

Renesas Synergy[™] プラットフォーム

R30AN0231JU0201 Rev.2.01 2016.11.18

ウェザーパネルアプリケーションのスタートガイド

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。

要旨

本アプリケーションノートでは、模擬ウェザーウェザーパネルアプリケーションを説明します。このウェザ ーウェザーパネルアプリケーションは、Renesas Synergy Software Package (SSP)を使用して、タッチスクリー ンの画像ヒューマンマシンインターフェース(HMI)のマルチスレッドアプリケーションがどのように開発 できるかをデモします。「簡単かつすぐに使える」を体験できるアプリケーションです。

本ウェザーウェザーパネルアプリケーションは、DK-S7G2、SK-S7G2 および PE-HMI1 を含む(図 1)さまざま な開発ボードの上で動作します。スクリーン解像度は各ボードで異なり、特長も若干違いがありますます。 例えば、DK-S7G2 および PE-HMI ボードは、LCD のバックライトを制御できますが、SK-S7G2 ボードに は、この機能は搭載されていません。本アプリケーションノートでは、各ボートでアプリケーションを動作 するために必要な変更点を示しながら PE-HMI1 ボートに焦点を当てていきます。



図 1 各開発ボード上のウェザーパネルアプリケーション

本アプリケーションは Synergy Software Package(SSP)を使用して開発されています。SSP は、 Express Logic 社の ThreadX リアルタイムオペレーティングシステム (RTOS)と連動した Synergy Arm Cortex M4/M0+ MCU における周辺機器へのドライバレベルのサポートを含めた統一かつ安定したフレームワークです。ThreadX



Renesas Synergy[™] プラットフォーム

に加えて、Express Logic 社の X-Ware スタック一式 (NetX, USBX, GUIX, and FileX)を通してフルスタックサ ポートに対応しています。このパワフルなツールセットにより複雑に組み込まれたアプリケーションを短時 間で開発できる包括的総合フレームワークの提供が可能になります。

本アプリケーションノートでは、ThreadXのようなRTOS上で、マルチスレッドアプリケーションのプログラムに関連した概念に精通しているユーザを想定しています。ThreadXに対する特定の知識があれば、コードをより簡単に理解することができますが、スレッド、メッセージキュー、セマフォおよびミューテックス等のRTOSの原理の経験があれば、本書の情報をより簡単に理解できます。ThreadXの特徴の詳細については、Synergy X-ware (ThreadX) ユーザーズマニュアルを参照してください。

ウェザーウェザーパネルアプリケーションは、Renesas e² studio 統合ソリューション開発環境(ISDE)を使用し て開発されています。IDE ベースの Eclipse は、ルネサスの Web サイトよりダウンロードできます。e² studio はマルチツールチェーンに対応していますが、本アプリケーションノートは、e²studio ISDE 環境と一緒に入 手できる無料の GCC コンパイルツールを使用してビルドされています。

類似のアプリケーションを開発するより、Renesas Synergy エコシステムは格段に早くアプリケーションを構 築できますが、複雑なマルチスレッドの HMI アプリケーションを構成するために必要な手順を理解するに は時間がかかります。本書では、下記の必要な手順を全て段階的に説明していきます。

- ボートの設定
- アプリケーション概要
- 画像スクリーンの使い方について
- GUIX Studio プロジェクトへの統合
- Synergy フレームワーク設定
- アプリケーション設計の重点
- メッセージ交換フレームワークを使用したスレッド間コミュニケーション
- 汎用 PWM タイマを使用した PWM バックライト制御信号
- プロジェクトのロードおよび作動

動作確認デバイス

Synergy PE-HMI 1 V2.0 ボート(Synergy S7G2 シリーズ MCU) Synergy DK-S7G2 V3.0 開発ボード(Synergy S7G2 シリーズ MCU) Synergy SK-S7G2 V2.0 開発ボード(Synergy S7G2 シリーズ MCU)



目次	
1. ボートの設定	4
2. アプリケーション概要	4
2.1 ウェザーパネルアプリケーションに使用されている Synergy S7G2 MCU の周辺回路	5
2.2 ヒューマンマシーンインタフェース(HMI)	6
2.3 ウェザーパネルスクリーン	7
2.3.1 スクリーン(大)デザイン	7
2.3.2 スクリーン(小)デザイン	8
3. GUIX Studio 概要	8
4. アプリケーションの分析	12
4.1 ソースコードレイアウト	12
4.2 スレッド概要	13
4.2.1 HMIスレッド	14
4.2.2 スレッドレイアウトおよび SSP	15
5. フレームワークコンフィグレーション	16
5.1 構成要素タプ	18
5.2 スレッドタブ	19
5.3 スレッドオブジェクト	22
5.4 モジュールコンフィグレーション	23
5.4.1 GLCD コンフィグレーション	23
5.4.2 TCON コンフィグレーション	24
5.4.3 フレームパッファ用の外部メモリ	27
5.4.4 e ² studio の便利な機能	29
6. アプリケーション設計の要点	31
6.1 スレッドおよび main	31
6.1.1 GUIX 初期化	33
6.1.2 イベントおよび GUIX メッセージ	33
6.2 LCD 制御	35
7. プロジェクトのインポートとビルド	37
8. 対象ボートへ実行プログラムのダウンロード	37
9. 参考資料	37



1. ボートの設定

PE-HMI1ボードは、本書に関連するファームウェアを動作する前に設定する、スイッチの設定がほとんどあ りません。このスイッチ設定の他、J-Link プログラミングインターフェースにアクセスするためのコネクタ がボードには含まれています。

供給されている J-Link LITE で、PE-HMI1の J12と PC (Synergy e²studio ソフトウェアをロード済み)の間を 接続します。J12 は図1で示されています。アプリケーションの適切な動作を確実にするために、表1の通 りに DIP スイッチを設定してください。

表1 PE-HMI1 のスイッチ設定

232	OFF
SLW	OFF
SPB	OFF
HALF	OFF
BOOT	OFF



図 2 ウェザーパネルアプリケーションの PE-HMI1 V2.0 ハードウェア詳細

2. アプリケーション概要

GUIX Studioを使用した複合 HMI スクリーン利用したアプリケーションのビルド方法をデモンストレーションすることがウェザーパネルアプリケーションの主な目的の1つです。ウェザーパネルアプリケーションの主な特長は以下のとおりです。

- GUIX Studio を使用した複合 HMI 設計
- ThreadX RTOS を使用したマルチスレッドアプリケーション

— キュー / ミューテックススレッドオブジェクト使用

- スレッド間コミュニケーション用の Synergy メッセージフレームワークの拡張使用
- スクリーンの各タイプ・サイズの GLCD 設定
 - 内外部メモリからのフレームバッファ動作
 - 外部メモリインターフェース



— ILI9341 画像コントローラ (SK-S7G2 ボード)の SPI 初期化

- タッチパネル、I2C タッチコントローラドライバ ft5x06
 - 外部 IRQ マッピング必須

ソフトウェア設計において、この問題を解決するには複数の方法があります。本書で紹介する解決法は、1 つのアプローチです。

2.1 ウェザーパネルアプリケーションに使用されている Synergy S7G2 MCU 周辺回路

ウェザーパネルアプリケーションは Synergy S7G2 MCU を使用しています。複雑な組み込みアプリケーションの開発には通常多くのステップのプロセスが必要となります。

- 最初のステップは、アプリケーションの要件を収集すること、そしてハードウェア構成要素に要件を関 連付けるシステムデザインから始まります。MCUを含み、アプリケーションをビルド / デバッグする ために必要なツールチェーンの条件を満たすことが必要となります。
- 次のステップは、対象 MCUのボード上で使用する周辺回路を決めることです。このステップでは、ボ ード上の周辺回路のレジスタマップを理解すること、またドライバコードの作成に多くの時間を当てる 必要があります。一方、Synergyのフレームワークでは、アプリケーション開発の効率化のため、ほと んど準備されています。
- 3. 次は、外部ハードウェアの制御方法です。例えば、DK-S7G2 および PE-HMI ボードは LCD スクリーン を搭載していますが、 ボート上の S7G2 MCU は画像 LCD コントローラ(GLCD)を介して直接制御するこ とが可能です。SK-S7G2 ボードは、初期化が必要になりますが、より小さく低コストの LDC をシリア ルインターフェースで、S7G2 MCU の GLCD が制御を行う前に制御しています。
- 4. 最後のステップは、初期要件を満たすために選択したハードウェアの上に、アプリケーションがいかに 構成されるかを、詳細に記述します。

ウェザーパネルアプリケーションの要件は、まず S7G2 MCUボード上にマッピングされます。下図にウ ェザーパネルアプリケーションで使用するハードウェアの周辺回路が全て示されています。本アプリケ ーションノートは、Synergy フレームワークを使用して各周辺回路がどう設計されているか、また各周 辺回路に対する注意事項の詳細についても説明します。



天気パネルアプリケーションのスタートガイド



図3 ウェザーパネルアプリケーションで使用される Synergy S7G2 周辺回路

2.2 ヒューマンマシーンインタフェース(HMI)

多くの HMI アプリケーションで困難な作業は GUI です。GUI は何を表示するかを決定するのではなく、いかに表示するかが重要です。何をいつ表示するかは、外部ロジックに頼ることになります。

要件をまとめ、トップレベルの設計が終了し、設計通りに実行できるハードウェアの特定が完了したら、グラフィカルユーザインタフェース(GUI)を迅速に作り上げます。システムの見栄えと使い勝手感を見せることはとても有益です。ここで、GUIX studioの出番になります。

SSPは Express logic が開発した GUIX に対応しています。ユーザは、スクリーン設計に直接 GUIX プリミティブコールを使用するか、または GUIX Studio を使用するかを選択できます。GUIX Studio は、単独のツールで、ポイントとクリックの環境を、組み込んだアプリケーションに必要なスクリーン全てを作成作成するために提供します。一旦設計が完了すれば、GUIX Studio は、ユーザがアプリケーションに統合する.c and .h



ファイルを出力します。ウェザーパネルアプリケーションのアプリケーションスクリーンは全て、GUIX studio を使用して構築されています。

2.3 ウェザーパネルスクリーン

スクリーンのデザインは通常、表示されるスクリーンのサイズに合わせて作成られます。これは、異なる LDC スクリーンサイズのボードにアプリケーションをポートする際、複数の画像デザインが必要になるた めです。この問題を解決するには2つのアプローチがあります。1つは、各スクリーンの解像度ごと個別の 静的ディスプレイを設計する作成方法です。GUIX studio を使用すれば時間をかけずに作成作成することが できます。本アプリケーションノートのアプローチです。2つ目は、動的にスクリーンを設計し、スクリー ンの解像度によって実行中にウィンドウ・ウェジェッドのサイズを決める方法です。GUIX は、このような ダイナミックスクリーンを作成することができる作成高度な API を有し、スクリーンを動的にビルドしま す。

ウェザーパネルアプリケーションには、大きいスクリーン(DK-S7G2ボードのような 4.3 インチと、PE-HMIボード上の7インチスクリーン)、そして小さなスクリーン(SK-S7G2上のボードのようにより小さい サイズ用)の2つのデザインがあります。



図 4 ウェザーパネルアプリケーションのスクリーンショット

2.3.1 スクリーン (大) 用デザイン

より大きいスクリーン(DK-S7G2 または PE-HMI ボード)で、ウェザーパネルアプリケーションは、図4の とおり、2つのメインスクリーンで構成され、以下の名前が付けられています。

スプラッシュページ 起動時に HMI 上に現れる最初のスクリーン メインページ ウェザーパネル設定の調整スクリーン(例:日付の設定、気温調整)

RENESAS

DK-S7G2 および PE-HMI ボートには 2 つの同じスクリーンが使われていますが、DK-S7G2 には、より小さ いスクリーン用にデザインを縮小する必要があるため、2 つの画像デザインを採用しています。GUIX Studio プロジェクトは、多種のリソースファイル(フォント、イメージ等)から作られていますが、多くの IDE の 場合のように、プロジェクト定義そのものは、実際には.gxp 拡張子の xml ファイルに保持されています。3 つのボードのデザインに対して個々の.gxp ファイルがあります。





図 5 SK-S7G2 ボートのスクリーンショット

図 5 は SK-S7G2 ボート用の小サイズのスクリーンのデザインを示します。このデザインでは 1 週間の表示 設定を 3 日に制限しています。

.-

3. GUIX Studio 概要

本章では、GUIX Studio を使用して GUIX スクリーンがデザインされ SSP アプリケーションに統合される概要を説明します。GUIX または GUIX Studio ドキュメントの代替になるものではありません。Synergy プラットフォーム用に画像インターフェースを設計する場合、GUIX および GUIX studio 用のドキュメントを全て読むことを推奨します。

GUIX Studio は、組み込んだアプリケーションに必要なスクリーンを全て短時間で作成するため、ポイント・アンド・クリックの環境を提供します。スクリーン解像度、色深度およびその他の各種パラメータを設定でき、デスクトップ PC の GUIX Studio スタイルをそのままスクリーンで見ることができます。

GUIX では、いくつかのフォント、ボタン制御のような基本的画像処理が標準装備になっています。凝った ディスプレイにする場合には、スクリーン作成の段階で、ユーザが用意した画像やフォントファイル等を GUIX に提供することもできます。GUIX Studio は、ストリングテーブルを使用することで、多言語ディス プレイも使えます。



図 6 GUIX Studio にデザインされたウェザーパネルページのスクリーンショット

GUIX Studioの IDE は、分かりやすい構成になっています。ターゲットビューとしてのセンタスクリーンには、デザインされたスクリーンが含まれています。

左上には[Project View]があります。このペインはプロジェクトに含まれるウィジェッドを示しています。プロジェクトにアイテムを追加する順序が最終的なスクリーンに描かれるアイテムの順序を決定するので、準備が必要となります。多くの画像デザイン環境で見られるように、スクリーンは階層化されており、通常、メインウィンドウは親ウィンドウで、親ウィンドウ内に含まれている画像オブジェクト全てが子ウィンドウになります。左下にある[Project View]は選択されているオブジェクトに関連したプロパティを表示します。オブジェクトは[Project View]またはターゲットビューから選択可能です。

GUIX Studioのスクリーンの一番右横には、プルダウンメニューでスクリーン作成に必要なリソース(色、フォント、イメージ(ピクセルマップ)およびストリング)にアクセスできます。GUIX はストリングテーブルを使用することで多言語デザインに対応しています。

画像デザインをインタラクティブにするには、適切な機能性を実行するイベント処理コードにイベント (例:スクリーンタッチ)を関連づけます。スクリーンをデザインする場合に、コールバック機能をウィジ ェッドに関連づけます。コールバック機能は、GUIイベントに応答するために、アプリケーションで必要と なるフックを提供します。

GUIX studio は描写およびイベントコールバックの両方を提供します。イベント機能は特定のイベント (例:タッチイベント)に応答します。描写機能は、カスタマイズした図面を追加できます。ウェザーパネ ルアプリケーションはイベント機能コールバックのみをトップレベルウィンドウで定義します。図7に示す 通り、コールバック機能名は[Properties View]の[Event function]のフィールドに入力されます。

ウェザーパネル GUIX デザインは下記の2つのイベント機能を定義しています。

mainpage_event splashscreen_event

Widget Type	window			50% Hui	marcy
Widget Name	main_screen	-			
Widget Id	177.040.7.407.6694.0.0799				
User Data					
Left	1			in an	
Тор	0			CLIN	MON
Width	800			SUN	MON
Height	480				
Border	No Border	•		Const .	2. Jo
Transparent				62 / 44	62 / 44
Draw Selected					
Enabled			1	1	
Accepts Focus					
Runtime Allocate					
Normal fill	WINDOW_FILL	•	1		
Selected fill	SELECTED_FILL	•			
Template					
Draw Function					
Event Function	MainScreenEventHandler	>			
Wallpaper	BG_SUNNY	•			
Tile Wallpaper			31		



Project Edit Insert C	onfigure Help	11
Project View weather	Project/Displays Languages Themes	<u>L</u> ±

図8 GUIX Studio プロジェクトコンフィグレーションビュー

図9は[Cofigure Project] ダイアログボックスです。基本的なディスプレイの設定、ビルドプロセスから作成 されるファイルを GUIX が探すのに必要なパス情報などのプロジェクトの特定情報を、このダイアログボッ クスから設定します。

プロジェクトをビルドする場合、GUIX Studio は c and h ファイルを作成しますが、このファイルには Synergy アプリケーションに組み込んだ LDC上の GUIX Studio で作成したスクリーンを実行するために必要 な情報が全て含まれています作成。[Directories]では、ソースおよびヘッダーファイル用のデフォルトの出力 ディレクトリを設定します。また、リソースファイル(例:イメージ)が全て保存されるディレクトリの場 所を設定できます。

Source Files	\\src\s7g2_pe	_hmi1\			browse	e
Header Files	\\src\s7g2_pe	_hmi1\			browse	e]
Resource Files	\src\s7g2_pe	_hmi1\			browse	e)
Target CPU R	nesas Synergy	✓ Advanced :	Settings			
Toolchain G	U					
		0				
Additional Heads Number of Displ Display Configu	ays 1	GUIX Lit	rary Vers	ion 5,	3 🍝 , 「	1
Additional Heads Number of Displ Display Configu Display Num x resoluti	ays 1	GUIX Lit Name <u>main_display</u> y resolution	rary Vers 480 p	ion 5,	3 📥 . 「	1
Additional Heads Number of Displ Display Configu Display Num x resoluti	rs ays 1 💼 ration per 1 📮 1 n 800 pixels	GUDX Lib Name <mark>main_display</mark> y resolution]grayscale	rary Vers 480 p	ion 5 , ixels	3 🧙 . 「 nat	1
Additional Heads Number of Displ Display Configu Display Num x resoluti	ays 1 -	GUIX Lib Name <mark>main_display</mark> y resolution]grayscale]invert polarity	rary Vers 480 p	ion 5, iixels 5:5:5 form	3 🔔 , 「 nat	1
Additional Heads Number of Displ Display Configu Display Num x resoluti 1 bpp 2 bpp 4 bpp 8 bpp	ays 1	GUIX Lit Name main_displa y resolution]grayscale]invert polarity]reverse byte order	rary Vers 480 p	ion 5, ixels 5:5:5 forr 4:4:4:4 fc 3:3:2 forr	3 💽 , nat rmat	1
Additional Heads Number of Displ Display Configu Display Num x resoluti 1 bpp 2 bpp 4 bpp 8 bpp 8 bpp	rs ays 1 mation per 1 0 pixels	GUIX Lit Name main_displa y resolution]grayscale] invert polarity]reverse byte order]packed format	rary Vers 480 p	ion 5, ixels 5:5:5 forr 4:4:4:4 fc 3:3:2 forr	3 🔔 , 「 nat rmat	1

図 9 GUIX プロジェクト設定ウィンドウ

ソース、ヘッダーおよびリソースファイルは、プロジェクトに関連した場所に保存するようにしてください。このことで、1つの場所から他の場所、または PC から他の PC にプロジェクトを移動するのが容易になります。ウェザーパネルアプリケーションの場合、src/s7g2_pe_hmi1 ディレクトリの下に全てのディレクトリが見えます。

1	GUIX Studio 5.3.0.1 - weather.gxp	
	Project Edit Insert Configure Help	
	New Project	
	Open Project	
	Save Project	
	Save Project As	
	Close Project	
	Recent Projects	
	Generate All Output Files	
	Generate Resource Files	
	Generate Specification Files	
	Exit	

図 10 GUIX プロジェクト:ソースファイルの作成

GUIX Studio のデザインを終了した場合、全出力ファイルの作成作成を GUIX Studio に指示します。 図 10 に 示す通り、[Project]メニュー [Generate All Output files]を選択します。



Renesas Synergy[™] プラットフォーム

基本的に、ウェザーパネルアプリケーションは、図 11 のとおり、オリジナルリソースファイルおよび weather.gxp ファイルを保持する guix_studio ディレクトリを有します。GUIX Studio が既にインストール済も 場合、weather.gxp ファイルをクリックして GUIX Studio を起動させます。



図 11 ウェザーパネルアプリケーションにおける GUIX プロジェクトのファイルビュー

前述したとおり、GUIX studioの出力はプロジェクトに統合される必要のある.c and .h ソースファイルです。 PE-HMI1 ウェザーパネルアプリケーションの GUIX プロジェクトには、各トップレベルのスクリーンに対し て、2つのイベントハンドラ機能が含まれています。以下のとおり、GUIX は weather_speci fi cati ons. h ファイル内に、コールバック関数のプロトタイプを自動的に作成します。

/* Declare event process functions, draw functions, and callback functions */
UINT MainScreenEventHandler (GX_WINDOW *widget, GX_EVENT *event_ptr);
UINT SplashScreenEventHandler (GX_WINDOW *widget, GX_EVENT *event_ptr);

GUIX Studio は、イベントハンドラ用にプロトタイプを定義していますが、各ハンドラに対して実際のコードを含むファイルを作成するかはユーザ次第です。図 12 のとおり、次章では、ウェザーパネルアプリケーションの実際のソースコードレイアウトを説明します。イベントハンドラコードは、 hmi_event_handl er.c ファイル内に保持されます。

4. アプリケーションの分析

HMIでは、HMIアプリケーションを理解することが大部分を占めます。Synergy アプリケーションを開発するには、e²studioでプロジェクトが物理的にどう構成されているのか、スレッドおよびそのリソースがプロジェクトにどう追加されるのか、スレッドのコミュニケーションの仕方やステートマシンの設計、ステートデータが連携するスレッド間でどう共有されているなどの理解が必要です。

4.1 ソースコードレイアウト

実際のアプリケーションコードに着手する前に、Synergy プロジェクトのソースコードレイアウト全般を最初に理解しておくことが大切です。Synergy アプリケーションは通常2つの異なるコード(ユーザ作成または自動作成)から構成されています。自動作成コードは、さらにSynergy フレームワークまたはGUIX Studio による自動作成かの、2つに分類できます。



図 12 ウェザーパネルプロジェクトのソースファイルレイアウト

図 12 は、PE-HMI1ボード用のソースコードレイアウトを示しています。フレームワーク自動作成コードは 赤色、GUIX 自動作成コードは黄色、ユーザ作成コードは緑色でそれぞれハイライトされています。GUIX Studio プロジェクトファイル、weather.gxp、HMIのイベントハンドラを含む hmi_event_handl er.c ファイ ルを除いては、ユーザ作成コードの多くが src ディレクトリに保持されています。

4.2 スレッド概要

概要で述べた通り、ウェザーパネルアプリケーションは ThreadX で動作するマルチスレッドアプリケーショ ンです。Synergy アプリケーションには、プログラマにより作成されるものとフレームワークによって作成 されるスレッドがあります。プログラマが作成したスレッドは明らかですが、フレームワークで作成された ものは必ずしも明らかではありません。SSP ユーザーマニュアルで説明していますが、Synergy アプリケー ションに追加する 2 つのモジュールの種類(ドライバモジュールとフレームワークモジュール)がありま す。ドライバモジュールは、RTOS を認識したモジュールですが、通常はは、RTOS オブジェクトを使用し ません。フレームワークレイヤモジュールは、セマフォ、ミューテックスのような RTOS オブジェクトを使 用し、必要に応じてスレッドも作成します。

ウェザーパネルアプリケーションはユーザ作成スレッドである HMI スレッドを使用します。スレッドは、 標準 ThreadX メッセージキューの最上階層の Snergy メッセージング交換フレームワークを介して通信しま す。HMI スレッドは、タッチメッセージ、GUIX イベントを処理します。次章のフレームワークコンフィグ レーションで、ユーザスレッドをアプリケーションに追加する方法を説明します。図 13 は、スレッドの高 レベルデザイン及びウェザーパネルアプリケーション上で動作するメッセージ交換を示します。HMI スレッ ド以外に GUIX 関連のスレッドやタッチコントローラもあります。



天気パネルアプリケーションのスタートガイド



図 13 ウェザーパネルアプリケーションメッセージシーケンスフロー

ソフトウェア構成要素に加え、多種のハードウェア構成要素もまた Synergy フレームワークによって提供されるハードウェアドライブを通してアクセスされます。クロック作成回路、タッチスクリーンコントローラ(I2C)、ARM コアプロセッサの外部割り込みユニットがこれらに含まれます。

4.2.1 HMI スレッド

HMI スレッドはウェザーパネルアプリケーションが使う GUIX を含む多くのサービスを初期化します。SK-S7G2 ボード上で、HMI スレッドは LCD スクリーンを初期化する必要があり、適切な RGB モードにスクリ ーンを設定します。S7G2 プロセッサの GLCD 周辺回路により制御が可能です。

この初期化が完了したら、HMIスレッドはタッチメッセージ、GUIXイベントを処理します。これらの入力のいずれかによりシステムの状態に変化を生じた場合、MHIスレッドは適切な更新メッセージをGUIXスレッドに送り、その結果、画像HMIを変化させます。図14のフローチャートは、HMIスレッドの高レベルのデザインおよびメッセージフローを示します。



図 14 HMI スレッドフロー図



Renesas Synergy[™] プラットフォーム

4.2.2 スレッドレイアウトおよび SSP

Synergy を初めてお使いのユーザは、多くの Synergy フレームワークで定義されたモジュール、アプリケーションにそのモジュールをどう追加していくのか、SSP スタックを形成するために、これらのモジュールがいかに階層化されているか、ということを習得することになります。

SSP ユーザーズマニュアルの第2章で記述しているとお通り、モジュールはSSPのコアビルディングブロックです。モジュールは上位に機能性を提供し、下位に機能性を要求するという基本概念を有します。SSP は、前もって定義された2つのレイヤ(ドライバレイヤおよびフレームワークレイヤ)と一緒に提供されます。ドライバレイヤモジュールは、RTOSを認知しているが、RTOSオブジェクトを使用せず、RTOS APIを呼び出さない周辺ドライバです。これは、ドライバレイヤモジュールがアプリケーションの中で、RTOSの使用/未使用に関わらず使うことができることを意味しています。フレームワークレイヤモジュールは、セマフォ等のRTOSオブジェクトを自由に使用することができ、必要であれば、RTOS APIを呼んだり、またはスレッドを作成もできます。

注:事前に SSP の命名慣例を理解しておくと Synergy アプリケーションを理解しやすくなります。ドライバレイヤモジュールは、必ず[r_]、フレームワークモジュールは、[sf_]の接頭語で必ず始まります。

 一番シンプルな SSP アプリケーションは、ユーザアプリケーション上に乗っている1つのモジュールで構成 されています。



図 15 SSP アプリケーション&モジュールレイヤ

例えば、SSPでは、ドライバの全てのインスタンスには名前があり、ドライバインスタンスにg_gptが付けられています。このケースでは、r_gptドライバとなります。r_gptには[r_]接頭語があるため、ドライバレベルのモジュールであることが最初に認識できます。フレームワークコンフィグレーションに関する章では、名前がドライバの特定インスタンスにどのように位置しているかを説明しています。



図 16 HMI スレッドモジュール表示.

HMIスレッドは多くのモジュールに依存しているのがわかります。中には重なってレイヤとなり SSPスタックを形成モジュールもあります。図 16 では、フレームワークモジュールは濃い青色で、ドライバモジュールは黄緑で表示しています。



多くの開発ボート上にあるタッチコントローラは、タッチ発生時に IRQ が作成されタッチの座標は I2C バス スで通信されます。図の中心を見ると、HMI スレッドでは sf_touch_panel_i2c モジュールを使用しているの がわかります。割り込みプロセスでは、このモジュールは sf_external_irq モジュールを必要としますが、 sf_external_irq モジュールは r_icu モジュールを必要とします。I2C 通信では、sf_touch_panel_i2c モジュール は r_iic モジュールを必要とします。



図 17 GUIX スレッドモジュール表示

図 17 は、SSP下でのアプリケーションアーキテクチャを完全に理解するために、内部を強調する変わった ケースを示しています。また、上記の HMI スレッド図の、小さな欠点も表しています。図 17 では GUIX ス レッドを明らかに示していますが、実際にはアプリケーションの中で GUIX スレッドを作成することはあり ません。アプリケーションにモジュールを追加した場合に、sf_el_gx モジュールが自動的にこのスレッドを 作成するためです。GUIX スレッドボックスの主な目的は、アプリケーションに追加するモジュールのプレ イスフォルダを用意することです。

上記の HMI スレッド図の欠点について、次章で説明していますが、sf_touch_panel_i2c モジュールが HMI スレッドの下に追加されていることに間違いはありません事実。

GUIX スレッドは、r_glcdドライバモジュールを含め数個のモジュールを使用します。S7G2 プロセッサには グラフィックス LCD(GLCD)コントローラが含まれます。このドライバモジュールは周辺回路を制御します が、モジュールの中では、理解するには複雑なモジュールです。このモジュールによって、スクリーン解像 度、フレームバッファがある所(例:内部メモリに対して外部メモリ)、ビデオ同期信号の配列等含む多く のプロパティを定義することができます。チームで画像ディスプレイ付きのの組み込むシステムを設計する 場合に、本モジュールを完全に理解しておくことが必要になります。

5. フレームワークコンフィグレーション

Synergy アプリケーションを作成する際に最初にしておくべきことの1つに、フレームワークの設定があり ます。フレームワークを適切に設定するには、実行するソフトウェアの設計と特定のソフトウェアを作動さ せるハードウェアの両方の詳しい知識をもっていることが求められます。ハードウェアに関しては、ハード ウェア上で使用する周辺回路の種類、配置する端子、プロセッサに対しの内部外部であるかどうか等を含み ます。ソフトウェアに関しては、スレッドをいくつ使用するか、どのスレッドがどのハードウェア構成要素 にアクセスする必要があるか、その他各スレッドが必要とする追加のソフトウェアオブジェクト(例:セマ フォ、キュー等)を確定する必要があります。上記が揃った時、特定のアプリケーションに必要なフレーム ワークを設定する準備が整います



ウェザーパネルアプリケーションでは、フレームワークコンフィグレーションは configuration.xmlのファイ ルの中に保管されています。このファイルをダブルクリックしてプロジェクトの Synergy コンフィグレーシ ョンタプを立ち上げます。e²studio が xml ファイルを処理するまでに数秒かかることがあります。

🔺 🚰 WeatherPanel [Debug]
🗈 🖑 Binaries
🖻 🗊 Includes
Þ 🐸 src
🕨 😂 synergy
🖻 🗁 Debug
🕨 🗁 board_config
🖻 🗁 guix_studio
🗈 🗁 script
🗈 🗁 synergy_cfg
🐵 configuration.xml
S7G2-PE-HMI1_WeatherPanel.pincfg
📄 WeatherPanel Debug.jlink
📄 WeatherPanel Debug.launch

図 18 プロジェクトペイン上の[configuration.xml]

ーからプロジェクトをビルドする場合、このコンフィグレーションタブより Synergy フレームワークの初期 設定をします。図 19のとおり、Synergy コンフィグレーション[WeatherPanel]ペインにはサマリ画面が含ま れており、設定する項目をまとめた[Summary]画面と現在このプロジェクトに選択されているスクロールウ ィンドウのソフトウェア構成要素を全てリストしています。スクロールウィンドウの下には、アプリケーシ ョンに特に必要となるフレームワークの調整ができるタブがあります。

本書の目的として、ウェザーパネルアプリケーションに関連するフレームワークコンフィグレーションの詳 細の一部をハイライトしています。Synergyフレームワークの設計に関する追加情報については、Synergyフ レームワークのドキュメントおよびアプリケーションノートを参照してください。

プロジェクトを適切に設定できたら、サマリースクリーンの上にある[Generatroject Content]の緑の矢印ボタンをクリックして、定義した構成要素を実装するために必要な自動作成ファイルを作成してくたさい。



🔋 [WeatherPanel] Synergy Configuration 🔀	
Summary	0
	Generate Project Conter
This editor allows you to modify the Synergy project settings stored in the configuration file (configuration.xml).	
BSP	
+ Allows board and device selection	
+ The board type is optional	
+ Board properties can be modified in the Properties view	
Clocks	
+ Allows configuration of the clock generation circuit	
Pins	
+ Allows editing of the projects pin configuration and set up	
Threads	
+ Allows configuring of threads within a Synergy project	
+ Synergy modules and objects can be added to individual threads	
+ Properties of each thread, module and object can be modified in the Properties view	
Messaging	
+ Allows configuration of the messaging framework	
ICU	
+ Allows configuration of interrupts	
Components	
+ Synergy components can be selected on this page	
+ Only those components suited to your selected board and device will be displayed	
BORIC 5762 PE-HMIL	<u> </u>
DEVICE INTERCENTED	
221.4212011 714	
Selected software components:	
* sf_el_gx V1.1.0 [Pack V1.1.0] [Renesas##Framework Services##all##sf_el_gx####11.0]	
* sf_external_irq V1.1.0 [Pack V1.1.0] [Renesas##Framework Services##all##sf_external_irq####1.1.0]	
* sf ipen. decode Y1.1.0.[Pack Y1.1.0]. [Renesas##Framework Services##all##sf ipen. decode####1.1.0]	
Jummary BSP Clocks Pins Threads Messaging ICU Components	

図 19 ウェザーパネルアプリケーションのスクリーンショット

5.1 コンポーネントタブ

[Components]タブは一番右に表示されていますが、設計しなくてはならない中の1つがこのタブです。 [Threads]タブで特定のスレッドにハードウェアリソースを配置するなど、後続の動作で最初に選択された構成要素が使用可能になります。選択された構成要素のみでコンパイルしてアプリケーション全体のサイズを 縮小できることは、Synergyフレームワークの利点の一つです。図20のとおり、構成要素は、7分類されま す。

Components
Component
> 🛷 BSP
> 🛷 Documentation
> 🔗 Express Logic
Framework Services
> 🔗 HAL Drivers
> 🔗 Projects
> 🛷 TES

図 20 コンポーネントタブ分類

分類名の左側にある矢印をクリックすると分類を拡張できます。

表2は、ウェザーパネルアプリケーションで使用している構成要素のリストです。[Components]タブの特長で、構成要素の機能および各構成要素の依存性の詳細を記述しています。例えば、sf_message構成要素は ThreadXを必要とします。構成要素を選択する時に、この依存性のリストにより、適切な依存する構成要素 を選択しなかった場合に生じるコンパイルタイムエラ を除外することができます。

分類	構成要素	バージ ョン	改訂内容
BSP	s7g2 pe hmi1	1.1.0	S7G2 PE HMI1ボードサポートパッケージ(BSP)
Express Logic	gx	1.1.0	Express Logic GUIX:Provides=[GUIX], Requires=[ThreadX]
	tx	1.1.0	Express Logic ThreadX:Provides=[ThreadX]
フレームワー クサービス	sf_el_gx	1.1.0	SF_EL_GX GUIX Adaption Framework:Provides=[SSP GUIX Adaptation Framework], Requires=[ThreadX, GUIX]
	sf_external_irq	1.1.0	Framework External IRQ:Provides=[Framework External IRQ], Requires=[External IRQ, ThreadX]
	sf_jpeg_decode	1.1.0	Framework JPEG Decode:Provides=[SF JPEG Decode] , Requires=[ThreadX ,JPEG Decode]
	sf_message	1.1.0	メッセージングフレームワーク: Provides=[Message] , Requires=[ThreadX]
	sf_tes_2d_drw	1.1.0	TES Dave/2d(DRW)
			Framework:Provides=[SF_TES_2D_DRW],
			Requires=[ThreadX ,TES Dave/2d]
	sf_touch_panel_i2c	1.1.0	I2C を使用したフレームワークタッチパネル:
	-		Provides=[Framework Touch Panel] ,
			Requires=[ThreadX ,Message ,I2C , Framework External IRQ]
HAL Drivers	r_cgc	1.1.0	クロック発生回路 Provides=[CGC]
	r_elc	1.1.0	イベントリンクコントローラ Provides=[ELC]
	r_glcd	1.1.0	グラフィック LCDCProvides=[Display]
	r_gpt	1.1.0	汎用 PWM タイマ Provides=[Timer ,GPT]
	r_icu	1.1.0	外部 IRQ: Provides=[External IRQ]
	r_ioport	1.1.0	I/O Port:Provides=[IO Port]
	r_jpeg_decode	1.1.0	JPEG デコード: Provides=[Key Matrix]
TES	dave2d	1.1.0	TES Dave/2d:Provides=[Dave/2d]

表 2 ウェザーパネルアプリケーションで使用する構成要素

5.2 スレッドタブ

スレッドタブはスレッドを追加・閲覧できるタブです。フレームワークは自動的にアプリケーションを作成 します。新規のスレッドを定義するには、 「ボタンをクリックして、新しいスレッドに名前をつけます。 新規スレッドを追加した後、スレッドが使うスレッドオブジェクトと一緒に使用するモジュールを定義しま す。

例えば、[Threads]をクリックして[HMI Threads]を1回クリックすると、図21のスクリーンショットのよう な画面が見えます。これは、HMI スレッドが複数のモジュールを必要とすること、また PE-HMI1のオンタ ッチセンサを読むのに i2c ドライバが使われることを示しています。



図 21 HMI スレッド プロパティ



[hreads						Generate Project Content		
Threads 🕢 🐑 💼	HMI Thread Stacks							
PHAL/Common Q_gc CCC Driver on r_gg g_clc ELC Driver on r_glc g_uptrut/20 Port Driver on r_joport	g_sf_touch_panel_i2c	_st_touch_panel_i2c Touch Panel Framework on st_touch_panel_i2c						
●HMI Thread g.sf.message Messaging Framework on sf.message g.sf.touch.panel_i2c Touch Panel Framework on sf_touch_panel_i2c GUIX on gx					24			
	g_J2c J2C Driver on r_tile	g_sf_external_irq External IRQ Framawork on sf_external_irq	g_sf_message Messaging Framework on sf_message	(j_st_el_gx0 GUIX Port on st_el_gx				
		g_external_irq External IRQ Driver on r_icu		D/AVE 2D Driver on dave2d	g_display Display Driver on r_glod	g_sf_peg_decode JPEG Decode Framework on sf_jpeg_decode		
HMI Thread Objects 💿 🛍				D/AVE 2D Port on sf_tes_2d_drw		g_jpeg_decode JPEG Decode Driver on r_jpeg		
					1			
				-10				

図 22 ウェザーパネルアプリケーション用 HMI スレッドモジュール

適切な構成要素を最初に選択できていなかった場合でも、フレームワークは自動的に構成要素を選択しま す。フレームワークがモジュール追加でエラーを探知した場合、モジュール名の上にマウスを重ねて、エラ ーを調べることができます。

				Generate Project	Content	1	2
HMI Thread Stac	ks				A A	A NC	P302
nel Framework on sf_touch_panel_i2c		GUIX on gx		Analog + Connectivity + Crypto + Graphics +	Driver Frame	work	, , , , , , , , , , , , , , , , , , ,
xternal_irq ıal IRQ work on ernal_irq	g_sf_message Messaging Framework on sf_message	g_sf_el_gx0 GUIX Port on sf_el_gx		Input Monitoring Power Storage System		E P610 F P614 G P813	P611 P612 PA15
al IRQ Driver on		D/AVE 2D D dave2d D/AVE 2D D Timer Driver on r_rtc Timer Driver on r_g Timer Driver on r_g	er on r_gpt_input_capture	Timers • Transfer •		J PA07	VSS PA06 P604 P601
		D/AVE 2D Port on sf_tes_2d_drw	g_jpeg_decode JPE Decode Driver on r_jpeg	G		M V55 N P102 P P101	VCC P103 P800
			III			R P100	P801

図 23 HMI スレッドへの PWM タイマの追加

Renesas Synergy[™] プラットフォーム

5.3 スレッドオブジェクト

ThreadX はイベントフラグ、ミューテックス、キュー、セマフォのような様々なオブジェクトに対応しています。[Threads]ウィンドウ内の[HMI Thread]をクリックすると、1つのキューオブジェクト[g_hmi_queue]がこのスレッド用に割り当てられたことがわかります。

左側のスレッドウィンドウからスレッドを選択することで、スレッドオブジェクトを追加することができま す。次に、スレッドオブジェクトウィンドウで新規

ボタンをクリックします。図 24 のとおり、スレッド オブジェクトウィンドウで新規

ボタンをクリックした後、プルダウンメニューが表示され、ThreadX が 対応する標準スレッドオブジェクトを追加することが可能になります。

HMI Thread Objects		Eve	ernal_irq nt Flags
g_hmi_queue Queue	•	Mut	ex.
	•	Que	ue
	•	Sen	naphore
			4
		1	•

図 24 スレッドオブジェクトの追加および ThreadX が対応するオブジェクト

[Properties]タブを通常有効にするスレッド、スレッドモジュール、またはスレッドオブジェクトの追加また はレビューでは、項目に関連するプロパティを調べたり変更したりできます。現在、[Properties]タブが表示 されていない場合、**以下(Window Show View Other... General**)の手順で[Properties]を選択します。図 24の例では、メッセージが1ワードかつキューサイズが20バイトのシステムキューを有していることを示 しています。これらの値を変更するには、[Properties View]から更新し、[Generate Project Content]ボタンを押 して、プロジェクトコードを新しい値に更新してください。



Synergy_cfg	HMI Thread Objects	a
 configuration.xml S7G2-PE-HMI1_WeatherPanel. WeatherPanel Debug.jlink WeatherPanel Debug.launch 	<mark>● g_hmi_queue Queue</mark>	h
۰ III • •	Summary BSP Clocks Pins Threads Messaging ICU Com	ponents
ৰ ।।। ■ Properties হয় 💽 Problems	Summary BSP Clocks Pins Threads Messaging ICU Com	ponents Pin Conf
ৰ আৰু ১০০০ বিজ্ঞানীয় বিজ্ঞানিক বিজ্ঞানিল বিজ্ঞানিক বিজ্ঞানিল বিজ্ঞানিক বিজ্ঞানিক বিজ্ঞানিক বিজ্ঞানিক বিজ্ঞানিল বিজ্ঞানিক বিজ্ঞানিল বিজ্ঞানিক বিজ্ঞানিক বিজ্ঞানিক বিজ্ঞানিক বিজে বিজ্ঞানিল বৈজে বিজে বিজ্ঞানিল বৈজে বি	Summary BSP Clocks Pins Threads Messaging ICU Com	ponents Pin Confi 0 items
 ✓ Ⅲ → Properties ☎ ₽ Problems Property Name 	Summary BSP Clocks Pins Threads Messaging ICU Com Threads Messaging ICU Com Threads Messaging ICU Com Threads Messaging ICU Com Threads Messaging ICU Com Messaging ICU Com Threads Messaging ICU Com Threa	ponents Pin Confi 0 items Descriptior
III ► Properties ☎ ₽ Problems Property Name Symbol	Summary BSP Clocks Pins Threads Messaging ICU Com Threads Messaging ICU Co	ponents Pin Confi 0 items Descriptior
III Properties ☎ Problems Property Name Symbol Message Size (Words)	Summary BSP Clocks Pins Threads Messaging ICU Com Value HMI Queue g_hmi_queue 1	ponents Pin Confi 0 items Description

図 25 HMI スレッドオブジェクト g_hmi キューのプロパティ

5.4 モジュールコンフィグレーション

ドライバまたはフレームワークモジュールをプロジェクトに追加した場合、モジュールのプロパティを設定 する必要があります。プロパティは追加したドライバに依存します。再度[Properties]タブを使って設定を行 います。また、ウェザーパネルアプリケーションは r_glcd ドライバモジュールを追加します。ARM Cortex M4 MCU の GLCD 周辺回路を設定するのにこのモジュールを使用します。開発ボードのプロパティが多少 異なっても、これらのプロパティを設定するプロセスは、通常全ての開発ボードで同じです。

5.4.1 GLCD コンフィグレーション

図 26 の通り、[HMI Thread Modules]ダイアログボックスの下の[g_display Display Driver on g_glcd]モジュール を選択すると、プロパティタブの下に関連のあるプロパティが表示されます。長いプロパティのリストです が、ICU とモジュールの 2 つのグループに分かれています。ICU グループでは、IRQ の優先順位を設定しま す。

モジュールグループでは、GLCDコントローラ自身を設定します。プロパティの設定は煩雑ですが、分割して作業していきます。モジュールのグループの中にいくつかの大まかな区分があります。

- 名前:モジュールのインスタンスにつけられた名前(デフォルトでは g_display)、ユーザ定義のコール バック機能(使われている場合)の名前。ウェザーパネルアプリケーションでは、コールバックは使用 しません。
- INPUT:このモジュールプロパティは、グラフィックスコントローラへのインプット、フレームバッフ ァのサイズ、ドットクロックのソース、フレームバッファのある場所等を定義します。本章では、2つ の画像スクリーンウェザーパネルアプリケーションを定義します。ウェザーパネルアプリケーションは 1つの画面しか使用しません。Input-Graphics Screen 1を[Used]に設定します。
- OUTPUT: GLCD のアウトプットプロパティを定義するのがこのエリアです。[total Horizontal]、[Video Cycles]、アクティブビデオサイクル(縦横)、フロント・バックポーチ期間等のプロパティも含みます。
- TCON: [Pins] タブと一緒にこのラインを使用して Horizontal Sync (Hsync)、 Vertical Sync (Vsync)および Data Enable 信号をマップします。また、 GCLDへのクロック入力を分周する LCD パネル分周クロック 除数を設定できます。この分周比は現在 1/1~1/32 の間です。



 カラー補正:いろいろなレベルでディスプレイのカラー補正(明るさ、コントラスト、ガンマ)をここ から行います。LCD 画面のカラー、コントラスト、ガンマの補正は、本アプリケーションノートの範囲 には含まれていませんが、これらの調整はこここら行います。

All Thomas det	Threads	Q 6	HMI Thread Stacks					Ð
Netx_SNTP WeatherPapel	HME Tappage st message Messaging Framework on st mes	sage	amework on _external_ing	Framework on sf_message				
# Binaries 과 Includes 문 과 src 과 synergy	C st_touch_panal_li2c Touch Panel Framework c GDX	on st_touch_panel 12	rexternal_ing ternal IRQ Driver on icu		D/AVE 2D Driver on dave2d	Orisplay Display Driver on r_glad	g_sf_peg_decode JPEG Decode Framswork on sf_peg_decode	
Debug Soard_config guix_studio	g_hmi_quete Quete	¥. 1.			D/AVE 2D Port on st tes 2d drw		ajpeg_decode JPEG Decode Driver on rjipog	
Synergy_rig configuration.sr			X	l.			A. 12)
	Summany BSP, Clocks, Pins, Threads, Messaging, II	"II Components						
perties 22 🐘 Problem		Dem n M	Pin Conflicts 😂 🔛 Con	sole 🥜 Search 💷 Ca	Il Hierarchy			
aty	Value	× Dit	orms.	,				
mmon		De	scription		Modula	Pin Loci	ation Resource	
Parameter Checking	Default (BSP)	E						
0								
GLODC LINE DETECT	Priority 3							
GLCDC UNDERFLOW 1	Priority 3							
CLODE UNDERFLOW 2	Disabled							
sdule								
Name	g_cisplay							
Name of display callba	ck function NULL							
Input - Panel clock sour	rce select Internal clock(GLCDCLK)							
Input - Graphics screen	1 Used							
Input - Graphics screen	1 frame bu fo_background							
Input - Number of Grap	thics screer 2							
input - Section where C	iraphics ser sdram							
Input - Graphics screen	1 input her 800							
Input - Graphics screen	1 input ver 480							
Input - Graphics screen	Lignut hor 800							
where and up brings	1 insulting 16 hits DORGET							
input - Granning screen	1 ICICICIE DE LEIDERN ISON DE DE							
Input - Graphics screen Input - Graphics screen	Linput line Notused							

図 26 Properties を使用した GLCD プロパティコンフィグレーション

5.4.2 TCON コンフィグレーション

[Properties]タブを少し下にスクロールすると4つの TCON プロパティがあります。1つは、パネルクロック 分周比に関連しています。クロックツリーの PLLOUT プランチから直接作動するドットクロックをさらに 分周できます。その他の3つは LCD 同期信号に関連しています。これらの信号が実際に接続されている端 子へ送られるか見てみます。

TCON - Hsync pin select	LCD_TCON0
TCON - Vsync pin select	LCD_TCON1
TCON - DataEnable pin select	LCD_TCON2
TCON - Panel clock division ratio	1/8

図 27 PPE-HMI1 ボード LCD 用 TCON 設定

いくつかの柔軟性を持たせるため、S7G2 MCUの GLCD コントローラは 2 つの端子群のオプションを有して います。オプション毎には MCU 上で異なる端子を使用し、LCD ディスプレイに接続するデータラインを動 作させます。どちらの端子群を使用するかはハードウェア設計者が設定します。どちらのグループでも選択 すると、ハードウェア設計のその他の部分で必要となる MCU の端子をフリーにします。

PE-HMI1 ボードの回路図のとおり、全ての端子が LCD データラインに接続されています。同期信号に接続 された4端子は赤で印しています。ハードウェア設計者が選んだデータラインはGLCD モジュールの下にあ る2つの端子群の1つと一致します。LCD 同期信号には若干予備の柔軟性を有しています。





図 28 回路図からの PE-HMI1 LCD 特定信号

[Synergy Configuration]ウィンドウボックスの[Pins]タブでこれを簡単に理解できます。図 28 のとおり、ポート、周辺回路、アナログ端子、その他の端子の選択肢があります。周辺回路ダイアログボックスを開くと、このスクリーンから設定が可能な ARM コア周辺回路の種類全てが表示されます。

LCD_GRAPHICS エントリまで下にスクロールして、横の小さな + サインをクリックすると 2 つのオプショ ン GLCD_Controller_Pin_Option_A および GLCD_Controller_Pin_Option_B が表示されます。 GLCD_Controller_Pin_Option_B の横に緑のチェックマークが表示され、端子グループが LCD ディスプレイ の起動に関連していることを示します。

TCON0 はポート 3 端子 15(P315)と関連しています。回路図(P3_15)でこの指定を見た場合、このスクリーンのデータイネーブル端子の LCD_DE に接続されていることがわかります。図 27 を再度参照すると、 TCON0 が選択されデータイネーブル信号を動作させることがわかります。

Pin Selection	Pin Configuration		
type filter text 🖉 🗐 🕀 🖃			
₽ ✓ Ports			
🗄 🖌 Peripherals			
🗄 🖌 🖌 Analog Pins			
🗄 🛛 Other Pins			
•			•
	Renard Benning and Articles and	55	

図 29 PE-HMI1 端子構成タブビュー

LCD データライン(例:LCD_DATA_DATA00)を見ると、接続されている端子が回路図で実際に接続された 端子と一致しています。ポートグループを選択して、特定のポート及び端子を選択した時と同様に、端子の 右側の矢印をクリックすると、直接、関連の端子構成のダイアログボックスが表示されます。

S7G2-PE-HMI1_WeatherPanel.pincfg	▼ Generate	e data: _g_bsp_pin_cfg		
Pin Selection	Pin Configuration			
type filter text 🖉 🗵 🖻				
▶ CTSU	LCD_DATA12_B:	▲ babe	•	
🕨 🖌 Debug	LCD_DATA13_B:	✓ P907	•	
ETHER_MII	LCD_DATA14_B:	✓ P908	•	
▷ ✓ ETHER_RMII	LCD DATA15 B	- P901	*	4
P GPT1		0512		
▶ KEY	LCD_DATAT6_B:	A 14372		
✓ LCD_GRAPHICS	LCD_DATA17_B:	~ P805	•	
GLCD_Controller_Pin_Option_A	LCD_DATA18_B:	~ PA11	•	-
 GLCD_Controller_Pin_Option_B PDC 	LCD_DATA19_B:	~ P914	+	
DSPI	LCD DATA20 B:	~ P915	•	
⊳ ∽ RIIC		2000		6
⊳ ✓ RSPI		• [P909		
▶ ✓ SCI0_2_4_6_8	LCD_DATA22_B:	~ P910	•	
	LCD_DATA23_B:	✓ P902	•	4
 SSI 	LCD_EXTCLK_B:	None	•	(d)
▶ TRACE	LCD_TCON0_B:	✓ P315	•	(\$)
▷ ✓ USB_FS	LCD TCON1 B	V P314	*	\$
▶ ✓ USB_HS	LCD TCOND R	(0212		6
Other Pins	LCD_ICONZ_B:	* Foto		
T	LCD_TCON3_B:	None	*	

図 30 コンフィグレータを使用した場合の LCD 端子構成

例えば、LCD_TCON0_B 端子の横の矢印をクリックすると、端子選択スクリーンは図 31 のように表示され ます。端子は[Peripheral Mode]に設定されています。本書の執筆時点では、デフォルトでは、[Pull Up]は無 し、[Drive Capacity]は Low に、[output type]は CMOS となっています。この画面の右の矢印ボタンをクリッ クすると、関連のある周辺回路画面に戻ります。

注: 本書の執筆時点では、LCD_GRAPHICS 周辺回路のAまたはBを選択した場合には、ディスプレイに 接続する各端子を全て手動で有効にしてください。周辺回路画面と端子構成画面の間を矢印ボタンを 使って切り替えると、このプロセスを簡単に行えます。



ins				
elect pin configuration				
57G2-PE-HMI1_WeatherPane	el.pincfg	🔹 🗹 Generate	e data: g_bsp_pin_cfg	
in Selection		Pin Configuration		
type filter text				
▲ ✓ Ports	-	Module name:	P315	
⊳ P0 ⊳ ✓ P1		Symbolic name:	GLCD_CONTROLLER_PIN_OP	
⊳ v P2		Comment:	*	
⊿ ✓ P3				
🖌 P300		P315 Configuration		
✓ P301		Mode:	Peripheral Mode	
✓ P302		Pull up:	None	
- P303	and the second s	Drive Canacity:	High	
✓ P304		Dire copacity.	(
✓ P305		Output type:	CMOS	
× P300		Chip input/output		
P308		P315:	- LCD TCON0 B	
✓ P309				
< P310				
✓ P311				
¥ P312				
🗸 P313				
✓ P314				
✓ P315				
⊳ v P4				
P ✓ P5				

図 31 Pin Configuration 構成タブ

5.4.3 フレームバッファ用の外部メモリ

SK-S7G2のような低コストの開発ボードと、よりコスト高の PE-HMI ボートの相違点の1つは、スクリーン バッファ用に外部メモリ領域があるかという点です。画面サイズおよび色深度が増した場合、もしくはより 高機能のディスプレイを使用した場合(例:ピンポンフレームバッファ)、マイコンの内部メモリ容量が十 分ではない可能性がありますます。このような場合には、通常、外部メモリデバイスをボートに追加しま す。

S7G2 MCU は、外部 SDRAM に対応しています。図 32 は、S7 シリーズユーザマニュアルの第4章から抜粋 した S7G2 メモリマップです。SDRAM アドレス空間は 9000 0000h ~ 9800 0000h のアドレスに割り当てられ ます。



Renesas Synergy[™] プラットフォーム



図 32 PE-HMI1 ボートの SDRAM メモリ領域

フレームバッファに外部 SDRAMを使用する場合、2つのことが必要になります。1つは、[Properties]タブのGLCDコントローラの以下のプロパティを探し、[bss]から[sdram]へ変更します。慣例により、[bss]はSSPに内部メモリのフレームバッファをbssセクションに設置するようにします。

Input - Graphics screen1	Used
Input - Graphics screen1 frame buffer name	fb_background
Input - Number of Graphics screen1 frame buffer	2
Input - Section where Graphics screen1 frame buffer allocated	sdram

図 33 フレームバッファ用の SDRAM 選択

2つ目の要件は、外部メモリインターフェースを設定することです。設定は、前述した GLCD コントローラの設定に似ています。[Synergy Configuration]ウィンドウボックスの[Pins]タブに戻り、[Peripherals]を開き、 さらに[BUS]を開き、[External_Memory_Interface]を選択します。[Operation Mode]を[Enabled]に変更し、 [Peripheral]と[Pin Configuration]ビューを端子名の右側の矢印ボタンを使用して切り替えながら、SDRAM で 使用する各ラインを手動で有効にします。

in Selection	Pin Configuration			
ype filter text 🖉 🕅	8			
±✓ Ports	×			
🗄 🖌 Peripherals	A01:	✓ P115	<u> </u>	52
E AGT	A02:	✓ P114		
External_Memory_Interface	A03:	✓ P113		4
	A04:	✓ P112		\Rightarrow
	A05:	✓ P111		4
E CTSU	100	10001		->

図 34 外部メモリインターフェースの設定

外部メモリバスを正しく設定し、GLCD プロパティを SDRAM に変更した後でも、ディスプレイ上には、大きな変化は見えないかもしれません。これは、現在のスクリーン解像度が内部メモリ内で適合したことによ



る可能性があります。外部メモリに切り替えた場合、以前と同様にスクリーンは問題なく動作しますが、 GLCD は外部 SDRAM からフレームバッファを使用しています。e²studioの便利な機能

e²studio IDE には、LCD スクリーンで見る画像が外部 SDRAM からのデータであることを確認できる便利な 機能があります。この機能を使うには、e²studio がボードに接続されていることを確認し、デバッガでプロ グラムを実行してください。[Console]ウィンドウ上の[Memory]タブが開かれていることを確認してください (通常では、デバッグビュー画面の下に表示)。小さな緑の + をクリックして、メモリモニタを追加しま す。この時点で、[Monitor Memory] ダイアログボックスが図 35 のとおり表示されます。上記の ARM メモ リマップから、SDRAM 領域と関連のある外部アドレス空間を 16 進法で(0x9000000)と入力し、[OK]をクリ ックします。

新規のタブが、メモリモニタ用に設定したメモリ領域の内容を表示している[Memory]タブに表示されます。

74 79	<pre></pre>	andler[]
80 0001fc10	{	
81 0001fc14	<pre>while(1);</pre>	
82	}	
83		
84		Monitor Memory
85	/* Main stack */	
86	<pre>static uint8_t g_main_stac</pre>	k[BSP_CF Enter address or expression to monitor:
87		
00	/# n+!. #/	0x90000000
🛛 Console 🎦 Prol	blems 🧟 Tasks 🔲 Properties 🚺 Mem	ory 🛛 🗿 🐘
Ionitors		Cancel
1973-1973-1976-1976-1976-1976-1976-1976-1976-1976		

図 35 SDRAM コンテンツを表示するためのメモリモニタ

SDRAM メモリ領域のコンテンツが、今作成したメモリモニタに表示されます。ディスプレイに表示される 各ピクセルの 16 進法の値が分かると想定した場合、画像が外部 SDRAM に保存されていることを、メモリ モニタを使って知ることができます。もちろんほどんどのユーザはピクセルに関連した 16 進法の値がわか らないので、さらにメモリモニタに働いてもらうことにしましょう。

Monitors	🕂 🗶 🆗	0x90000000 : 0x9000000	0 <hexintege< th=""><th>r> 🛿 🔶 🕂 Ne</th><th>w Renderings</th><th></th></hexintege<>	r> 🛿 🔶 🕂 Ne	w Renderings	
♦ 0x90000000		Address	0 - 3	4 - 7	8 - B	C - F
		0000000090000000	01F001F1	01F10211	02110211	01F00211
		0000000090000010	01F101F1	021001F0	02100210	021101F1
		0000000090000020	01F101F1	02110211	02110211	02110211
		0000000090000030	02110211	021101F0	02110211	02110211
		0000000090000040	02100211	021101F1	02100211	02110211

図 36 SDRAM コンテンツビュー

今作成したメモリモニタの隣の[New Rederings...]のタブを選択し、オプションリストから[Raw Image]を選択 します。それから、スクリーンの右側にある[Add Rendering(s)]ボタンをクリックします。 Renesas Synergy[™] プラットフォーム

Monitors + X & 0x9000000 <hex integer=""> + New Renderings Ox90000000 Select rendering(s) to create: Fixed Point Image Raw Image</hex>				💽 🎲 🍪 eiei giot 🔚 📋	
Ox9000000 Memory Monitor: 0x9000000 : 0x9000000 Select rendering(s) to create: Fixed Point Image Raw Image Raw Image	🔶 🗙 🔆	0x90000000 <hex integer=""></hex>	💠 New Renderings		
Fixed Point Image Raw Image		Memory Monitor: 0x9000000 Select rendering(s) to create:	0 : 0x9000000		
Raw Image		Fixed Point Image			Add Rendering(s)
		Raw Image			
		◆ X ½	X X Ox9000000 <hex integer=""> Memory Monitor: 0x9000000 Select rendering(s) to create: Fixed Point Image Raw Image Floating Point</hex>	X X Ox90000000 <hex integer=""> Wew Renderings Memory Monitor: 0x90000000 : 0x90000000 Select rendering(s) to create: Fixed Point Image Raw Image Floating Point</hex>	X X Ox90000000 <hex integer=""> New Renderings Memory Monitor: 0x90000000 : 0x90000000 Select rendering(s) to create: Fixed Point Image Raw Image Floating Point</hex>

図 37 GUIX レンダリング形式選択ビュー

この時点で、[Raw Image Format]ダイアログボックスが表示され、スクリーン解像度の[Width]および [Height]、また、[Encoding] (この場合: 16bpp (5:6:5))を設定します。

Dimensions	
Width: 800	
Height: 480	
Encoding	ne
RGB:	[16bpp (5:6:5)
🔘 BGR:	32bpp (8:8:8:8)
O YCbCr:	24bpp (4:4:4 chunky) (y0, Cb, Cr)
Start Position Top Bottom	
?	OK Cancel

図 38 PE-HMI1 におけるウェザーアプリの Raw Image フォーマット

[OK]ボタンをクリックすると、メモリモニタに入力したパラメータを元にメモリアドレスに表示される画像 が表示されます。この時点でメモリモニタタブへ戻り、メモリの位置を変更して、メモリモニタと実際の LCD スクリーンで画像が変更することができます。内部メモリの容量がなくなってしまった場合にも同様 の手順を実行することができますが、最初にリンカーマップを参照して、.bss section のどこにスクリーンメ モリがマップされているかを特定し、その位置でメモリモニタを開きます。 それから上記手順を繰り返し 行います。



図 39 e²studio メモリモニタを使用した GUIX 実行



図 40 e²studio メモリモニタを使用した GUIX 実行

6. アプリケーション設計の要点

本章では、ウェザーパネルアプリケーションの要点について詳説します。ウェザーパネルアプリケーションの目的は、SSPで ThreadX および GUIX を使用しながら、より複雑なマルチスレッド HMI アプリケーションを開発方法を示すことです。

SSPは、多くの周辺機能のインターフェースに関する複雑さを集約して、出来る限り早く、より複雑なアプリケーションを構築することにユーザが楽に集中できるようにすることを目指しています。

6.1 スレッドおよび main

ThreadX を SPP 環境で使用するには、いくつかの微妙な違いがあります。典型的な ThreadX アプリケーションでは、main()が tx_kernal_enter()を呼び、それから、tx_kernal_enter()が tx_application_define()を呼びます。 ユーザが Synergy で作業する以前に ThreadX アプリケーションのプログラム経験がある場合には、主なアプ リケーションスレッドの作成作業、tx_application_define()でアプリケーションが使用するその他のリ ソース(例:キュー、セマフォ等)の定義作業にに慣れてきているでしょう。

Synergy フレームワークでは、main()は、自動的に作成されたファイルで、そのコードは下記に類似したものになります。この場合、tx_application_define()は、フレームワーク設定の間に指定したスレッドのためにスレッドエントリ関数を呼びます。

```
void tx_application_define(void* first_unused_memory)
{
    hmi_thread_create ();
#ifdef TX_USER_ENABLE_TRACE
    TX_USER_ENABLE_TRACE;
```

#endif

}

g_hal_init ();

tx_application_define_user (first_unused_memory);

```
void main(void)
{
    __disable_irq ();
    tx_kernel_enter ();
}
```

[Threads]タブを使いスレッドを作成する場合、フレームワークは実際にいくつかのファイルを作成します。 例えば、HMIスレッドが追加される時に、フレームワークは3つのファイルを作成します。図41に示すと おり、hmi_thread.h、hmi_thread.c、hmi_thread_entry.cの3つのファイルです。

初めの2つのファイルは自動作成であるため、synergy_genフォルダに置かれています。

hmi_thread_entry.cファイルは、HMIスレッドへのエントリポイントで、ここにアプリケーションプロ グラムのコードを格納します。自動作成ファイルはプロジェクトが構築される度、または[Generate Project Content]ボタンをクリックする際に再生されるため、ユーザは更新しません。自動作成ファイルの一番上に [do not edit]の類のメッセージが必ず含まれています。



🗈 🗱 Binaries	
🕞 🔂 Includes	
🔺 😂 src	
🕨 🗁 s7g2_pe_hmi1	
🔺 🗁 synergy_gen	
Image: Common_data.c	
👂 🖻 common_data.h	
▶ 🖻 hal_data.c	
▷ <a>b hal_data.h	
Image:	
Image: http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.image.com/ http://www.i	
👂 🧟 main.c	
Image: big message_data.c	
pin_data.c	
🕨 🚾 hal_entry.c	
Imi_thread_entry.c	
Image:	

図 41 フレームワークが作成したソースファイル構成

6.1.1 GUIX 初期化

本章では、GUIX 初期化について詳述します。GUIX システムはフレームワークにより自動的に初期化され ません。GUIX を初期化して、実際に描かれる初期化のキャンバスを作成するにはいくつかの呼出が必要で す。hmi_thread_entry.c file 内にある hmi_thread_entry()関数の最初にこの初期化コードがあります。

```
status = qx_system_initialize();
if(TX_SUCCESS != status)
{
    while(1);
}
/* Initializes GUIX drivers.*/
err = g_sf_el_gx0.p_api ->open (g_sf_el_gx0.p_ctrl, g_sf_el_gx0.p_cfg);
if(SSP_SUCCESS != err)
{
    while(1);
}
gx_studio_display_configure ( MAIN_DISPLAY,
                               g_sf_el_gx0.p_api ->setup,
                               LANGUAGE ENGLISH,
                               MAIN_DISPLAY_THEME_1,
                               &p_window_root );
err = g_sf_el_gx0.p_api ->canvasInit(g_sf_el_gx0.p_ctrl, p_window_root);
if(SSP_SUCCESS != err)
{
    while(1);
}
```

6.1.2 イベントおよび GUIX メッセージ

ウェザーパネルアプリケーションのスクリーンにタッチすると、GUIX が GUIX studioのスクリーン用に定義した特定のコールバック関数を起動させます。GUIX はコールバック関数にイベントを発生させるウィン

ドウおよび実際に発生したイベントの特定の情報を提供します。現在、GUIX により認識されるイベントは 46 種類もあります。このイベントは gx_api.h ファイル内で定義され、便宜上、ここで再生されます。

/* Define the pre-defined Widget event types.*/

#define	GX_EVENT_TERMI NATE	1
#define	GX EVENT REDRAW	2
#define	GX_EVENT_SHOW	3
#define	GX EVENT HIDE	4
#define	GX_EVENT_RESIZE	5
#define	GX_EVENT_SLIDE	6
#define	GX_EVENT_FOCUS_GAI NED	7
#define	GX_EVENT_FOCUS_LOST	8
#define	GX_EVENT_HORI ZONTAL_SCROLL	9
#define	GX_EVENT_VERTI CAL_SCROLL	10
#define	GX_EVENT_TIMER	11
#define	GX_EVENT_PEN_DOWN	12
#define	GX_EVENT_PEN_UP	13
#define	GX_EVENT_PEN_DRAG	14
#define	GX_EVENT_KEY_DOWN	15
#define	GX_EVENT_KEY_UP	16
#define	GX_EVENT_CLOSE	17
#define	GX_EVENT_DESTROY	18
#define	GX_EVENT_SLI DER_VALUE	19
#define	GX_EVENT_TOGGLE_ON	20
#define	GX_EVENT_TOGGLE_OFF	21
#define	GX_EVENT_RADI 0_SELECT	22
#define	GX_EVENT_RADI 0_DESELECT	23
#define	GX_EVENT_CLI CKED	24
#define	GX_EVENT_LI ST_SELECT	25
#define	GX_EVENT_VERTICAL_FLICK	26
#define	GX_EVENT_HORI ZONTAL_FLICK	28
#define	GX_EVENT_MOVE	29
#define	GX_EVENT_PARENT_SI ZED	30
#define	GX_EVENT_CLOSE_POPUP	31
#define	GX_EVENI_ZOOM_IN	32
#define	GX_EVENI_ZOOM_OUI	33
#define	GX_EVENI_LANGUAGE_CHANGE	34
#define	GX_EVENT_RESOURCE_CHANGE	35
#define	GX_EVENT_ANIMATION_COMPLETE	36
#derine	GX_EVENT_SPRITE_COMPLETE	37
	GX_EVENI_IEXI_EDIIED	40
#define	UX_EVENT_IX_IIMEK	41
		42
#uerine	UX_EVENT_FUCUS_PREVIOUS	43
#define	UX_EVENI_FUCUS_UAIN_NUTIFY	44
	GA_EVENI_SELECT	45
#detine	GX_EVENT_DESELECT	46

ウェザーパネルアプリケーションは、GX_EVENT_CLICKED のようなイベントの一部を使用します。GUIX はGX_EVENT 構造体としてイベントを受け渡します。構造体の最初の要素はイベントタイプです。 GX_EVENT は、メッセージの一環としてデータを送ることができます。最終フィールドの gx_event_payload は、多種のデータを結合したものです。ウェザーパネルアプリケーションはこのペイロードを使用して現在 のステータスデータ構造体にポインタを送ります。

```
/* Define Event type.Note: the size of this structure must be less than or equal to
the constant
   GX_EVENT_SIZE defined previously.*/
typedef struct GX_EVENT_STRUCT
{
    ULONG
             gx_event_type;
                                  /* Global event type
                                                                                 */
    USHORT
             gx_event_sender;
                                                  /* ID of the event sender
*/
                                                    /* ID of event destination
    USHORT
             gx_event_target;
*/
    ULONG
             gx_event_di spl ay_handl e;
    uni on
    {
        UINT
                    gx_event_timer_id;
        GX_POI NT
                    gx_event_pointdata;
                    gx_event_uchardata[4];
        GX_UBYTE
        USHORT
                    gx_event_ushortdata[2];
        ULONG
                    gx_event_ul ongdata;
        GX BYTE
                    gx_event_chardata[4];
                    gx_event_shortdata[2];
        SHORT
        INT
                    qx event intdata[2];
        LONG
                    gx_event_l ongdata;
    } gx_event_payl oad;
GX_EVENT;
```

6.2 LCD 制御

PE-HMI1ディスプレイには、ウェザーパネルアプリケーションから駆動するデジタル制御がいくつかあります。最初に、ハードウェアの依存性を特定し、適切なドライバを設定します。

本章で紹介する 2 つの信号は LCD_ON および LCD_BLEN (ブランクイネーブル)です。図 42 は、PE-HMI1 V2.0 回路図からの抜粋ですが、J5 コネクタを示しています。LCD スクリーンがプラグインするコネクタで す。図のとおり、2 つの信号が PA5 および PA3 の関連する MCU の端子を表記しています。です。LCD_ON 信号は単純に LCDを ON / OFF する Hi/Low 状態が必要となります。LCD_BLEN 信号にはディスプレイの輝 度を変調する PWM 信号が必要です。



図 42 PE-HMI1 の LCD On and バックライト端子の詳細

図 43 は、 PE-HMI1 ウェザーパネルアプリケーションの[Pins]タブを示します。端子の設定をする場合、最 初にスクリーン左側にある [Pin Selection]ダイアログボックスより、ポートを選択します。この場合 PA(ポー ト A)です。ポートの選択から I/O ポートに関連する端子が開きます。

Select pin configuration				
S7G2-PE-HMI1_WeatherPanel.pi	incfg	▼	Generate data: g_bsp_pir	n_cfg
Pin Selection		Pin Configuration		
type filter text	<u>&</u> 🗄 🖻			
⊿	*	Module name:	PA03	
PA00		Symbolic name:	GPI01	
🛩 PA01		Comment:	1	
PA02		854.033.03.03		
(PA03)		PA03 Configuration		
PA04		Mode:	Output mode	-
V PAUS		initiode.	ouparmode	
PA00 PA07	111	Pull up:	None	
✓ PA08		Drive Capacity:	Low	•
PA09		Output type:	смоя	*
🗸 PA10		100		
PA11		IKQ:	None	•
PA12		Chip input/output		
PA13		PA03:	V GPIO	•
PA14)
PA15	*			

図 43 PE-HMI1 の LCD ON 端子設定

この場合、GPIO 端子のもっとも簡単な設定は[Mode]は、[Output mode]に、[Chip input/output]は、[GPIO]に 設定します。モジュール名は PA03 で、Synergy 端子設定で見られる命名慣例です。I/O 端子をこの様に命名 すると、Synergy フレームワークが自動的に作成される hal_data.c ファイルの中に端子のインスタンスを 作成します。

```
const ioport_instance_t g_ioport =
{ .p_api = &g_ioport_on_ioport, .p_cfg = NULL };
```

これにより、端子へドライバがアクセス可能になりますが、端子状態を設定するコードを書くのはユーザで す。この場合、ウェザーパネルアプリケーションに必要なのは、端子をハイレベルに設定し、LCDディス プレイを ON にすることだけです。これは、hmi_thread_entry.cファイル内でコードを初期化中に完了す る作業です。

```
/* Controls the GPIO pin for LCD ON. */
err = g_ioport.p_api->pinWrite(IOPORT_PORT_10_PIN_03, IOPORT_LEVEL_HIGH);
if (err)
{
    while(1);
}
```

このデモアプリケーションでは、g_i oport. p_api ->pi nWri te()の呼出しからエラーが返ってきた場合、 エラー処理は単に[while (1)]条件でループします。このループは HMI スレッドの応答を停止させ、pinWrite コールからエラーが戻ってきます。

7. プロジェクトのインポートとビルド

e²studio にウェザーパネルアプリケーションを取り入れるには、以下の手順を行ってください。

- 1. e²studio ISDE (Version 5.0.0.43 以降)を起動
- 2. ワークスペースランチャー内で、選択したワークスペースの場所を探す
- 3. [Welcome]ウィンドウを閉じる
- 4. ISDE で、メニューから[File] [Import]を選択
- 5. [Import]ダイアログボックスで、[Existing Projects into Workspace]を選択
- 6. ワークスペースのルートディレクトリ(プロジェクトが保管されている場所)を選択
- 7. インポートするプロジェクトを選択し、[Finish]をクリック

8. 対象ボートへ実行するプログラムのダウンロード

コードを実行するには、次の手順を行ってください。

- 1. **PE-HMI1** のクリックスタートガイドを参照し、PC から対象ボード上の JTAG コネクタへ **J-Link** デバッ ガ接続をセットアップ
- 2. [Run] [Debug configurations]を選択
- 3. [Debug]をクリック、リセットハンドラでプログラムを中断
- 4. ISDE が要求してきたら、[Yes]をクリックして、[Debug perspective]に切り替える
- 5. [**Resume**]を2回クリック

9. 参考資料

- PE-HMI1 v2.0 ユーザーズマニュアル:ハードウェア編
- PE-HMI1-V2.0 回路図
- Renesas Synergy Software Package データシート
- Synergy_X-ware $F = J \times V F(GUIX, ThreadX)$
- Synergy[™] Software Package (SSP) v1.1.0 ユーザーズマニュアル
- e²studio ヘプロジェクトをインポートしプロジェクトをビルドに関する Synergy プロジェクトインポート ガイド(r11an0023 j u0100_synergy_ssp.pdf)



ホームページとサポート窓口

Renesas Synergy[™] ギャラリー:

https://synergygallery.renesas.com/support

テクニカルサポート窓口:

- •米国: <u>https://renesas.zendesk.com/anonymous_requests/new</u>
- •ヨーロッパ: <u>https://www.renesas.com/en-eu/support/contact.html</u>
- •日本: <u>https://www.renesas.com/ja-jp/support/contact.html</u>

すべての商標および登録商標は,それぞれの所有者に帰属します。



改訂記録

		改訂内容	
Rev.	発行日	ページ	ポイント
1.0	2015.10.09	-	初版
1.10	2015.12.04	2	プロジェクトの作成とビルドの章で、最適化レベルの変更手 順を削除。この手順は SSP version 1.0.0-beta.3 以降には、使 用されなくなっている。
1.11	2016.01.11	-	テンプレートの参照先を削除。
2.00	2016.08.23	全体	PE-HMI および ISDE 5.0.0.43 の参照先を追加。
2.01	2016.11.18	全体	書式変更

	~ 注 辛聿之
	こ注息者で
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、その他の不適切に使用しないでくたさい。かかる改造、改変、複製寺により生した損害に関し、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、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、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できません。たとえ、意図し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。また、当社製品および技術 を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル(無人航空機を含みます。)の開発、設計、製造、使用もし くは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、か つ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
	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.	. お客様の転売、貸与等により、本書(本ご注意書きを含みます。)記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は一切その 責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
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.	. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。
注1	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その総株主の議決権の過半数を 直接または関接に保有する会社をいいます。
注2	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.3.0-1 2016.11)

RENESAS

ルネサスエレクトロニクス株式会社

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/

■営業お問合せ窓口

© 2017 Renesas Electronics Corporation. All rights reserved. Colophon 5.0