Renesas Synergy™ プラットフォーム

GUIX™ Synergy ポートフレームワーク モジュールガイド

R11AN0217JU0102
Rev.1.02
2019.06.03

(注1)本資料は英語版を翻訳した参考資料です。内容に相違がある場合には英語版を優先します。資料によっては英語版のバージョンが更新され、内容が変わっている場合があります。日本語版は、参考用としてご使用のうえ、最新および正式な内容については英語版のドキュメントを参照ください。

(注 2)本資料の第 6 章まで(要旨除く)の日本語訳は、「Synergy™ Software Package (SSP) v1.5.0 ユーザーズマニュアル モジュール概要編(参考資料)」の第 4 章「モジュールの概要」に掲載されていますのでそちらを参照ください 。

要旨(Introduction)

本モジュールガイドは、GUIX™ Synergy ポートフレームワークモジュール(GUIX™ Synergy Port Framework Module)を効果的に使用してシステムが開発できるようになることを目的としています。このモジュールガイドを習得することで、開発システムへのモジュールの追加とターゲットアプリケーション向けの正確な設定(configuration)ができ、さらに付属のアプリケーションプロジェクトコードを参照して、効率的なコード記述が行えるようになります。

より詳細な API や、より高度なモジュール使用法を記述した他のアプリケーションプロジェクト例もルネサス WEB サイト(本書末尾の「参考文献」の項を参照)から入手でき、より複雑な設計に役立ちます。

Express Logic の GUIX Synergy ポートモジュールである SF_EL_GX は、Synergy MCU グループに対応した Express Logic GUIX™ の適応レイヤ(adaptation layer)であり、グラフィクスエンジンである GLCDC や DRW (2DG エンジン)、または JPEG デコードエンジンを持っています。この API は GUIX 用にセットアップされたグラフィックスハードウェアエンジンをサポートしているほか、このハードウェアエンジンによるアクセラレーションによる、グラフィックス描画と表示もサポートしています。このモジュールは、全てのGUIX ローレベル(low-level)ディスプレイドライバ関数を定義します。これらの関数は、DRW (2DG エンジン:2DG engine) または JPEG によるアクセラレーションによるグラフィックス描画や、GLCDC を使用したグラフィックスの表示を行います (『GUIX User Guide』の「Chapter 5: GUIX Display Drivers」を参照)。このモジュールは、ハードウェアアクセラレーションによるグラフィックス描画に対応するほか、ハードウェアアクセラレーションを使用しないソフトウェア処理にも対応します。

目次

1. **GUIX Synergy ポートフレームワークモジュールの機能（GUIX Synergy Port Framework Module Features）**
2. **GUIX Synergy ポートフレームワークモジュール API の概要（GUIX Synergy Port Framework Module APIs Overview）**
3. **GUIX Synergy ポートフレームワークモジュールと JPEG デコード HAL モジュールの動作の概要（GUIX Synergy Port Framework Module and JPEG Decode HAL Module Operational Overviews）**
4. **アプリケーションへの GUIX Synergy ポートフレームワークモジュールの組み込み（Including the GUIX Synergy Port Framework Module in an Application）**
5. **GUIX Synergy ポートフレームワークモジュールの設定（Configuring the GUIX Synergy Port Framework Module）**
6. **アプリケーションでの GUIX Synergy ポートフレームワークモジュールの使用**
7. **GUIX Synergy ポートフレームワークモジュールのアプリケーションプロジェクト（The GUIX Synergy Port Framework Module Application Project）**

このモジュールガイドに関連するアプリケーションプロジェクトは、設計全体の手順を示します。このプロジェクトは、このドキュメントの末尾にある「参考情報」の章に掲載されているリンクにあります。ISDE でアプリケーションプロジェクトをインポートして開き、GUIX Synergy ポートフレームワークモジュールに対応する設定項目を表示することができます。また、完成した設計における GUIX Synergy ポートフレームワークモジュール API を理解するために、my_gui_thread_entry.c 内のコードを確認することもできます。

このアプリケーションプロジェクトのメインスレッドのエントリは、GUIX システムと、ディスプレイおよび LCD の各ハードウェアに対応するローレベルドライバを初期化します。メインスレッドは次に、モードが LCD_TEST と GUIX_TEST のどちらに設定されているかに応じて、2 種類のテストのうちどちらかを実行します。モードが GUIX_TEST に設定されている場合、GUIX Studio が生成したリソースファイルと仕様ファイル（specification file）を使用します。モードが LCD_TEST に設定されている場合、シンプルな 3 本のカラーバーイメージ（color bar image）を作成し、lcd_setup.c ファイル内の LCD サービスを呼び出して、LCD 画面に対してイメージを描画します。このアプリケーションは、成功以外のステータスを返した API や、エラーを示す GUIX コールバックを対象にするエラーカウンタを保持しています。SEMI_HOSTING が定義されている場合、デバッグの printf 出力は、successful (成功) または failed (失敗) のテスト結果を示します。

表 1　　このアプリケーションプロジェクトが使用するソフトウェアとハードウェアのリソース

リソース	リビジョン	説明
e² studio	7.3.1 またはそれ以降	統合ソリューション開発環境 (ISDE)
SSP	1.6.0 またはそれ以降	Synergy ソフトウェアプラットフォーム
IAR EW for Renesas Synergy	8.23.3 またはそれ以降	IAR Embedded Workbench for Renesas Synergy
SSC	7.3.0 またはそれ以降	Synergy Standalone Configurator
SK-S7G2	v3.0 と v3.1	スタータキット
GUIX Studio	5.4.0	ビットマップを作成するための Windows 向けスタンドアロンツール

以下の図は、このアプリケーションプロジェクトの簡単なフローを示します。

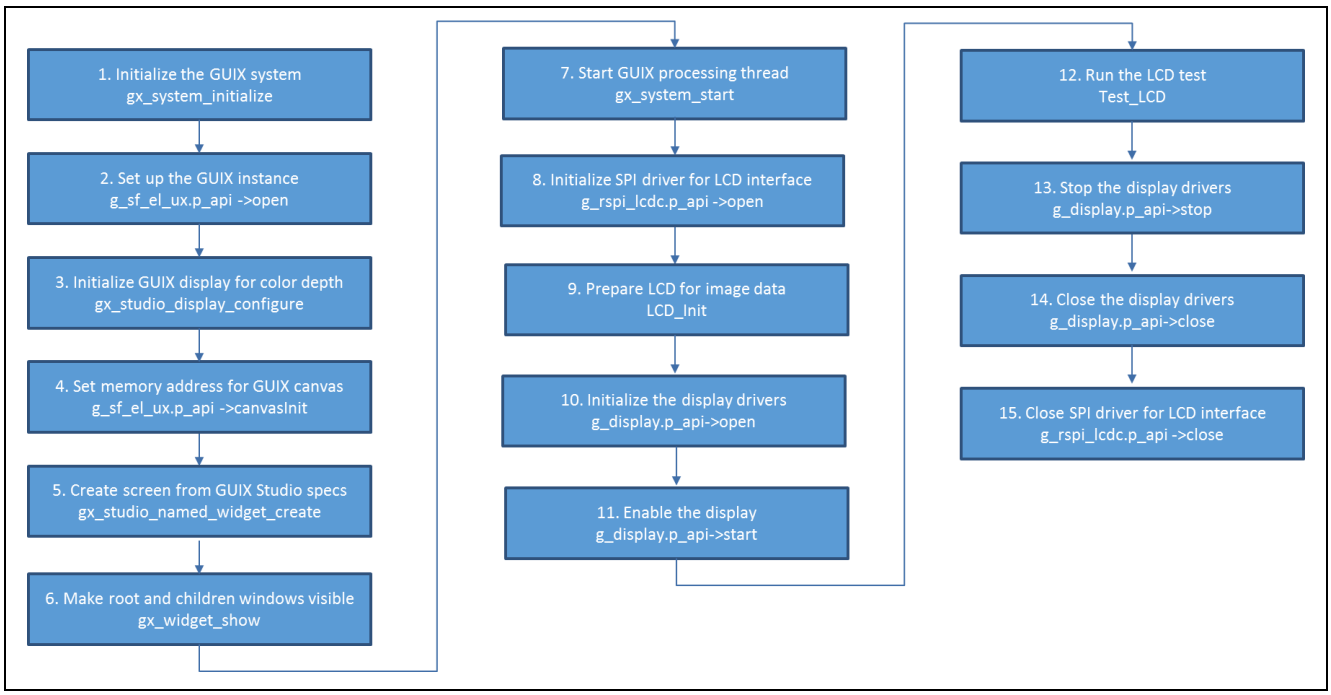

図 1　LCD_TEST モードでの GUIX Synergy ポートフレームワークモジュール アプリケーションプロジェクトのフロー

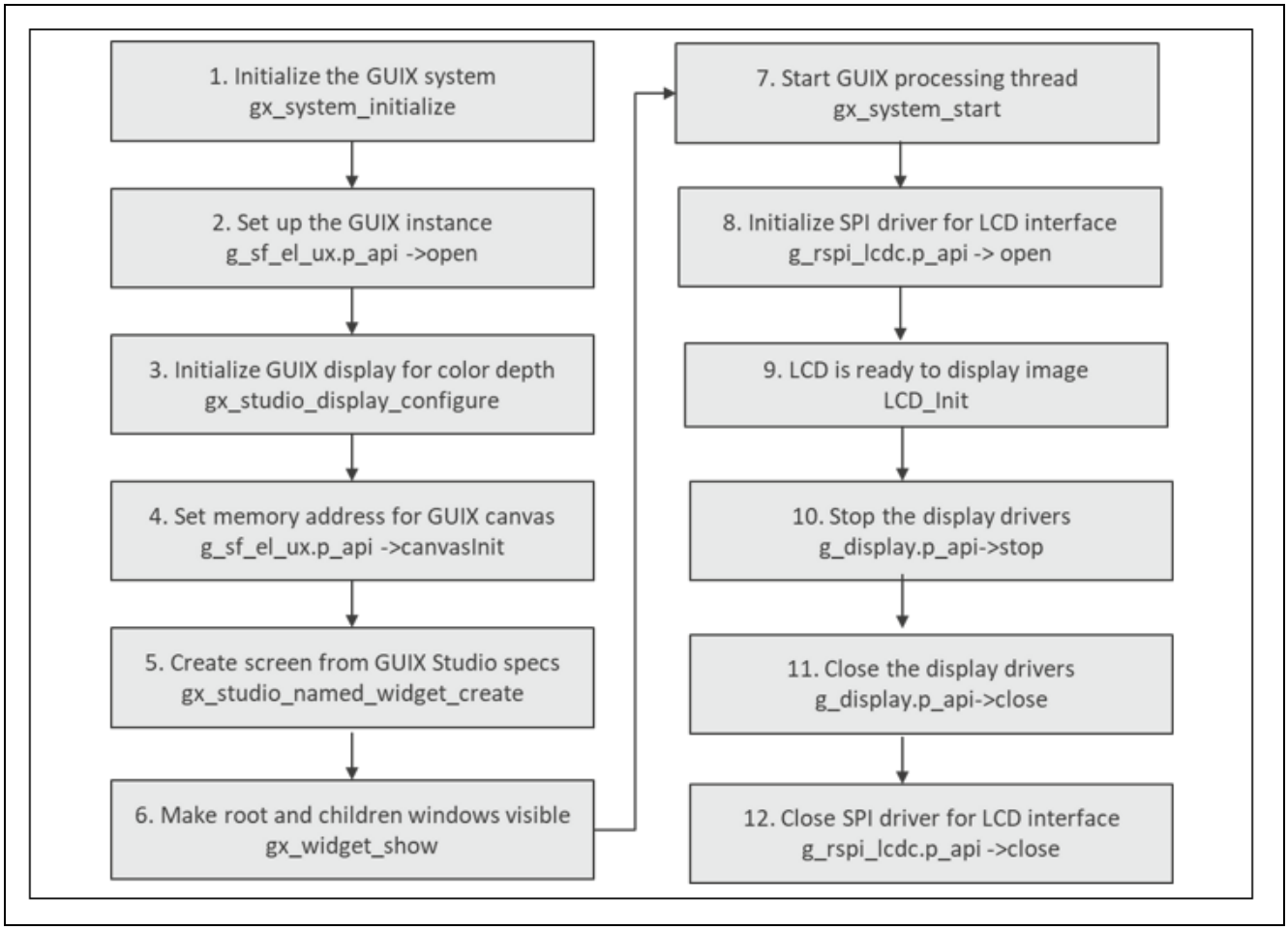

図 2　GUIX_TEST モードでの GUIX Synergy ポートフレームワークモジュールアプリケーションプロジェクトのフロー

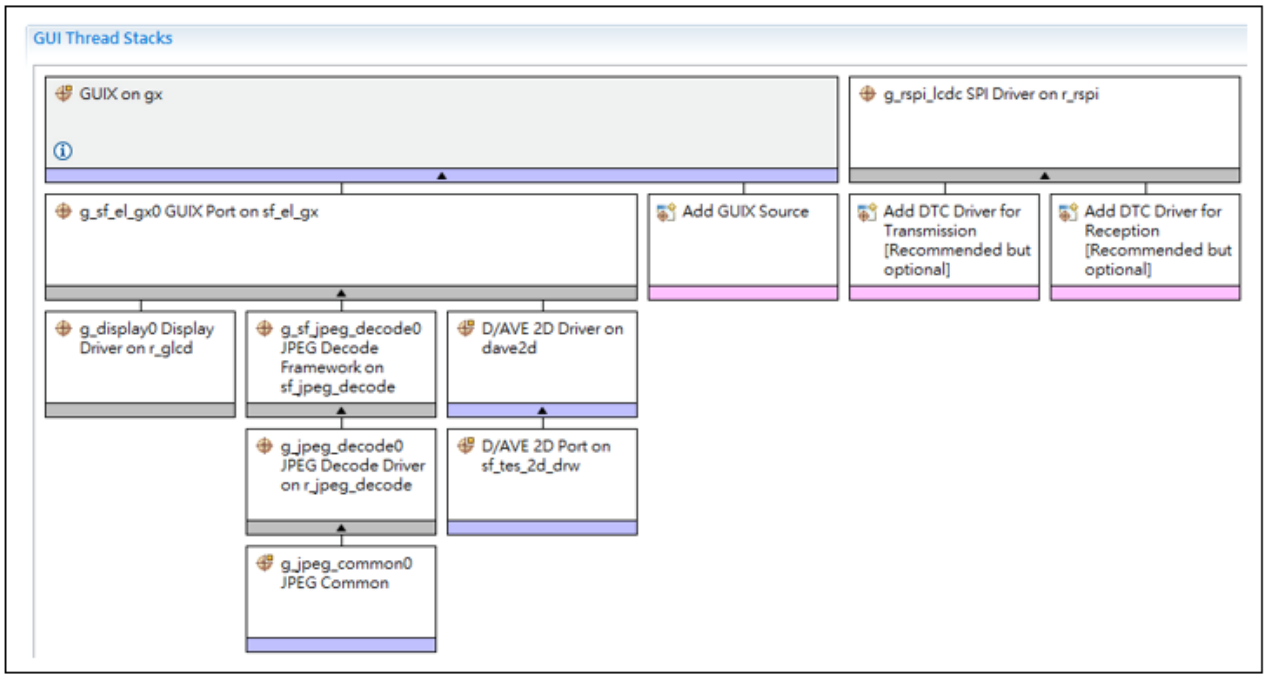

図 3 LCD ハードウェアへのインタフェースとして機能する GUIX Synergy ポートフレームワークモジュール アプリケーションプロジェクトの各種コンポーネント

SPI ドライバの r_rspi_lcdc インスタンスは、my_lcd_spi_callback というコールバックを使用するように定義してあります。この関数は guix_driver_sf_el_gx_mg_ap.c 内で定義してあります。この関数は呼び出された時点でセマフォを解放し、LCD が処理を続行します。このコールバックは、SPI が spi_callback_args_t ポインタ入力を通じて、イベント転送に関する情報をアプリケーションに返します。my_lcd_spi_callback は e² studio の **[Thread Stack]** (スレッドスタック) ペイン内で作成したセマフォを使用して、SPI ドライバインスタンスに対する読み取りと書き込みを制御します。

my_guix_thread_entry.c、guix_driver_sf_el_gx_mg_ap.c、guix_driver_sf_el_gx_mg_ap.h、my_gui_event_handler.c、lcd.h、lcd_setup.c の各ファイルは、このプロジェクトを ISDE にインストールした後、プロジェクト内に配置されます。ISDE でこれらのファイルを開き、API の主な使い方を確認するための説明を参照できます。

my_guix_thread_entry.c ファイルには、スレッドのエントリ関数があります。この関数は、printf のデバッグ出力を初期化します。次に、guix_driver_sf_el_gx_mg_ap.c 内で定義されている guix_driver_test_image_draw を 2 回呼び出します。1 回はモードを LCD_TEST に設定した状態、もう 1 回はモードを GUIX_TEST に設定した状態です。その後、これらのテストの結果を出力し、何らかのエラーが発生したかどうか、また GUIX がアプリケーションに通知した表示イベントの数を示します。

GUIX_TEST

このテストモードでは、GUIX Studio を使用してピクセルマップデータを保持できるリソースファイルと初期の画面状態を定義する仕様ファイルを作成し、プロジェクトの src フォルダにそれらのファイルを保存します。このプロジェクトでは、以下のような 4 つの GUIX Studio ファイルを、プロジェクトの src フォルダに配置する必要があります。

- guix_for_mg_ap_resources.c
- guix_for_mg_ap_resources.h
- guix_for_mg_ap_specifications.c
- guix_for_mg_ap_specifications.h

これらのファイル名は、GUIX Studio 内のプロジェクト名に基づくものであり、このプロジェクトの場合は guix_for_mg_ap でした。

guix_driver_sf_el_gx_mg_ap.c は、GUIX Studio が生成した各種ヘッダファイルを持っています。gx_api.h は GUIX サービスに対応し、lcd.h は LCD イメージ処理に対応しています。my_gui_thread.h は [Thread Stack] (スレッドスタック) コンポーネントに基づいて Synergy が生成したヘッダファイルです。また、このファイルは、GUIX システムに対応するコールバックである my_guix_callback を含め、いくつかの関数プロトタイプ（function prototype）を定義しています。GUIX はこのコールバックを使用して、GUIX の実行中に発生したイベントとエラーをアプリケーションに通知できます。このコールバックを設定するプロパティは、プロジェクトの g_sf_el_gx0 コンポーネント内にある **[Name of User Callback]** (ユーザコールバック関数の名前) プロパティです。

guix_driver_sf_el_gx_mg_ap.c は p_window_root というルートウィンドウへのポインタを定義します。このポインタは、GUIX と GUIX Studio の API で使用します。可視キャンバスを作成するたびに、そのキャンバスに対応するルートウィンドウを作成する必要があります。この特別なウィンドウは基本的に、すべてのトップレベル（top-level）アプリケーションウィンドウ（application window）とウィジェットに対するコンテナ（container）として機能します。詳細については、「参考情報」の章に掲載の『*GUIX User Guide*』を参照してください。

guix_driver_test_image_draw は、LCD パネルに対応するイメージを以下のように処理する関数です。

1. gx_system_initialize API を呼び出して、最初に GUIX システムを初期化します。
2. GUIX のインスタンスである g_sf_el_gx0 モジュールは、セットアップを実行します (コンテキストの更新中にミューテックスを作成して GUIX ドライバをロック、描画と表示を同期するためにセマフォを作成、GUIX キャンバスメモリ、画面の回転、およびその他の詳細を確認、さらに g_sf_el_gx0 制御ブロックを初期化)。
   a. g_sf_el_gx0.p_api->open()
3. 次に、gx_studio_display_configure API はディスプレイドライバを初期化します。テーマ（theme）、カラーパレット（color palette）、フォント、言語を含め、イメージに関する各種コンポーネントを集約して表示オブジェクト（display object）内に配置します。
   この API は、要求された GUIX リソースのテーマをインストールします。テーマは、フォント、カラー、ピクセルマップのデータを指定するもので、それぞれに Resource ID (リソース ID) が関連付けられています。この API はさらにアプリケーションの文字列テーブル（string table）をインストールし、現在アクティブな言語を定義します。MAIN_DISPLAY パラメータは、現在どのディスプレイに設定されているかを識別します (アプリケーションが複数のハードウェアディスプレイを定義している場合)。MAIN_DISPLAY_THEME_1 パラメータは、インストールするリソースのテーマを定義します。アプリケーションは複数のリソースのテーマをインストールできるからです。最後のLANGUAGE_ENGLISH パラメータは、最初にアクティブにする言語を定義します。GUIX アプリケーションは複数の言語を定義して、実行時（runtime）にアクティブな言語を選択できるからです。

   これらのリソースは、前述の GUIX Studio プロジェクトファイルの中で定義されています。
4. canvasInit API は、GUIX キャンバスに対応するメモリアドレスを設定します。これは、GUIX ディスプレイドライバによる描画の対象になるバッファ領域です。
5. gx_studio_named_widget_create API は、GUIX Studio が生成した仕様ファイルの中で渡されたデータを使用して、「splash_screen」ウィジェットを作成します。これはプライマリ画面になります。
6. gx_widget_show API は、ルートウィンドウと、すべての子ルートウィンドウを可視化します。
7. 次に、アプリケーションは gx_system_start API を呼び出します。この API は、GUIX システムのスレッドを起動します。GUIX キャンバスのリフレッシュ、または変更が発生したダーティ領域（dirty area）の再描画が必要なときに、このスレッドは、GUIX イベントキューに投稿されたイベントをディスパッチする役割を果たします。
8. 以下のように open 関数を呼び出して、シリアルポートドライバを初期化します。
   a. err = g_rspi_lcdc.p_api->open(g_rspi_lcdc.p_ctrl, g_rspi_lcdc.p_cfg);
9. その後、(lcd_setup.c 内で定義されている) LCD_init サービスを呼び出して、イメージデータを受け取るための LCD を準備します。

ここで、GUIX エンジンはイメージを LCD パネルに送信できます。

この期間のうちに、GUIX Studio プロジェクト内で作成したウィジェットの各種コンポーネントに対応するユーザ定義コールバックを呼び出すことができます。my_guix_event_handler.c ファイルは、このプロジェクトに対応する複数のコールバック関数を記述しています。特に、SplashScreenEventHandler コールバックです。このコールバックは、GUIX と GUIX Studio のサービスを使用して、実行時にどのようにウィジェット機能を制御するかを示します。このコールバックはイベントタイプを指定します。タイプとして、ボタンのクリック、またはタイマの期限切れを使用できます。スプラッシュ画面が表示されている時の GX_EVENT_SHOW イベントタイプの場合、このコールバックは 100 ティック（tick）で期限切れになるように設定したタイマを開始し (これは、100 ティック/秒の周期に設定したシステムの場合、1 秒の長さとなります)、このタイマは 1 回のみ期限切れになります (期限切れの時点でリセットされません)。

次に、制御を GUIX システムに返して、splash_screen (スプラッシュ画面) の表示処理を完了させます。このコールバックは、基礎となる（underlying）GUIX エンジンから GX_EVENT_TIMER イベントタイプを受け取った時点で、スプラッシュ画面内でシンプルな変更を実行します。メッセージを「Version 5.3.3」(バージョン 5.3.3) から「Hello World」(ハローワールド) に上書きします。

このイメージが数秒表示された後、以下のように guix_driver_test_image_draw サービスを数回呼び出して、ディスプレイドライバとシリアルポートドライバを閉じる必要があります。

1. ディスプレイドライバを停止します。
   g_display0.p_api->stop(g_display0.p_ctrl);
2. ディスプレイドライバインタフェースを閉じます。
   g_display0.p_api->close(g_display0.p_ctrl);
3. SPI ドライバインタフェースを閉じます。
   err = g_rspi_lcdc.p_api->close (g_rspi_lcdc.p_ctrl);

my_guix_thread_entry 関数に対して、エラーの数を返します。0 は、テストに成功したことを表します。

LCD_TEST

LCD_TEST は、SPI ドライバとディスプレイドライバ、および LCD ハードウェアのテストを行います。このテストモードは GUIX の各種サービスを使用しません。

このテストを実行するには、LCD SPI コールバックを定義しておく必要があります。ただし、GUIX のどのサービスも使用しないので、my_guix_callback を使用しません。LCD_TEST モードでは、シンプルなカラーバーを表示する test_LCD 関数を LCD_Setup.c 内で定義しています。

LCD_TEST モードの guix_driver_test_image_draw サービスは、ディスプレイドライバを直接開き、GUIX_TEST モードの場合と同じディスプレイドライバおよびシリアルポートインタフェースドライバを使用します。

1. 以下のように open を呼び出して、シリアルポートドライバを初期化します。
   err = g_rspi_lcdc.p_api->open(g_rspi_lcdc.p_ctrl, g_rspi_lcdc.p_cfg);
2. 以下のように open を呼び出して、ディスプレイドライバを初期化します。
   g_display0.p_api->open(g_display0.p_ctrl, g_display0.p_cfg)’
3. その後、LCD_init サービスを呼び出して、イメージデータを受け取るための LCD ハードウェアを準備します。
4. 以下の関数を呼び出してディスプレイを有効にします。
   g_display0.p_api->start(g_display0.p_ctrl)
5. イメージ描画関数である test_LCD を呼び出します。赤、緑、青 (RGB) という 3 色のカラーバーが LCD 画面に表示されます。

このイメージが数秒表示された後、以下のように guix_driver_test_image_draw サービスを呼び出して、ディスプレイドライバとシリアルポートドライバを閉じる必要があります。

1. ディスプレイドライバを停止します。
   g_display0.p_api->stop(g_display0.p_ctrl);
2. ディスプレイドライバインタフェースを閉じます。
   g_display0.p_api->close(g_display0.p_ctrl);
3. SPI ドライバインタフェースを閉じます。
   err = g_rspi_lcdc.p_api->close (g_rspi_lcdc.p_ctrl);

my_guix_thread_entry 関数に対して、エラーの数を返します。0 は、テストに成功したことを表します。

my_guix_thread_entry 関数内で SEMI_HOSTING が定義されている場合、両方のテストを実行した後、エラーの合計数と表示イベントの合計数がデバッグコンソールに出力されます。

注記：この説明は、Synergy ソフトウェアパッケージ内のデバッグコンソールで printf() を使用する方法をユーザが理解していることを想定しています。このような経験がない場合は、下記 WEB サイトの FAQ 2000008「Synergy ソフトウェアパッケージのデバッグコンソールで Printf_使用方法」という記事を参照してください。デバッグモードで変数ウォッチ機能を使用して結果を表示することもできます。

https://ja-support.renesas.com/knowledgeBase/17792531

このアプリケーションプロジェクトではターゲットボードや MCU の必須の操作（required operation）と物理プロパティ（physical property）をサポートするために、いくつかの重要なプロパティを設定しています。以下のプロパティとこのプロジェクトにおける設定値の表において、それら重要なプロパティを**太字**で示します。このアプリケーションプロジェクトを開き、[Properties] (プロパティ) ウィンドウでこれらの設定を確認することもできます。

表 2　GUIX Synergy Port Framework Module on GUIX on gx (gx の GUIX 上の GUIX Synergy ポートフレームワークモジュール) の設定

ISDE のプロパティ	値	説明
Enable Synergy 2D Drawing Engine Support (Synergy 2D 描画エンジンのサポートを有効にする)	Yes	[Synergy 2D Drawing Engine (DRW) Support] (Synergy 2D 描画エンジン (DRW) サポート) が有効になっている場合、Synergy DRW はテクスチャマッピング（texture mapping）付きで回転を処理します。有効になっていない場合、ソフトウェアによるコピーで回転を処理します。
Enable Synergy JPEG Support (Synergy JPEG サポートを有効にする)	Yes	この項目を有効にすると、JPEG イメージのサイズ、幅の単位（width modulus）、フォーマット、出力の整列などに制約が課されます。また、JPEG イメージをフレームバッファに直接デコードする処理に限定されます。(一方、ソフトウェアデコーダの場合、これらの制約はいずれも課されません。ただし、当然ですが、より多くの CPU 時間を消費する結果になります)

表 3　GUIX Synergy Port Framework Module on GUIX on sf_el_gx (sf_el_gx 上の GUIX Synergy ポートフレームワークモジュール) の設定

ISDE のプロパティ	値	説明
Parameter Checking (パラメータチェック)	Enabled	パラメータチェックを有効にします。
Name (名前)	g_sf_el_gx0	ISDE が生成する SF_EL_GX インスタンスの名前。名前は有効な C シンボルである必要があります。
Name of Display Driver Run-time Configuration (Must be a valid symbol) (ディスプレイドライバの実行時設定の名前 (有効なシンボルである必要があります))	g_display0_runtime_cfg_bg	[Synergy Configuration] (Synergy 設定) の中で DISPLAY モジュールに対して割り当てた実行時設定の名前を指定します。名前は有効な C シンボルである必要があり、NULL は許容されません。
Name of Frame Buffer A (Must be a valid symbol) (フレームバッファ A の名前 (有効なシンボルである必要があります))	g_display0_fb_background[0]	フレームバッファの名前を指定します。[Synergy Configuration] (Synergy 設定) 内の DISPLAY モジュールの設定は、作成するフレームバッファの名前を保持しています。そのフレームバッファの名前をここで設定します。名前は有効な C シンボルである必要があり、NULL は許容されません。
Name of Frame Buffer B (NULL allowed if consisting a single frame buffer system) (フレームバッファ B の名前 (単一フレームバッファシステムを使用している場合は NULL が使用可能))	g_display0_fb_background[1]	別のフレームバッファの名前を指定します。単一フレームバッファを使用するグラフィックスシステムを設計する場合、NULL をこのパラメータに割り当てます。それ以外の場合、[Name of Frame Buffer A] (フレームバッファ A の名前) パラメータを使用して、フレームバッファの名前を設定します。「Tearing in Single Buffer Designs」(単一バッファの設計で発生する画面遷移のティアリング) を参照してください。
Name of User Callback function (ユーザコールバック関数の名前)	my_guix_callback	イベントが発生したときに、このモジュールが起動する（呼び出す）ユーザコールバック関数の名前 (オプション)。

Screen Rotation Angle (Counterclockwise) (画面の回転角度 (反時計回り))	0	画面の回転角度 (「度」単位)。0 以外の値を選択した場合、画面の回転が有効になります。GUIX は画面イメージをフレームバッファに描画し、反時計回りで指定した角度だけイメージを回転します。
GUIX Canvas Buffer (required if rotation angle is not zero) (GUIX キャンバスバッファ (回転角度が 0 でない場合は必須))	Not used (使用しない)	画面の回転機能を有効にしない場合、このパラメータを [Not Used] (使用しない) に設定します。
Size of JPEG Work Buffer (valid if JPEG hardware acceleration enabled) (JPEG 作業バッファのサイズ (JPEG ハードウェアアクセラレーションが有効になっている場合はこのパラメータも有効))	81920	バイト単位の JPEG 作業バッファサイズ。値として、有効な整数の値を指定する必要があります。JPEG アクセラレーションを使用しない場合、0 に設定することも許容されます。バッファサイズを大きくすると、描画時間が短くなります。「Size of JPEG Work Buffer」(JPEG 作業バッファのサイズ) を参照してください
Memory section for GUIX Canvas Buffer (GUIX キャンバスバッファのメモリセクション)	bss	GUIX キャンバスバッファの割り当て先として使用するメモリセクションの名前。リンカスクリプトファイル（linker script file）で定義済みの有効なセクション名を入力します。名前は有効な C シンボルである必要があります。
Memory section for JPEG Work Buffer (JPEG 作業バッファのメモリセクション)	bss	JPEG 作業バッファの割り当て先として使用するメモリセクションの名前。リンカスクリプトファイル（linker script file）で定義済みの有効なセクション名を入力します。名前は有効な C シンボルである必要があります。

表 4　r_glcd 上の GLCD HAL モジュールの設定

ISDE のプロパティ	値	説明
Parameter Checking (パラメータチェック)	Enabled	パラメータチェックを有効または無効にします。
Name (名前)	g_display0	GLCDC モジュール制御ブロックインスタンスに使用する名前。この名前は、他の可変インスタンスのプレフィックスにも使用されます。
Name of display callback function to be defined by user (ユーザが定義する表示コールバック関数の名前)	NULL	名前は有効な C シンボルである必要があります。
Input - Panel clock source select (入力 - パネルクロックソースの選択)	Internal clock (GLCDCLK) (内部クロック (GLCDCLK))	開発中システムに応じて、パネルクロックソースを選択します。

Input - Graphics screen1 (入力 – グラフィックス画面 1)	Used (使用する)	グラフィックス画面 N を使用する場合は、[Used] (使用する) を指定します。その場合、ISDE が、グラフィックス画面 1 に対応するフレームバッファである「display_fb_background」、グラフィックス画面 2 に対応する「display_fb_foreground」を自動的に生成します。どちらかのグラフィックス画面を使用しない場合は、[Not used] (使用しない) を指定します。その場合、フレームバッファは作成されません。[Not used] (使用しない) を指定した場合、フレームバッファに対するメモリ読み取りアクセスは発生しないので、バス帯域幅の使用量が減ることに注意してください。
Input - Graphics screen1 frame buffer name (入力 – グラフィックス画面 1 のフレームバッファの名前)	fb_background	フレームバッファのカスタム名。
Input - Number of Graphics screen1 frame buffer (入力 - グラフィックス画面 1 のフレームバッファ数)	2	グラフィックス数の選択。
Input - section where Graphics screen1 frame buffer allocated (入力 - グラフィックス画面 1 のフレームバッファの割り当て先セクション)	bss	フレームバッファの割り当て先となるセクションの名前を指定します。この項目が有効になるのは、[Input - Graphics screen1] (入力 – グラフィックス画面 1) を [Used] (使用する) に設定した場合です。
Input - Graphics screen1 input horizontal size (入力 - グラフィックス画面 1 の入力水平サイズ)	256	水平ピクセル数を指定します。
Input - Graphics screen1 vertical size (入力 - グラフィックス画面 1 の垂直サイズ)	320	垂直ピクセル数を指定します。
Input - Graphics screen1 input horizontal stride (not bytes but pixels) (入力 - グラフィックス画面 1 の入力水平ストライド (バイト単位ではなくピクセル単位))	256	各水平ラインのメモリストライドを指定します。この値は、実際のバイト数ではなくピクセル数で指定する必要があります。通常、このパラメータは、[input horizontal size] (入力水平サイズ) と同じ値に設定します。
Input - Graphics screen1 input format (入力 - グラフィックス画面 1 の入力フォーマット)	16bits ARGB1555, 16bits	グラフィックス画面の入力フォーマットを指定します。CLUT フォーマットを選択する場合、開始する前に、clut を使用して CLUT データを書き込む必要があります。デフォルト設定は、RGB565 フォーマットのイメージをサポートします。

Input - Graphics screen1 input line descending (入力 - グラフィックス画面 1 の入力ライン降順)	Not used (使用しない)	フレームバッファ内で、イメージデータが下端のラインから上端のラインへ向かう降順になっている場合、[On] (オン) を指定します。通常は、[Off] (オフ) を指定します。
Input - Graphics screen1 input line repeat (入力 – グラフィックス画面 1 の入力ライン反復)	Off (オフ)	LCD パネルサイズより小さいラスタイメージを反復して読み取ることが予期される場合、[On] (オン) を指定します。通常は、[Off] (オフ) を指定します。詳細については、「Line Repeating」(ライン反復) 機能の説明を参照してください。
Input - Graphics screen1 input line repeat times (入力 - グラフィックス画面 1 の入力ライン反復回数)	0	1 つのフレーム内で 1 つのラスタイメージを反復して読み取る場合、反復回数を指定します。
Input - Graphics screen1 layer coordinate X (入力 – グラフィックス画面 1 のレイヤ座標 X)	0	バックグラウンド画面からグラフィックス画面までの水平オフセットをピクセル単位で指定します。
Input - Graphics screen1 layer coordinate Y (入力 – グラフィックス画面 1 のレイヤ座標 Y)	0	バックグラウンド画面からグラフィックス画面までの垂直オフセットをピクセル単位で指定します。
Input - Graphics screen1 layer background color alpha (入力 - グラフィックス画面 1 のレイヤにおけるバックグラウンドカラーのアルファ)	255	アルファ値に基づいて、グラフィックス画面 2 (フォアグラウンドグラフィックス画面) がグラフィックス画面 1 (バックグラウンドグラフィックス画面) にブレンドされるか、グラフィックス画面 1 がモノクロームのバックグラウンド画面にブレンドされるかが決まります。
Input - Graphics screen1 layer background color Red (入力 – グラフィックス画面 1 のレイヤにおけるバックグラウンドカラーの赤)	255	グラフィックス画面 N のバックグラウンドカラーを指定します。
Input - Graphics screen1 layer background color Green (入力 – グラフィックス画面 1 のレイヤにおけるバックグラウンドカラーの緑)	255	グラフィックス画面 N のバックグラウンドカラーを指定します。
Input - Graphics screen1 layer background color Blue (入力 – グラフィックス画面 1 のレイヤにおけるバックグラウンドカラーの青)	255	グラフィックス画面 N のバックグラウンドカラーを指定します。

Input - Graphics screen1 layer fading control (入力 – グラフィックス画面 1 のレイヤフェード制御)	None	グラフィックス画面のフェードインを実行する場合、[On] (オン) を指定します。透明な画面が徐々に不透明に変化します。グラフィックス画面のフェードアウトを実行する場合、[Off] (オフ) を指定します。不透明な画面が徐々に透明に変化します。GLCDC ハードウェアによるアクセラレーションがこの処理に適用され、一度開始すると停止できないことに注意してください。statusGet を使用して、遷移のステータス（transition status）を監視できます。
Input - Graphics screen1 layer fade speed (入力 – グラフィックス画面 1 のレイヤフェード速度)	0	フェードの遷移が完了するまでのフレーム数を指定します。
Input - Graphics screen2 (入力 – グラフィックス画面 2)	Not used (使用しない)	グラフィックス画面 N を使用する場合は、[Used] (使用する) を指定します。その場合、ISDE が、グラフィックス画面 1 に対応するフレームバッファである「display_fb_background」、グラフィックス画面 2 に対応する「display_fb_foreground」を自動的に生成します。どちらかのグラフィックス画面を使用しない場合は、[Not used] (使用しない) を指定します。その場合、フレームバッファは作成されません。[Not used] (使用しない) を指定した場合、フレームバッファに対するメモリ読み取りアクセスは発生しないので、バス帯域幅の使用量が減ることに注意してください。
Input - Graphics screen2 frame buffer name (入力 – グラフィックス画面 2 のフレームバッファの名前)	fb_foreground	フレームバッファのカスタム名。
Input - Number of Graphics screen2 frame buffer (入力 - グラフィックス画面 2 のフレームバッファ数)	2	グラフィックス数の選択。
Input - section where Graphics screen2 frame buffer allocated (入力 - グラフィックス画面 2 のフレームバッファの割り当て先セクション)	sdram (SDRAM)	フレームバッファの割り当て先となるセクションの名前を指定します。この項目が有効になるのは、[Input - Graphics screen1] (入力 – グラフィックス画面 1) を [Used] (使用する) に設定した場合です。
Input - Graphics screen2 input horizontal size (入力 - グラフィックス画面 2 の入力水平サイズ)	800	水平ピクセル数を指定します。デフォルト値は、800x480 ピクセルのイメージに対応するサイズです
Input - Graphics screen2 vertical size (入力 - グラフィックス画面 2 の垂直サイズ)	480	垂直ピクセル数を指定します。デフォルト値は、800x480 ピクセルのイメージに対応するサイズです。

Input - Graphics screen2 input horizontal stride (not bytes but pixels) (入力 - グラフィックス画面 2 の入力水平ストライド (バイト単位ではなくピクセル単位))	800	各水平ラインのメモリストライド（memory stride）を指定します。この値は、実際のバイト数ではなくピクセル数で指定する必要があります。通常、このパラメータは、[input horizontal size] (入力水平サイズ) と同じ値に設定します。デフォルト値は、800x480 ピクセルのイメージに対応するサイズです。
Input - Graphics screen2 input format (入力 - グラフィックス画面 2 の入力フォーマット)	16bits RGB565 (16 ビット RGB 565)	グラフィックス画面の入力フォーマットを指定します。CLUT フォーマットを選択する場合、開始する前に、clut を使用して CLUT データを書き込む必要があります。デフォルト設定は、RGB565 フォーマットのイメージをサポートします。
Input - Graphics screen2 input line descending (入力 - グラフィックス画面 2 の入力ライン降順)	Off (オフ)	フレームバッファ内で、イメージデータが下端のラインから上端のラインへ向かう降順になっている場合、[On] (オン) を指定します。通常は、[Off] (オフ) を指定します。
Input - Graphics screen2 input line repeat (入力 – グラフィックス画面 2 の入力ライン反復)	Off (オフ)	LCD パネルサイズより小さいラスタイメージを反復して読み取ることが予期される場合、[On] (オン) を指定します。通常は、[Off] (オフ) を指定します。詳細については、「Line Repeating」(ライン反復) 機能の説明を参照してください。
Input - Graphics screen2 input line repeat times (入力 - グラフィックス画面 2 の入力ライン反復回数)	0	1 つのフレーム内で 1 つのラスタイメージを反復して読み取る場合、反復回数を指定します。
Input - Graphics screen2 layer coordinate X (入力 - グラフィックス画面 2 のレイヤ座標 X)	0	バックグラウンド画面からグラフィックス画面までの水平オフセットをピクセル単位で指定します。
Input - Graphics screen2 layer coordinate Y (入力 - グラフィックス画面 2 のレイヤ座標 Y)	0	バックグラウンド画面からグラフィックス画面までの垂直オフセットをピクセル単位で指定します。
Input - Graphics screen2 layer background color alpha (入力 - グラフィックス画面 2 のレイヤにおけるバックグラウンドカラーのアルファ)	255	アルファ値に基づいて、グラフィックス画面 2 (フォアグラウンドグラフィックス画面) がグラフィックス画面 1 (バックグラウンドグラフィックス画面) にブレンドされるか、グラフィックス画面 1 がモノクロームのバックグラウンド画面にブレンドされるかが決まります。
Input - Graphics screen2 layer background color Red (入力 – グラフィックス画面 2 のレイヤにおけるバックグラウンドカラーの赤)	255	グラフィックス画面 N のバックグラウンドカラーを指定します。
Input - Graphics screen2 layer background color Green (入力 – グラフィックス画面 2 のレイヤにおけるバックグラウンドカラーの緑)	255	グラフィックス画面 N のバックグラウンドカラーを指定します。

Input - Graphics screen2 layer background color Blue (入力 – グラフィックス画面 2 のレイヤにおけるバックグラウンドカラーの青)	255	グラフィックス画面 N のバックグラウンドカラーを指定します。
Input - Graphics screen2 layer fading control (入力 – グラフィックス画面 2 のレイヤフェード制御)	None	グラフィックス画面のフェードインを実行する場合、[On] (オン) を指定します。透明な画面が徐々に不透明に変化します。グラフィックス画面のフェードアウトを実行する場合、[Off] (オフ) を指定します。不透明な画面が徐々に透明に変化します。GLCDC ハードウェアによるアクセラレーションがこの処理に適用され、一度開始すると停止できないことに注意してください。statusGet を使用して、遷移のステータス（transition status）を監視できます。
Input - Graphics screen2 layer fade speed (入力 – グラフィックス画面 2 のレイヤフェード速度)	0	フェードの遷移が完了するまでのフレーム数を指定します。
Output - Horizontal total cycles (出力 – 水平合計サイクル数)	320	各水平ライン内の合計サイクル数を指定します。開発中システムの LCD パネルシートのデータシートで定義されているのと同じサイクル数に設定してください。デフォルト値は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Horizontal active video cycles (出力 – 水平アクティブビデオサイクル数)	240	各水平ライン内のアクティブビデオサイクル数を指定します。開発中システムの LCD パネルシートのデータシートで定義されているのと同じサイクル数に設定してください。デフォルト値は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Horizontal back porch cycles (出力 – 水平バックポーチサイクル数)	6	各水平ライン内のバックポーチサイクル数を指定します。Hsync サイクル開始の時点で、バックポーチが開始されます。つまり、バックポーチサイクルは、複数の Hsync サイクルで構成されています。開発中システムの LCD パネルシートのデータシートで定義されているのと同じサイクル数に設定してください。デフォルト値は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Horizontal sync signal cycles (出力 - 水平同期信号サイクル数)	4	Hsync 信号のアサート（assertion）サイクル数を指定します。開発中システムの LCD パネルシートのデータシートで定義されているのと同じサイクル数に設定してください。デフォルト値は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Horizontal sync signal polarity (出力 - 水平同期信号の極性)	Low active	開発中システムに合わせて、Hsync 信号の極性（porality）を選択します。デフォルト設定は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。

Output - Vertical total lines (出力 - 垂直合計ライン数)	328	各フレーム内の合計ライン数を指定します。開発中システムの LCD パネルシートのデータシートで定義されているのと同じライン数に設定してください。デフォルト値は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Vertical active video lines (出力 - 垂直アクティブビデオライン数)	320	各フレーム内のアクティブビデオライン数を指定します。開発中システムの LCD パネルシートのデータシートで定義されているのと同じライン数に設定してください。デフォルト値は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Vertical back porch lines (出力 - 垂直バックポーチライン数)	4	各フレーム内のバックポーチライン数を指定します。Vsync ライン開始の時点で、バックポーチが開始されます。つまり、バックポーチラインは、複数の Vsync ラインで構成されています。開発中システムの LCD パネルシートのデータシートで定義されているのと同じライン数に設定してください。デフォルト値は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Vertical sync signal lines (出力 - 垂直同期信号ライン数)	4	各フレーム内の Vsync 信号アサートライン数を指定します。開発中システムの LCD パネルシートのデータシートで定義されているのと同じライン数に設定してください。デフォルト値は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Vertical sync signal polarity (出力 - 垂直同期信号の極性)	Low active	開発中システムに合わせて、Vsync 信号の極性（porality）を選択します。デフォルト設定は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Format (出力 - フォーマット)	16bits RGB565 (16 ビット RGB 565)	開発中 LCD パネルに合わせて、グラフィックス画面の出力フォーマットを指定します。デフォルト設定は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Endian (出力 - エンディアン)	Little endian (リトルエンディアン)	LCD パネルに出力する信号のデータエンディアン（data endian）を選択します。デフォルト設定は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Color order (出力 - 色の順序)	RGB	LCD パネルに出力する信号のデータ順序を選択します。必要な場合、青 (B) と赤 (R) の順序を入れ替えることができます。デフォルト設定は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Data Enable Signal Polarity (出力 - データイネーブル信号の極性)	High active	開発中システムに合わせて、Data Enable (データイネーブル) 信号の極性を選択します。デフォルト設定は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。
Output - Sync edge (出力 - 同期エッジ)	Rising Edge (立ち上がりエッジ)	開発中システムに合わせて、Sync 信号の極性 (立ち上がりと立ち下がりのどちらでトリガするか) を選択します。デフォルト設定は、S7G2 PE-HMI1 ボードの LCD パネルに一致しています。

Output - Background color alpha channel (出力 - バックグラウンドカラーのアルファチャネル)	255	バックグラウンド画面のバックグラウンドカラーを指定します。
Output - Background color R channel (出力 - バックグラウンドカラーの R チャネル)	0	バックグラウンド画面のバックグラウンドカラーを指定します。
Output - Background color G channel (出力 - バックグラウンドカラーの G チャネル)	0	バックグラウンド画面のバックグラウンドカラーを指定します。
Output - Background color B channel (出力 - バックグラウンドカラーの B チャネル)	0	バックグラウンド画面のバックグラウンドカラーを指定します。
CLUT	Not used (使用しない)	グラフィックス画面の入力フォーマットとして CLUT フォーマットを選択する場合、[Used] (使用する) を指定します。その場合、ISDE が自動生成するソースファイルの中で、CLUT ソースデータに対応する「CLUT_buffer」という名前のバッファが生成されます。
CLUT - CLUT buffer size (CLUT - CLUT バッファサイズ)	256	CLUT ソースデータバッファに対応するエントリの数を指定します。各エントリは、4 バイト (1 ワード) を使用します。ISDE が自動生成するソースファイルの中で、このパラメータで指定したワード数の CLUT ソースデータが生成されます。
TCON - Hsync pin select (TCON - 水平同期端子の選択)	LCD_TCON0	開発中システムに合わせて、Hsync 信号で使用する TCON 端子を選択します。デフォルト設定は、S7G2 PE-HMI1 ボードの LCD パネルに対応しています。
TCON - Vsync pin select (TCON - 垂直同期端子の選択)	LCD_TCON1	開発中システムに合わせて、Vsync 信号で使用する TCON 端子を選択します。デフォルト設定は、S7G2 PE-HMI1 ボードの LCD パネルに対応しています。
TCON - DataEnable pin select (TCON - データイネーブル端子の選択)	LCD_TCON2	開発中システムに合わせて、DataEnable 信号で使用する TCON 端子を選択します。デフォルト設定は、S7G2 PE-HMI1 ボードの LCD パネルに対応しています。
TCON - Panel clock division ratio (TCON - パネルクロック分周比)	1/32	クロックソース分周器の値を選択します。ピクセルクロックについては、この表の末尾にあるソースクロックに関する注記を参照してください。
Color correction - Brightness (色補正 - 輝度)	Off (オフ)	輝度制御を行う場合、[On] (オン) を指定します。[Off] (オフ) を指定する場合、以下の設定は出力カラーに影響を及ぼしません。
Color correction - Brightness R channel (色補正 - 輝度 R チャネル)	512	出力カラーのレベルは以下のように計算されます。出力カラーレベル = 入力カラーレベル ± 512。R、G、B の各チャネルに対して値を設定してください。

Color correction - Brightness G channel (色補正 - 輝度 G チャネル)	512	出力カラーのレベルは以下のように計算されます。出力カラーレベル = 入力カラーレベル ± 512。R、G、B の各チャネルに対して値を設定してください。
Color correction - Brightness B channel (色補正 - 輝度 B チャネル)	512	出力カラーのレベルは以下のように計算されます。出力カラーレベル = 入力カラーレベル ± 512。R、G、B の各チャネルに対して値を設定してください。
Color correction - Contrast (色補正 - コントラスト)	Off (オフ)	コントラスト制御を行う場合、[On] (オン) を指定します。[Off] (オフ) を指定する場合、以下の設定は出力カラーに影響を及ぼしません。
Color correction - Contrast(gain) R channel (色補正 - コントラスト (ゲイン) R チャネル)	128	出力カラーのレベルは以下のように計算されます。出力カラーレベル = 入力カラーレベル x (/128)。R、G、B の各チャネルに対して値を設定してください。
Color correction - Contrast(gain) G channel (色補正 - コントラスト (ゲイン) G チャネル)	128	出力カラーのレベルは以下のように計算されます。出力カラーレベル = 入力カラーレベル x (/128)。R、G、B の各チャネルに対して値を設定してください。
Color correction - Contrast(gain) B channel (色補正 - コントラスト (ゲイン) B チャネル)	128	出力カラーのレベルは以下のように計算されます。出力カラーレベル = 入力カラーレベル x (/128)。R、G、B の各チャネルに対して値を設定してください。
Color correction - Gamma correction(Red) (色補正 - ガンマ補正 (赤))	Off (オフ)	R、G、B の各チャネルを制御します。赤 (R) チャネルでガンマ補正を実行する場合、[On] (オン) を指定します。[Off] (オフ) を指定する場合、ゲインとしきい値（threshold）に関する設定は出力カラーに影響を及ぼしません。
Color correction - Gamma gain R[0-15] (色補正 - ガンマゲイン R[0-15])	0	ガンマ補正曲線の領域 N で、赤 (R) チャネルに対応するゲインの値を設定します。領域 N に対応するゲイン設定は、カラーレベルが ((Gamma threshold R[N-1])<<2) と ((Gamma threshold R[N])<<2) の間である入力データに対して適用されます。出力値は以下のように計算されます。出力カラーレベル = 入力カラーレベル / 1024 (/128)。
Color correction - Gamma threshold R[0-15] (色補正 - ガンマしきい値 R[0-15])	0	ガンマ補正曲線の領域 N で、赤 (R) チャネルに対応するしきい値を設定します。領域 N に対応するゲイン設定は、カラーレベルが Gamma threshold R[N-1] (ガンマしきい値 R[N-1]) と Gamma threshold R[N] (ガンマしきい値 R[N]) の間である入力データに対して適用されます。出力値は以下のように計算されます。出力カラーレベル = 入力カラーレベル / 1024 (/128)。
Color correction - Gamma correction(Green) (色補正 - ガンマ補正 (緑))	Off (オフ)	R、G、B の各チャネルを制御します。緑 (G) チャネルでガンマ補正を実行する場合、[On] (オン) を指定します。[Off] (オフ) を指定する場合、ゲインとしきい値（threshold）に関する設定は出力カラーに影響を及ぼしません。

Color correction - Gamma gain G[0-15] (色補正 - ガンマゲイン G[0-15])	0	ガンマ補正曲線の領域 N で、緑 (G) チャネルに対応するゲインの値を設定します。領域 N に対応するゲイン設定は、カラーレベルが ((Gamma threshold G[N-1])<<2) と ((Gamma threshold G[N])<<2) の間である入力データに対して適用されます。出力値は以下のように計算されます。出力カラーレベル = 入力カラーレベル / 1024 (/128)。
Color correction - Gamma threshold G[0-15] (色補正 - ガンマしきい値 G[0-15])	0	ガンマ補正曲線の領域 N で、緑 (G) チャネルに対応するしきい値を設定します。領域 N に対応するゲイン設定は、カラーレベルが Gamma threshold G[N-1] (ガンマしきい値 G[N-1]) と Gamma threshold G[N] (ガンマしきい値 G[N]) の間である入力データに対して適用されます。出力値は以下のように計算されます。出力カラーレベル = 入力カラーレベル / 1024 (/128)。
Color correction - Gamma correction(Blue) (色補正 - ガンマ補正 (青))	Off (オフ)	R、G、B の各チャネルを制御します。青 (B) チャネルでガンマ補正を実行する場合、[On] (オン) を指定します。[Off] (オフ) を指定する場合、ゲインとしきい値（threshold）に関する設定は出力カラーに影響を及ぼしません。
Color correction - Gamma gain B[0-15] (色補正 - ガンマゲイン B[0-15])	0	ガンマ補正曲線の領域 N で、青 (B) チャネルに対応するゲインの値を設定します。領域 N に対応するゲイン設定は、カラーレベルが ((Gamma threshold B[N-1])<<2) と ((Gamma threshold B[N])<<2) の間である入力データに対して適用されます。出力値は以下のように計算されます。出力カラーレベル = 入力カラーレベル / 1024 (/128)。
Color correction - Gamma threshold B[0-15] (色補正 - ガンマしきい値 B[0-15])	0	ガンマ補正曲線の領域 N で、青 (B) チャネルに対応するしきい値を設定します。領域 N に対応するゲイン設定は、カラーレベルが Gamma threshold B[N-1] (ガンマしきい値 B[N-1]) と Gamma threshold B[N] (ガンマしきい値 B[N]) の間である入力データに対して適用されます。出力値は以下のように計算されます。出力カラーレベル = 入力カラーレベル / 1024 (/128)。
Dithering (ディザリング)	Off (オフ)	ディザリング (色の遷移をできるだけ滑らかにする意図的なノイズ混入) を有効にします。RGB666 または RGB565 という出力フォーマットを選択した状況でカラーバンド (色ごとの帯) が発生しやすくなりますが、その影響を緩和するためのディザ効果を適用する場合は [On] (オン) を指定します。変換を行う際にディザリングを適用することもできます。[Off] (オフ) を指定する場合、ディザリングに関する以下の設定は出力に影響を及ぼしません。ディザ効果の詳細については、ハードウェアマニュアルの「Output Control Block Panel Dither Correction Register (OUT_PDTHA)」(出力制御ブロックパネルのディザ補正レジスタ (OUT_PDTHA)) を参照してください。

Dithering - Mode (ディザリング - モード)	Truncate (切り捨て)	ディザリングモードを指定します。詳細については、ハードウェアマニュアルの「Output Control Block Panel Dither Correction Register (OUT_PDTHA)」(出力制御ブロックパネルのディザ補正レジスタ (OUT_PDTHA)) を参照してください。
Dithering - Pattern A (ディザリング - パターン A)	Pattern 11 (パターン 11)	2X2 のパターンモードに対応するディザリングパターンを指定します。詳細については、ハードウェアマニュアルの「Output Control Block Panel Dither Correction Register (OUT_PDTHA)」(出力制御ブロックパネルのディザ補正レジスタ (OUT_PDTHA)) を参照してください。
Dithering - Pattern B (ディザリング - パターン B)	Pattern 11 (パターン 11)	2X2 のパターンモードに対応するディザリングパターンを指定します。詳細については、ハードウェアマニュアルの「Output Control Block Panel Dither Correction Register (OUT_PDTHB)」(出力制御ブロックパネルのディザ補正レジスタ (OUT_PDTHB)) を参照してください。
Dithering - Pattern C (ディザリング - パターン C)	Pattern 11 (パターン 11)	2X2 のパターンモードに対応するディザリングパターンを指定します。詳細については、ハードウェアマニュアルの「Output Control Block Panel Dither Correction Register (OUT_PDTHC)」(出力制御ブロックパネルのディザ補正レジスタ (OUT_PDTHC)) を参照してください。
Dithering - Pattern D (ディザリング - パターン D)	Pattern 11 (パターン 11)	2X2 のパターンモードに対応するディザリングパターンを指定します。詳細については、ハードウェアマニュアルの「Output Control Block Panel Dither Correction Register (OUT_PDTHD)」(出力制御ブロックパネルのディザ補正レジスタ (OUT_PDTHD)) を参照してください。
Misc - Correction Process Order (その他 - 補正プロセスの順序)	Brightness and Contrast then Gamma (輝度とコントラスト、次いでガンマ)	必要な場合、色補正（color correction）の処理順序（processing order）を指定します。
Line Detect Interrupt Priority (ライン検出割り込みの優先順位)	Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX)	ドライバは、割り込みの優先順位に関する有効な設定を必要とします。無効な場合、ドライバは動作しません。
Underflow 1 Interrupt Priority (アンダーフロー 1 割り込みの優先順位)	Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX)	ドライバは、割り込みの優先順位に関する有効な設定を必要とします。無効な場合、ドライバは動作しません。
Underflow 2 Interrupt Priority (アンダーフロー 2 割り込みの優先順位)	Disabled	ドライバは、割り込みの優先順位に関する有効な設定を必要とします。無効な場合、ドライバは動作しません。

表 5　sf_jpeg_decode の JPEG デコードフレームワークモジュールの設定項目

ISDE のプロパティ	値	説明
Parameter Checking (パラメータチェック)	Enabled	パラメータチェックを有効または無効にします。
Name (名前)	g_sf_jpeg_decode0	JPEG デコードフレームワークモジュールインスタンスに使用する名前。

表 6　r_jpeg 上の JPEG デコード HAL モジュールの設定

ISDE のプロパティ	値	説明
Parameter Checking (パラメータチェック)	Enabled	パラメータエラーチェックを有効または無効にします。
Name (名前)	g_jpeg_decode0	JPEG デコードモジュールインスタンスに使用する名前。
Byte Order for Input Data Format (入力データフォーマットのバイト順序)	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8)	入力データのバイト順序を指定します。順序は 8 バイトごとに指定に従ってスワップされます。
Byte Order for Output Data Format (出力データフォーマットのバイト順序)	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8)	出力データのバイト順序を指定します。順序は 8 バイトごとに指定に従ってスワップされます。
Output Data Color Format (出力データのカラーフォーマット)	Pixel Data RGB565 format, Pixel Data ARGBB888 format デフォルト: Pixel Data RGB565 format (ピクセルデータ RGB 565 フォーマット)	出力データフォーマットを指定します。
Alpha value to be applied to decoded pixel data (only valid for ARGB8888 format) (デコード済みピクセルデータに適用するアルファ値 (ARGB 8888 フォーマットでのみ有効))	255	出力データフォーマットのアルファ値を指定します (ARGB 8888 フォーマットでのみ有効)。
Name of user callback function (ユーザコールバック関数の名前)	NULL	ユーザコールバック関数の名前を指定します。
Decompression Interrupt Priority (解凍割り込みの優先順位)	Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX)	解凍割り込みの優先順位の選択。
Data Transfer Interrupt Priority (データ転送割り込みの優先順位)	Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX)	データ転送割り込みの優先順位の選択。

表 7　dave2d 上の D/AVE 2D ドライバの設定

ISDE のプロパティ	値	説明
可能な設定はありません		

表 8　sf_tes_2d_drw 上の D/AVE 2D ポートの設定

ISDE のプロパティ	値	説明
Work memory size for display lists in bytes (ディスプレイリストで使用する作業メモリのバイト単位のサイズ)	32768	ディスプレイリストで使用する作業メモリのバイト単位のサイズに関する選択
DRW Interrupt Priority (DRW 割り込みの優先順位)	Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX)	DRW INT の選択

RSPI バスの端子を設定する必要があります。[Thread Stack] (スレッドスタック) ペインで、[Pins] タブ -> [Peripherals] – [Connectivity:SPI] -> [SPI0] ([端子] タブ -> [周辺装置] – [接続:SPI] -> [SPI0]) を選択します。[Operation Mode] (動作モード) を [Enable] (有効) に変更します。端子割り当てが以下の図のようになっていることを確認します。

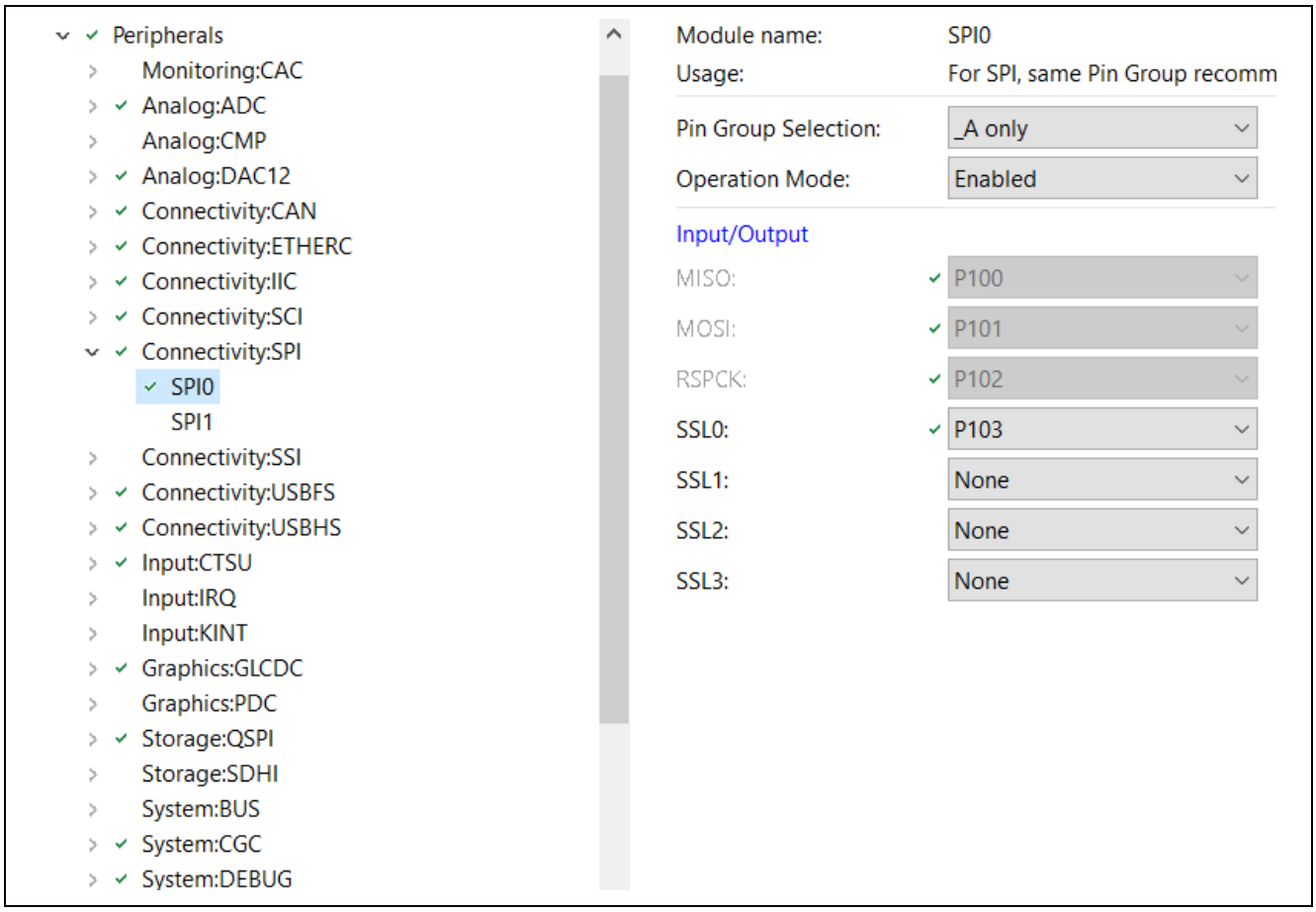

図 4　RSPI バス端子の設定

アプリケーションの制御下にある GPIO 端子を使用して、複数の LCD パネル信号を制御します。RSPI は Clock Synchronous (クロック同期) 動作に設定してあるので、SPI スレーブセレクト（slave select）信号は、アプリケーションから 1 つの GPIO 端子という形で制御する必要があります。LCD のスレーブセレクトは P611 です。また、P610 はリセット端子（reset pin）で、P115 はコマンド端子（command pin）です。これら 3 本の端子を GPIO 出力として有効にする必要があります。

[Pins] (端子) -> [P6] -> [P610] を選択します。**[Mode]** (モード) を、以下の図のように **[Output mode (Initial High)]** (出力モード (初期はハイ)) に設定します。P610 と P115 に対して、同じ操作を反復します。

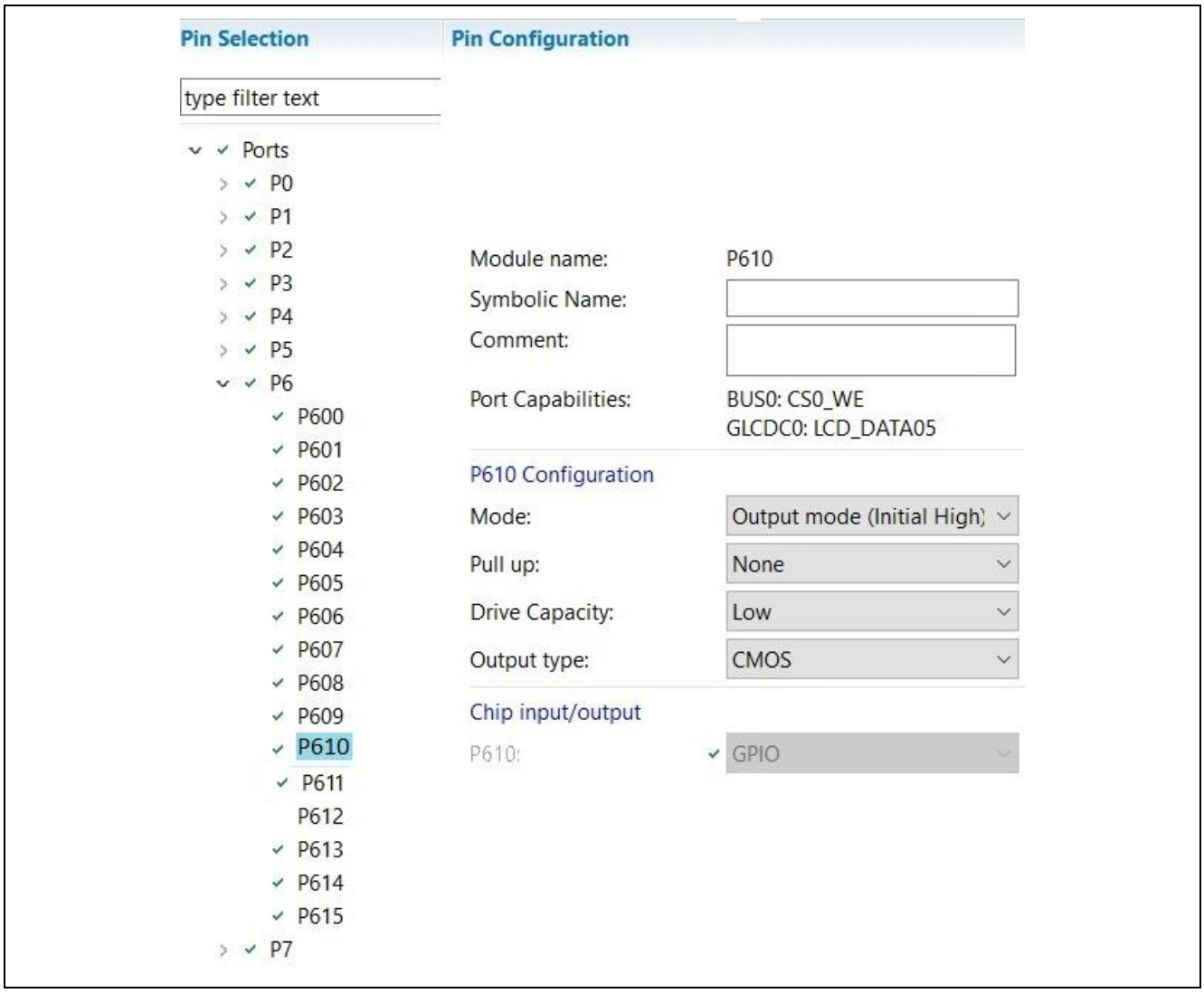

図 5　LCD パネルに対応する GPIO 端子の設定

8. ターゲットアプリケーションに対応する GUIX Synergy ポートフレームワークモジュールのカスタマイズ（Customizing the GUIX Synergy Port Framework Module for a Target Application）

このプロジェクトは、既存のスプラッシュ画面やメモリリソースに合わせてカスタマイズできます。例えば、画面の回転角度を変更し、縦長または横長のイメージに方向を合わせることができます。速度を最適化する目的、またはソフトウェアデコーダを使用するのにメモリが課題の場合は、このプロジェクトでハードウェア JPEG デコーダを使用することができます。これらの作業を行う方法の詳細については、「Synergy™ Software Package (SSP) v1.5.0 ユーザーズマニュアル モジュール概要編（参考資料）」の第4章「モジュールの概要」参照してください。

GUIX Studio を使用せずに、アプリケーションを作成することもできます。その場合、アプリケーションは単純に gx_<widget_type>_create() 関数を直接呼び出し、すべての子ウィジェットも手動で作成することになります。この場合は多くの工数が必要になるので、高度な特化型アプリケーションや小さな点まで制御するアプリケーションを除き、GUIX Studio を使用する方法を推奨します。このプロジェクト内の LCD_TEST は、手動でイメージを描画する例です。

GUIX システムを理解し、GUIX ソースコードのロジックをステップ実行するには、図12で示すように [Add GUIX Source] (GUIX ソースを追加] をクリックしてソースコードを追加します。GUIX SourceのWarningは無視でき、プロパティの**Show linkage warning**を設定することでdisableにできます。プロジェクトを生成し、再ビルドした後、アプリケーションのデバッグを実行します。コードのステップ実行で最善の結果を得るには、最適化レベル（optimization level）を -O0 (no optimization、最適化なし) に設定します。プロジェクトを右クリックし、**[Properties] -> [C/C++ Build (expand)] -> [Settings] -> [Optimization]** ([プロパティ] -> [C/C++ のビルド (展開)] -> [設定] -> [最適化]) を選択します。

注：DTCモジュールはオプションです。追加された場合は削除してください。

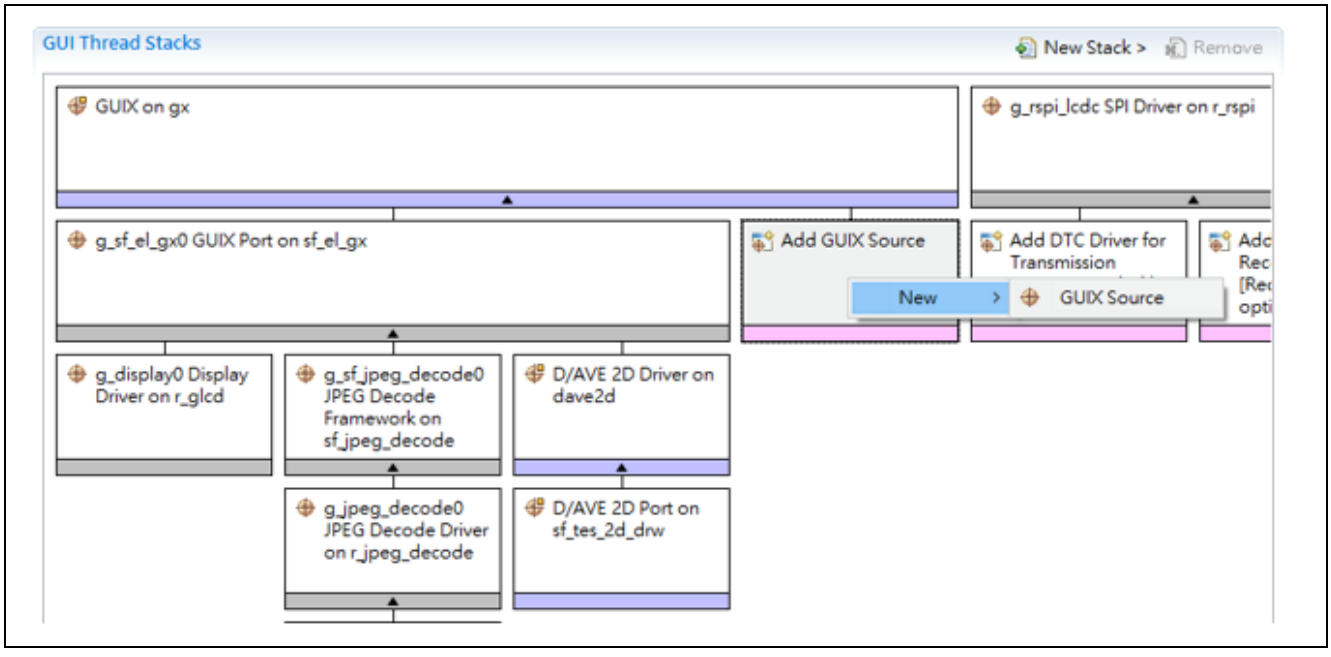

図 12　GUIX ソースコードの追加

9. GUIX Synergy ポートフレームワークモジュールのアプリケーションプロジェクトの実行（Running the GUIX Synergy Port Framework Module Application Project）

GUIX Synergy ポートフレームワークモジュールのアプリケーションプロジェクトの動作を確認するために、ターゲットキットで ISDE にこのプロジェクトをインポートし、コンパイルしてデバッグを実行することができます。

ISDE にこのプロジェクトをインポートし、コンパイルしてデバッグを実行するだけで、SPI フレームワークのアプリケーションプロジェクトを実行させ、対象キットでその動作を観察するためができます。

e² studio または IAR Embedded Workbench® for Renesas Synergy™ にプロジェクトをインポートし、アプリケーションをビルドして実行する手順については、『Synergy プロジェクトインポートガイド』（下記WEB）を参照してください。

英語版：

https://www.renesas.com/jp/ja/doc/products/renesas-synergy/apn/r11an0023eu0121-synergy-ssp-import-guide.pdf

日本語版（参考資料）：

https://www.renesas.com/jp/ja/doc/products/renesas-synergy/apn/r11an0023ju0121-synergy-ssp-import-guide.pdf

新しいプロジェクト内で GUIX Synergy ポートフレームワークモジュールアプリケーションを実装するには、ターゲットキットで定義、設定、ファイルの自動生成、コードの追加、コンパイル、デバッグを行う、以下の手順に従います。このガイドに示す手順に従うことでSSPでの開発プロセスをより実践的に習得するのに役立ちます。

注記：Synergy 開発プロセスの基本的な流れを経験したことのあるユーザにとって、以下の手順は十分詳細なものです。これらの手順をまだ理解していない場合、これらの手順を実行する説明について、『SSP ユーザーズマニュアル』の最初にあるいくつかの章を参照してください。

GUIX Synergy ポートフレームワークモジュールのアプリケーションプロジェクトを作成し、実行するために、以下の手順に従ってください。

1. GUIX_sf_el_gx_AP という名称で、S7G2-SK に対応する新しい Renesas Synergy プロジェクトを作成します。[BSP] のみを選択します。
2. **[Threads]** (スレッド) タブを選択します。
3. HAL/Common スレッドスタックに対して CGC が追加されたかどうか確認します。追加されていない場合は、追加します。
4. my_guix_thread というスレッドを追加し、[priority] (優先順位) を 10 に設定します。
5. GUIスレッドの下にセマフォ g_my_gui_semaphoreを追加します。
6. 以下の方法で、GUIX ドライバのインスタンスを追加します。**[Thread Stack]** (スレッドスタック) ペイン内の **(+)** アイコンをクリックし、**[X-ware] -> [GUIX] -> [GUIX]** でgxを選択します。この結果、GUIX ドライバ sf_el_gx、ディスプレイドライバ r_glcd、D/AVE 2D ドライバ、および JPEG デコーダフレームワークが追加されます。各インスタンスをデフォルトの名前のままにします。
7. LCD 画面にシリアルインタフェースを追加するために、**[Driver] -> [Connectivity] -> [SPI Driver]**でr_rspiを選択して、g_rspi_lcdc という名前に変更します。
8. **[Generate Project Content]** (プロジェクトコンテンツの生成) ボタンをクリックします。
9. 付属のプロジェクトファイル guix_driver_sf_el_gx_mg_ap.c と guix_driver_sf_el_gx_mg_ap.c からコードを追加するか、生成されたスレッドエントリファイル（thread entry file）ファイルに上書きする形でロジックをコピーします。スレッドに my_guix_thread という名前を付けた場合、Synergy が生成したスレッドエントリファイルは、my_guix_thread_entry.c という名前になります。7 章の「GUIX_TEST」ですでに説明したように、GUIX Studio が生成した 4 個のファイルを追加します。
10. このプロジェクトには、メインウィンドウである SplashScreenHandler に対応する、オプションのイベントハンドラがあります。GUIX Studio の仕様ファイルでそのようなコールバックを指定した場合、アプリケーションでそのコールバックを定義する必要があります。このコールバックに対応するために、my_guix_event_handler.c プロジェクトファイル内にあるイベントハンドラ関数 SplashScreenEventHandler を使用します (または、独自の関数を作成します)。

11. [Thread Stack] (スレッドスタック) コンポーネント (例えば、g_sf_el_gx0 ドライバインスタンスと r_lcd ディスプレイドライバ) のプロパティを、7 章の表で示した値に設定します。
12. 7 章の説明に従って、端子と周辺回路を設定します。
13. guix_driver_test_image_draw サービスを呼び出し、希望のモードとして LCD_TEST または GUIX_TEST を指定して、スレッドエントリ関数 my_guix_thread_entry を使用して LCD パネルに渡すイメージを処理します (開発中スレッドが my_guix_thread という名前の場合)。
14. printf のデバッグ出力を使用する場合、スレッドエントリ関数内、またはプロジェクト全体に対し、#define SEMI_HOSTING を記述してセミホスト機能を定義します。後者の場合、プロジェクトを右クリックし、**-> [Settings] -> [C/C++ Build (expand the list)] -> [Cross ARM C Compiler] -> [Preprocessor]** を選択します。**(+)** をクリックし、「SEMI_HOSTING」と入力します。IAR の場合、プロジェクトを右クリックし、**[Options] -> [C/C++ compiler] -> [Preprocessor]**を選択して、「SEMI_HOSTING」を追加します。
15. アプリケーションのデバッグを開始します。
16. 出力は、Renesas Debug Console (Renesas デバッグコンソール) に表示されます。エラーが発生しない場合、出力は以下のようになります。

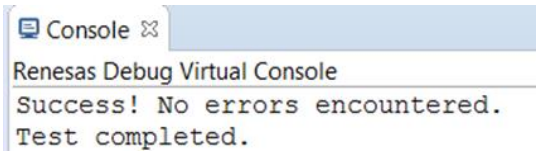

図 13　GUIX Synergy ポートフレームワークのアプリケーションプロジェクトのサンプル出力

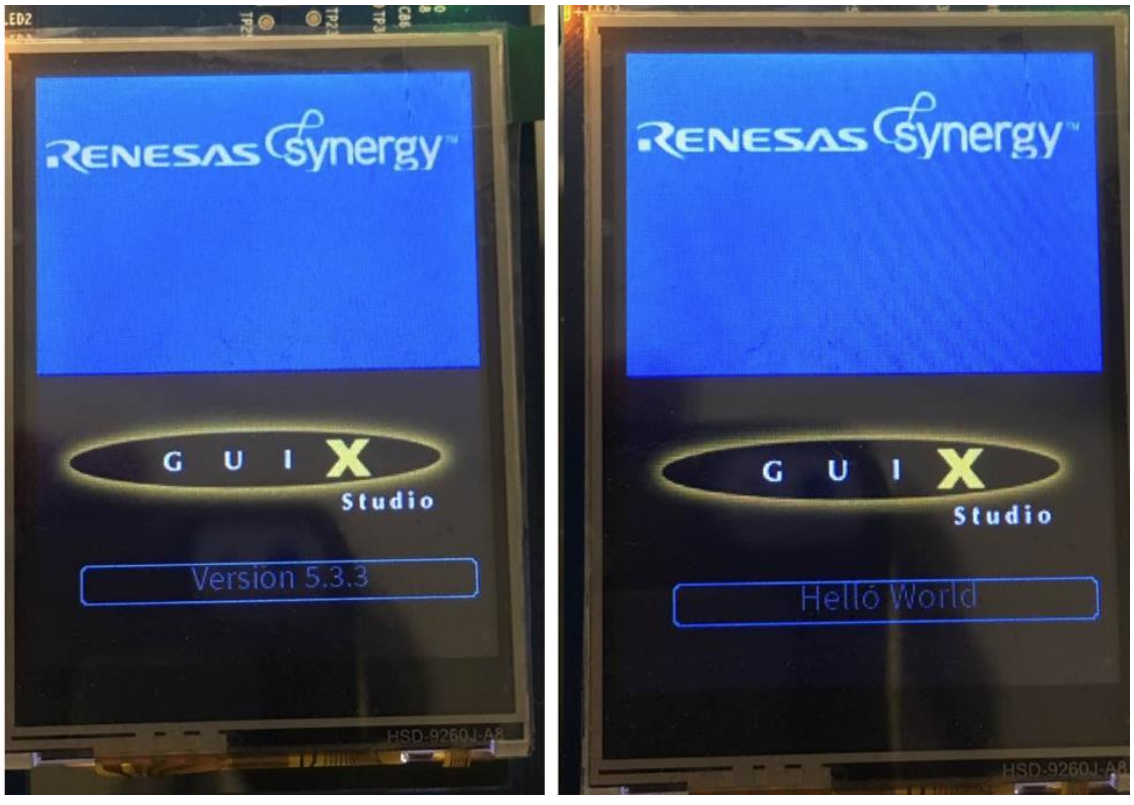

図 14　GUIX_TEST モードの GUIX SF_EL_GX Driver アプリケーションプロジェクトを実行したときの LCD 画面

最初は左側の画面がおよそ 1 秒間表示されます。次に、右に示すように、画面下端のテキストが「Hello World」(ハローワールド) に変化します。

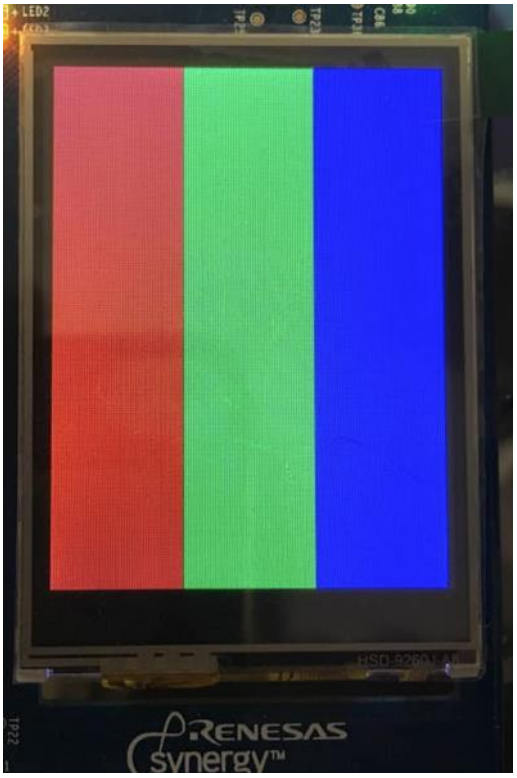

図 15　LCD_TEST モードの GUIX SF_EL_GX Driver アプリケーションプロジェクトを実行したときの LCD 画面

10. GUIX Synergy ポートフレームワークモジュールのまとめ（GUIX Synergy Port Framework Module Conclusion）

このモジュールガイドは、サンプルプロジェクトでモジュールの選択、追加、設定、使用を行うために必要な背景情報全般を説明しました。このプロジェクトは、e² Studio のプロジェクトペインで GUIX システムとハードウェアドライバを初期化し、描画コンポーネントのプロパティと端子割り当てを設定する手順をユーザ向けに紹介しました。このプロジェクトは、LCD_TEST モードでイメージを LCD 画面に直接描画する方法を示しました。これは、GUIX システムとハードウェアのシンプルなテストでもあります。このプロジェクトは、より複雑な LCD 画面を表現するために、GUIX Studio でリソースファイルと仕様ファイルを作成する方法も示しました。

このモジュールガイドは、サンプルプロジェクトでモジュールの選択、追加、設定、使用を行うために必要な背景となる情報全般を説明しました。従来の組み込みシステムでは、これらの手順を理解することに多くに時間を必要とし、また間違いが起こりやすい操作でした。Renesas Synergy プラットフォームにより、これら手順の所要時間が短くなり、設定項目の競合や、ローレベルドライバの誤った選択など、誤りが防止できるようになりました。アプリケーションプロジェクトで示したように、ハイレベル API を使用することで高いレベルの開発からスタートし、ローレベルドライバを作成するような従来の開発環境で必要とされる時間が不要になり、開発時間を短縮できます。

11. GUIX Synergy ポートフレームワークモジュールの次の手順（GUIX Synergy Port Framework Module Next Steps）

シンプルな GUIX モジュールのプロジェクトをマスターした後、より複雑なサンプルを確認できます。

GUIX Studio を使用して生成したこれらのリソースファイルと仕様ファイルは、かなりシンプルな画面です。それ以外に、ボタン、ドロップダウンリスト、子ウィンドウ、その他の画面オブジェクトを追加することもできます。より複雑なハンドラやタイマを追加して、アプリケーションの機能を強化することもできます。

GUIX キャンバスのサイズとメモリの配置先、および JPEG イメージのデコード方法を最適化して、性能を向上させることもできます。

プロジェクトにタッチスクリーンコンポーネントを追加し、タッチスクリーンイベントに対応するハンドラを記述することもできます。

12. GUIX Synergy ポートフレームワークモジュールの参考情報（GUIX Synergy Port Framework Module Reference Information）

『SSP ユーザーズマニュアル』: SSP ディストリビューションパッケージの一部として HTML 形式が入手できるほか、Renesas Synergy™ WEBサイトのSSPページ
https://www.renesas.com/jp/ja/products/synergy/software/ssp.htmlから pdf を入手することもできます。

最新版のsf_el_gx モジュールの参考資料やリソースへのリンクは、以下の Synergy WEBサイトから入手できます。

https://www.renesas.com/jp/ja/products/synergy.html

ホームページとサポート窓口

サポート:　https://synergygallery.renesas.com/support

テクニカルサポート:

- アメリカ:　https://www.renesas.com/en-us/support/contact.html
- ヨーロッパ:　https://www.renesas.com/en-eu/support/contact.html
- 日本:　https://www.renesas.com/ja-jp/support/contact.html

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.02	2019.06.03		・初版 ・英文版（R11AN0217EU0102、Rev.1.02、2019.May.06 ）の巻頭と第7章以降を翻訳

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。お客様の機器・システムの設計において、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には、お客様の責任において行ってください。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、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、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。
5. 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。
6. 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。
7. 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。
8. 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。
9. 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。
10. 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。
11. 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。
12. 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.4.0-1 2017.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/