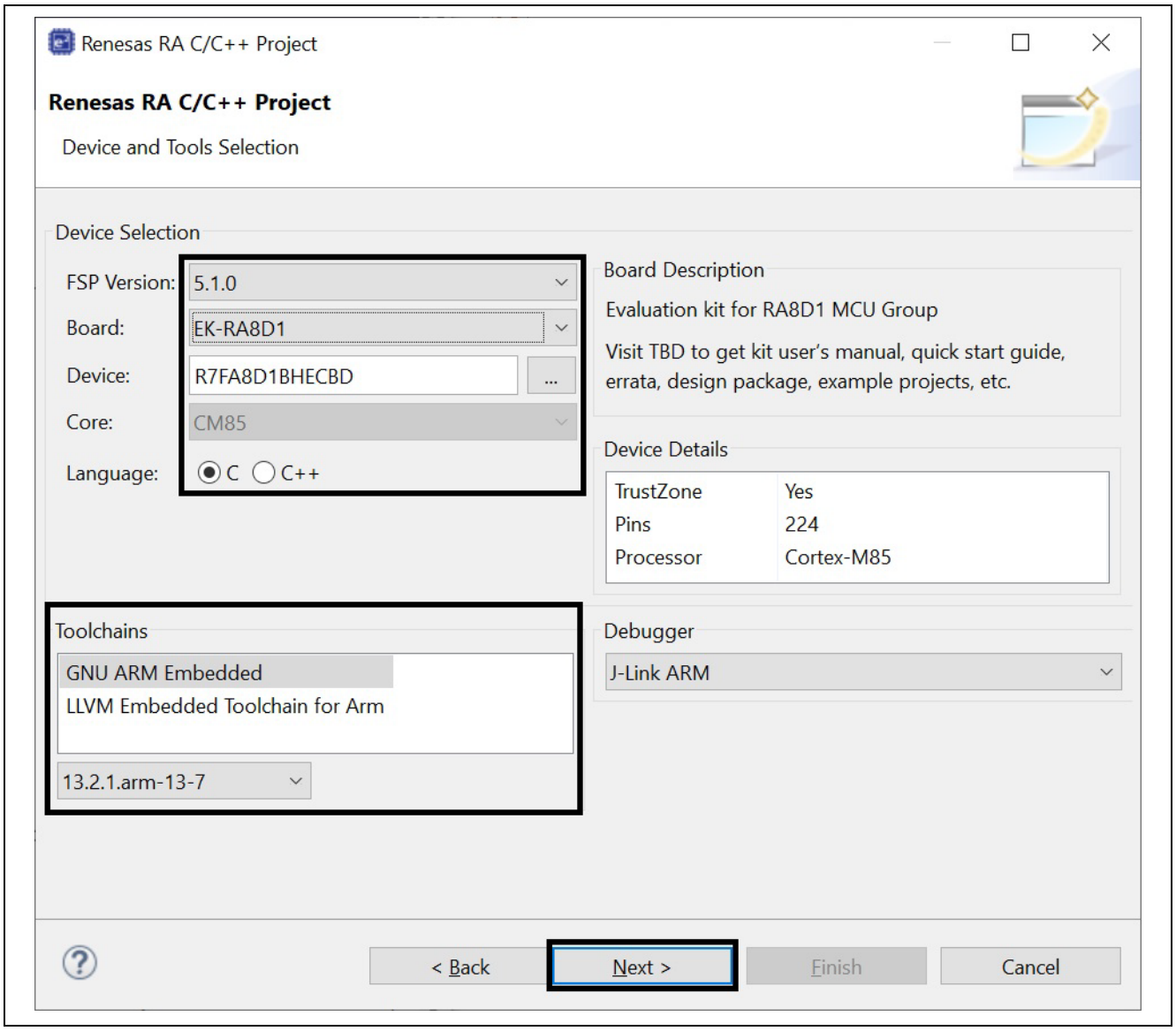


図 5.　評価ボードの選択・設定

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。

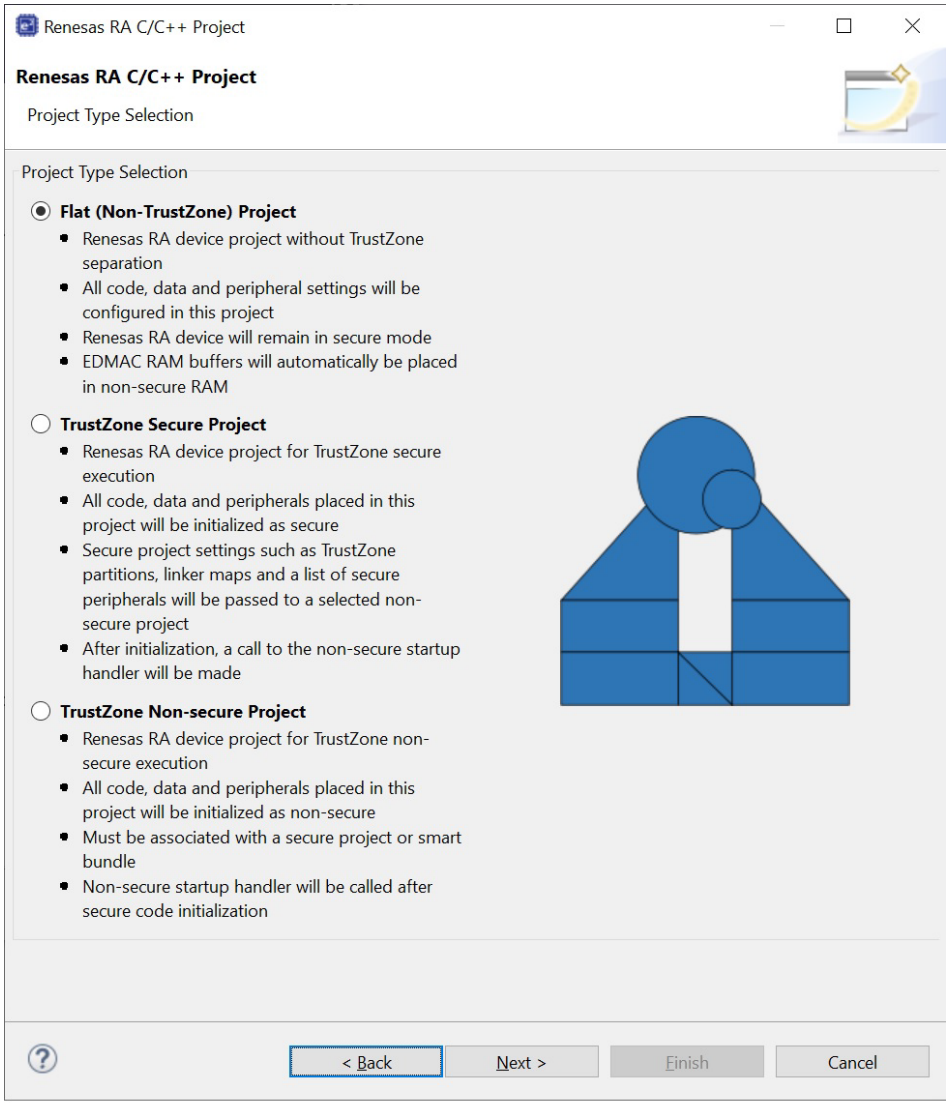


図 6.　Azure RTOS ThreadX の選択

4. **Azure RTOS ThreadX-Minimal** テンプレートを使用します。

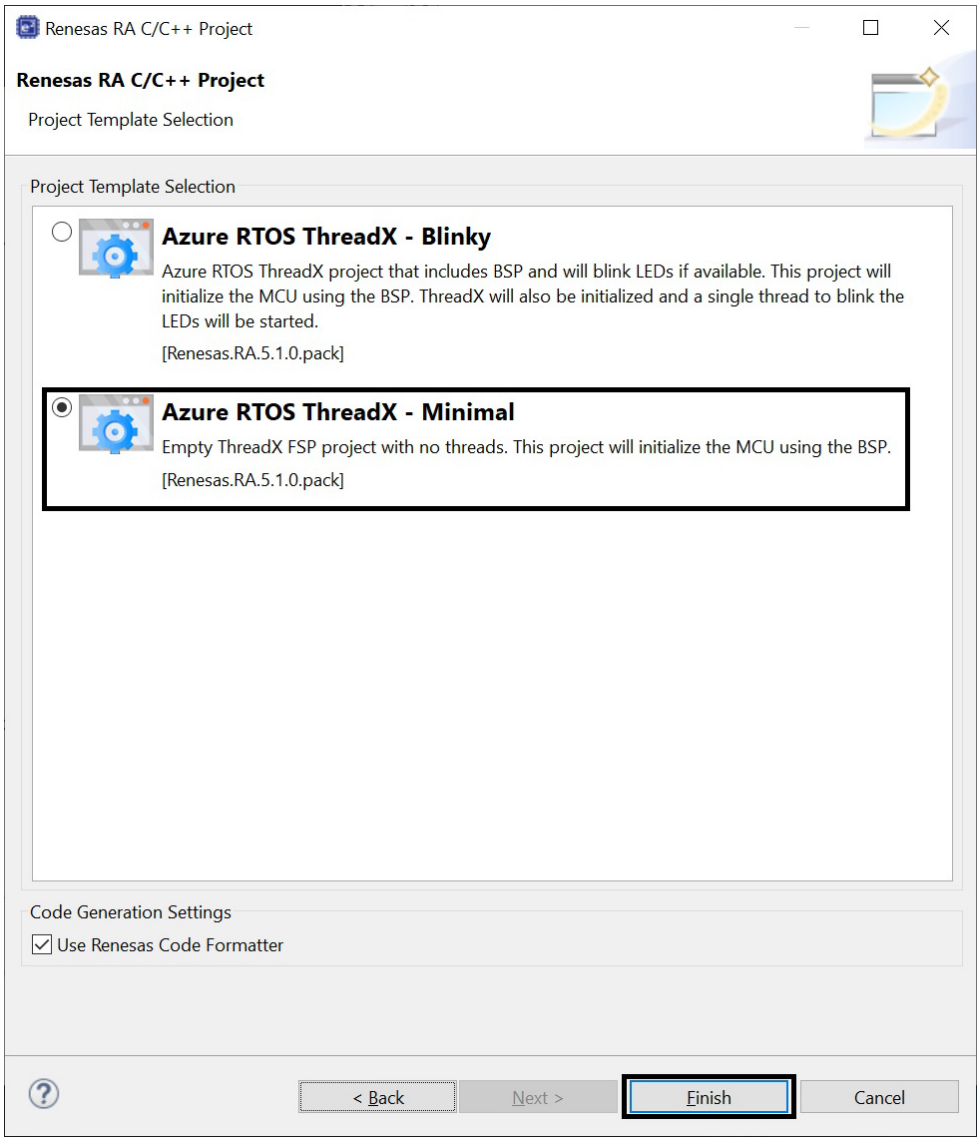


図 7. Minimal Template の選択

5. プロジェクト構成を開くには、**configuration.xml** configuration.xml ファイルをダブルクリックして、e² studio のビューを FSP Configuration に変更します。**BSP** タブに移動し、**Properties** タブを選択し、**Main stack size(bytes)**と **Heap size (bytes)**を **0x2000** に変更します。

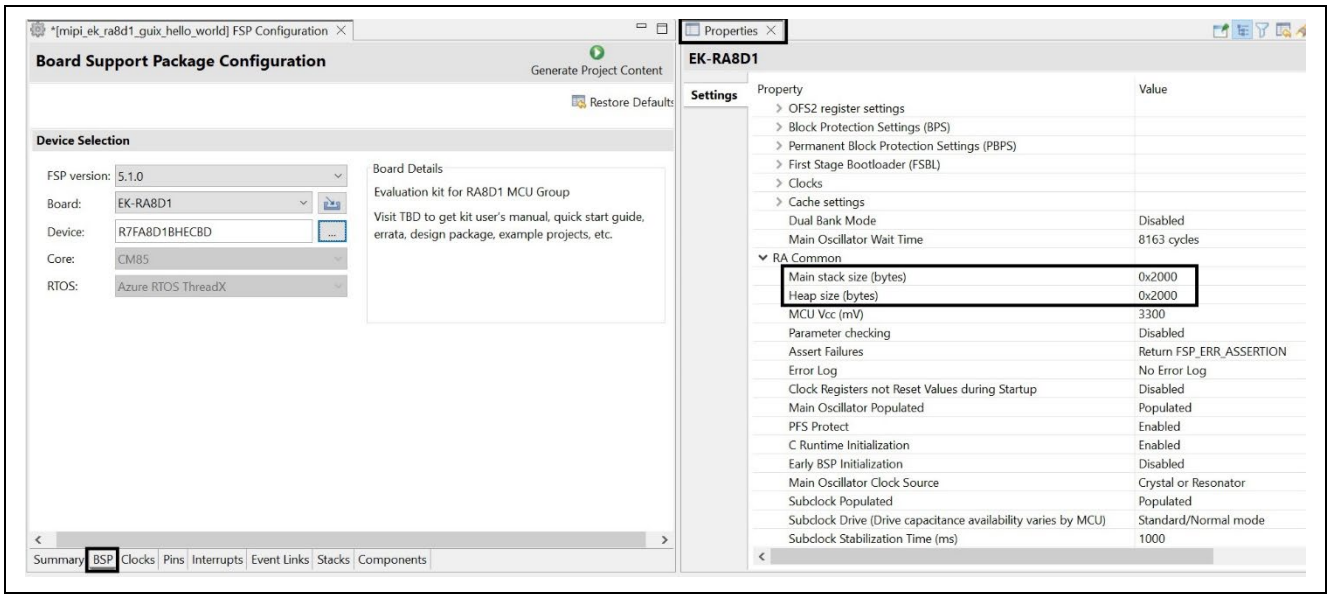


図 8.　Heap Size の変更

6. **Clocks** タブをクリックし、**LCDCLK** ソース入力を **PLL1P** に設定します。

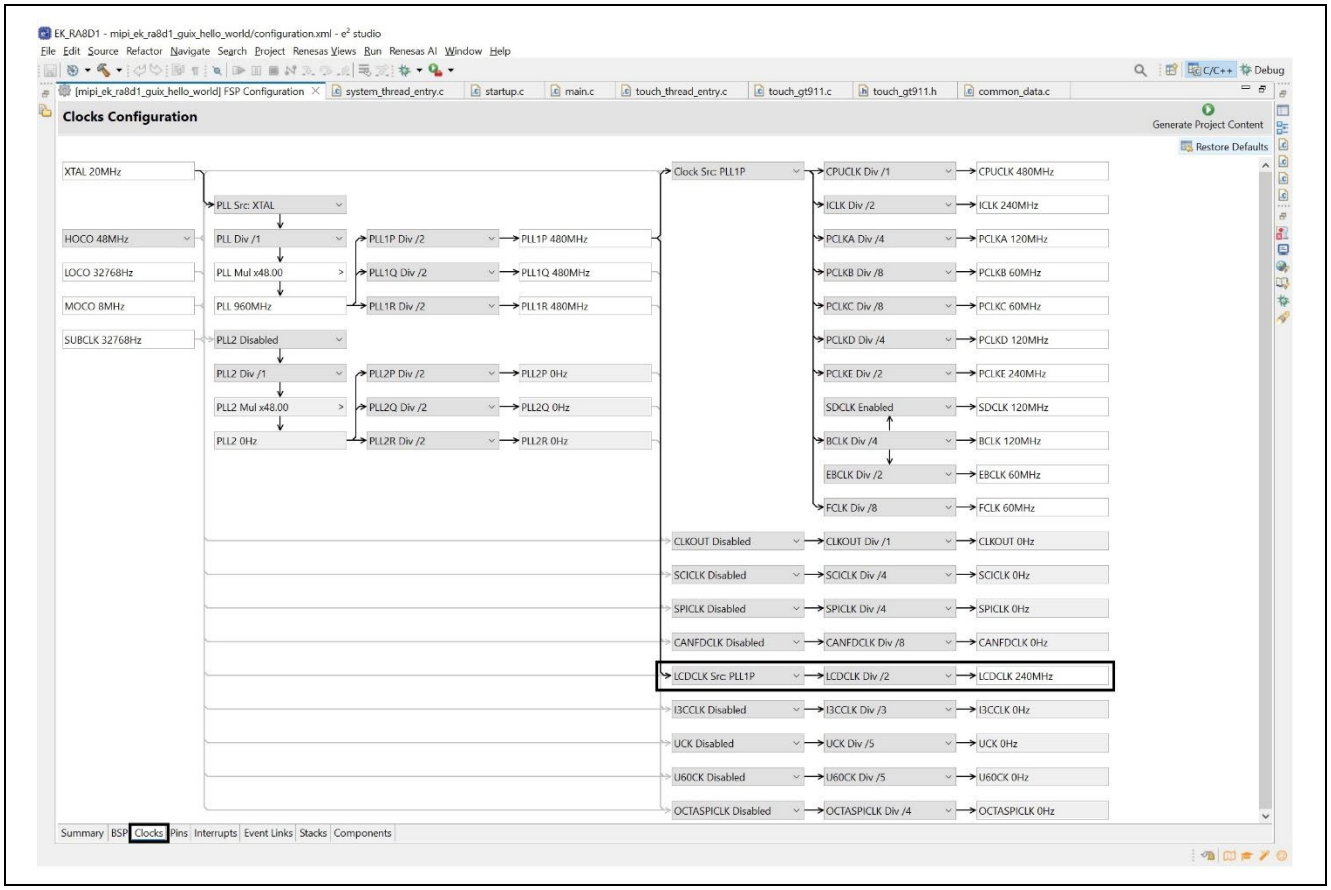


図 9.　PLL1P への LCD クロックの有効化

7. **Stacks** タブに移動し、現在のプロジェクトのスレッドとモジュールを表示します。**New thread** を追加し、以下の**プロパティ**設定で **System Thread** と名前を付けます。

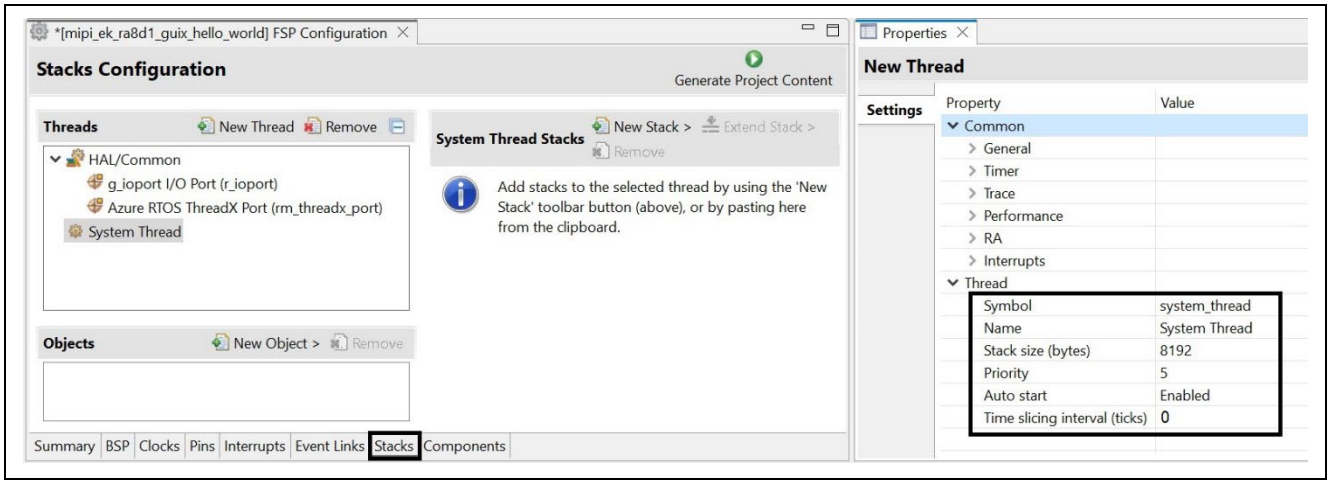


図 10. システムスレッドの追加

8. スレッドウィンドウ内で System Thread がハイライトされていることを確認します。**New Stack** をクリックし、**Graphics>Azure RTOS GUIX** をシステムスレッドに追加します。

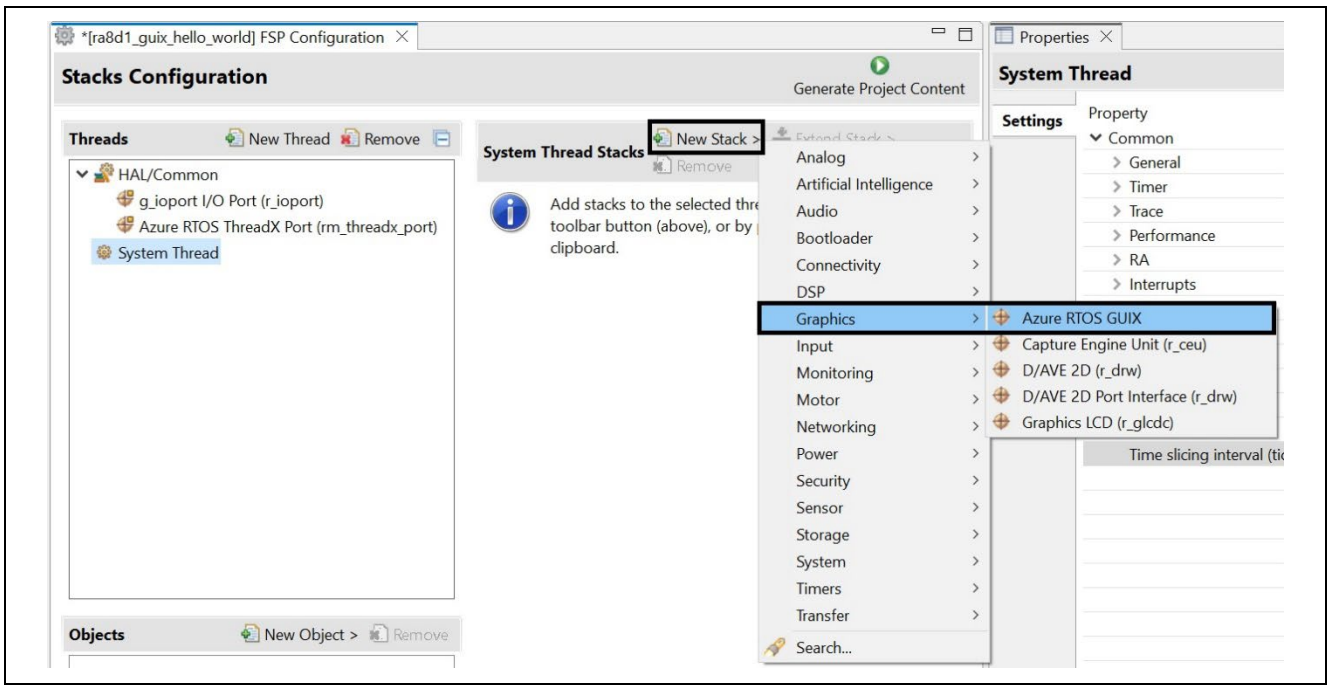


図 11. Azure RTOS GUIXの追加

9. **Azure RTOS GUIX (rm_guix_port)**と **GLCDC Display** のプロパティ設定

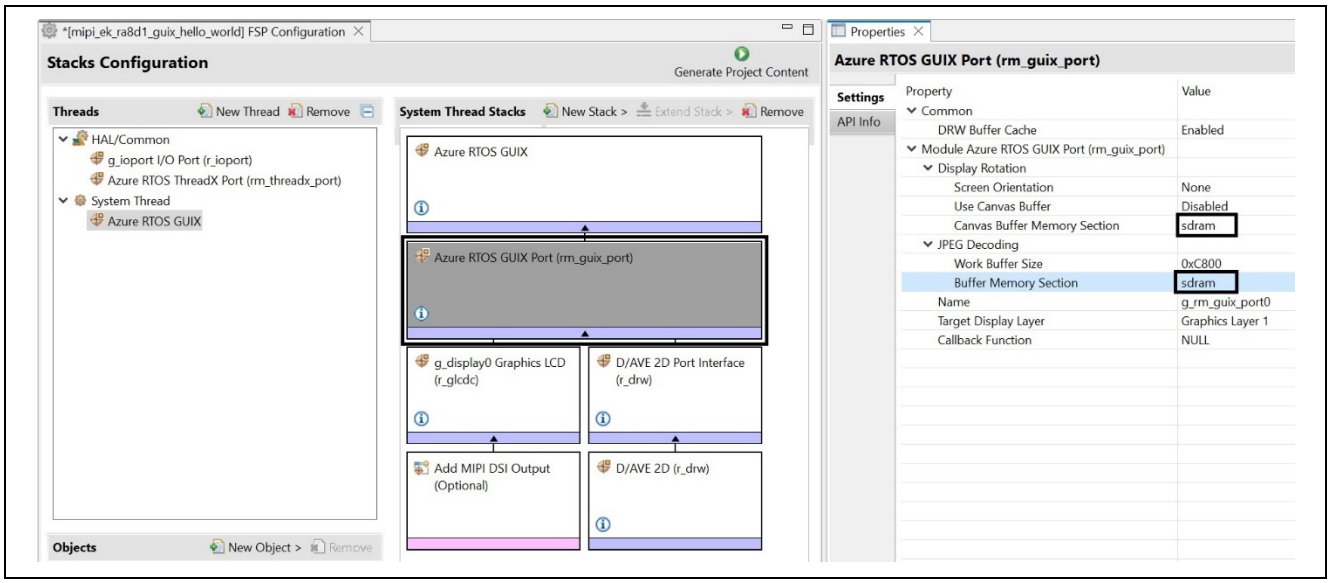


図 12.　ハードウェア用プロパティの設定と確認

10. **グラフィックス LCD のプロパティ設定**

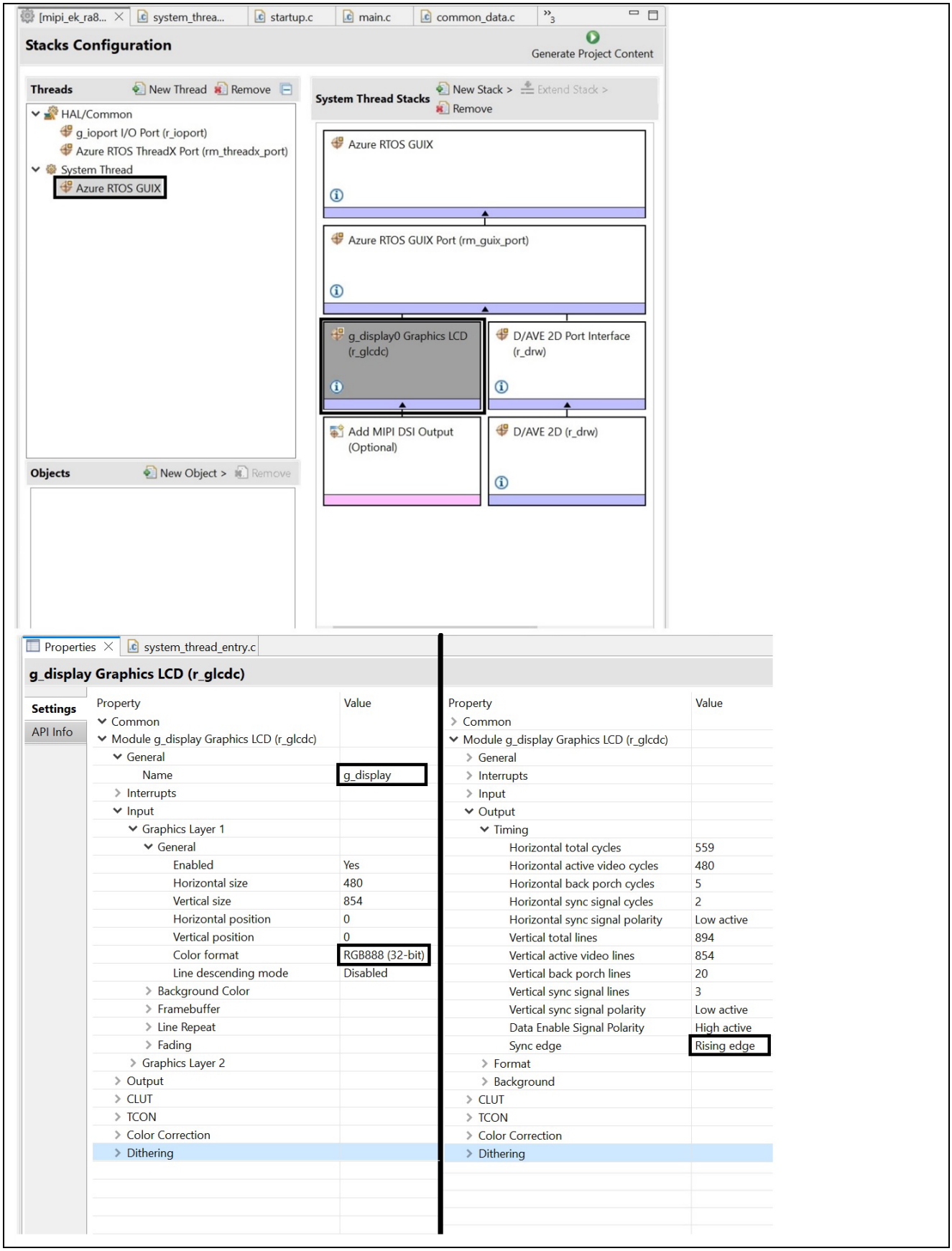


図 13.　GLCDC DISPLAY のプロパティ設定

11. r_glcdc スタックに添付されている Add オプションをクリックして MIPI DSI Output を追加し、**MIPI Graphics LCD(r_mipi_dsi)**モジュールのプロパティ設定を入力します:

注:　他の設定はすべて LCD ピクセル 480x854 のデフォルト値に設定されているため、ここでは詳細な説明は行いません。

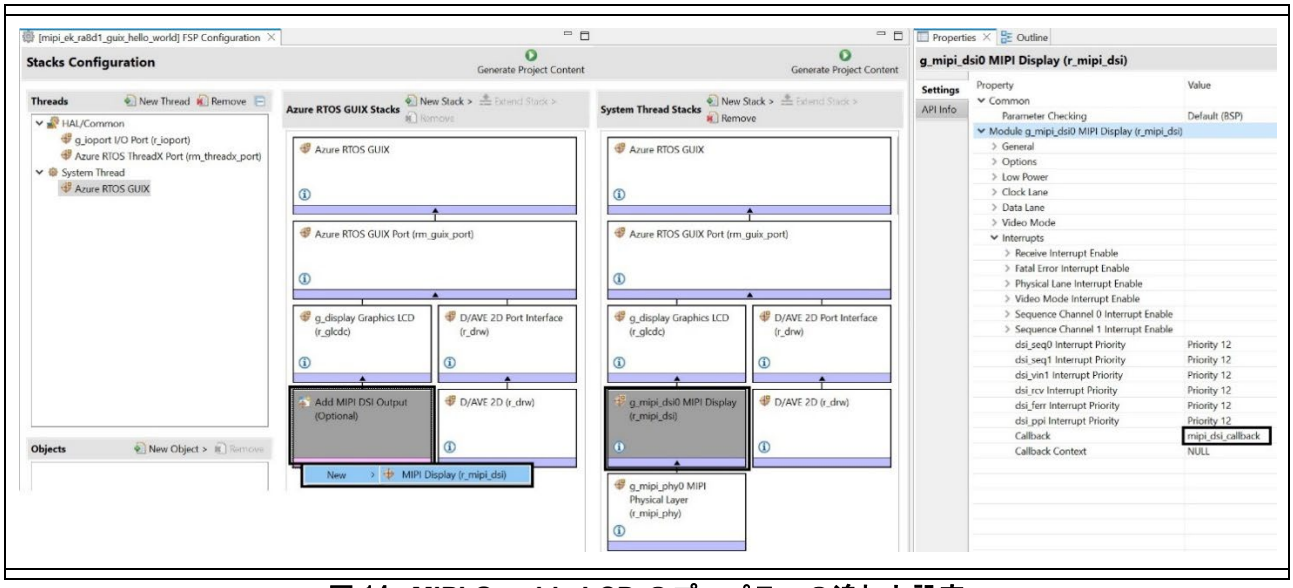


図 14.　MIPI Graphic LCD のプロパティの追加と設定

12.**New Stack**をクリックし、**Timer > PWM Timer (r_gpt)**を追加します。

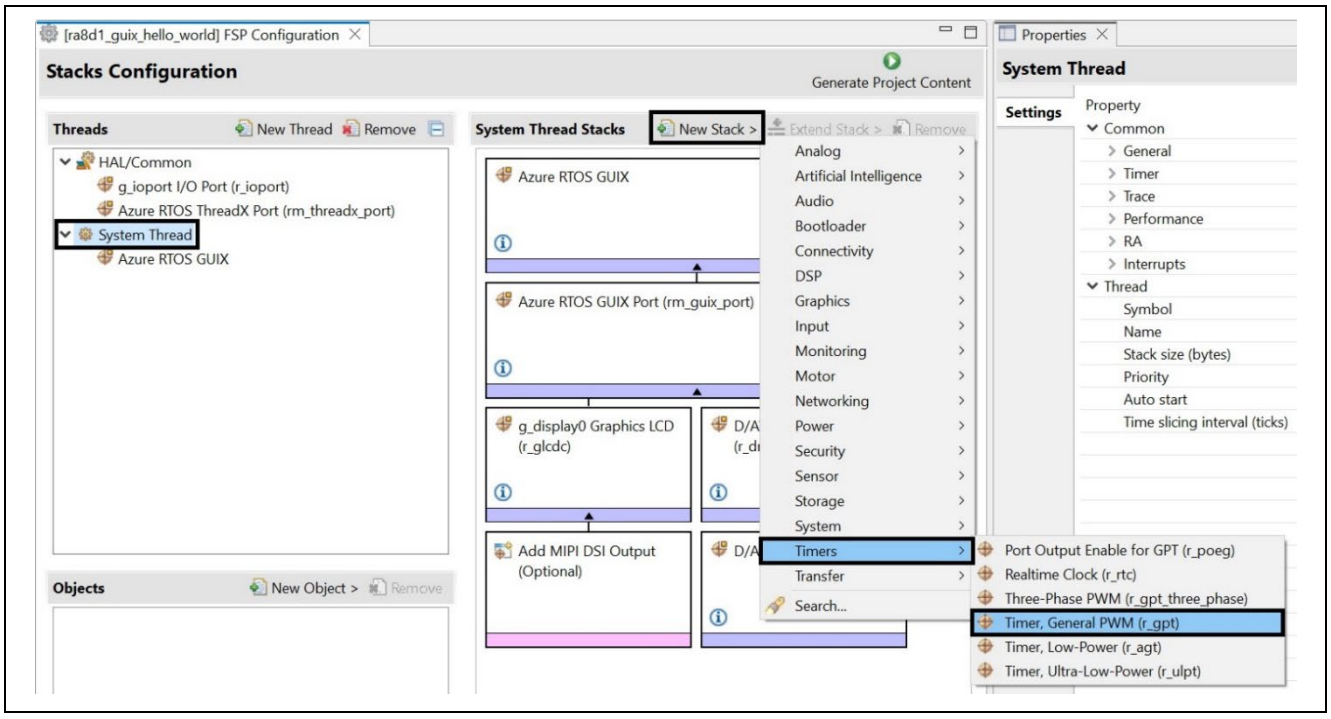


図 15.　タイマの追加

13.**タイマ**および **General PWM (r_gpt)**モジュールのプロパティを以下のように設定します。

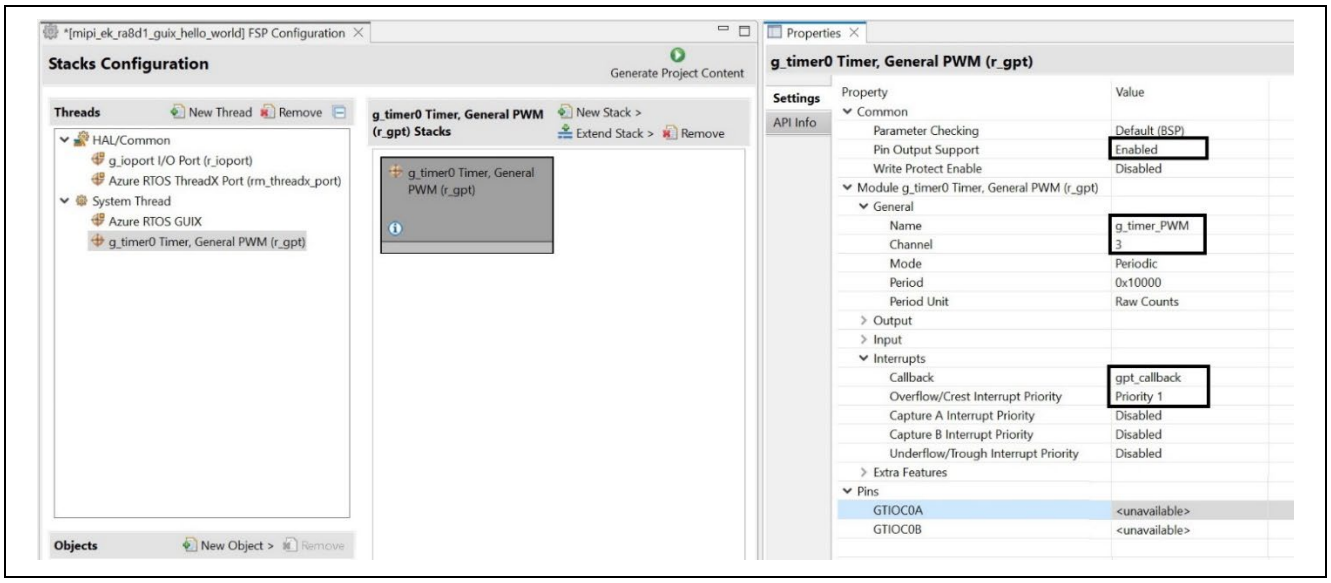


図 16. タイマのプロパティ設定

14.プロジェクトの出力ピンを設定します。**Pins** タブに移動します。**P404** に LCD パネルの **DISP_BLEN** 信号を設定し、**Mode** を **Output mode(Initial High)**に設定します。

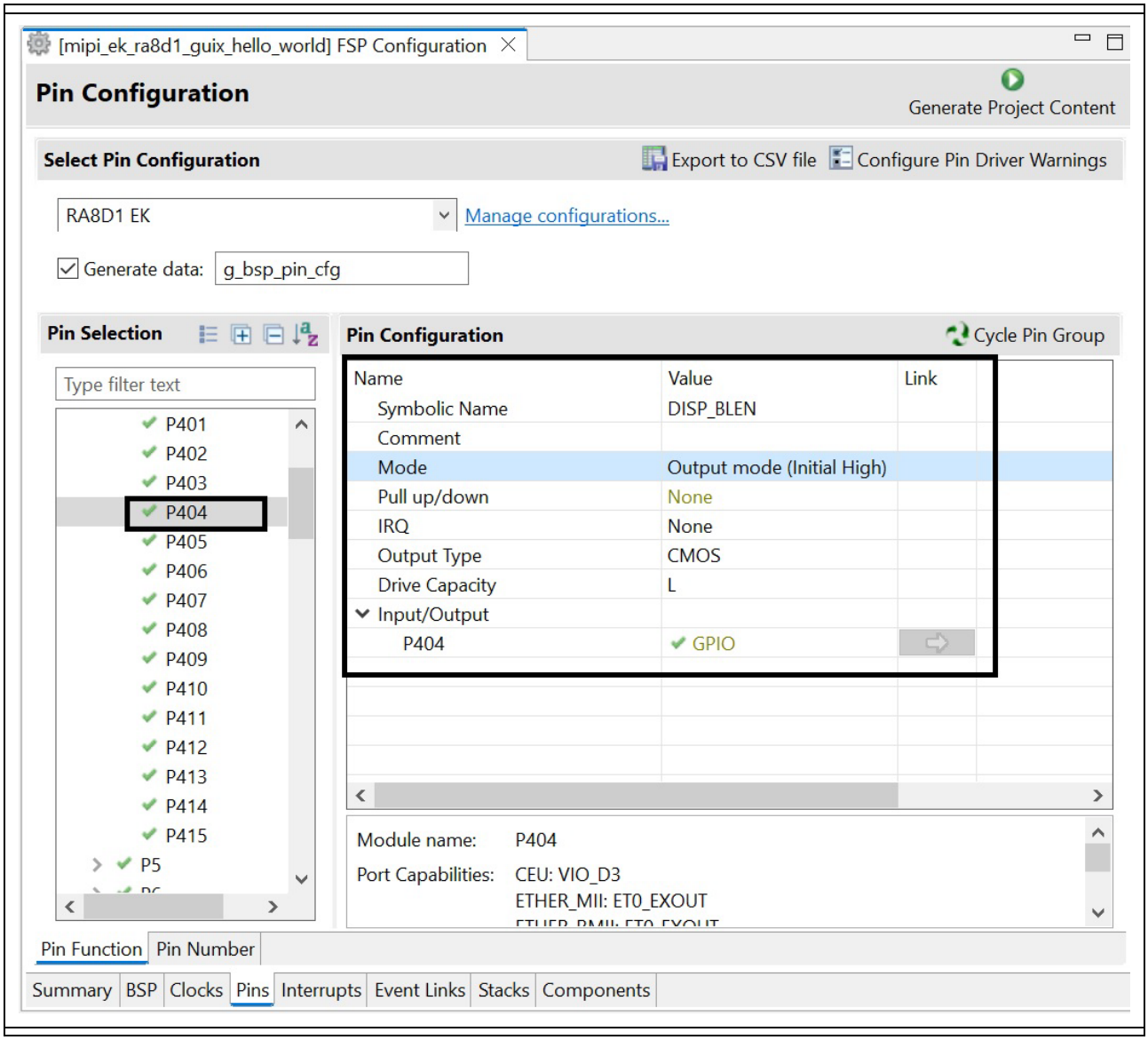


図 17. P404 に DISP_BLEN を設定

15. LCD パネルの LCD_BLEN ピンは、RA8D1 ボードの P404 DISP_BLEN に接続されます。

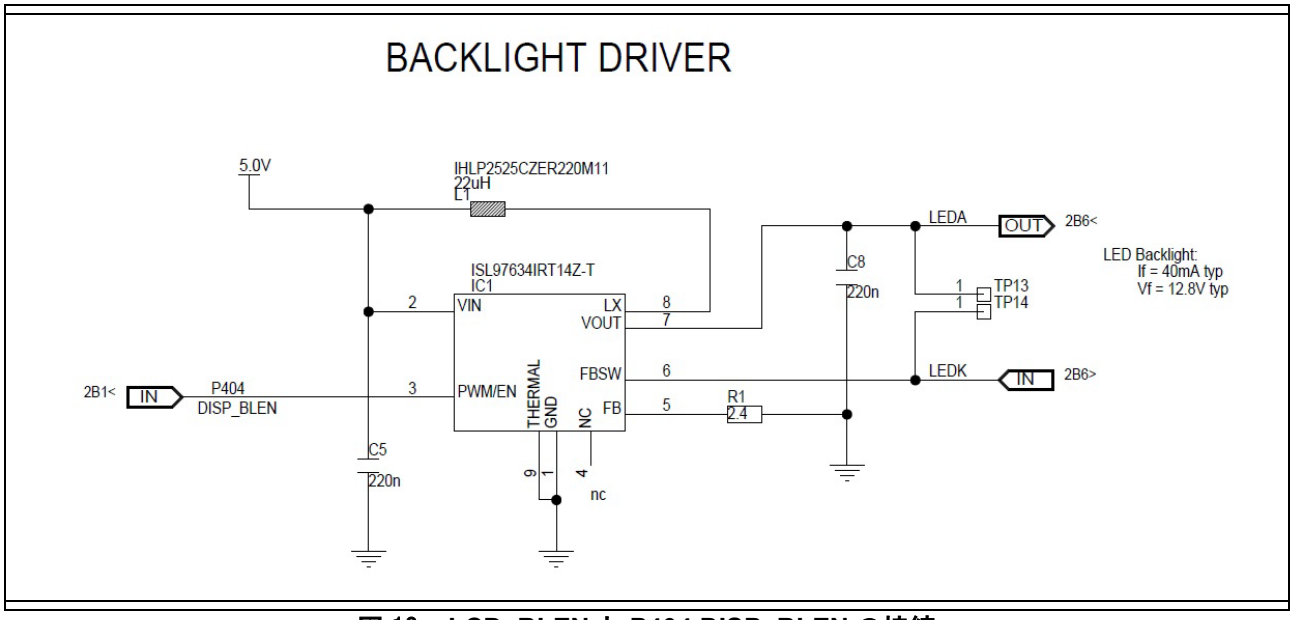


図 18.　LCD_BLEN と P404 DISP_BLEN の接続

16. システムスレッドの詳細については、**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。

```
/* 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);
```

図 19. system_thread_entry.c の gpt_timer_PWM

3. gt911 タッチ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。

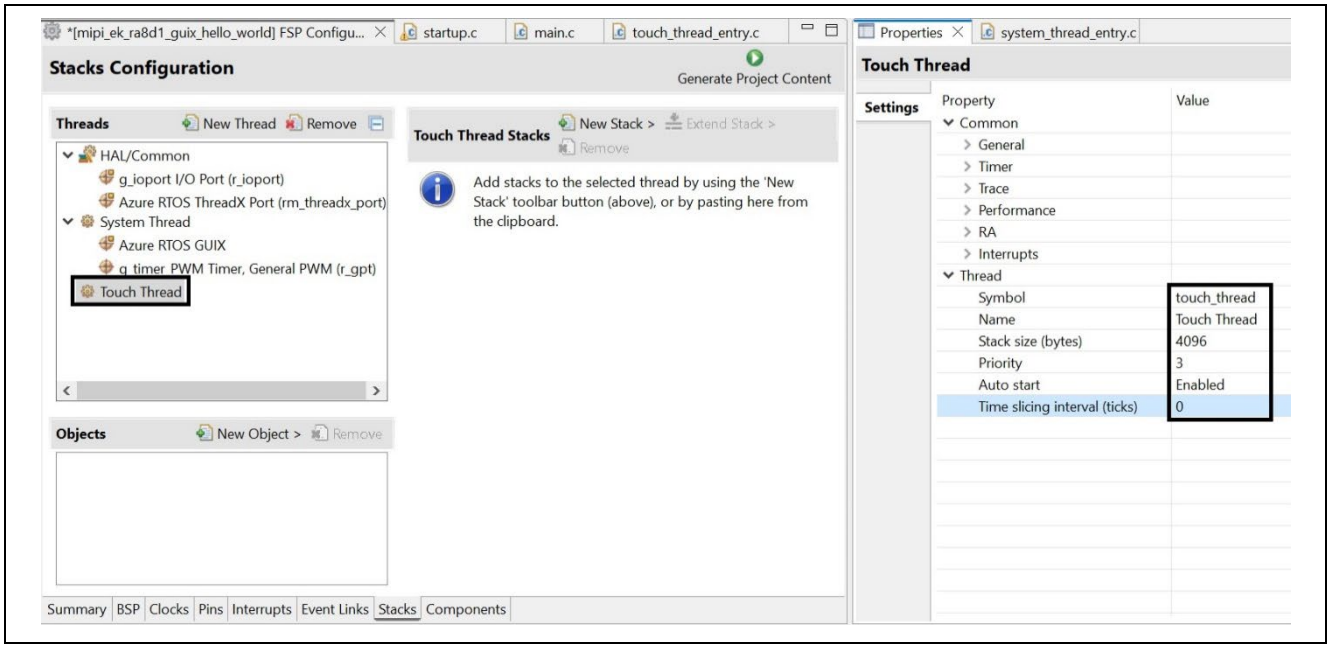


図 20. タッチスレッドプロパティの追加と設定

2. Threads ウィンドウから、HAL/Common を選択し、**New Stack** をクリックして、**Input > External IRQ (r_icu)**モジュールを追加します。

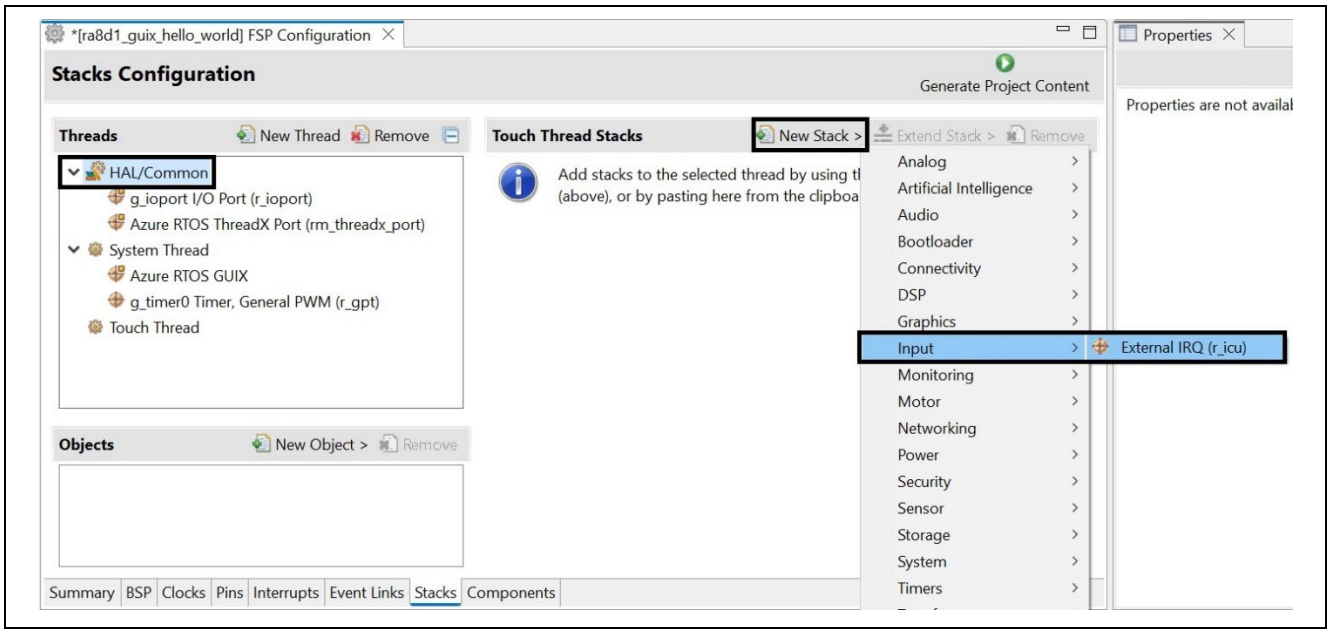


図 21. External IRQ の追加

3. モジュール名を **g_int_gt911** とし、以下の画像のプロパティに従ってデフォルト設定を変更します。矢印  をクリックして、**Pins** タブに移動し P510 の設定をします。

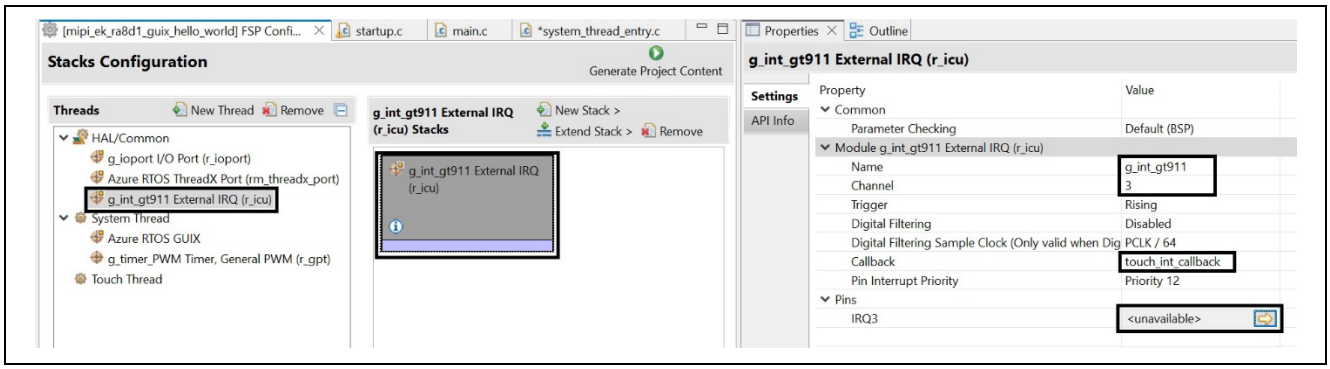


図 22. External IRQ のプロパティ設定

4. **DISP_INT** 信号の **P510** の設定を変更し、Operation Mode が **Custom**、IRQ3 が **P510** になるようにします。

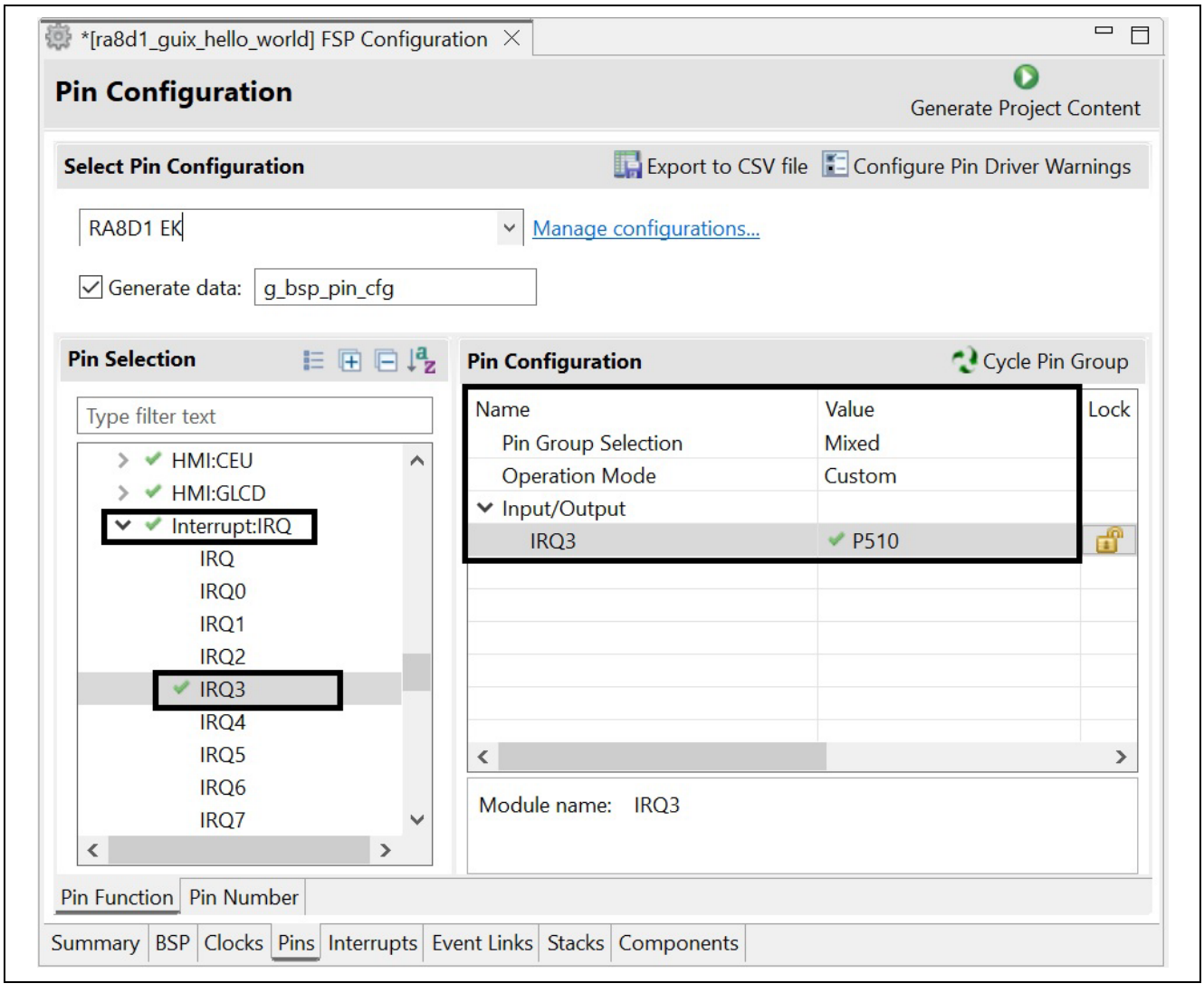


図 23.　P510 に IRQ3 を割り当て

5. **Stacks** タブに戻ります。Threads ウィンドウから HAL/Common を選択し、**New Stack** をクリックして **Connectivity > I2C Master (r_iic_master)**モジュールを追加します。

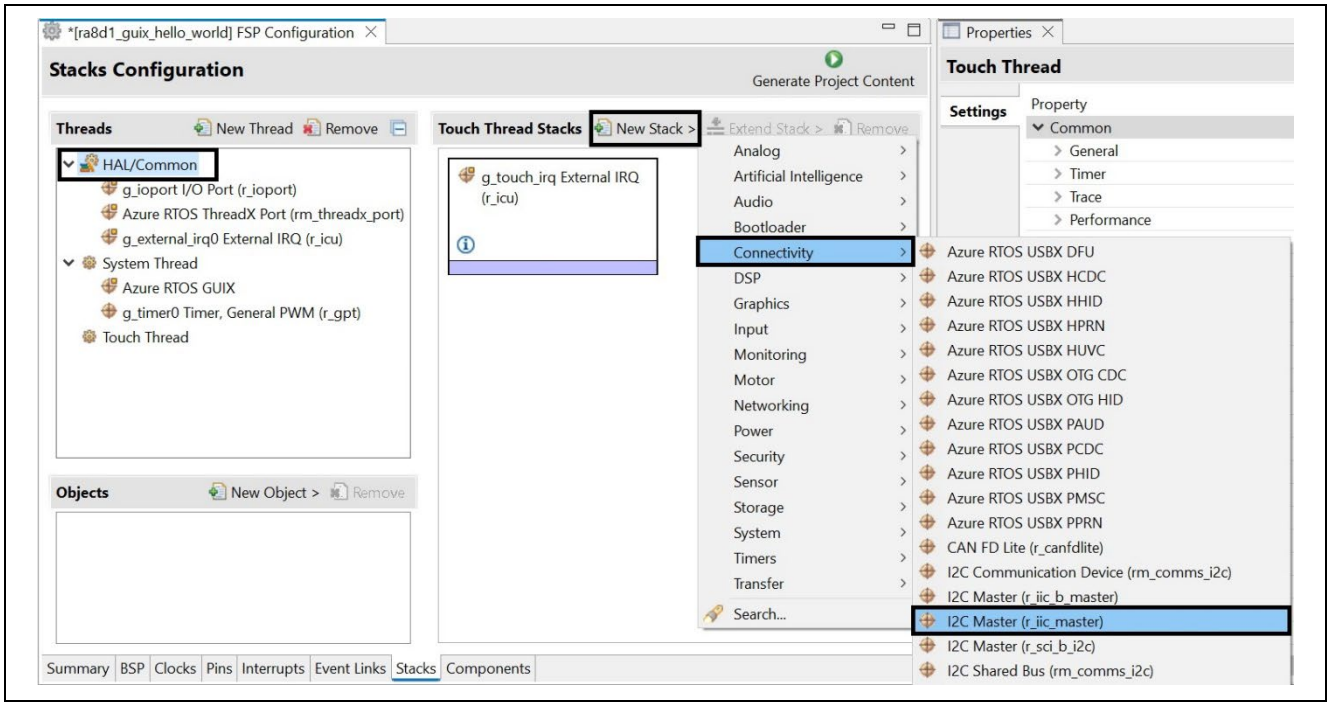


図 24.　タッチ用 I2C マスタを追加

6. モジュール名を **g_i2c_gt911** とし、以下の画像のプロパティに従ってデフォルト設定を変更します。矢印  をクリックして、**Pins** タブに移動し、**SCL1** と **SDA1** を設定します。

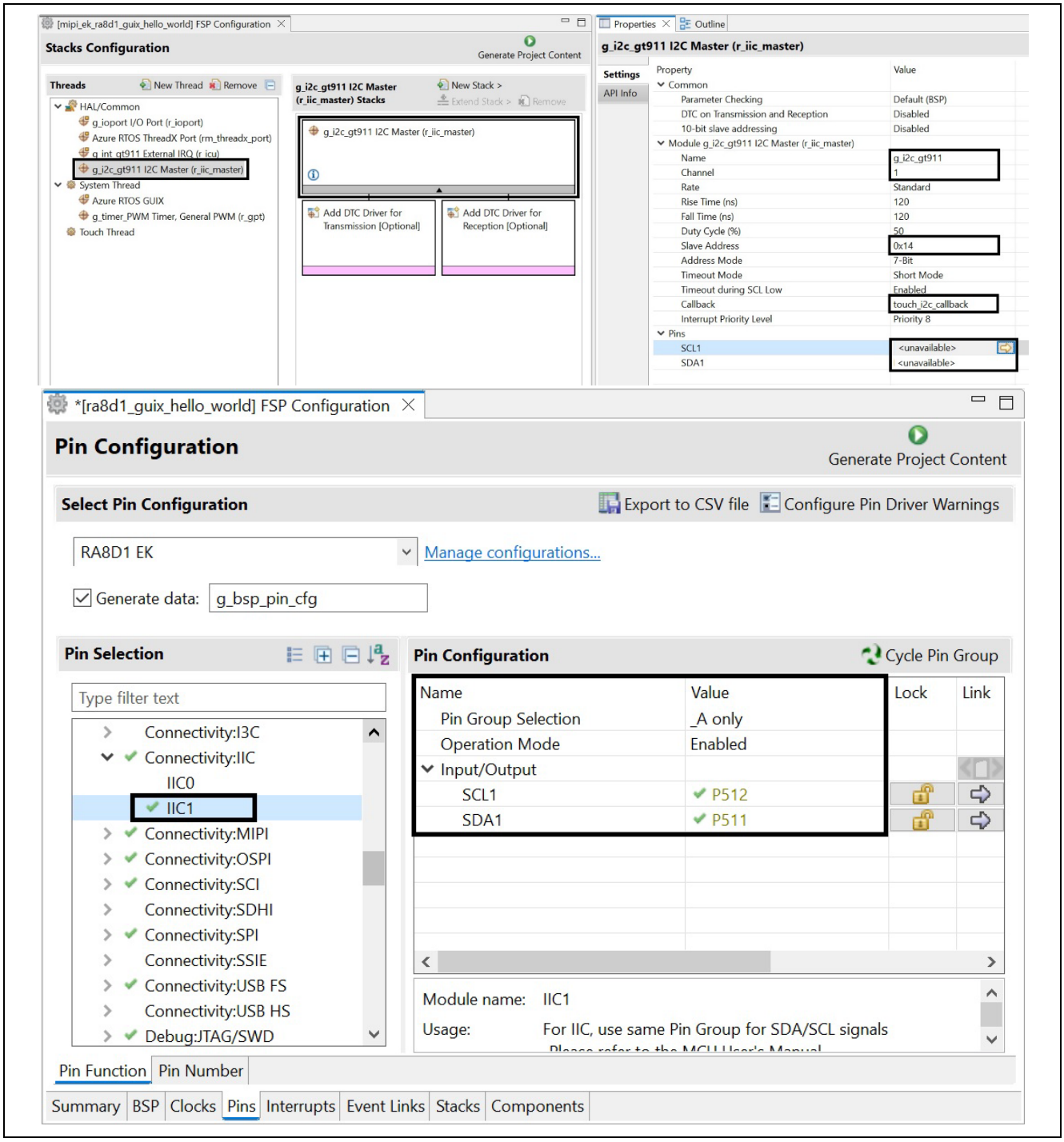


図 25.　モジュール名とプロパティの設定

7. **New Object** をクリックし、**セマフォ**を追加します。このセマフォは、gt911 タッチ割り込みが発生したことを示します。

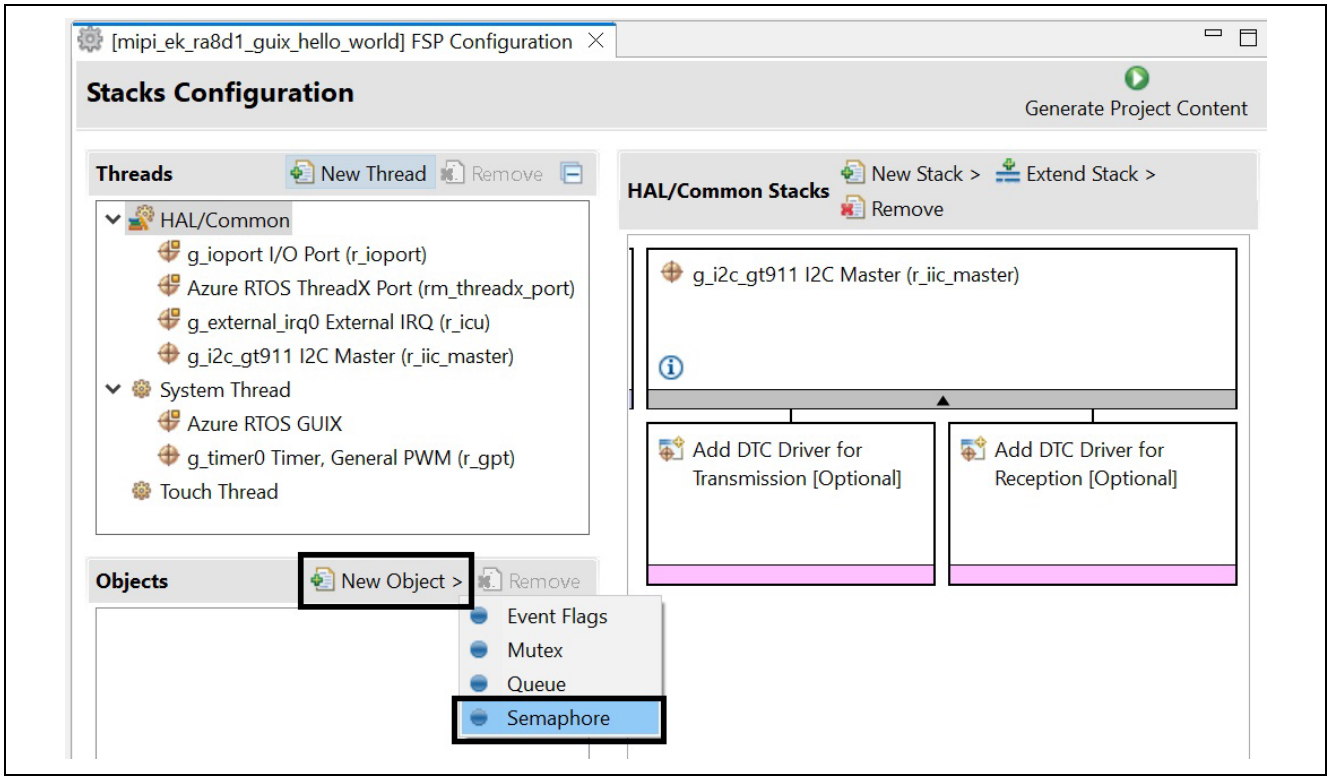


図 26.　新規セマフォの追加

8. **g_semaphore_gt911_irq** と名前を付け、次のプロパティを設定します。

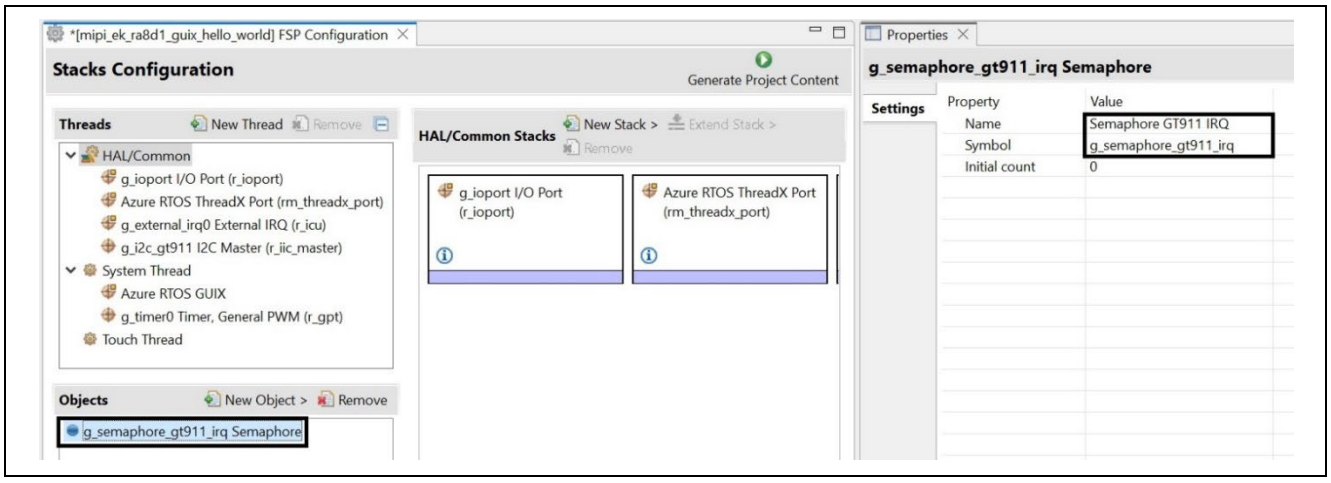


図 27.　タッチ セマフォの追加と名前の設定

9.**New Object** をクリックして別の**セマフォ**を追加します。このセマフォは i2c のタイミング制御に使用されます。

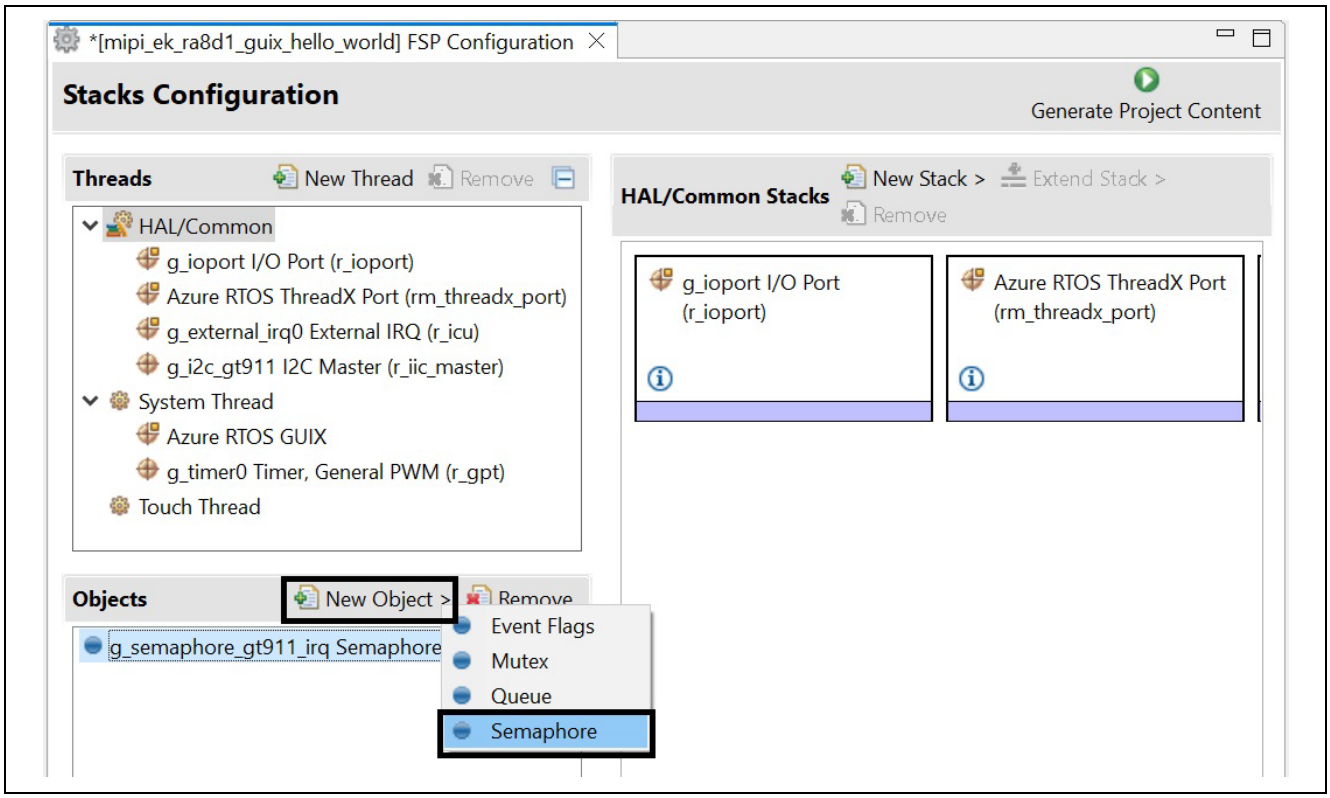


図 28.　別の新規セマフォの追加

10. **g_semaphore_gt911_i2c** と名前を付け、次のプロパティを設定します。

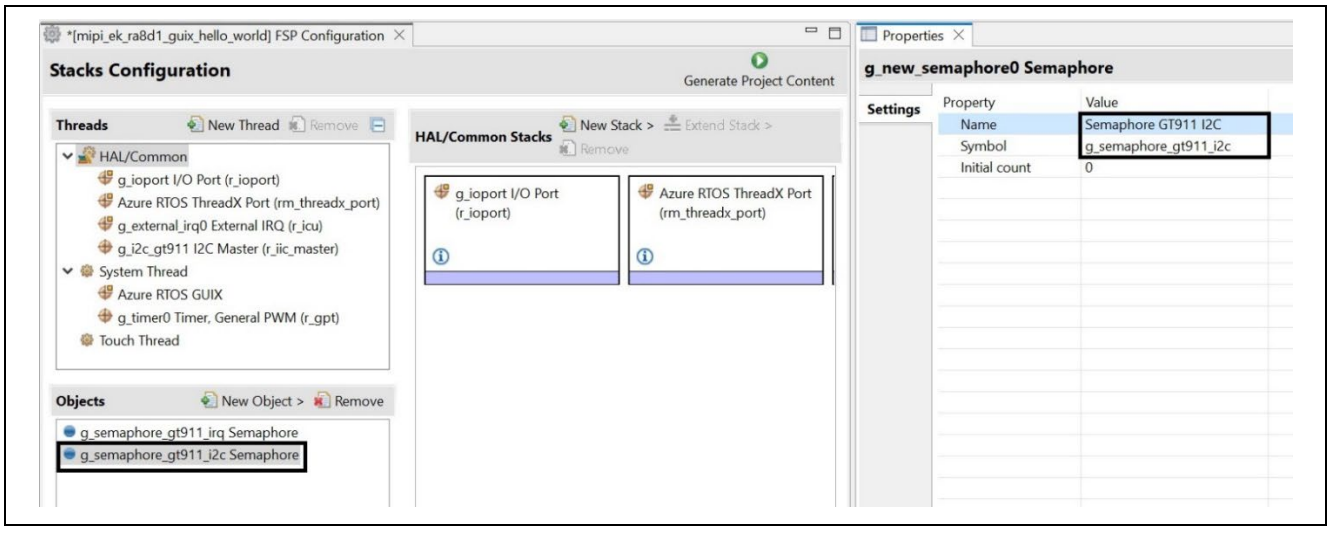


図 29.　I2C セマフォの追加と名前設定

11. **New Object** をクリックし、**Event Flags** を追加します。このイベントフラグは gt911 ドライバによって使用されます。

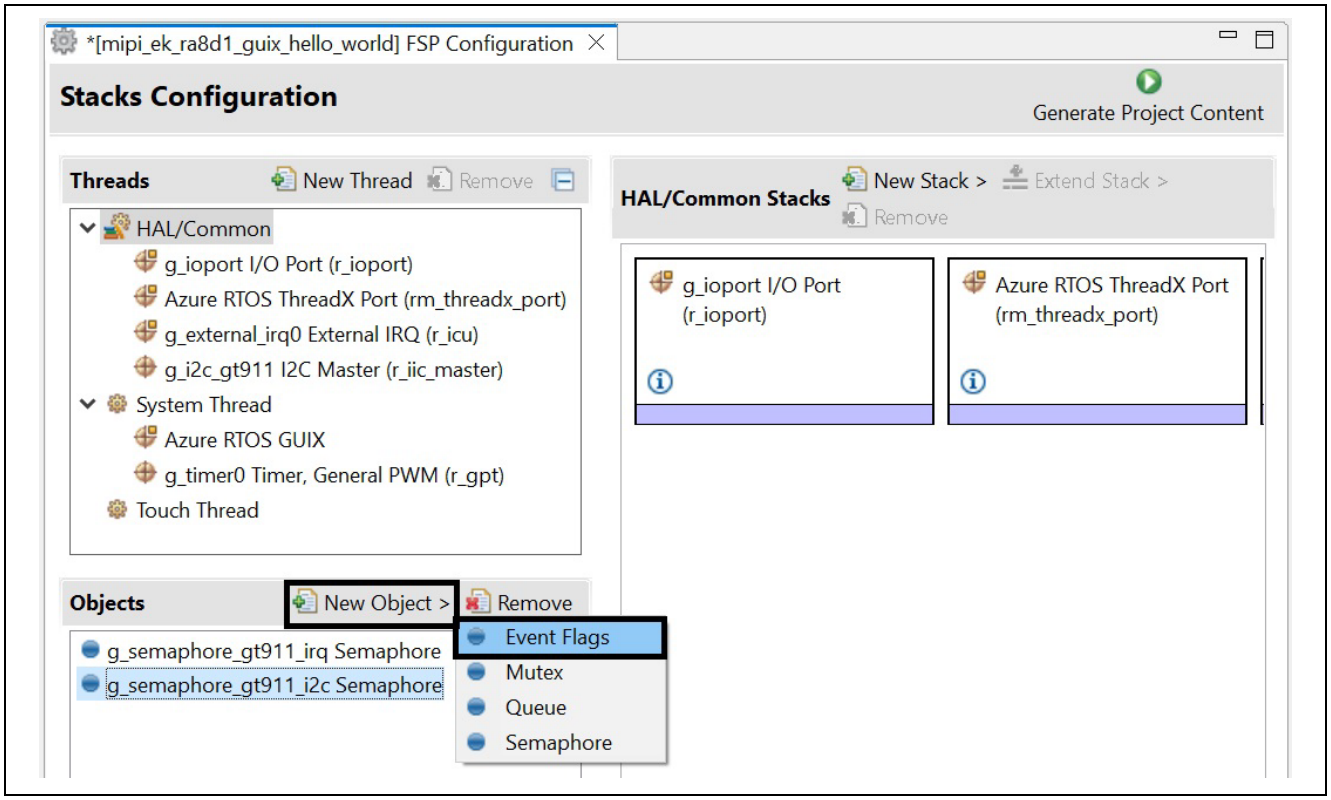


図 30. イベントフラグの追加

12. **g_event_flags_gt911** という名前を付け、以下のプロパティを設定します。

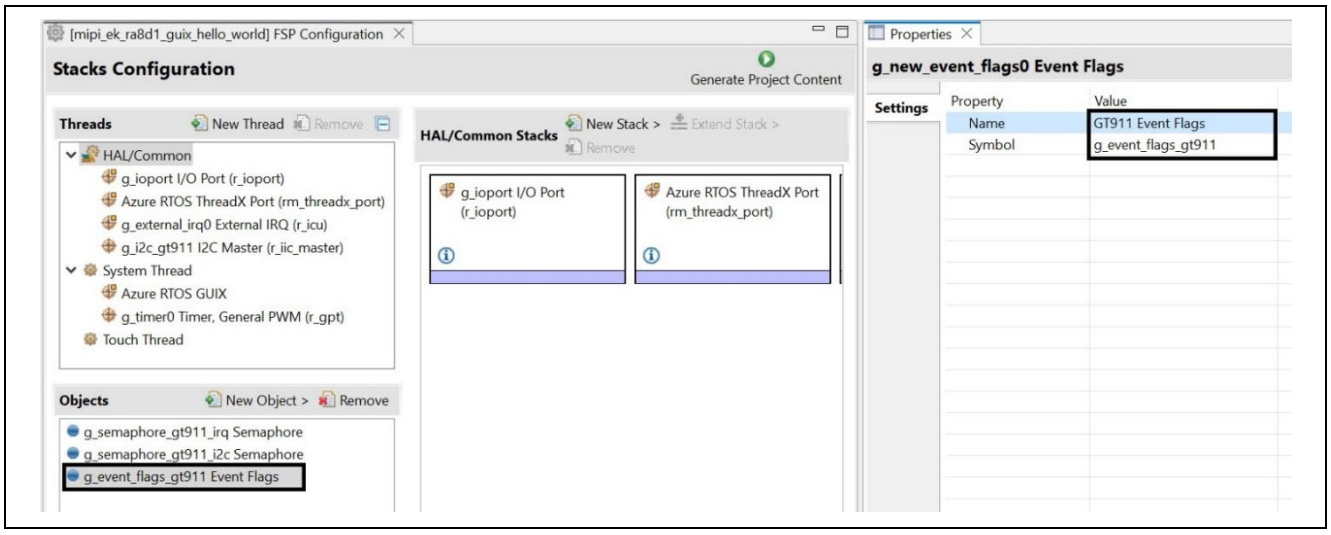


図 31.　イベントフラグの名前とプロパティ設定

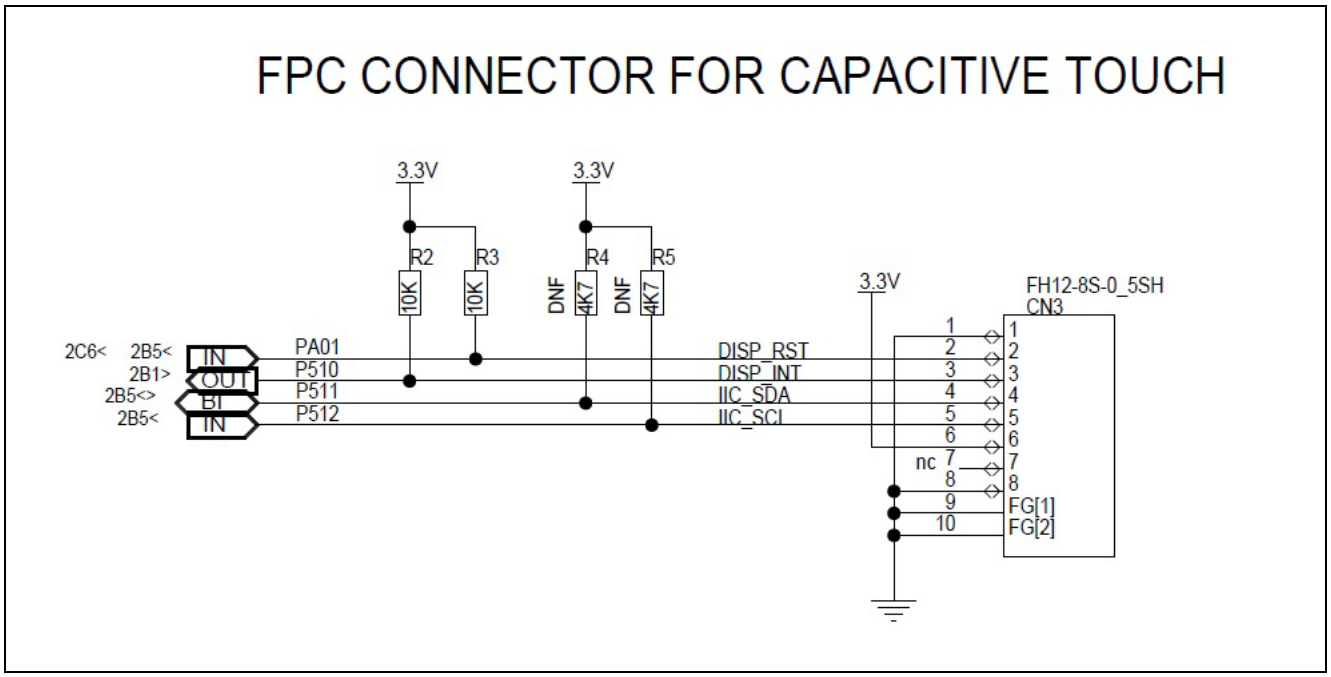


図 32. MIPI LCD とタッチピンの接続図

図 32に示されているピンは、MIPI 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_gt911 は提供のソースフォルダ内にあります。
- PA01 は MIPI 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. 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

```
/* 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);
}
```

図 33.　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。

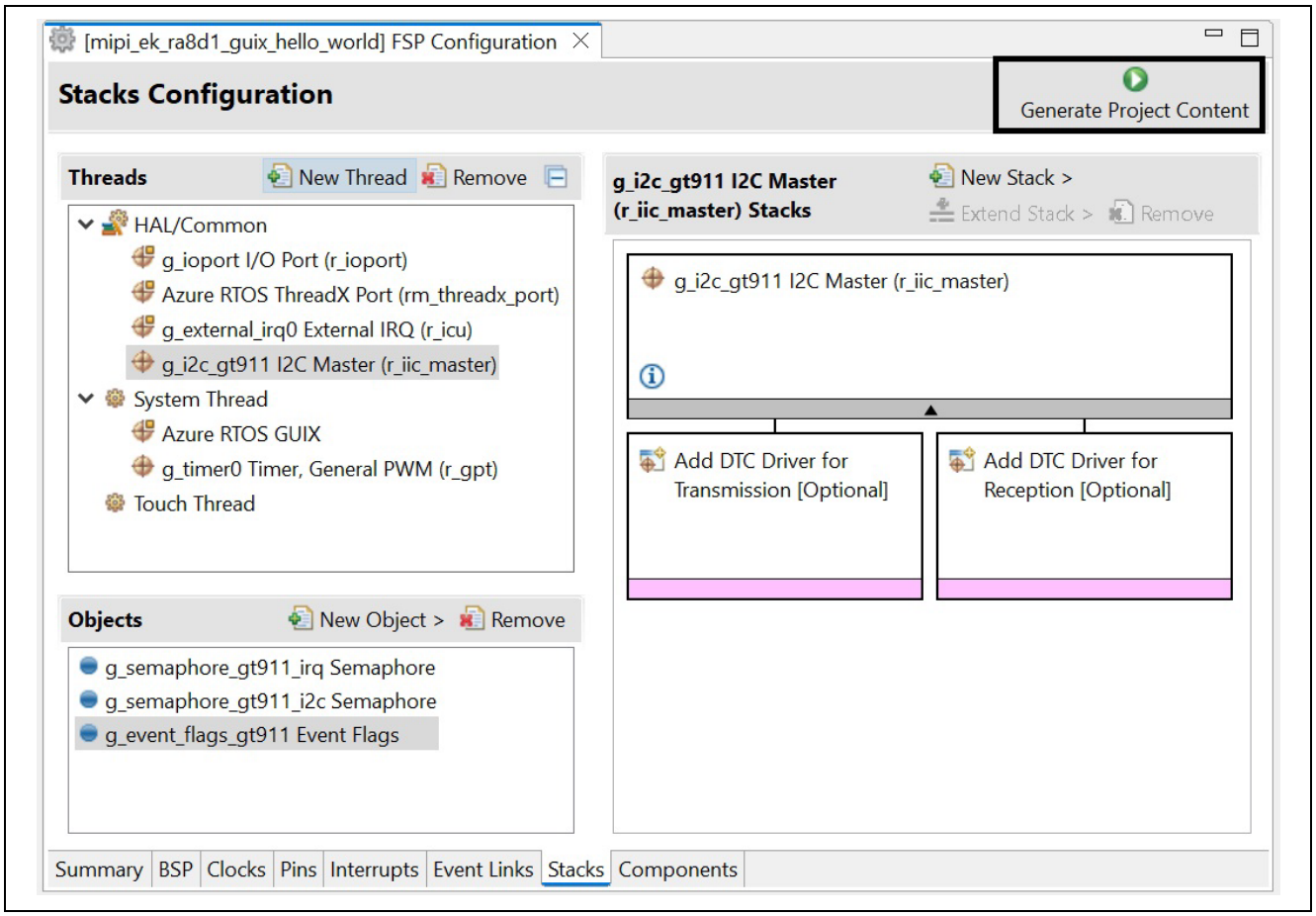


図 34. プロジェクトコンテンツの生成

15. 提供のフォルダ Source.zip を解凍し開きます。6 つのファイルと 2 つのフォルダ SEGGER_RTT、touch_gt911 をプロジェクト **mipi_ek_ra8d1_guix_hello_world** のフォルダ src にコピーします。既存のファイルを上書きするように求められた場合は、YES を選択してください。
    SEGGER_RTT フォルダ
    - touch_gt911 folder
    - common_utils.h
    - hal_entry.c
    - system_thread_entry.c
    - system_thread_entry.h
    - touch_thread_entry.c
    - windows_handler.c

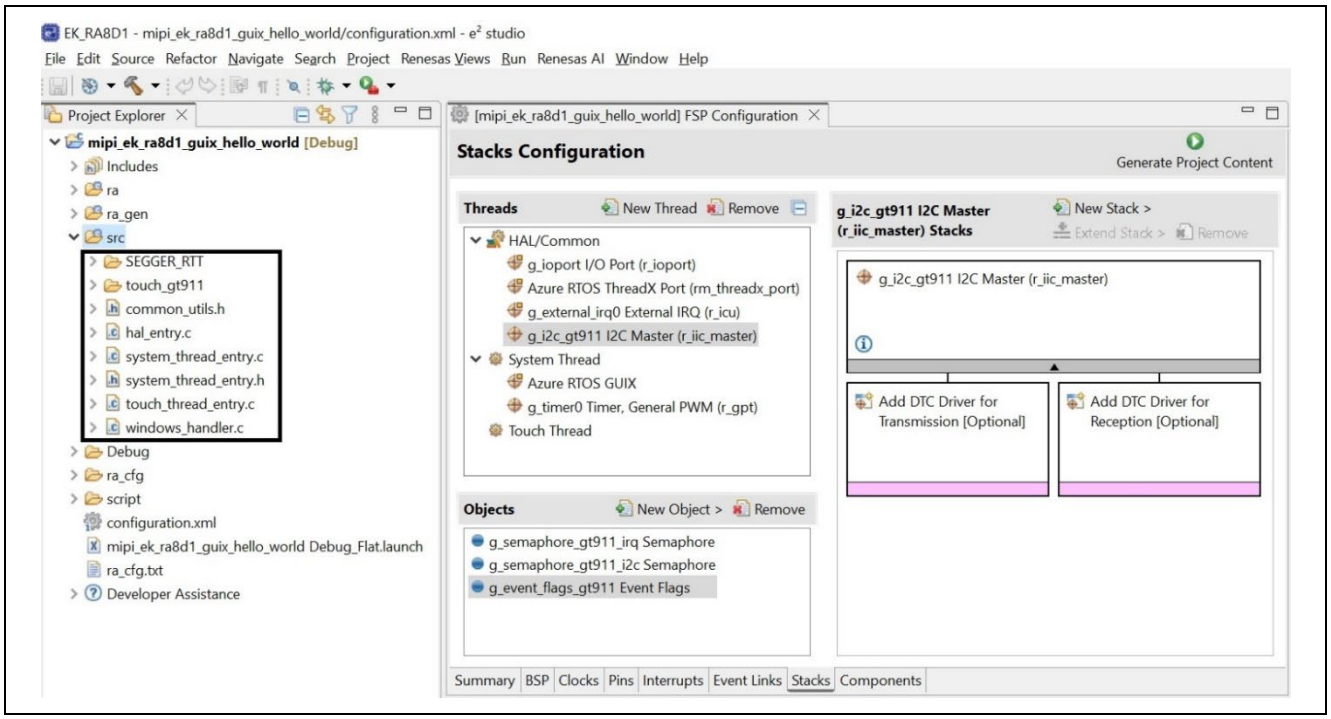


図 35.　e² studio プロジェクトの src フォルダに Source.zip からフォルダとファイルのコピー

4. Azure RTOS GUIX Studio project 用のフォルダの作成

1. mpi_ra8d1_guix_hellow_world プロジェクト内に Azure RTOS GUIX Studio project 用のフォルダを作成します。
2. 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。このフォルダには、GUIX Studio グラフィックアプリケーションが生成するファイルが保存されます。

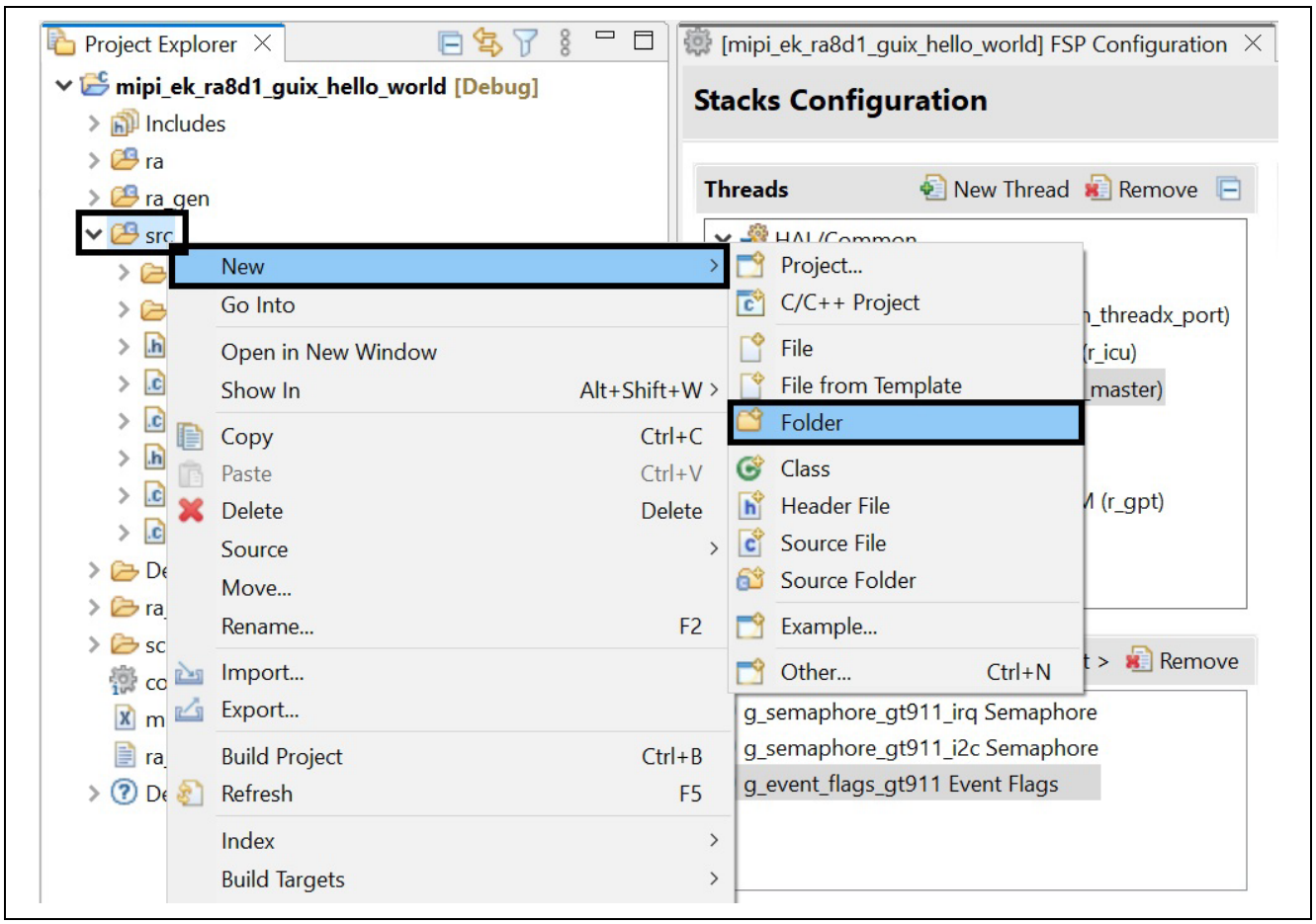


図 36.　フォルダ guix_gen の作成

3. 最上位のプロジェクトレベルに新規フォルダを作成し、下の図に従って guix_studio という名前を付け、**Finish** をクリックします。このフォルダに GUIX Studio プロジェクトが保存されます。

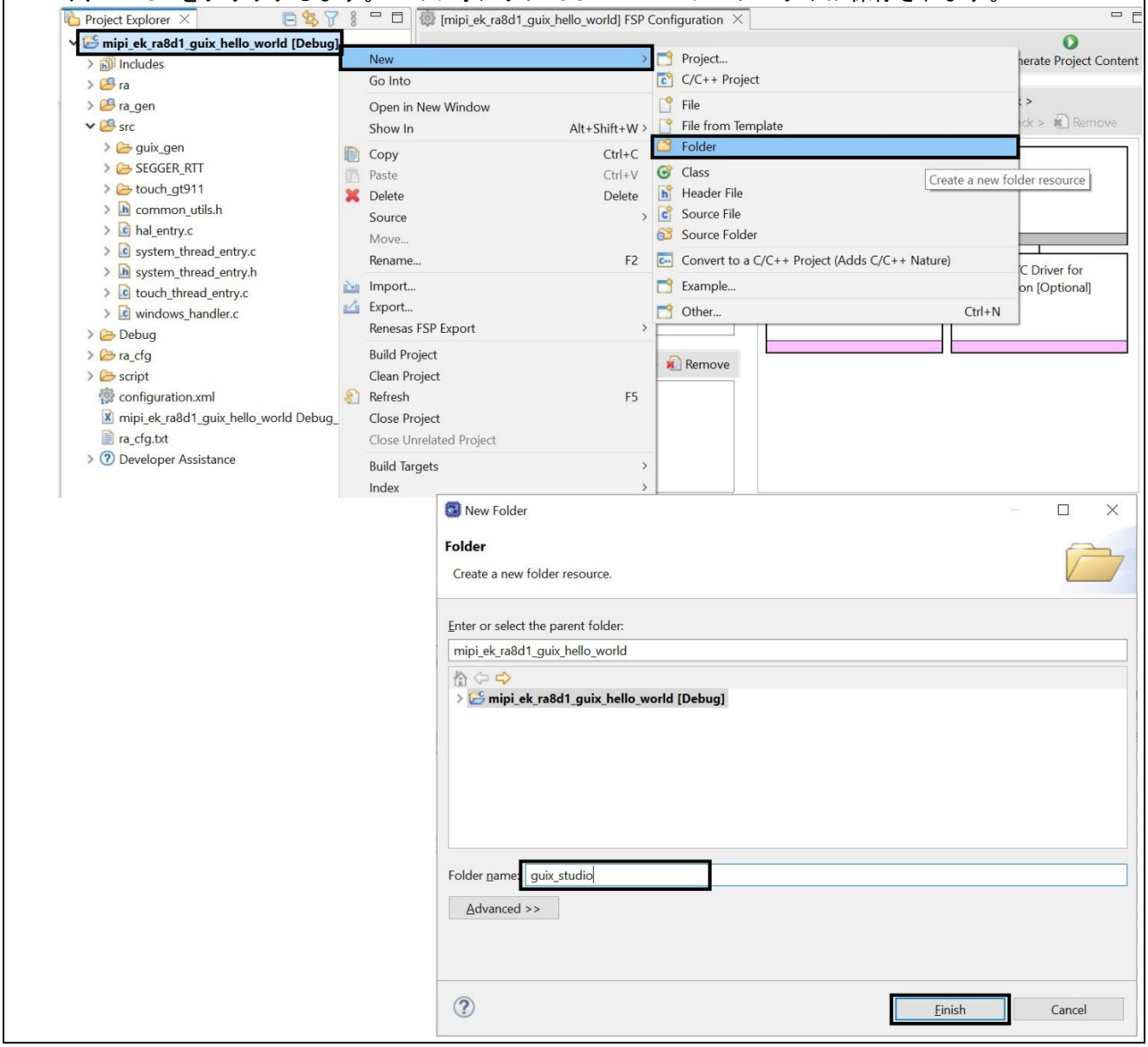


図 37.　フォルダ guix_studio の作成

4. 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。

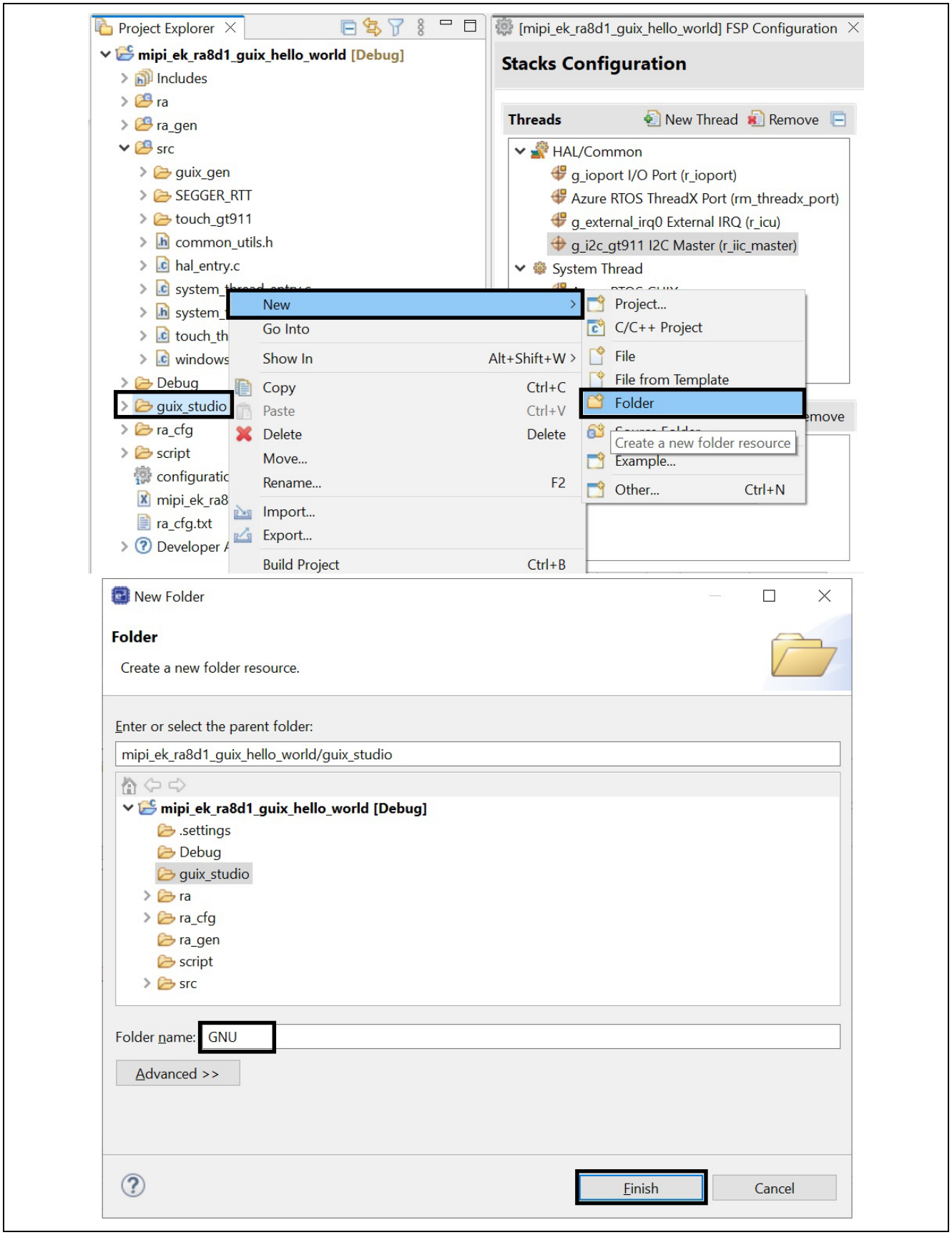


図 38.フォルダ GNU の作成

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。

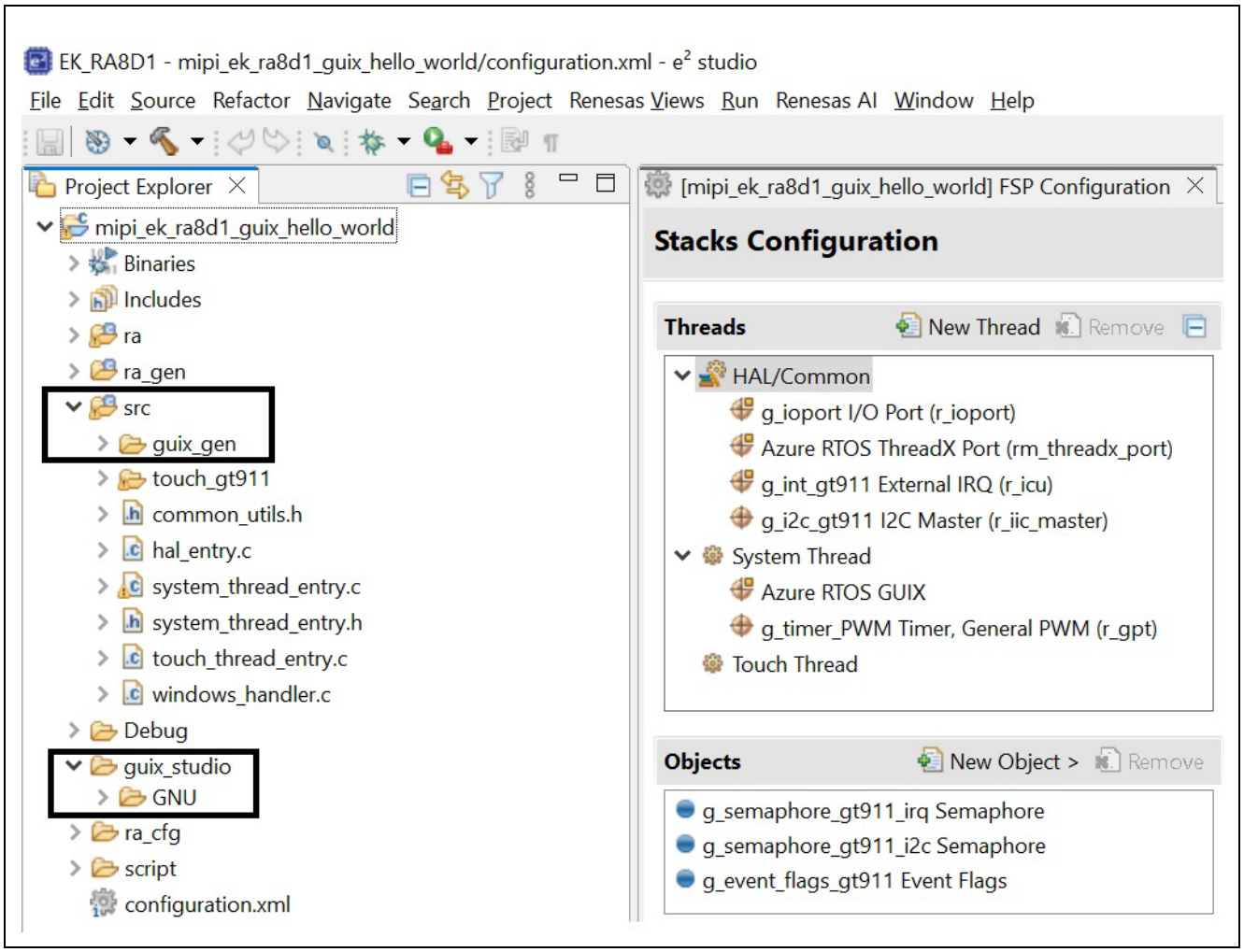


図 39.　プロジェクトのフォルダ構造

5. Azure RTOS GUIX Studio を使用して GUI ウィンドウの作成

1. Azure RTOS GUIX Studio v6.3.0.1 以降を開きます。

図 40. GUIX Studio アイコン

2. 次のサブステップでは、Hello World という新規プロジェクトを作成する手順を説明します。
   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:e² studio で作成したフォルダの場所を参照します。
   mipi_ek_ra8d1_guix_hello_world\ guix_studio\GNU を以下の画像のように指定します。
   D. **OK** ボタンをクリックし、**Save** ボタンをクリックして選択内容を確定します。

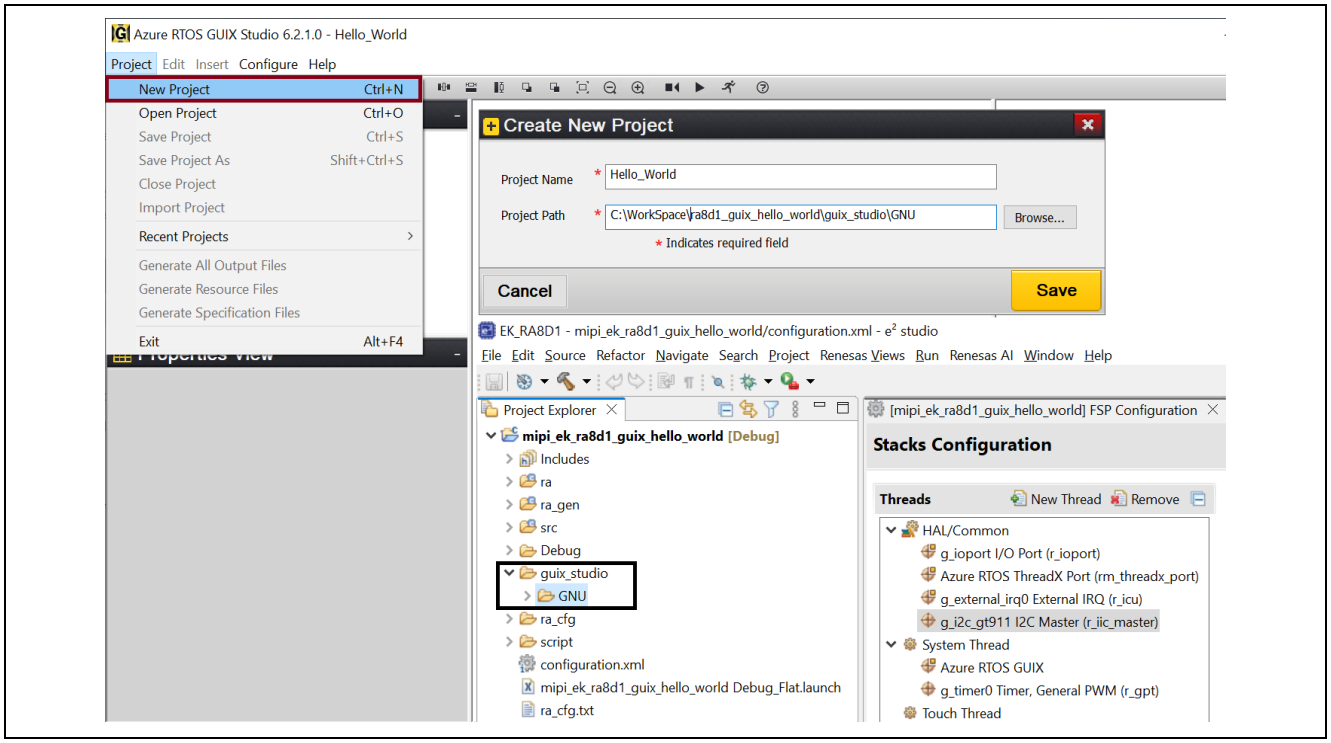


図 41. GUIX Studio で Hello World プロジェクト作成

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 42** 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、プロジェクト設定が以下の画像に反映されている内容と一致したら、**Save** をクリックします。

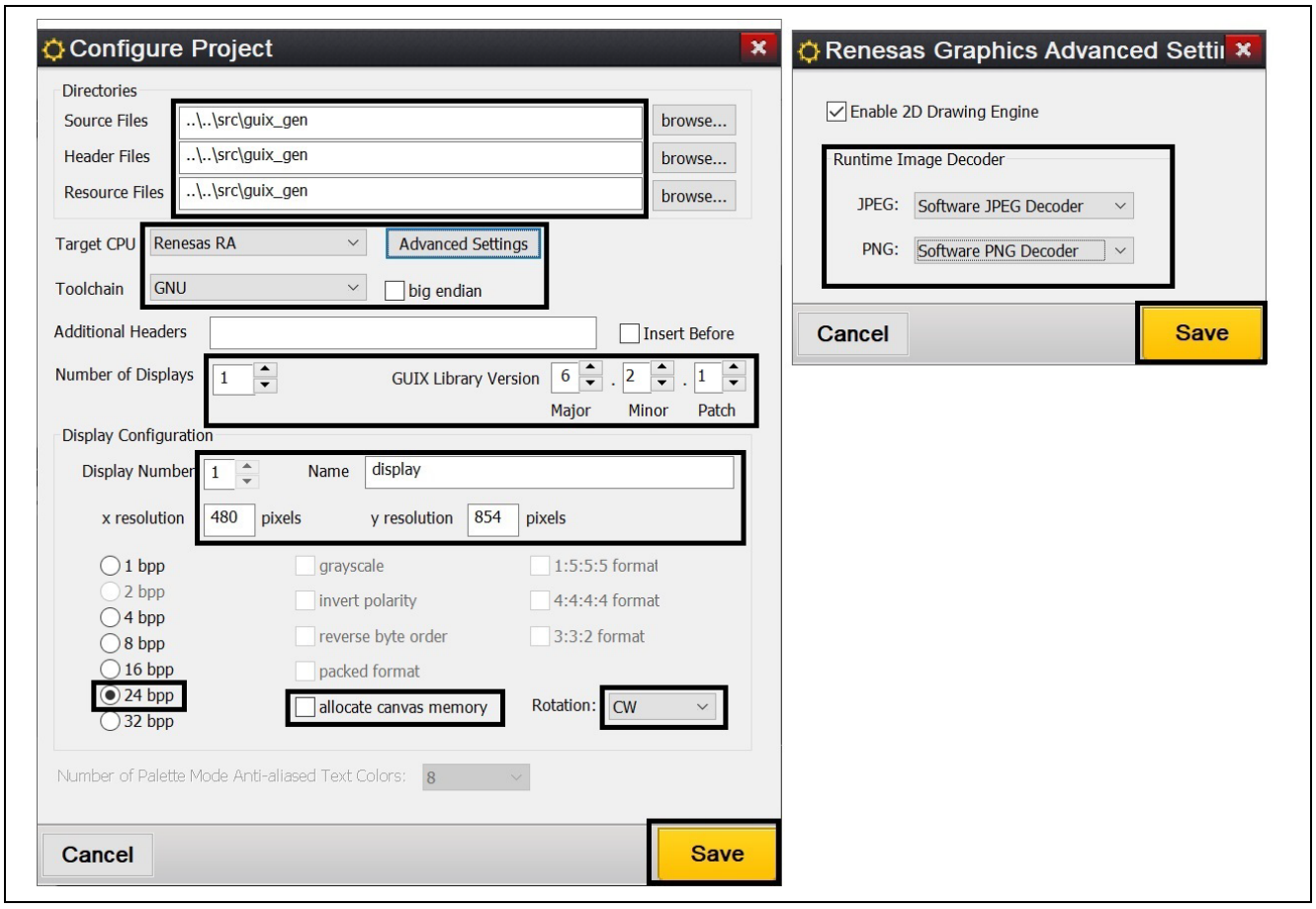


図 42.　新規プロジェクト Hello World 詳細設定

4. 新規プロジェクト Hello World のデフォルトビューは以下の画像のようになります。

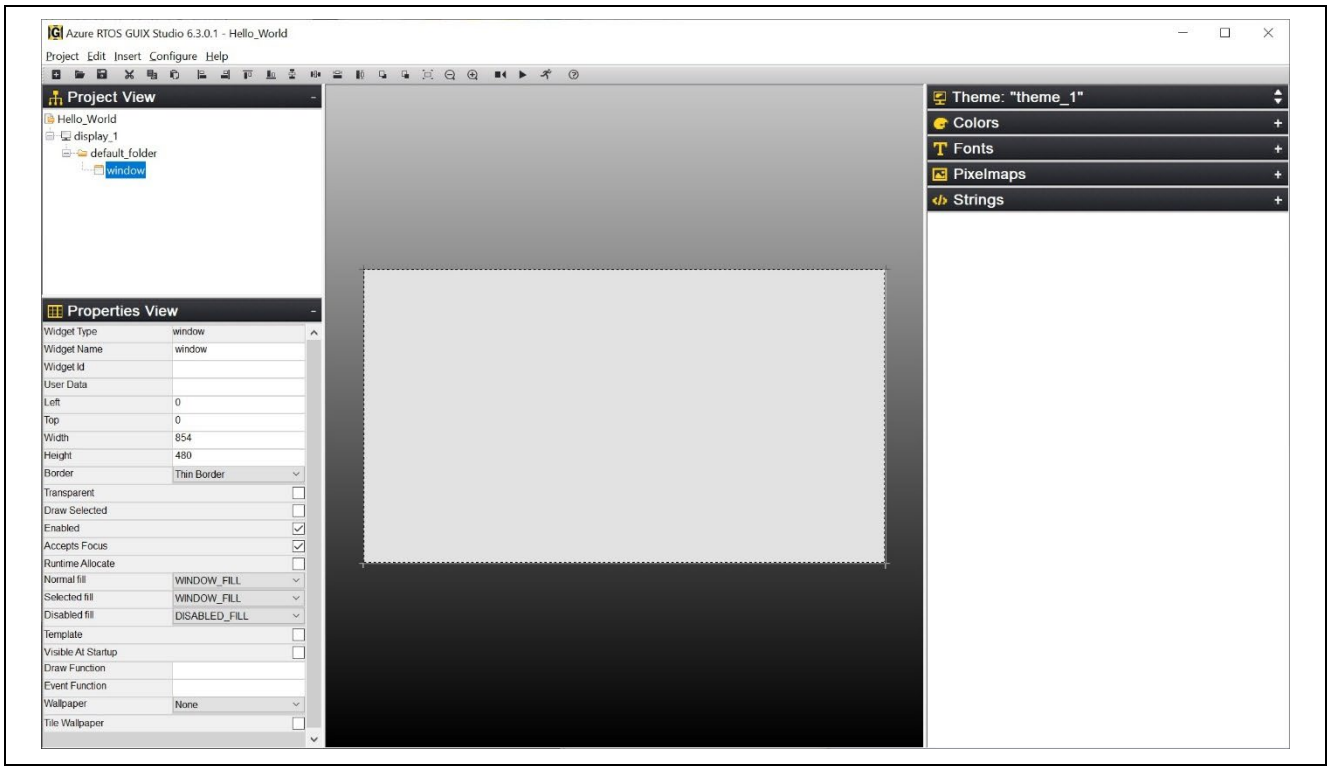


図 43. 新規プロジェクト Hello World 設定後のデフォルトビュー

5. **Project View** で **Window1** を選択し、**Properties View** で **Window1** の設定が以下の画像と一致することを確認します。

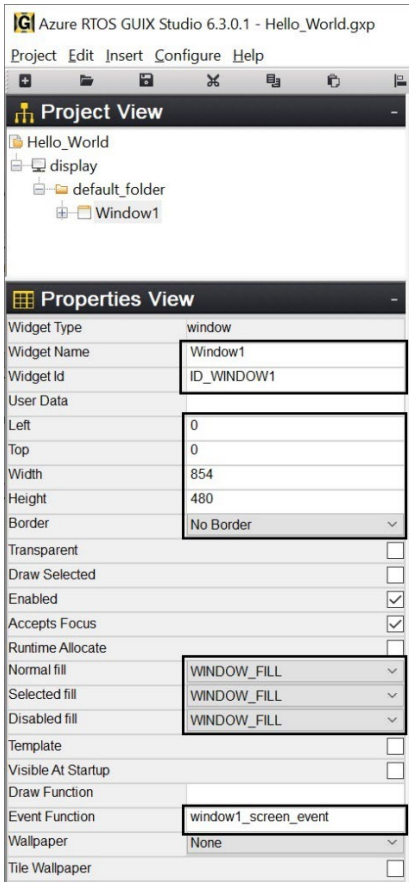


図 44.　Window1 のプロパティ設定

6. 以下の画像のように、String ID を追加するには、**Strings** をクリックします。

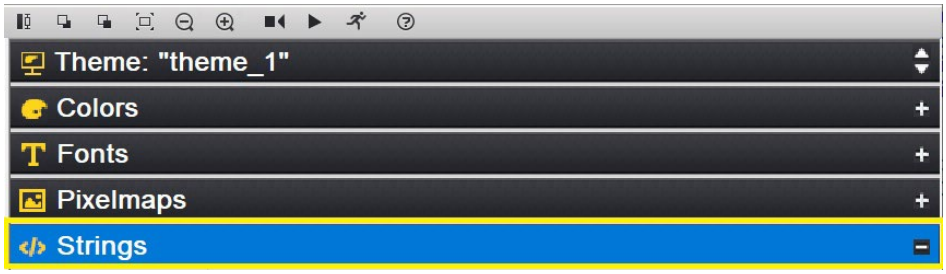


図 45.　Strings

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

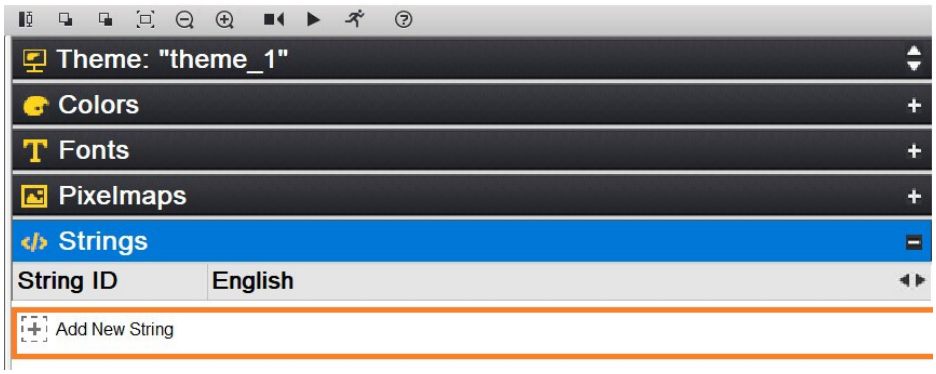


図 46.　新規 String の追加

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。

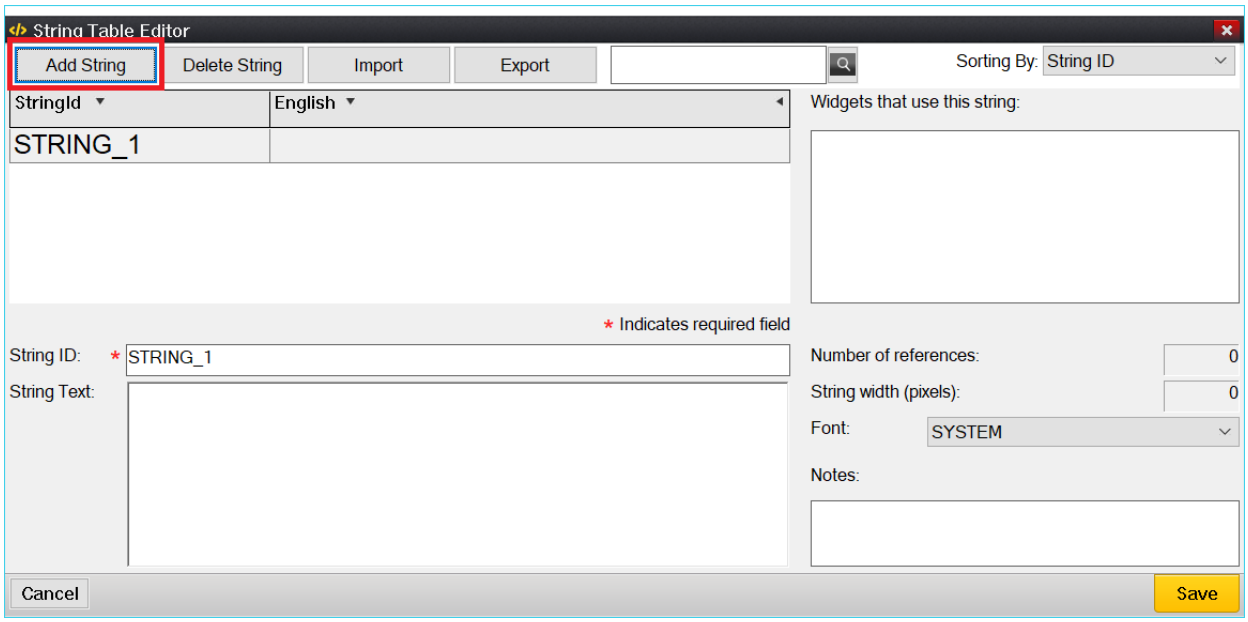


図 47.　String Table Editor

9. **String ID**と**String Text**を編集します。

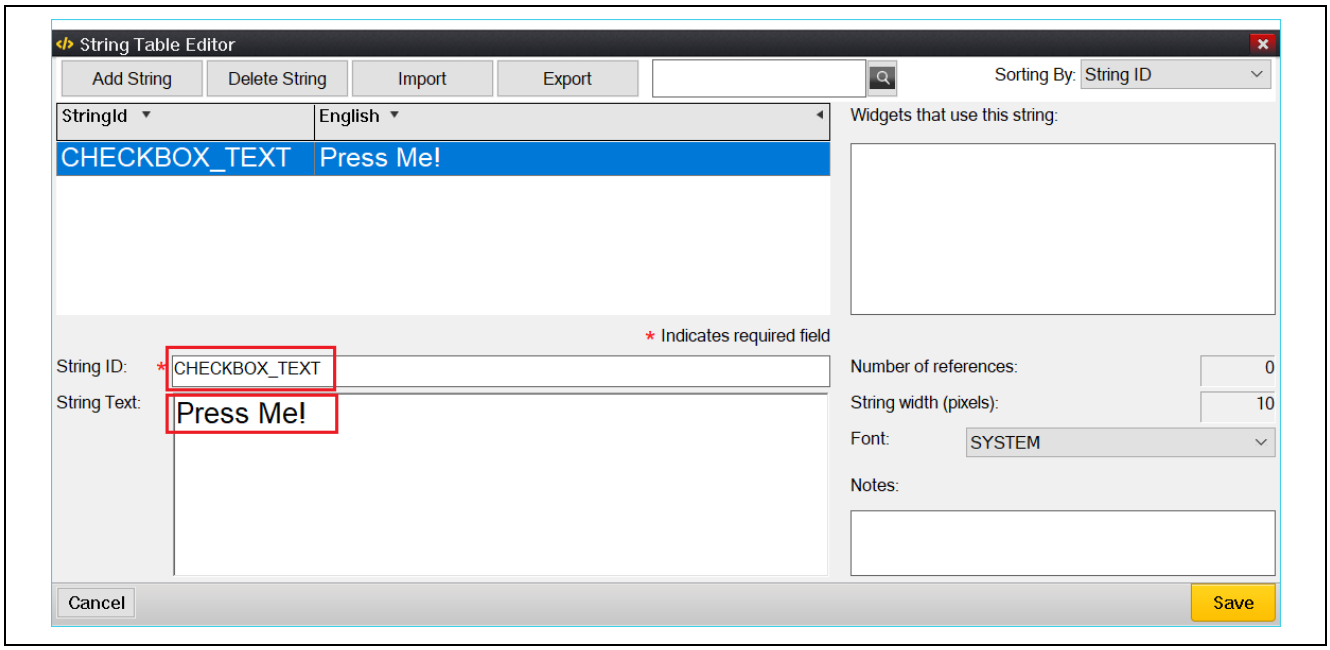


図 48.　String ID と String Text の編集

10. 続けて **Add String** をクリックし、表が**図 49** のように表示されるまで、各々の新しいエントリに対応する **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。

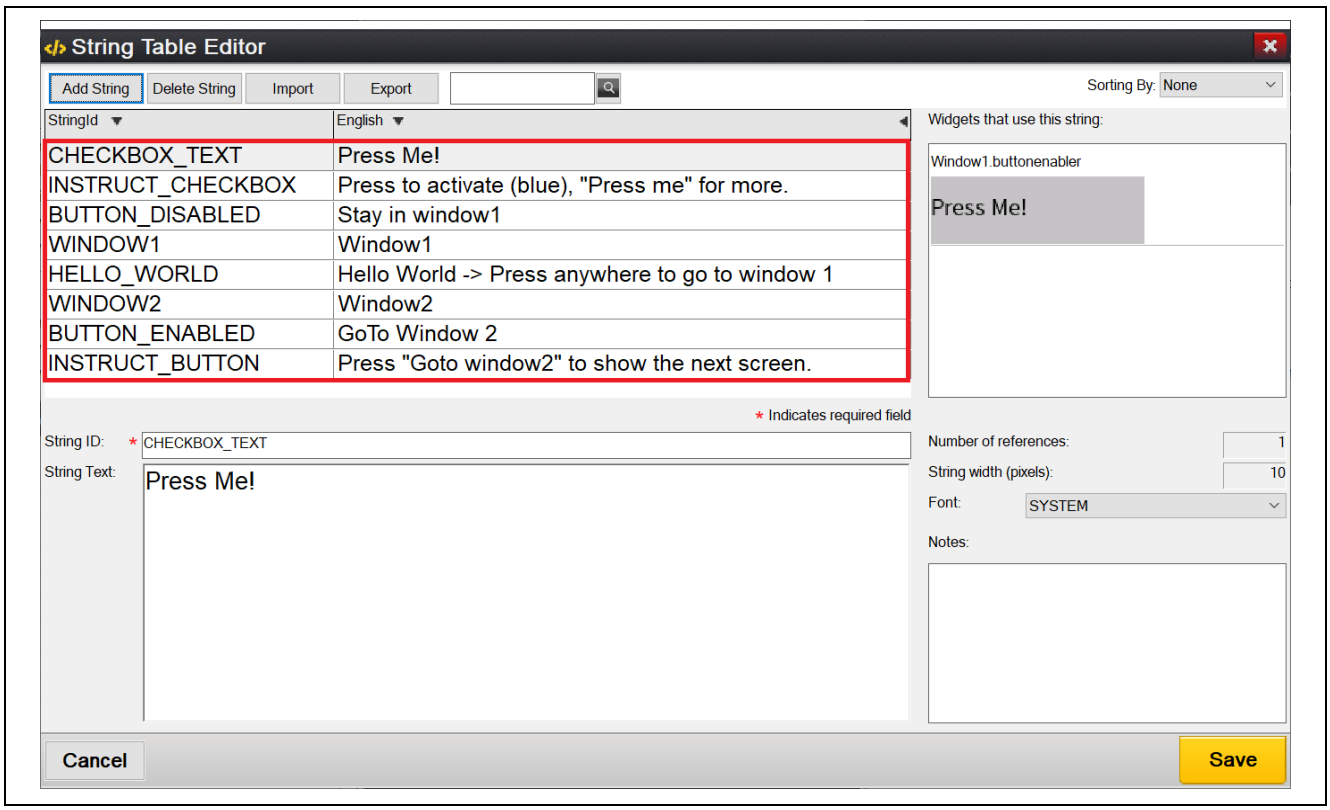


図 49.　すべての Strings

11.**図 50 のように、Project View** で **Window1** を右クリックし **Text Button** を挿入します。

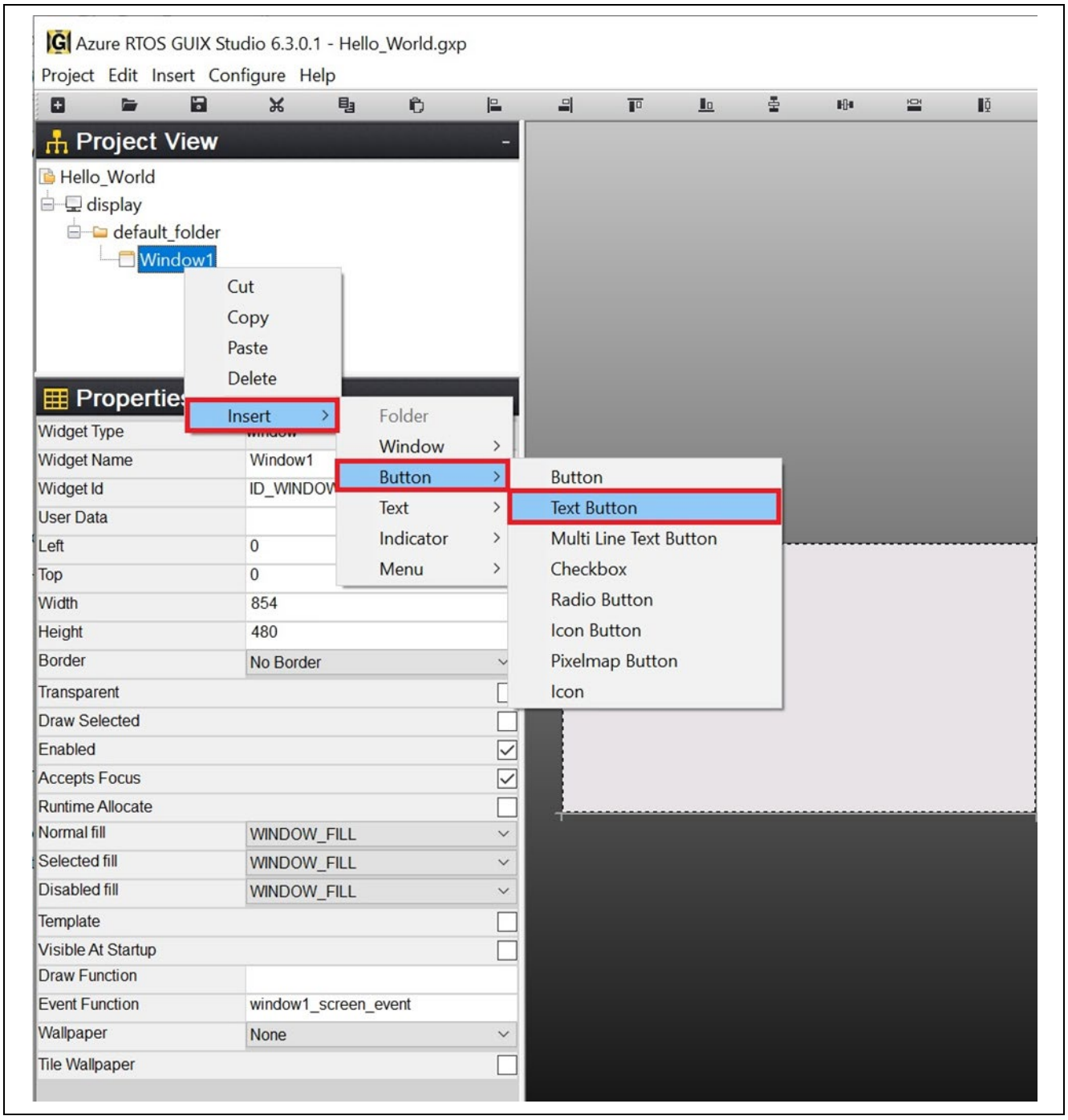


図 50.　Text Button の挿入

12.**text_button のプロパティ**を設定します。

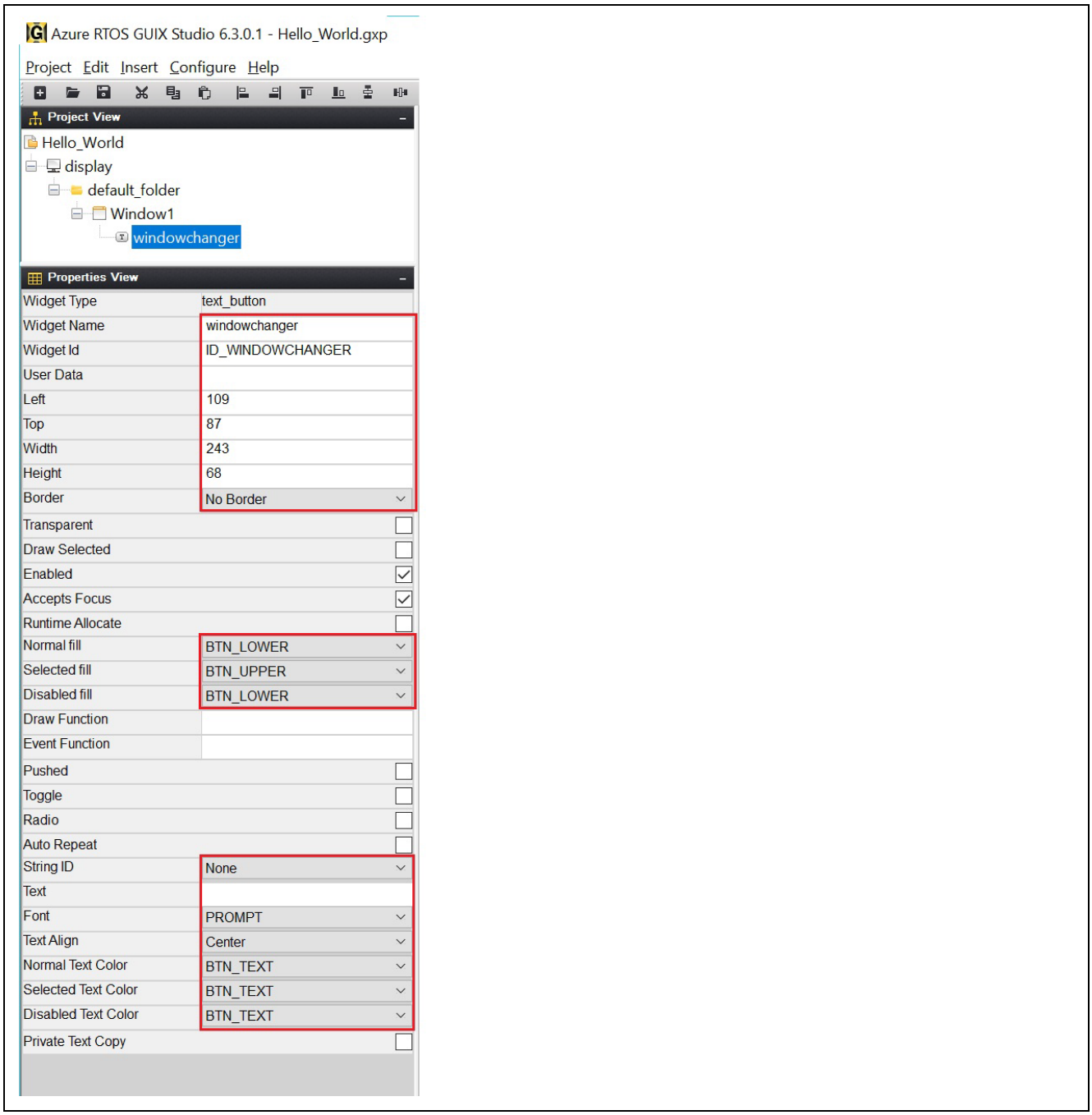


図 51.　text_button のプロパティの設定

13.**図 52**のように、**windowchanger**を右クリックして**Prompt**を挿入します。

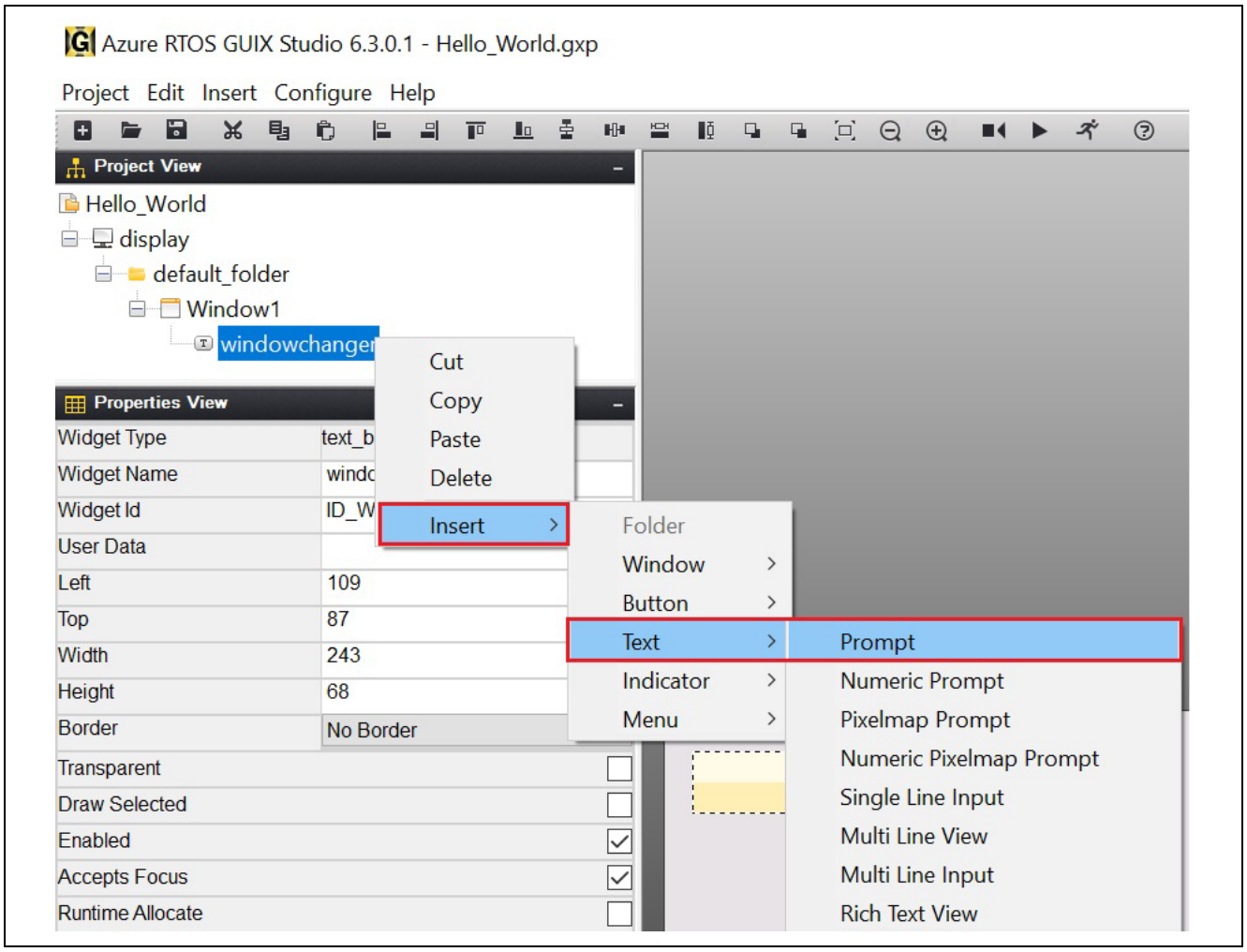


図 52.　Prompt の挿入

14.**Promptのプロパティ**を設定します。

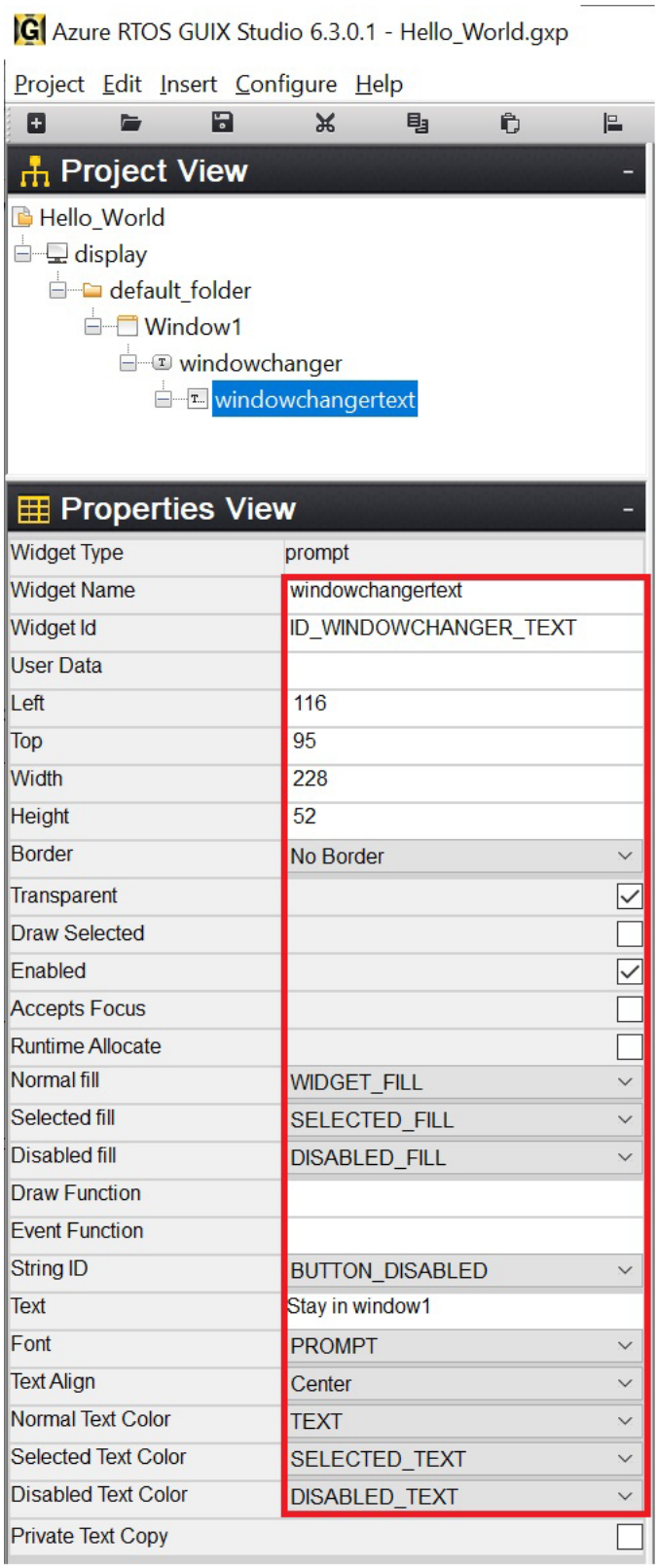


図 53. Prompt のプロパティの設定

15. 図 54 のように、**windowchangertext** を右クリックして新規 **Button** を挿入します。

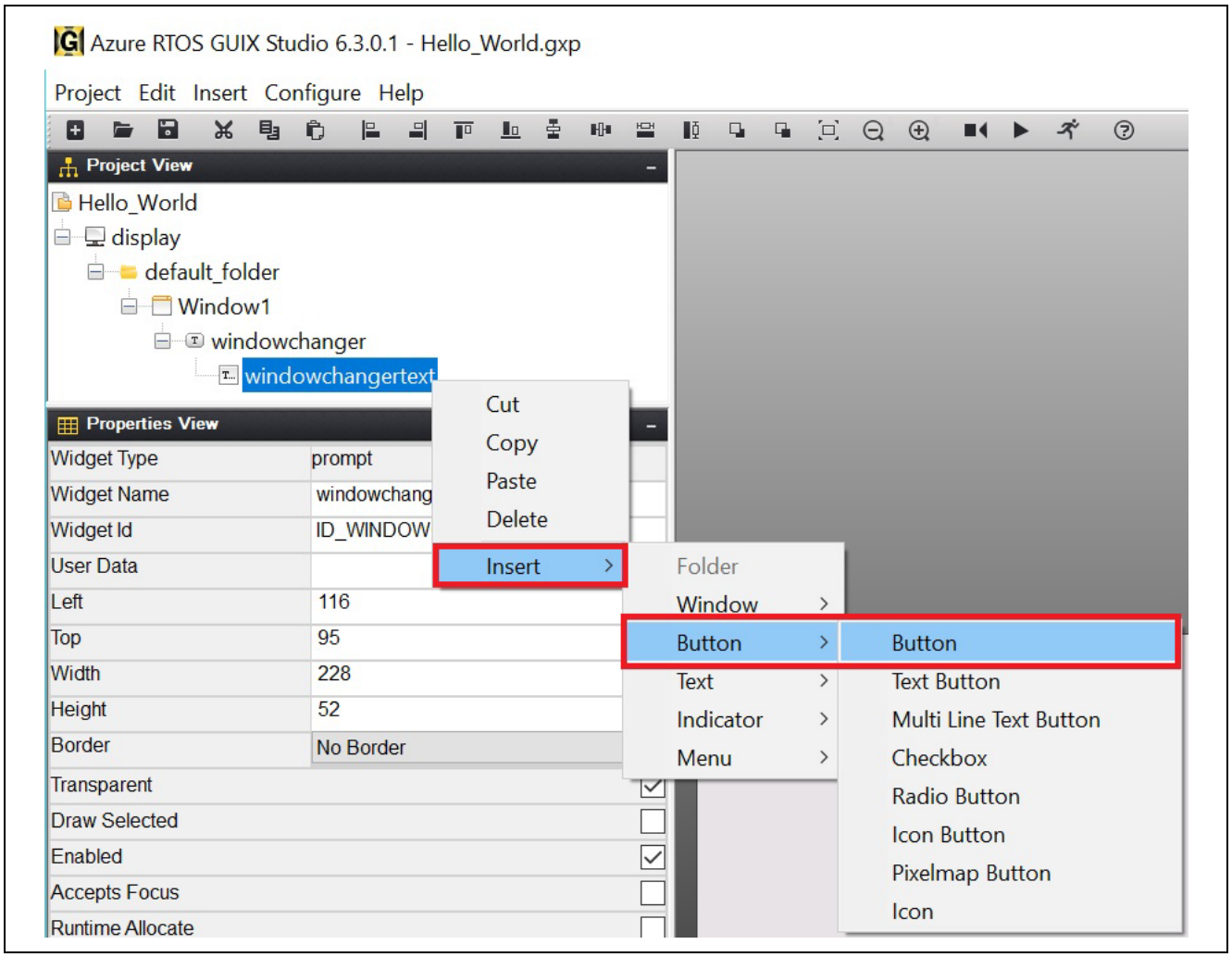


図 54.　Button の挿入

16.**buttonのプロパティ**を設定します。

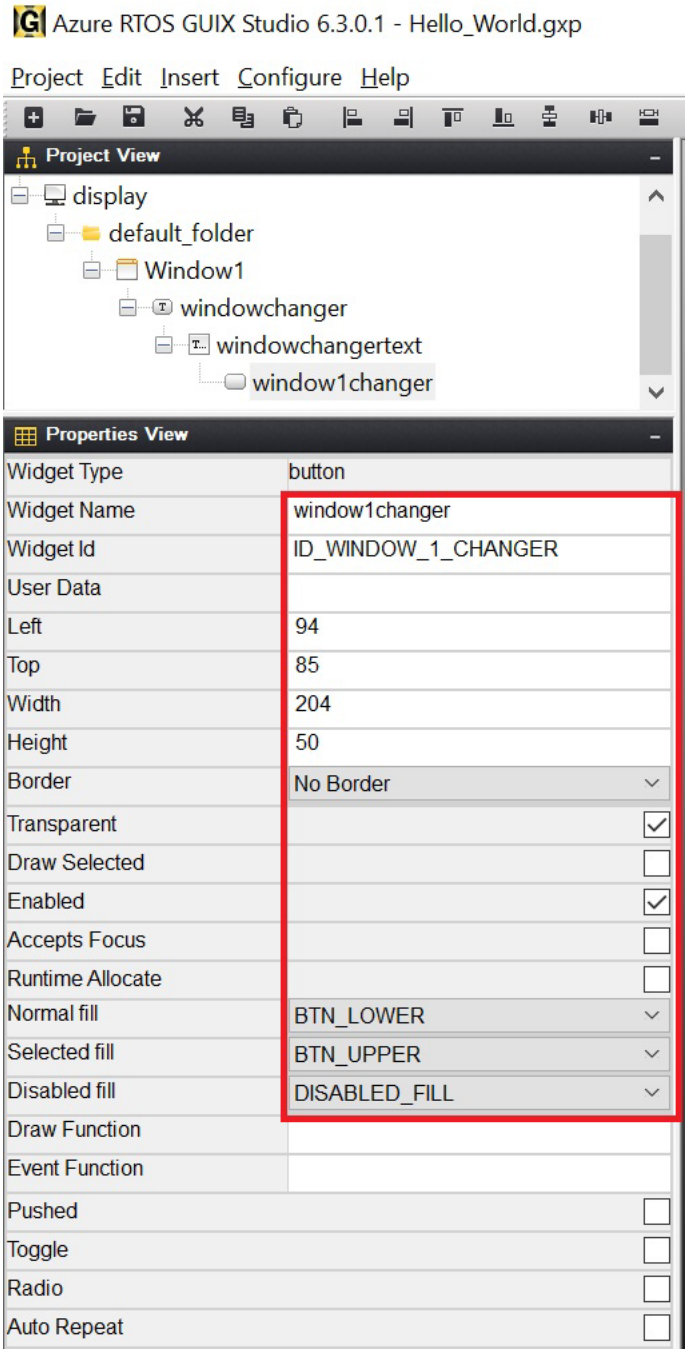


図 55. button のプロパティ設定

17. **図 56** 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。

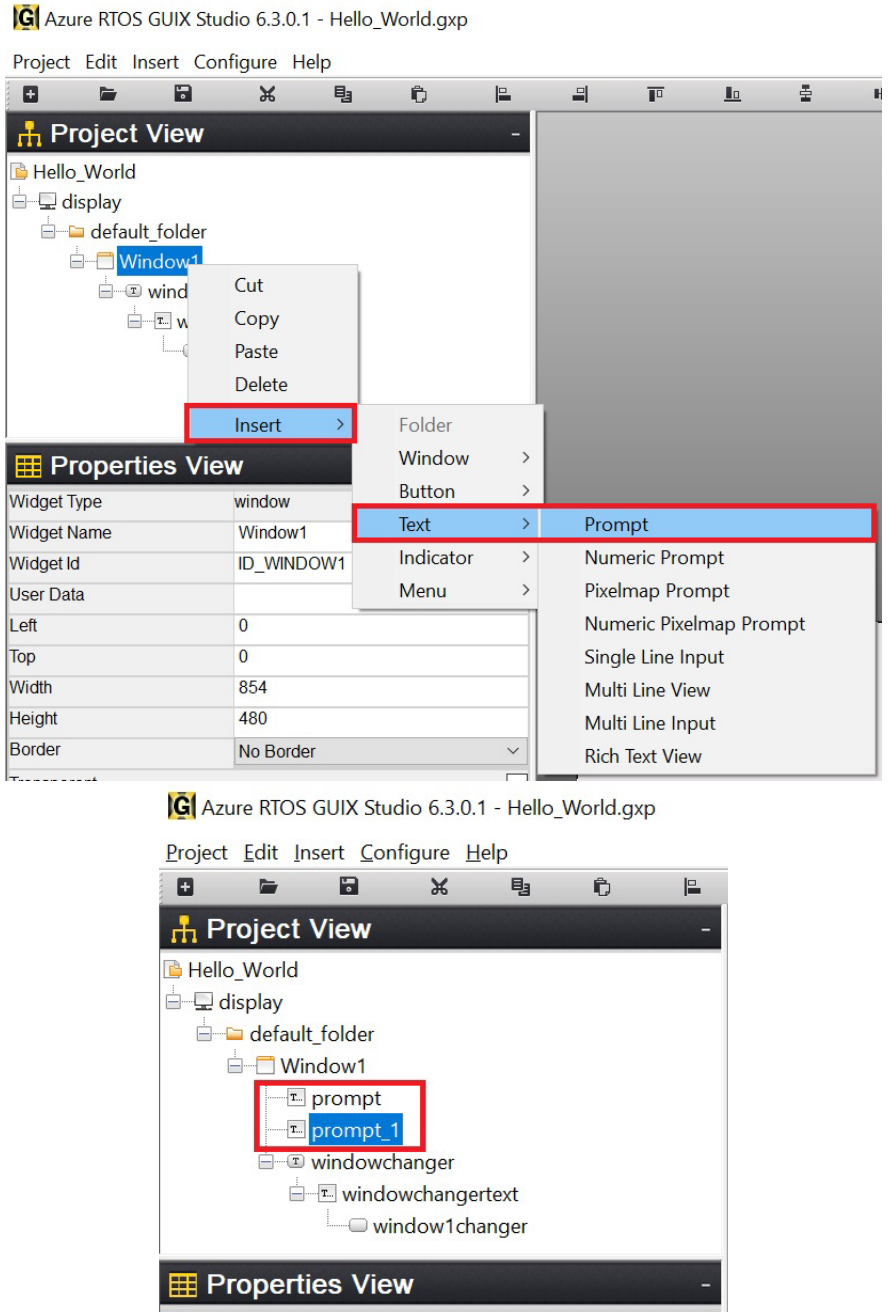


図 56.　Prompt の挿入

18.**Prompt のプロパティ**を設定します。

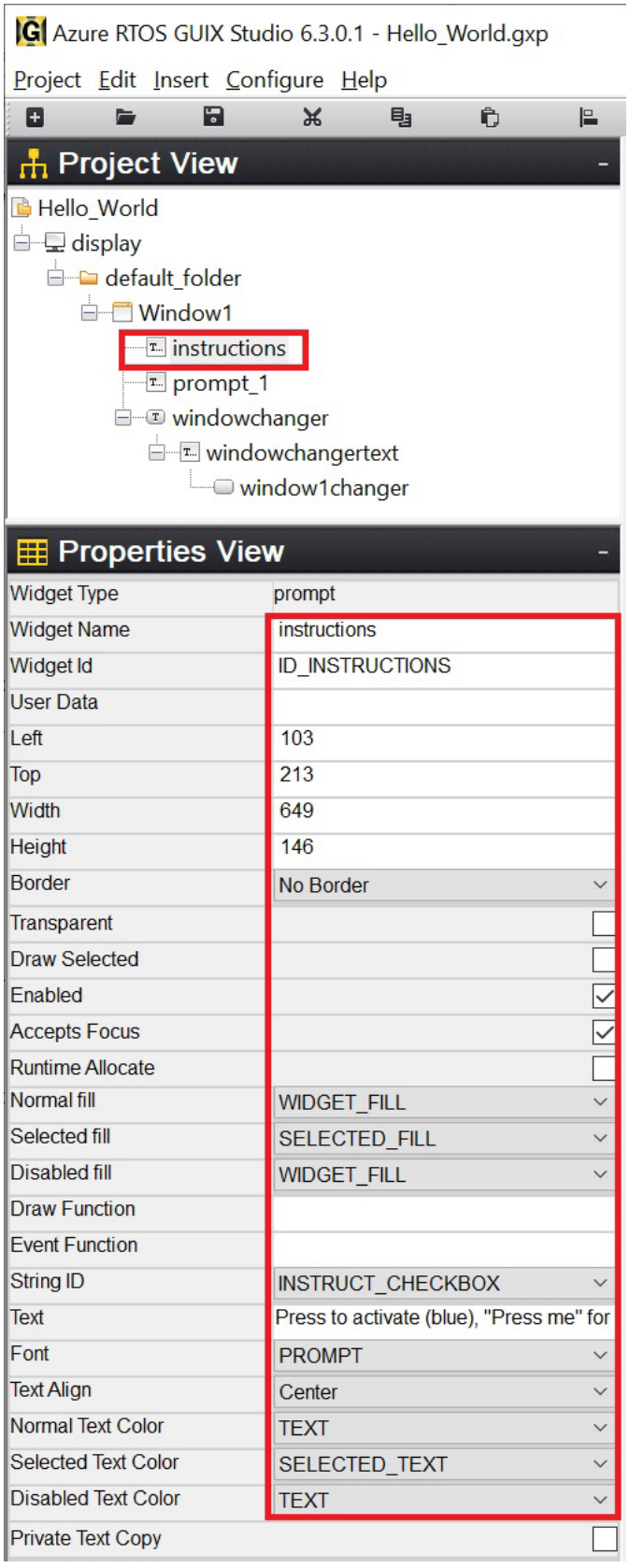


図 57.　Prompt のプロパティ設定

19.**Prompt1 のプロパティ**を設定します。

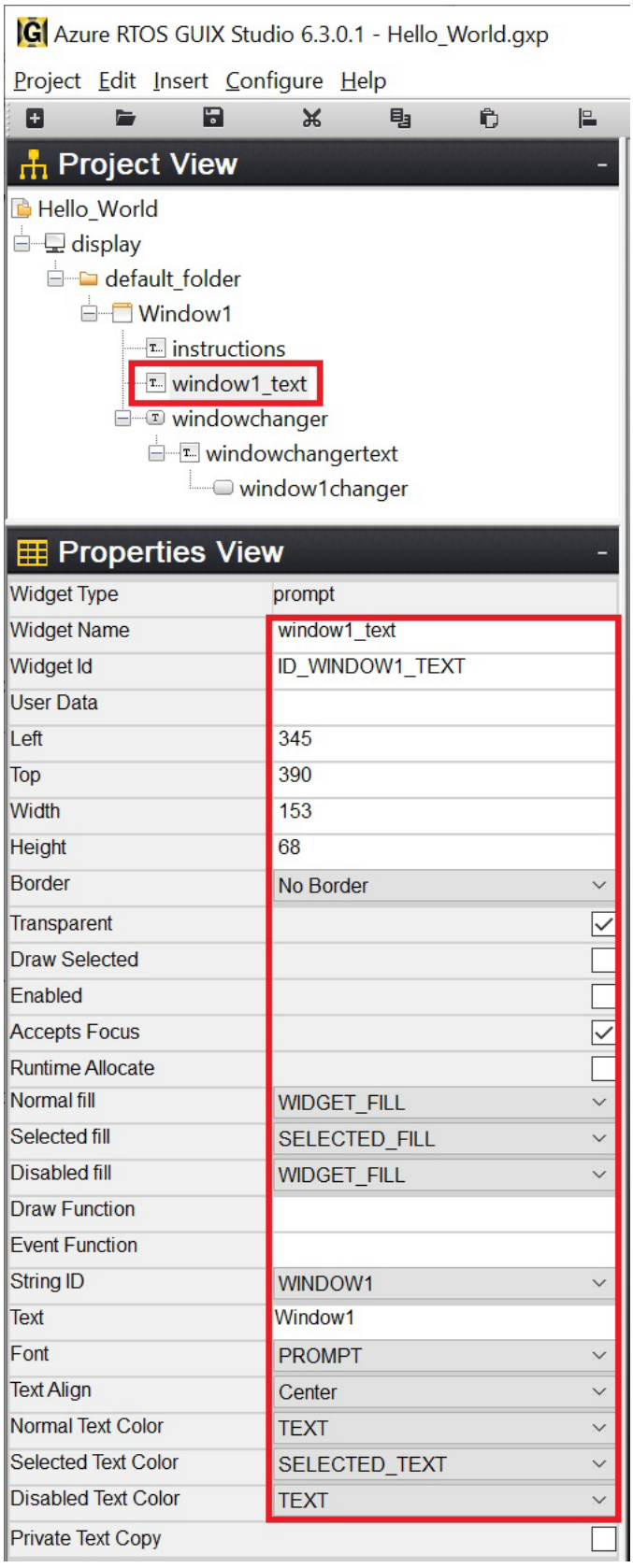


図 58.　Prompt1 のプロパティ設定

20.**図 59**のように、**Window1**を右クリックして、**Click Insert > Button > Checkbox**をクリックします。

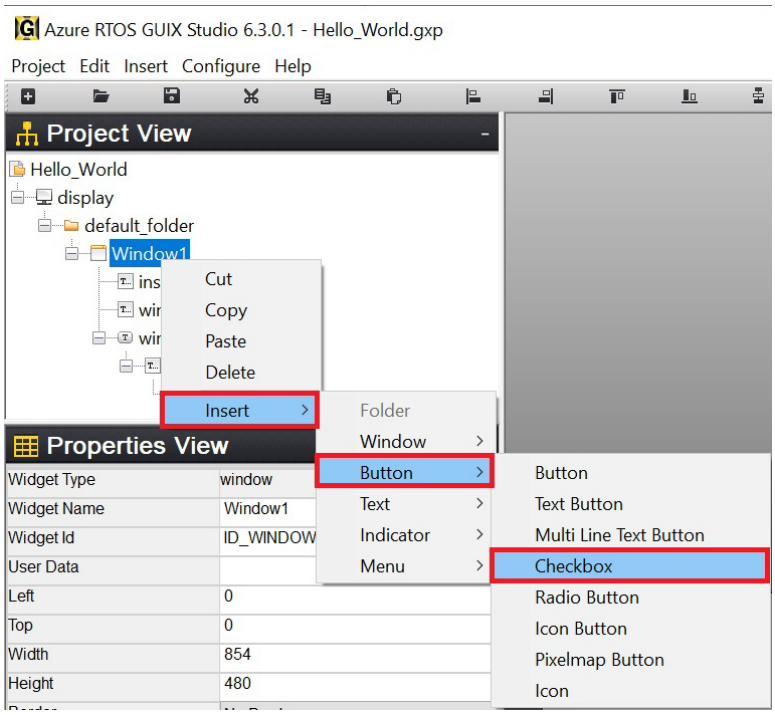


図 59.　Button Checkbox の挿入

21.**Checkboxのプロパティ**を設定します。

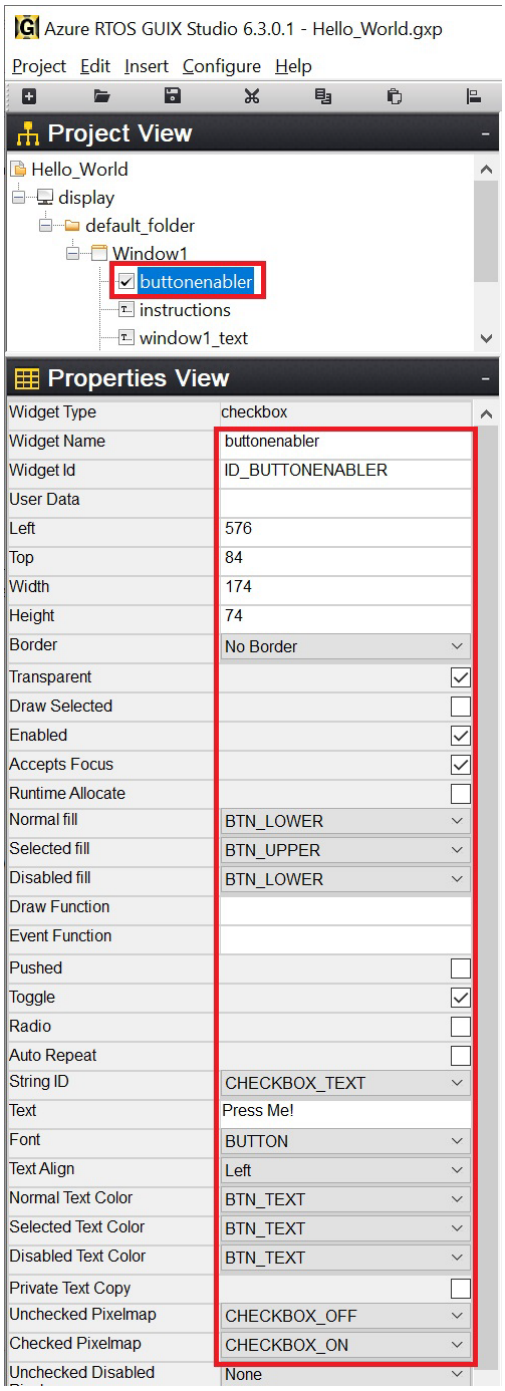


図 60.　Checkbox のプロパティ設定

22.**Window1** の作成が完了すると、以下の画像のようになります。

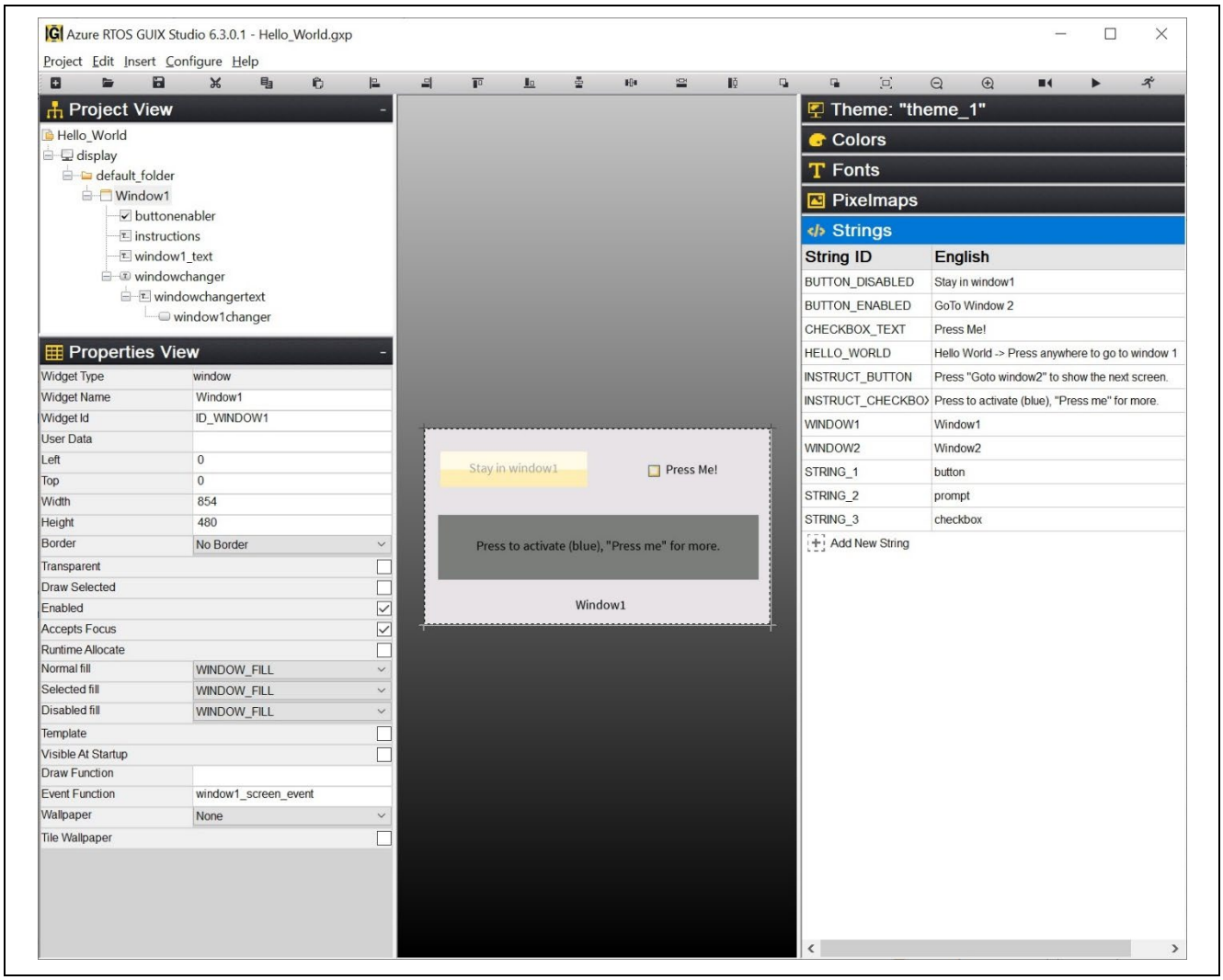


図 61.　作成した Window1

23.GUIX Studio プロジェクトに別のウィンドウを追加します。**図 62** のように、**Project View** で **default_folder** を右クリックし、**Insert > Window > Window** の順にクリックします。

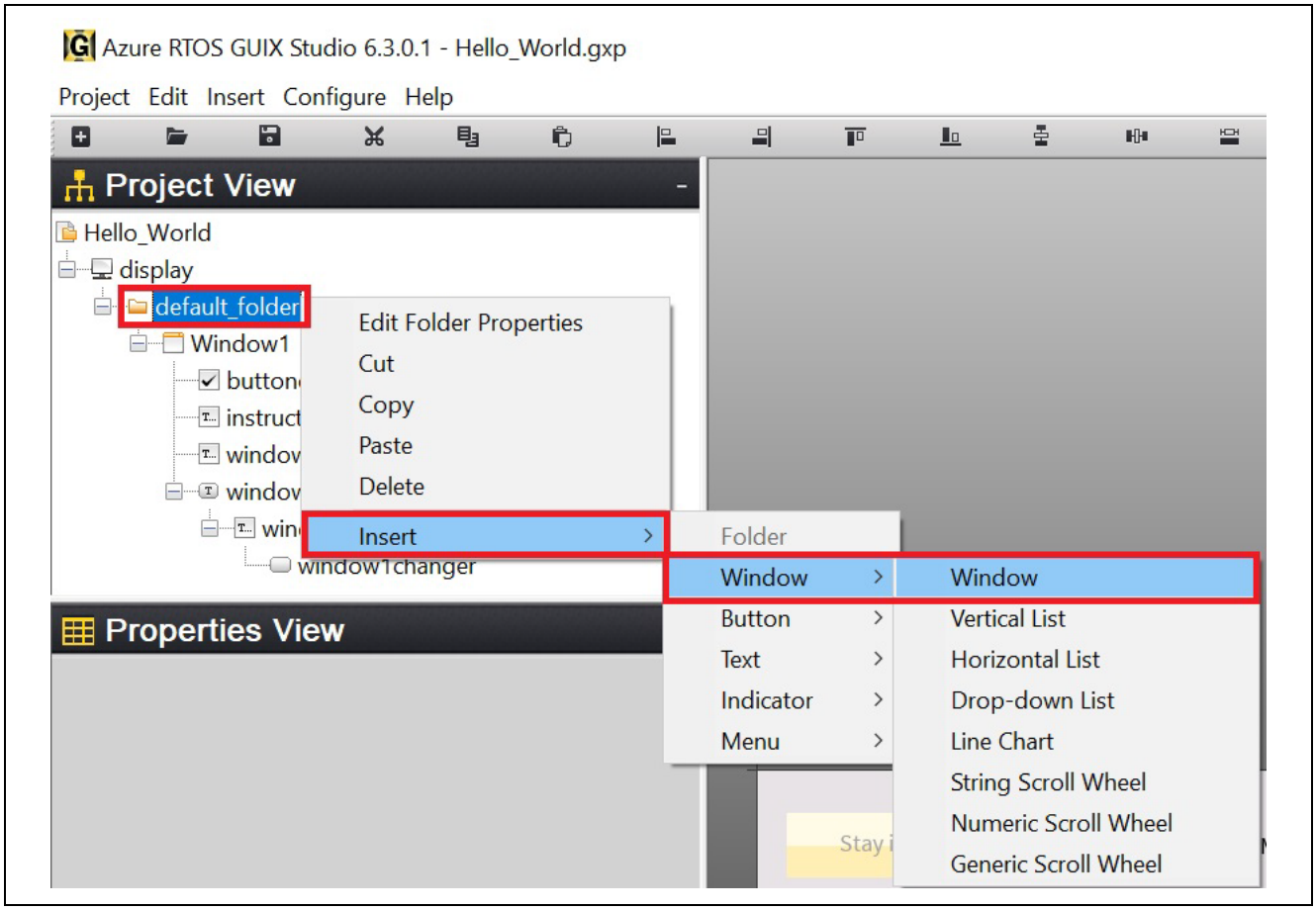


図 62.　Window2 の挿入

24.**Window2のプロパティ**を設定します。

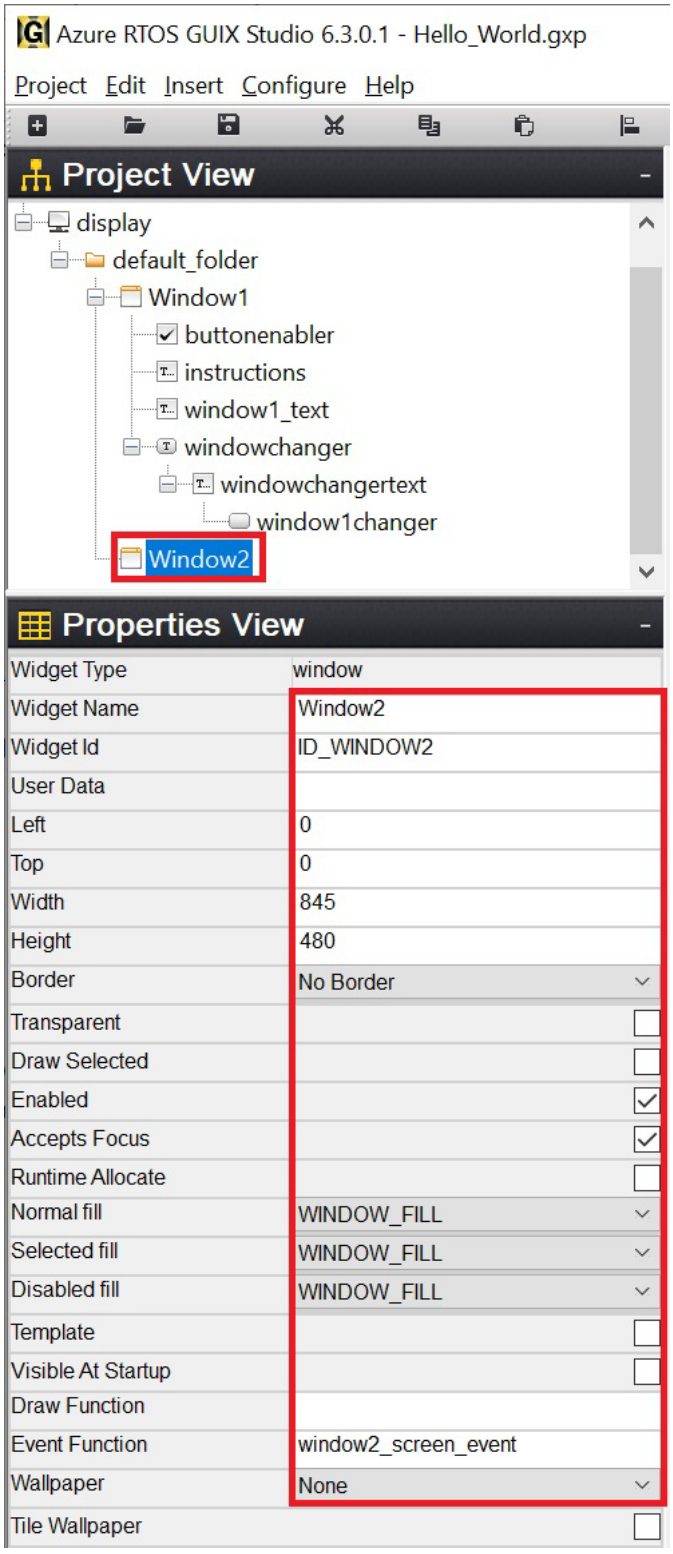


図 63.　Window2 のプロパティ設定

25.**図 64** のように、Window2 を右クリックし、**Prompt** を挿入します。**Insert > Text > Prompt** をクリックし
ます。

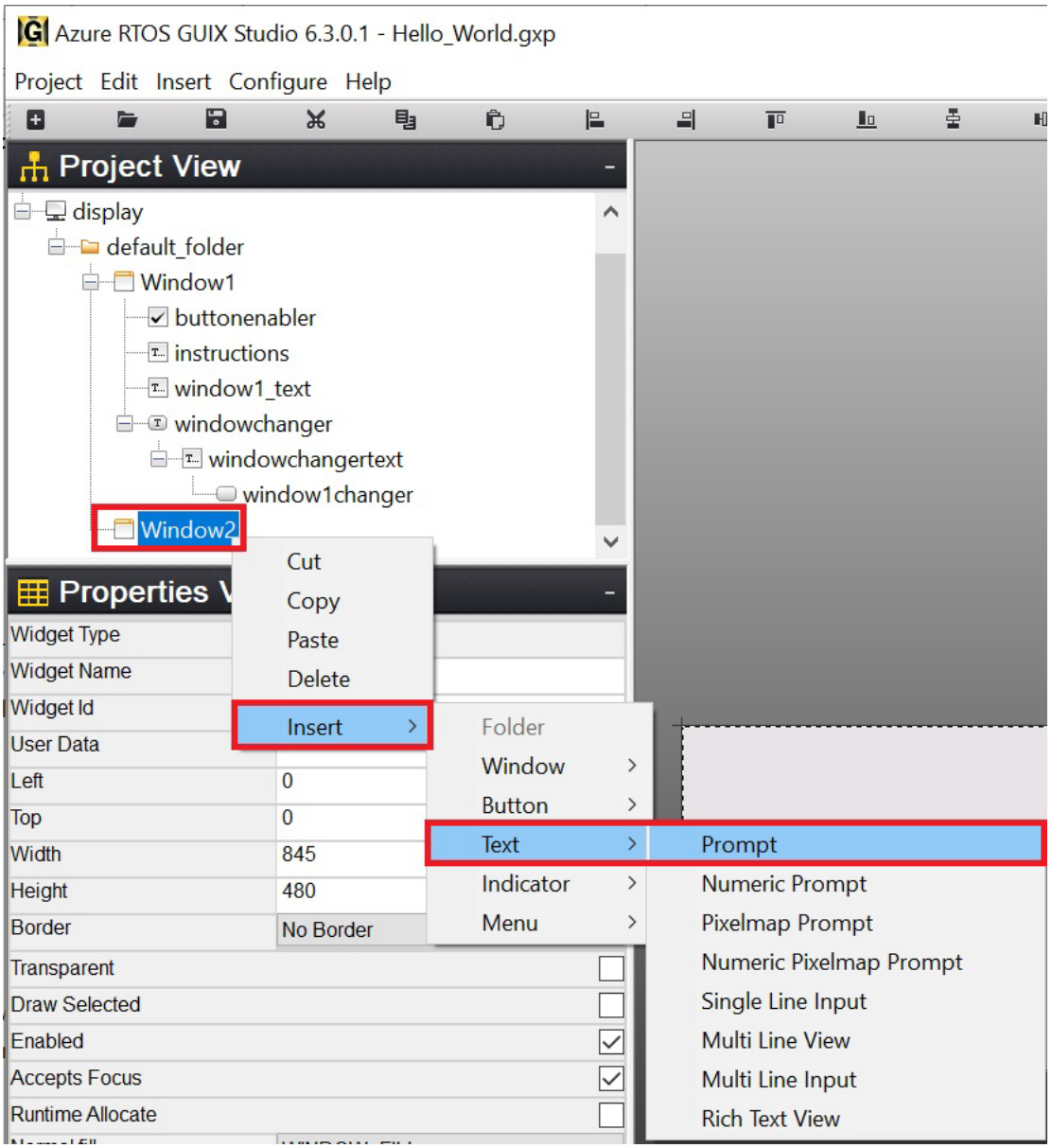


図 64.　Window2へPromptを挿入

26. **Prompt**のプロパティを設定します。

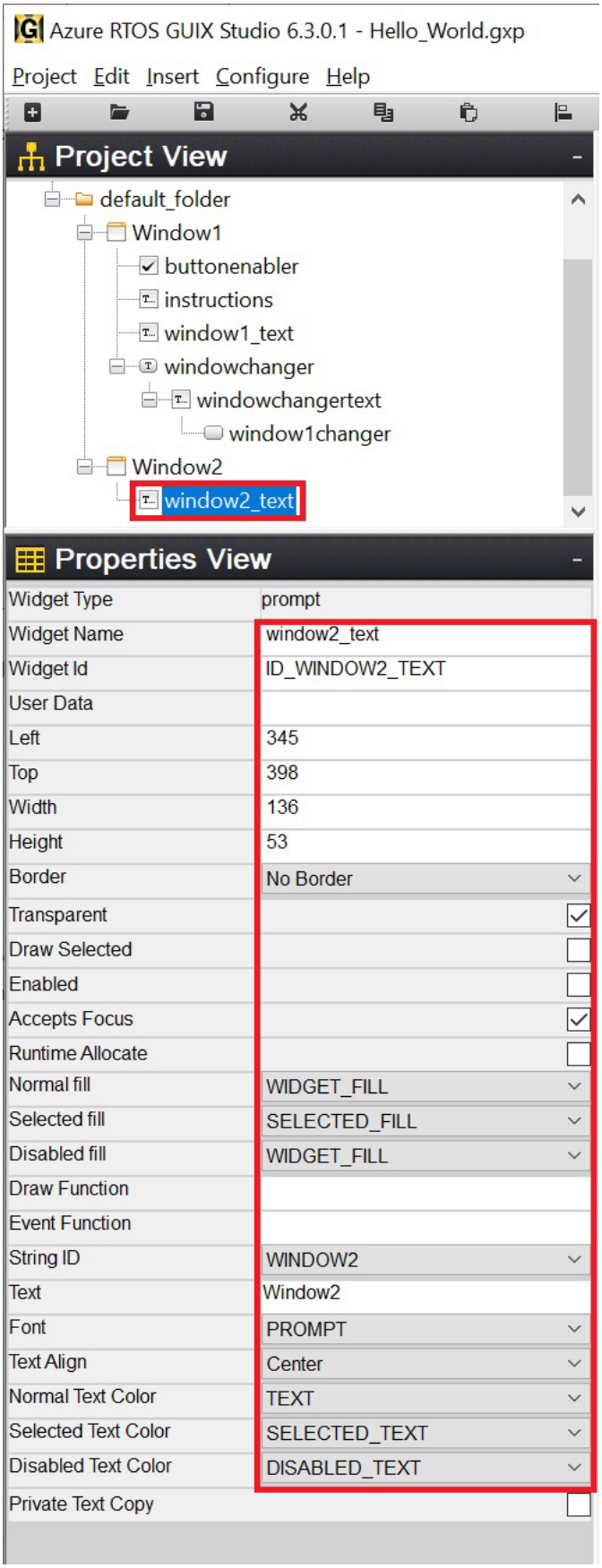


図 65.　Windows2_text のプロパティ設定

27. **図 66** のように、Window2 を右クリックし、**text_button** を挿入します。**Insert > Button > Text Button** をクリックします。

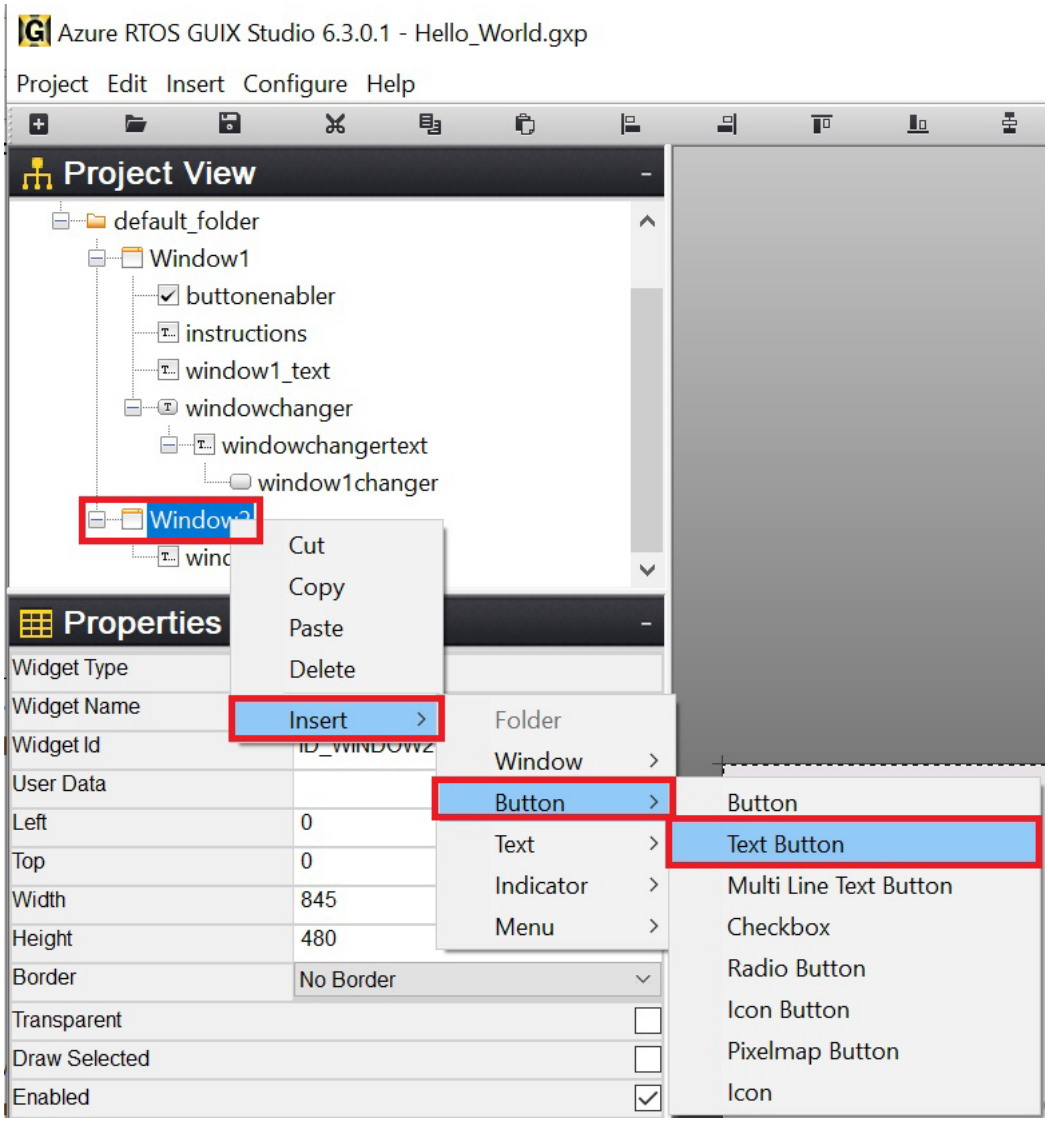


図 66.　Window2のText Buttonの挿入

28. **図 67**のように、**text_button**のプロパティの設定をします。

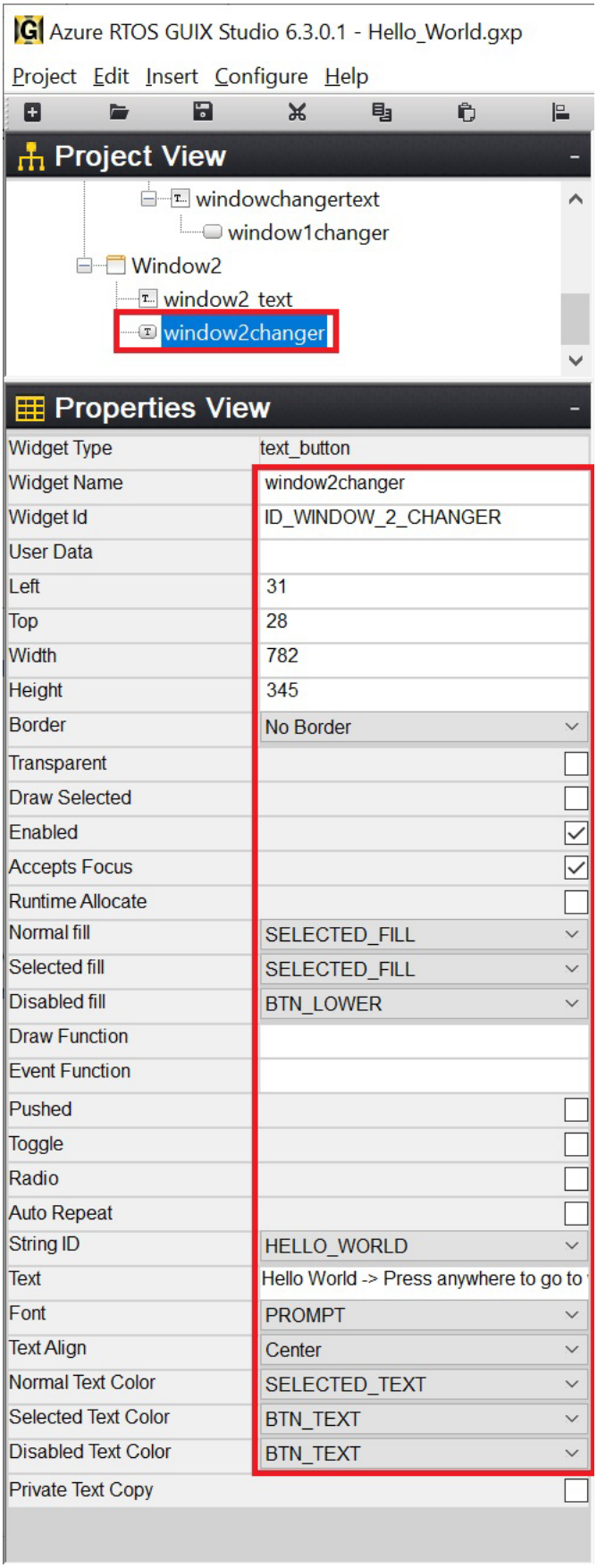


図 67.　text_button のプロパティ設定

29. 挿入と設定終了後の Window2 を**図 68** に示します。

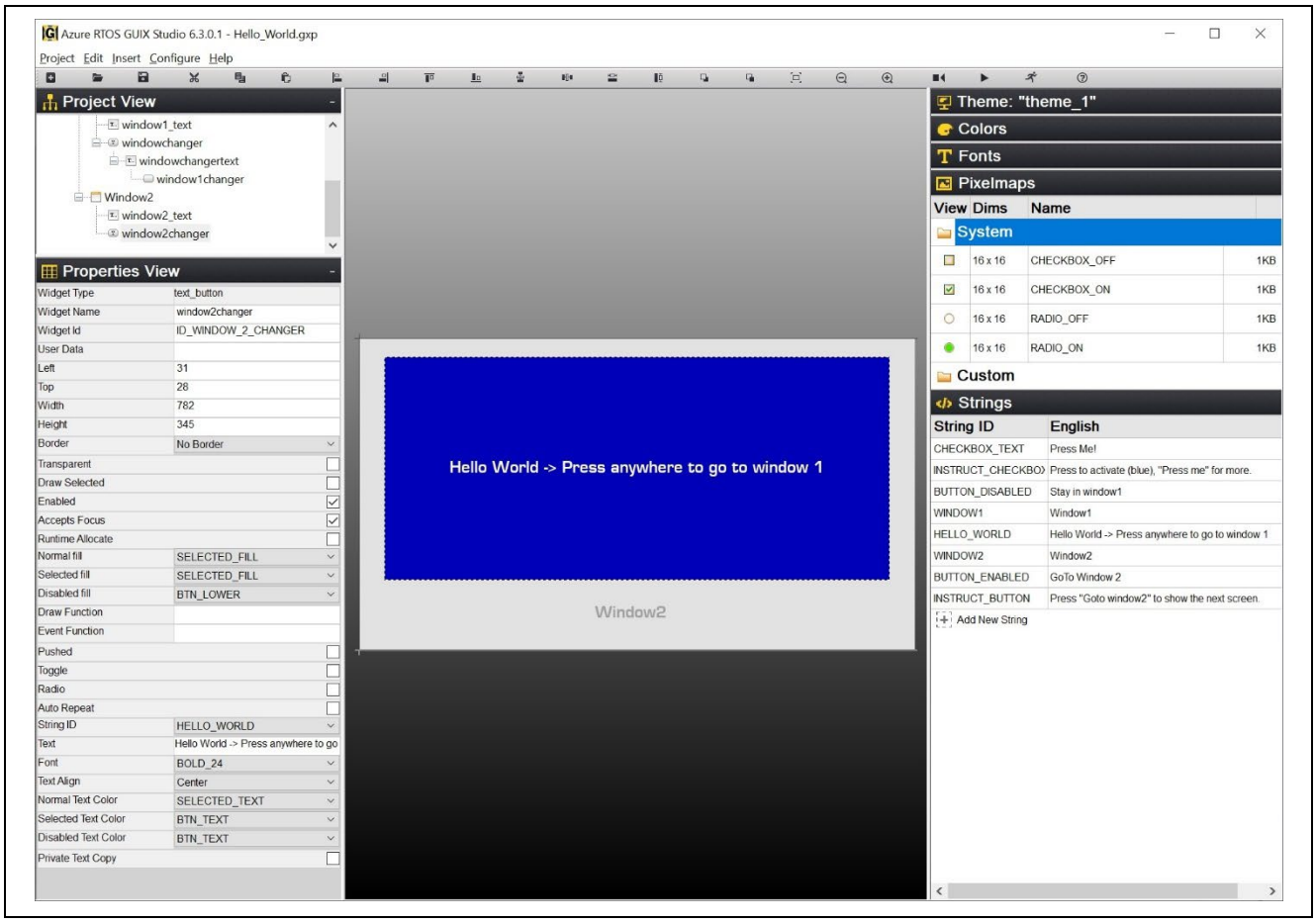


図 68.　Window2

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。

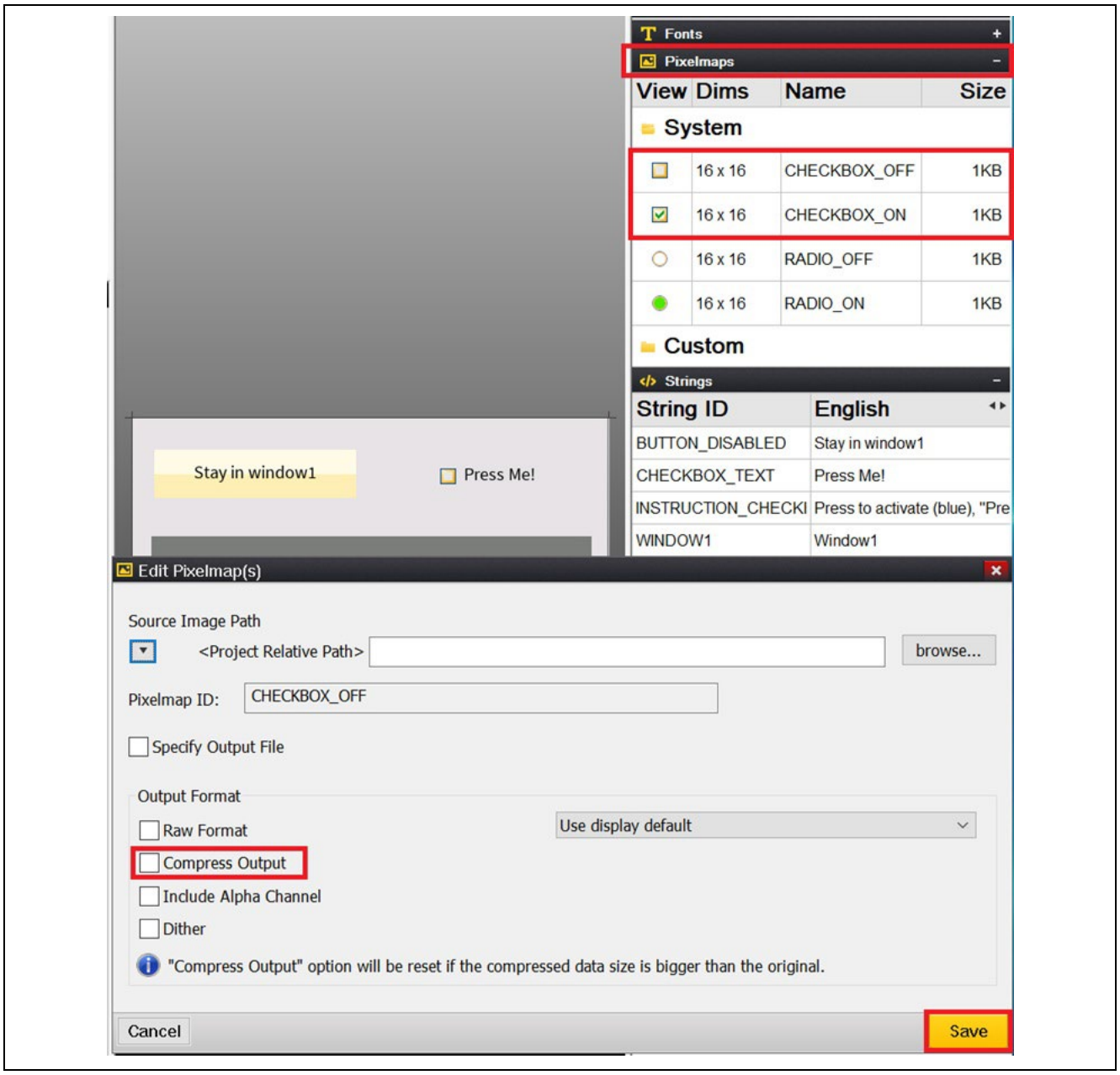


図 69.　Pixelmap の設定

31. ここで、**プロジェクト**のドロップダウンリストにある **Save Project** と **Generate All Output Files** をクリックできるようになりました。これにより、GUIX Hello World の作成とプロジェクトへのエクスポートが完了しました。

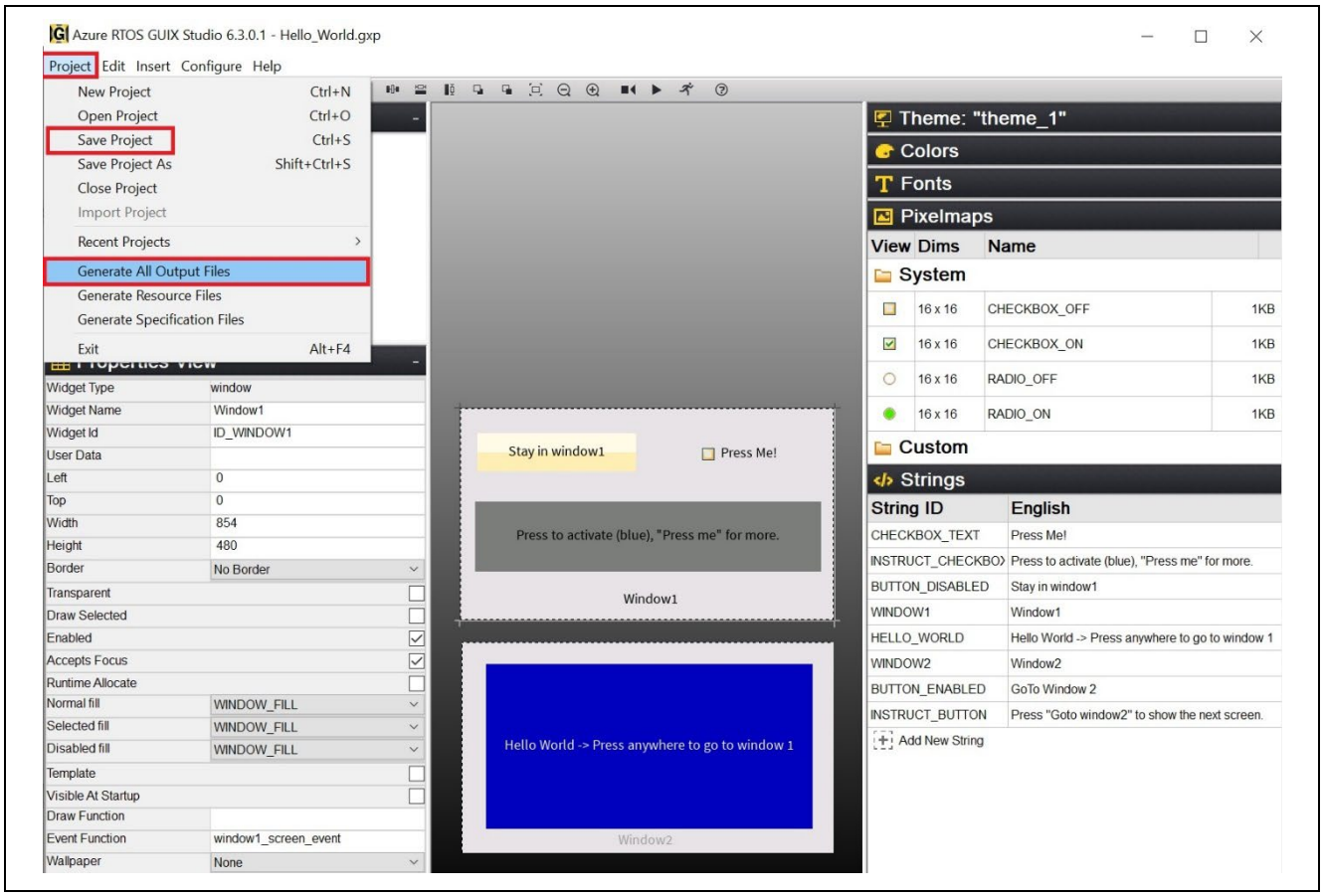


図 70. プロジェクトの保存と生成

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。

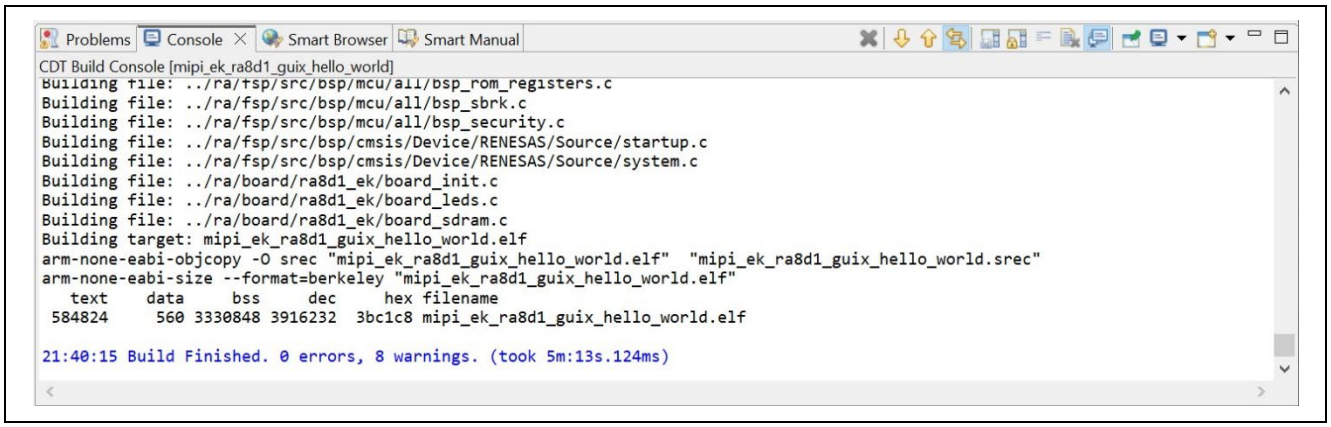


図 71. コードのビルド

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。
    mipi_ek_ra8d1_guix_hello_world プロジェクトをダウンロードして実行します。

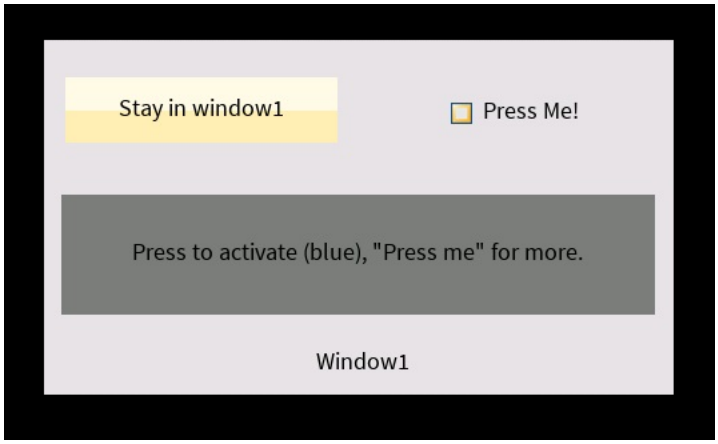


図 72.　Window1 の表示

6. 既存プロジェクトの概要

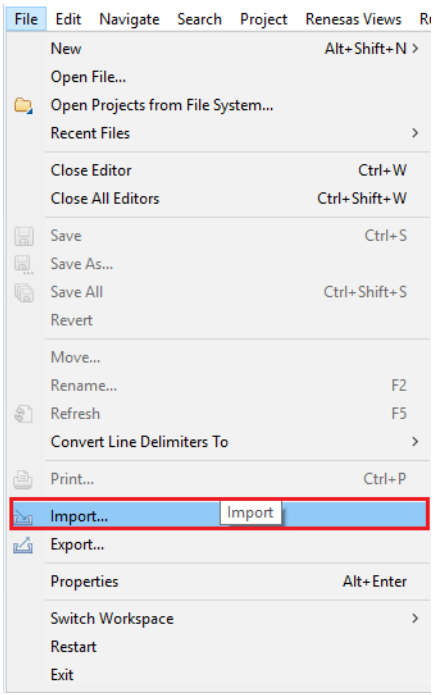
6.1 概要

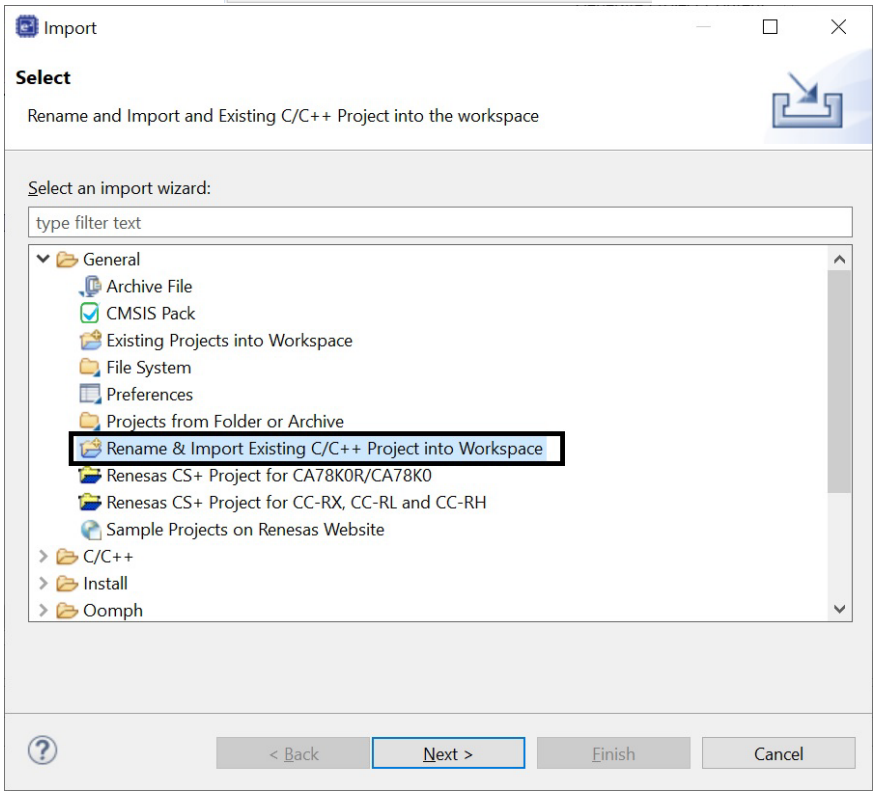
この章では、弊社で作成済のプロジェクト**mipi_ek_ra8d1_guix_hello_world**のインポートの実行方法を説明します。

-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。

6.2 手順

1. 提供されているプロジェクト **mipi_ek_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。


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

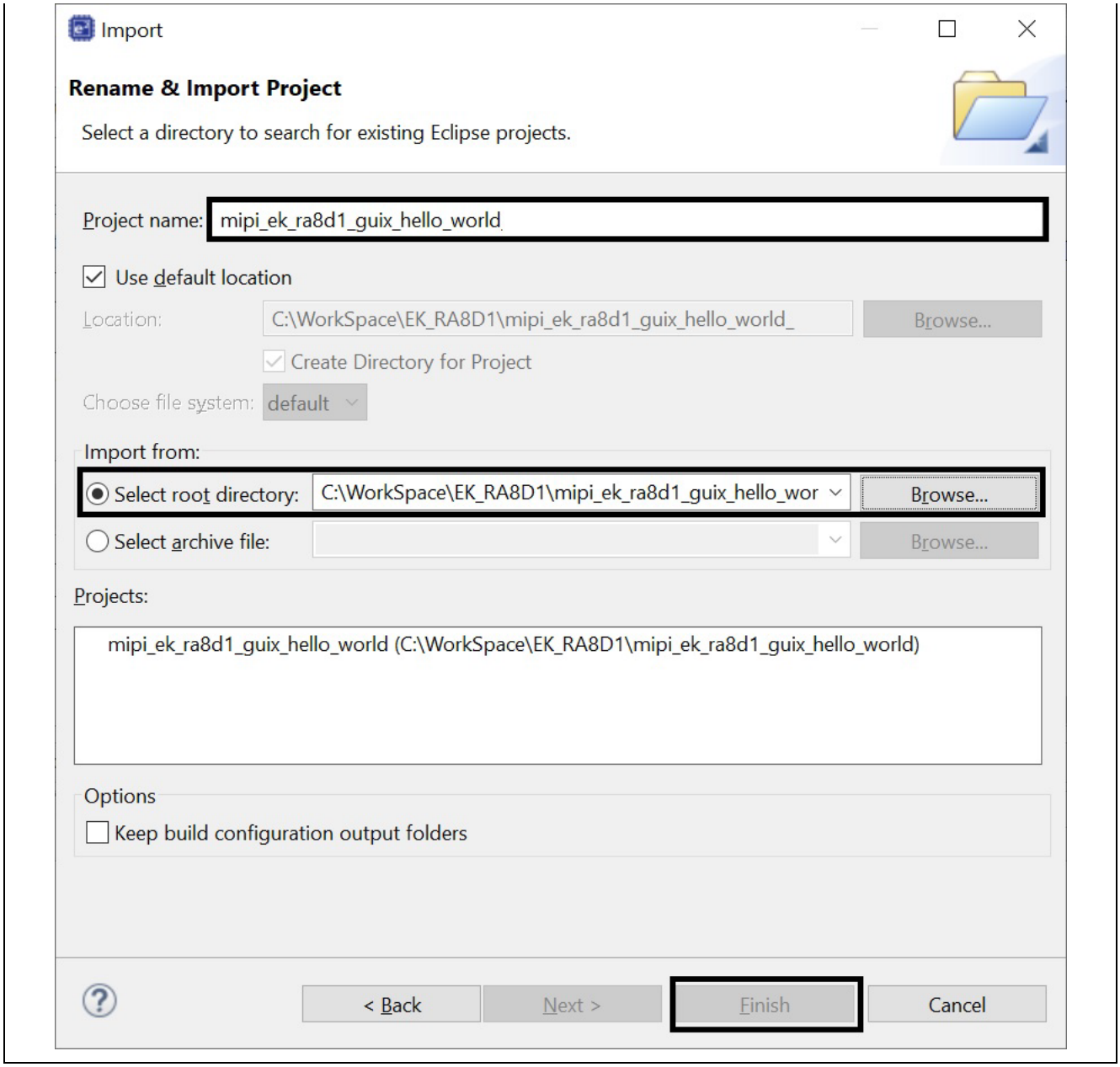


図 73. 既存プロジェクトのインポート

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

改訂履歴

Rev.	日付	説明	
		ページ	概要
1.00	2024 年 10 月 1 日	—	初版発行

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

商標

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/