

## 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。例えば、DK-S7G2およびPE-HMI1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、各ボード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ボード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

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ミング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 e<sup>2</sup> 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。e<sup>2</sup> 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、e<sup>2</sup> studio 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。

- ポートの設定
- 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 ボード (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 構成要素タブ.....	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 フレームバッファ用の外部メモリ.....	27
5.4.4 e <sup>2</sup> 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. 対象ポ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. ポートの設定

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 e<sup>2</sup>studio 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

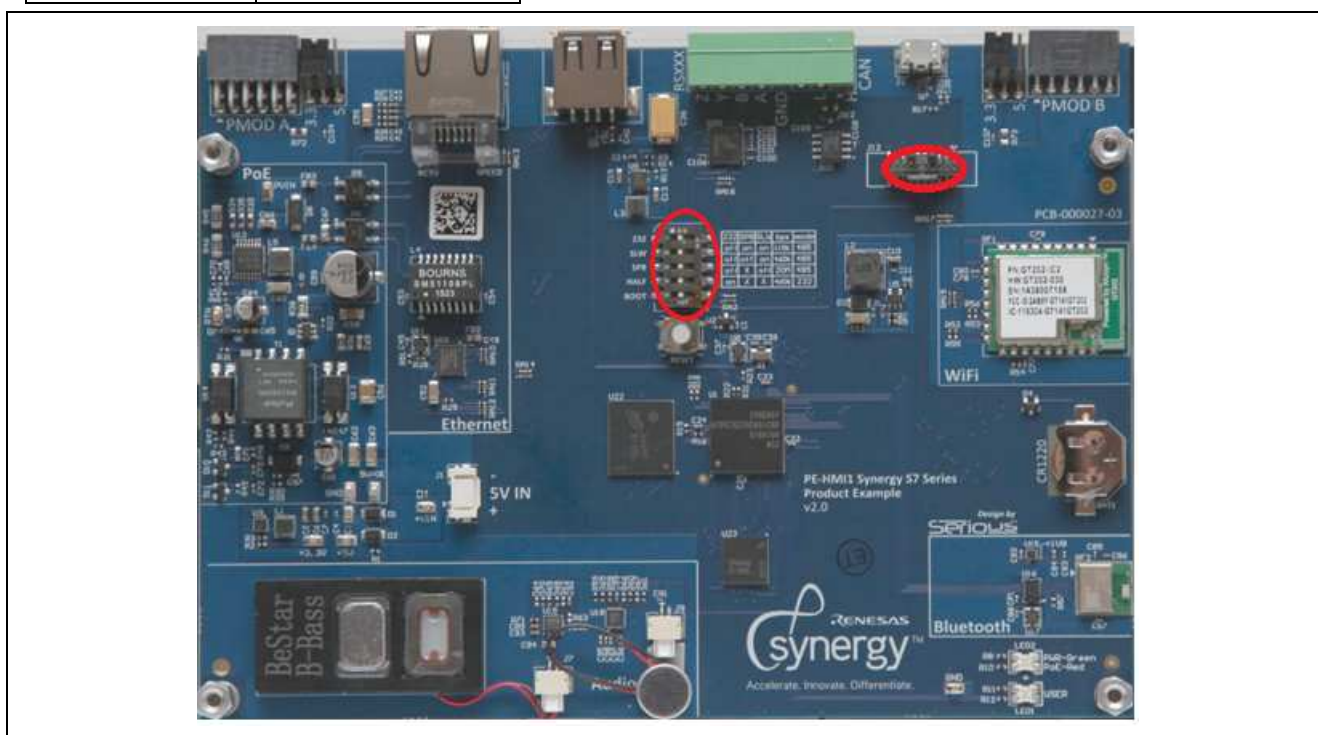


図 2 ウェザーパネルアプリケーションの PE-HMI1 V2.0 ハードウェア詳細

## 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。

1. 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。
2. 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、ポート上の 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。





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されます。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・ウェジェット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。

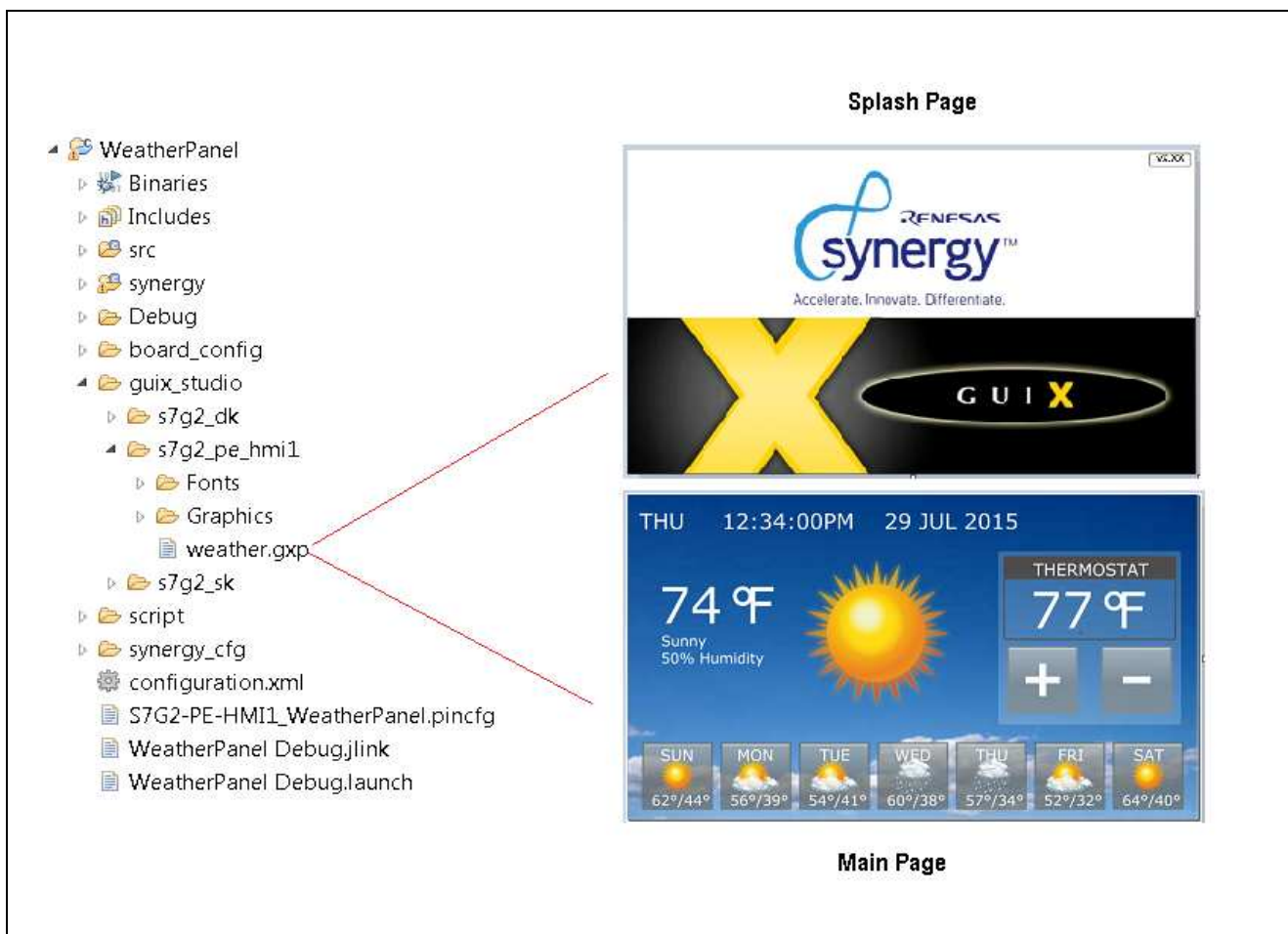


図4 ウェザーパネルアプリケーションのスクリーンショット

#### 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）

DK-S7G2 および PE-HMI ポートには 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。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。

### 2.3.2 スクリーン（小）デザイン

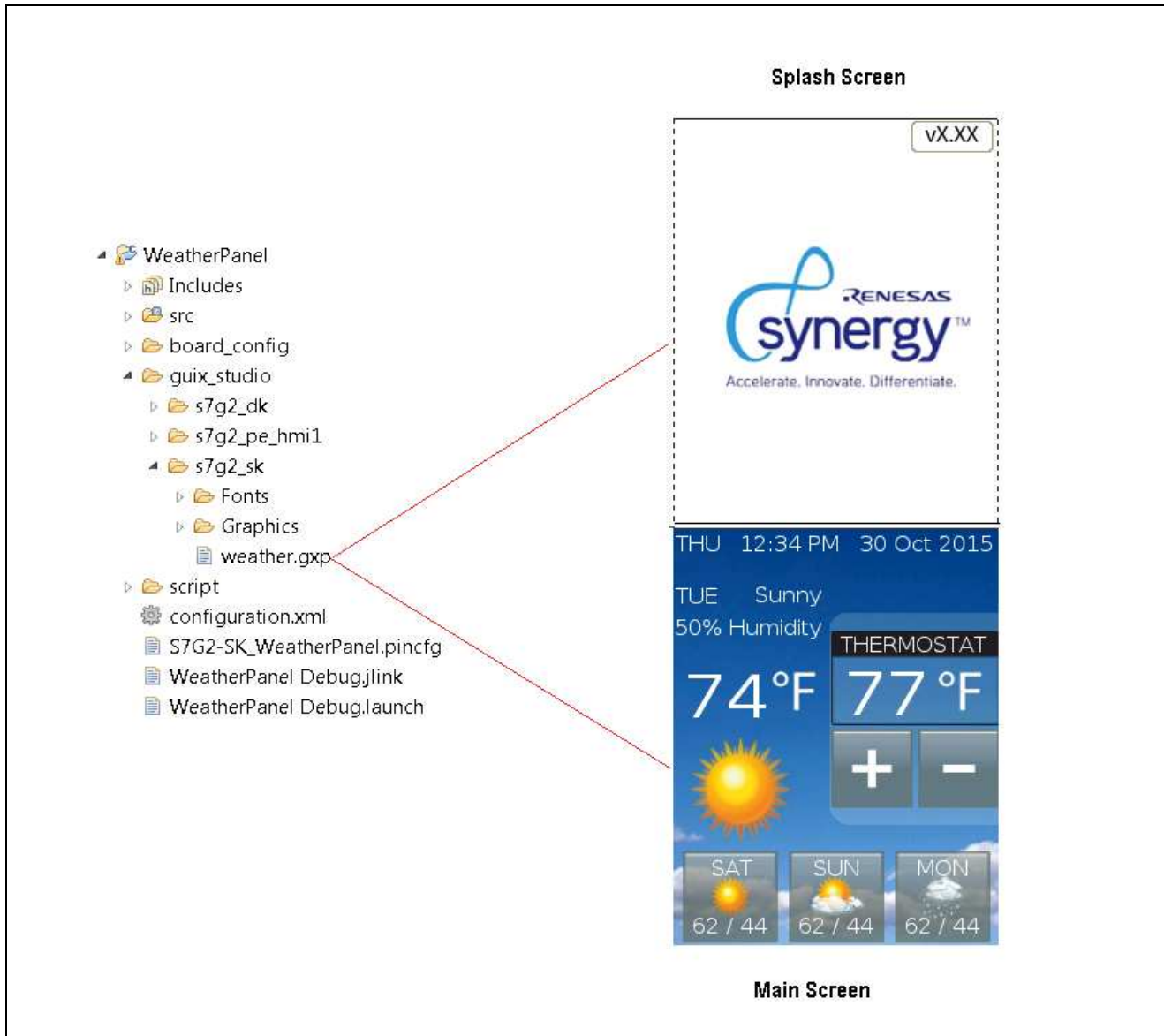


図 5 SK-S7G2 ポートのスクリーンショット

図 5 は SK-S7G2 ポ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トを示しています。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は、コールバック機能をウィジェットに関連づけます。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
```

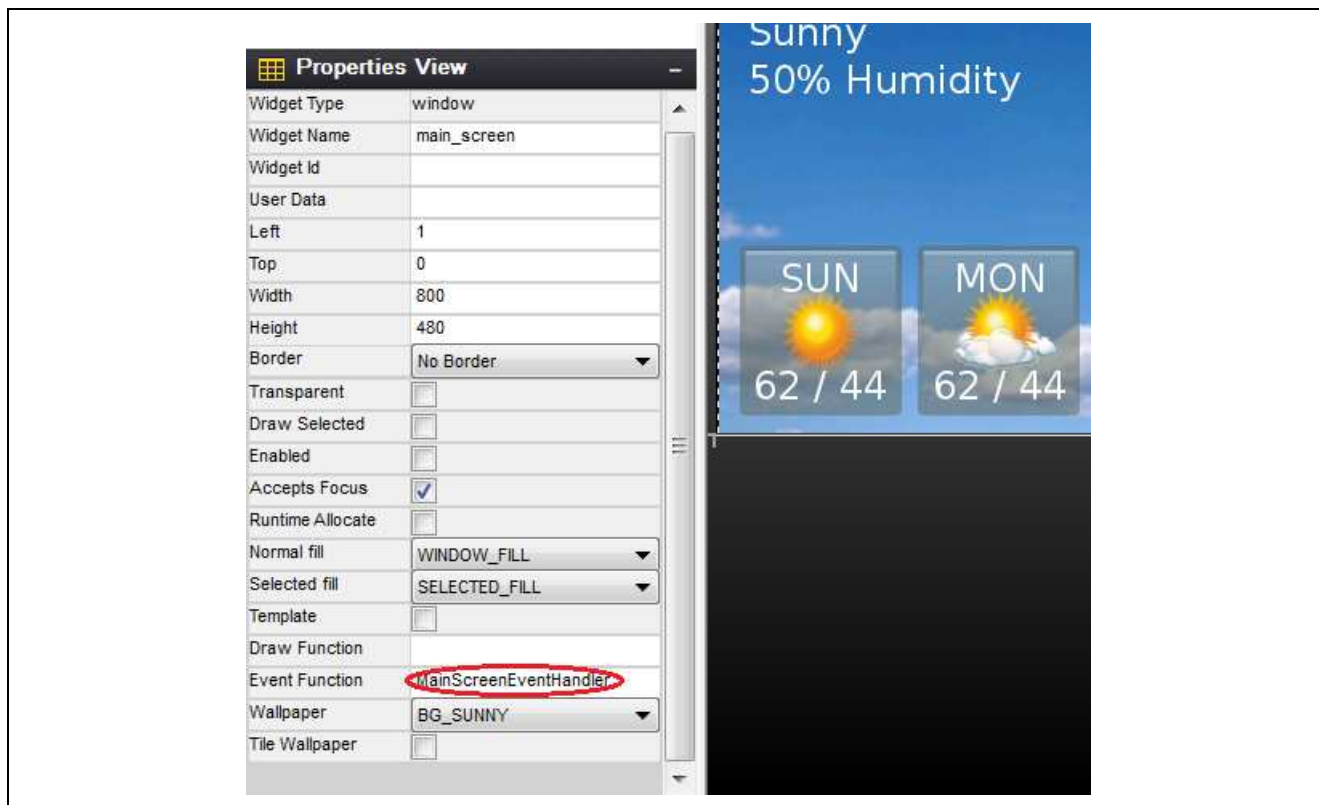


図 7 GUIX Studio スクリーンプロパティビュー

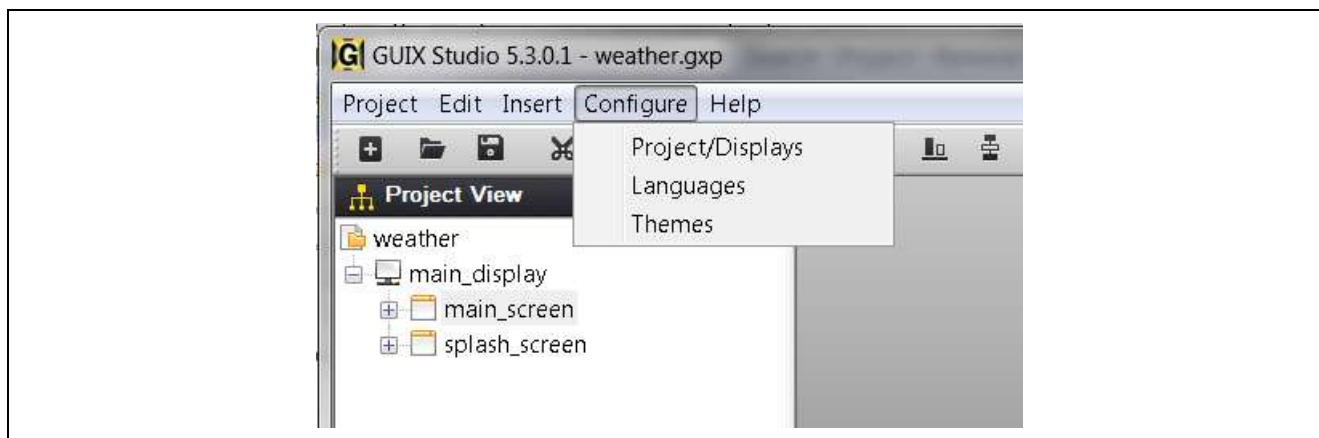


図 8 GUIX Studio プロジェクトコンフィグレーションビュー

図 9 は[Configure 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。

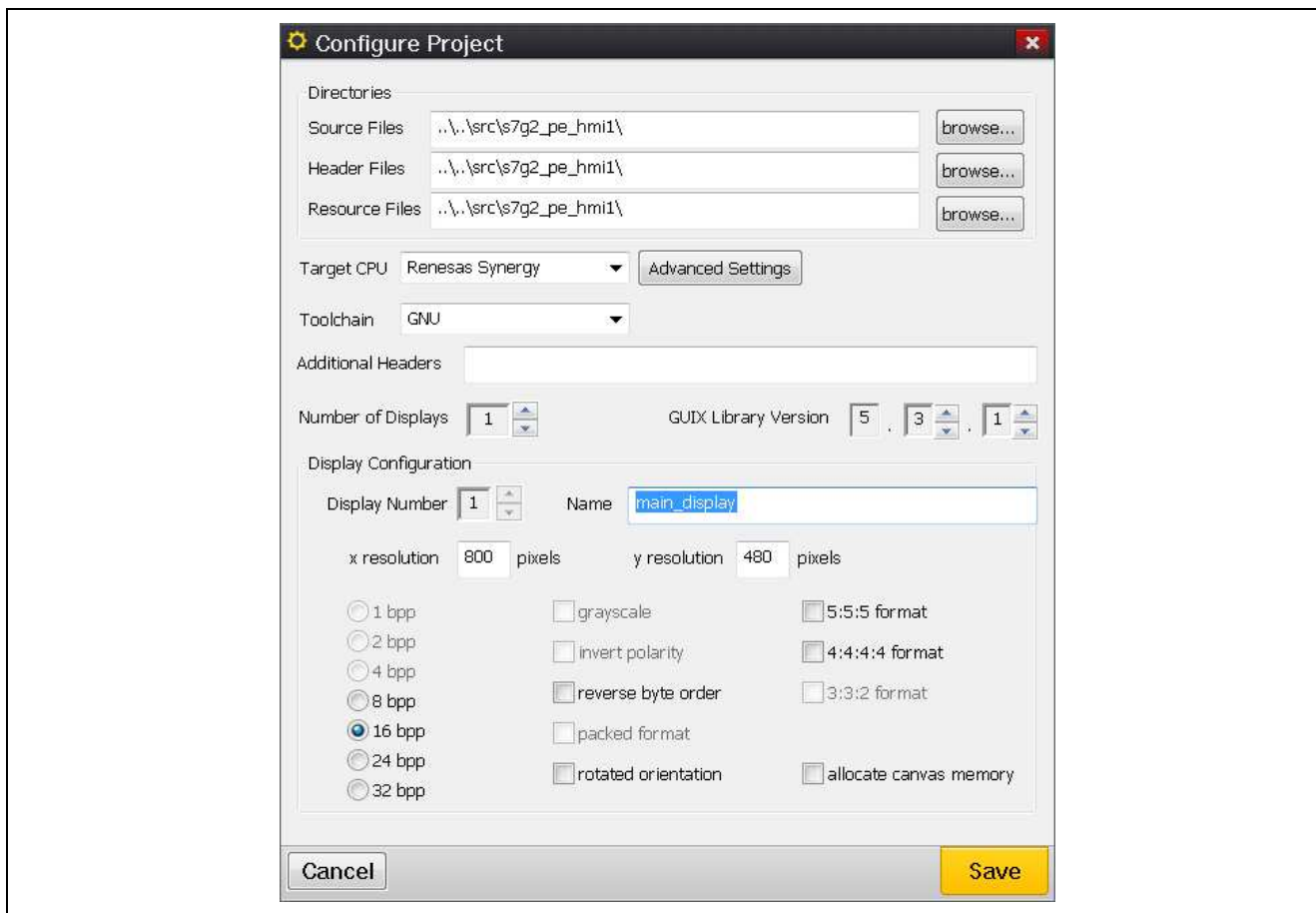


図 9 GUIX プロジェクト設定ウィンドウ

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。

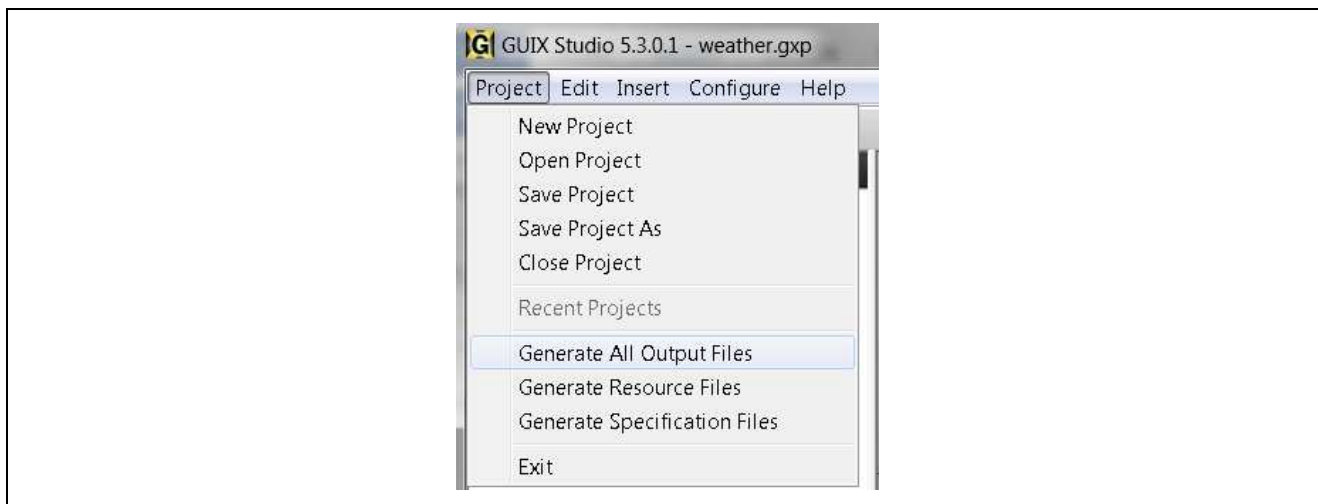


図 10 GUIX プロジェクト：ソースファイルの作成

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。

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み場合、weather.gxp ファイルをクリックして GUIX Studio を起動させます。

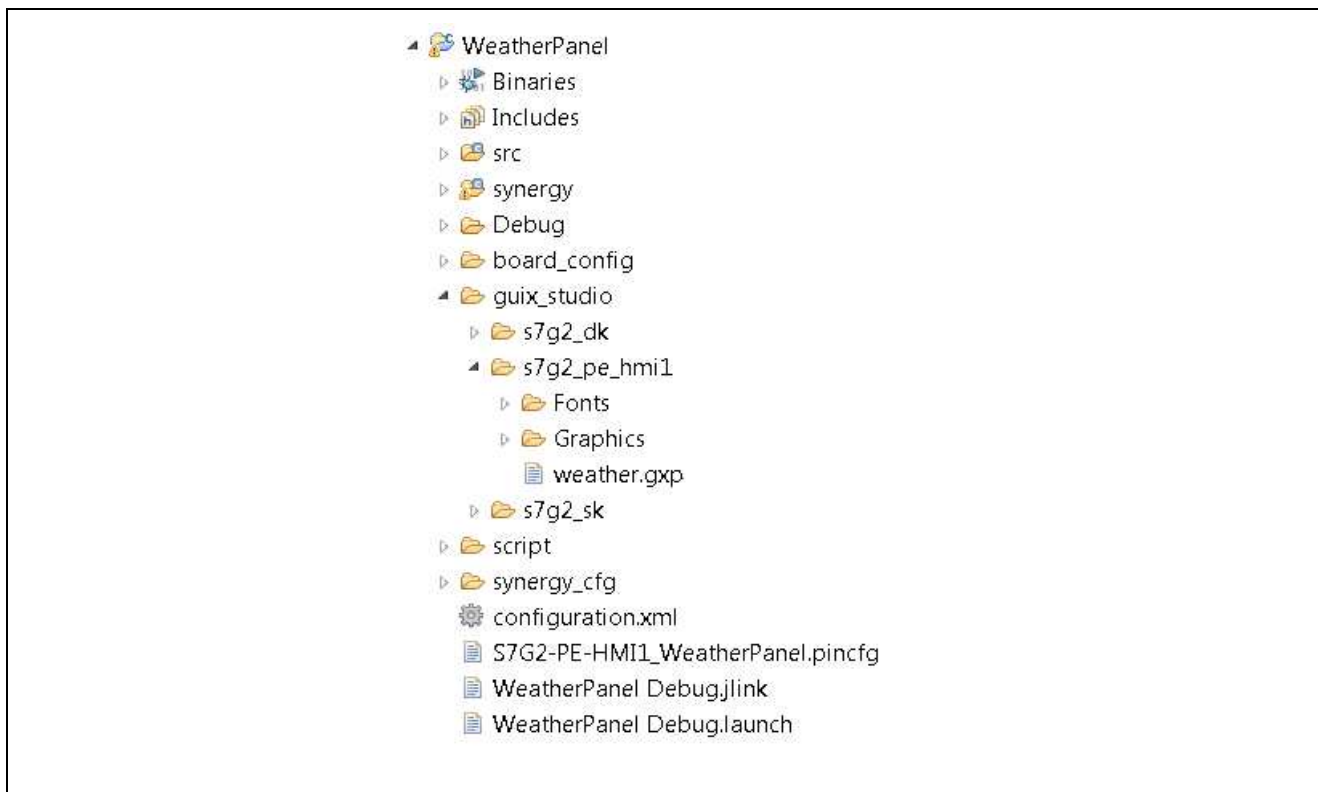


図 11 ウェザーパネルアプリケーションにおける GUIX プロジェクトのファイルビュー

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\_specifications.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\_handler.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。



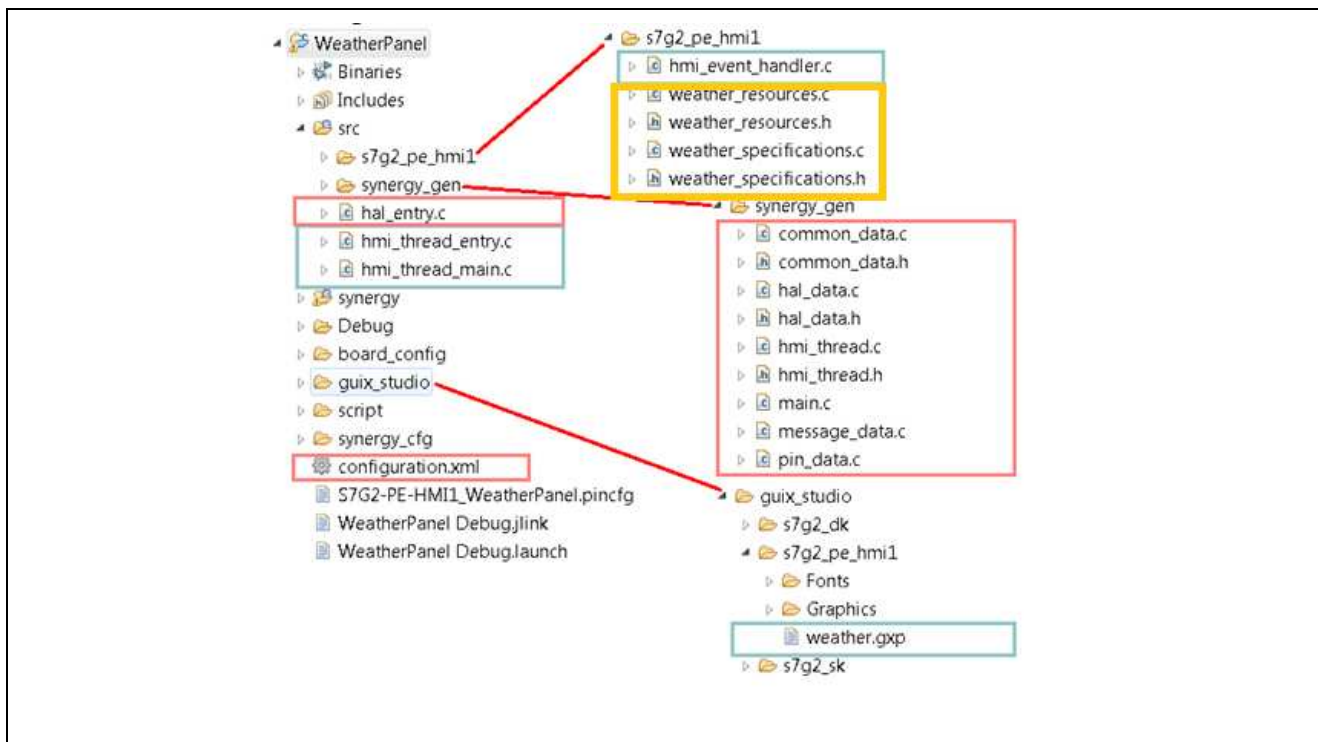


図 12 ウェザーパネルプロジェクトのソースファイルレイアウト

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\_handler.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、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 Synergy メッセージ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ギュレーションで、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。

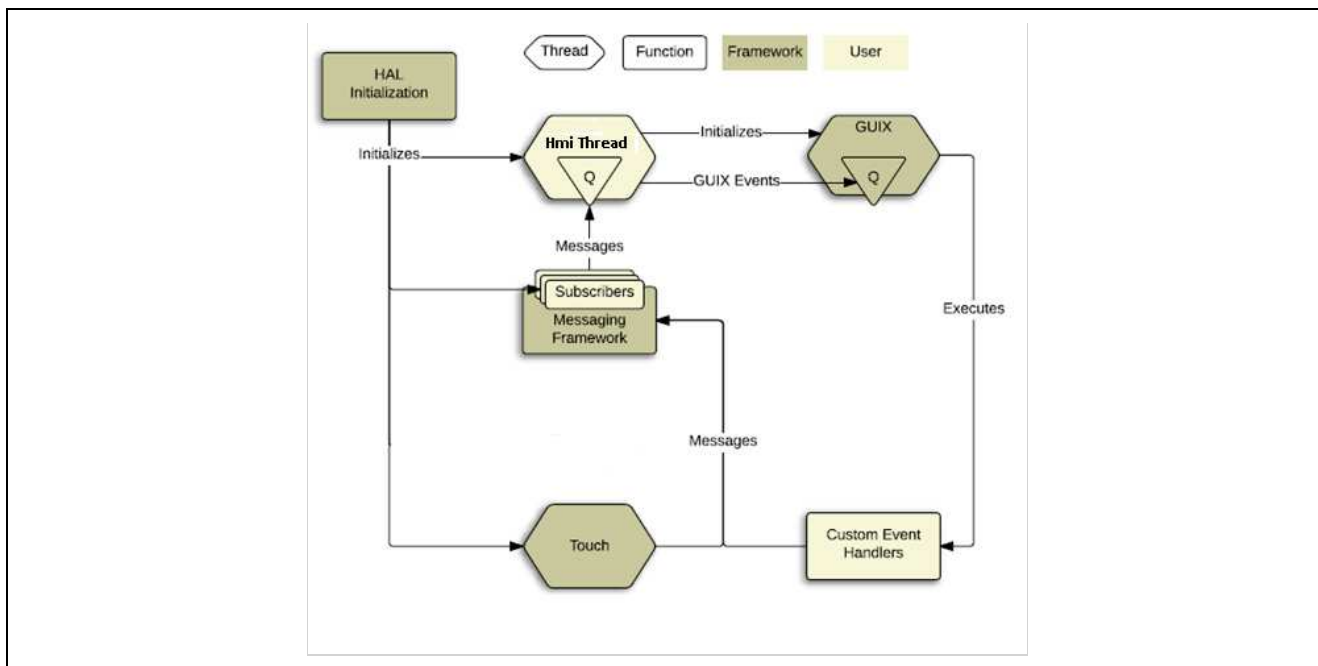


図 13 ウェザーパネルアプリケーションメッセージシーケンスフロー

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。これらの入力の内いずれかによりシステムの状態に変化を生じた場合、HMI 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。

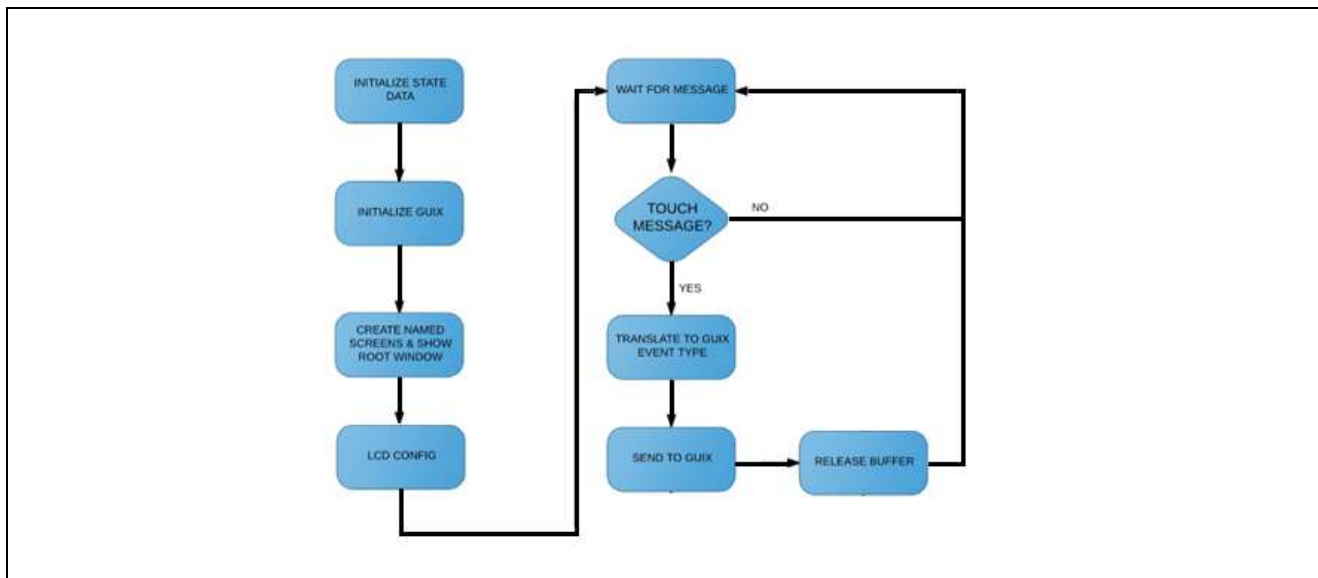


図 14 HMI スレッドフロー図

### 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、ということを知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ことで 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。

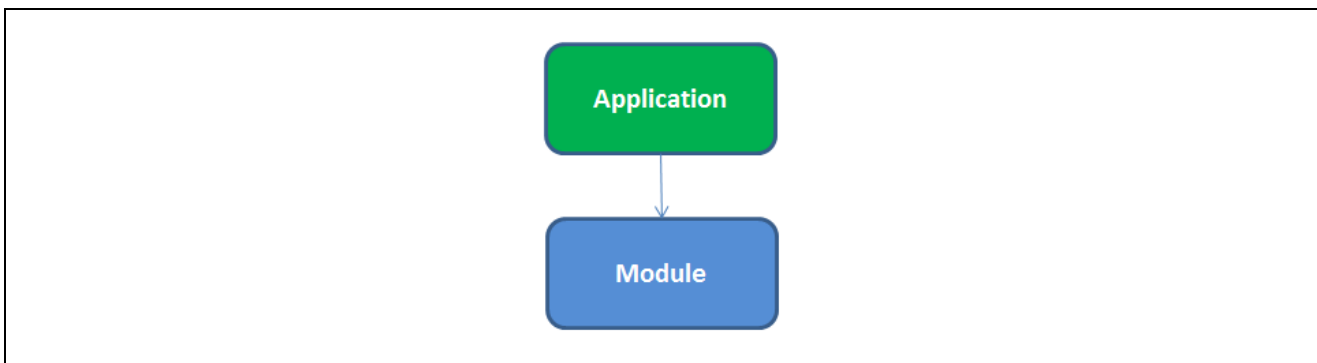


図 15 SSP アプリケーション & モジュールレイヤ

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。

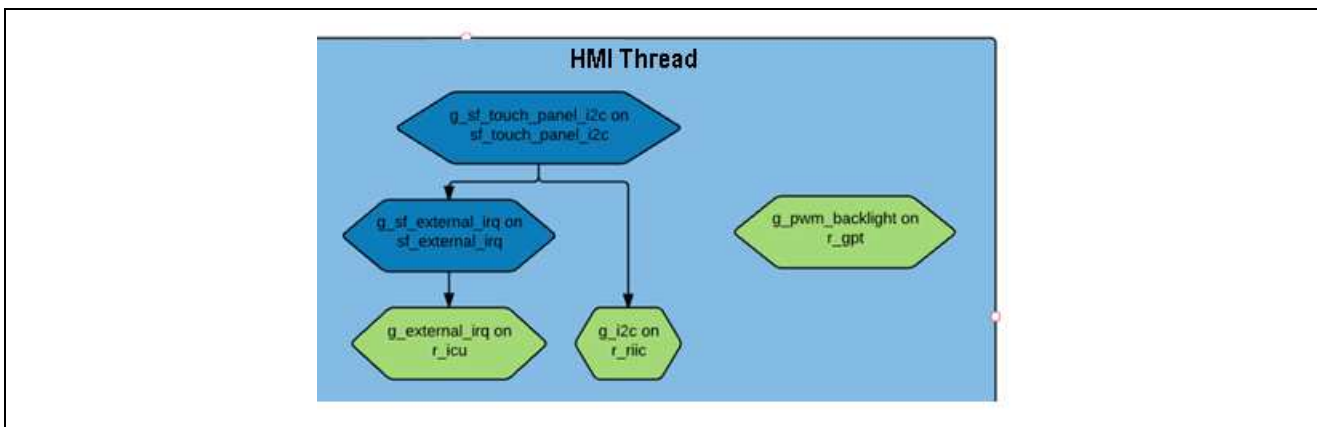


図 16 HMI スレッドモジュール表示.

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。

多くの開発ポ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。

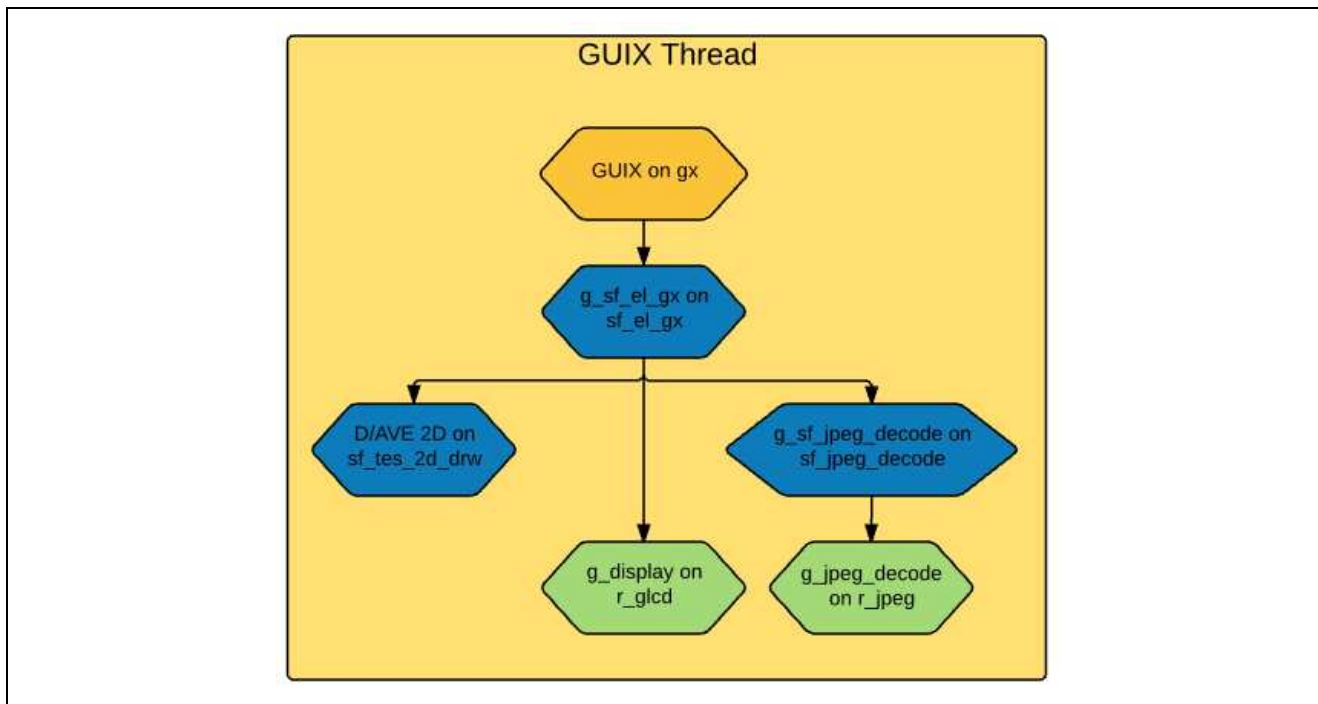


図 17 GUIX スレッドモジュール表示

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動作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、プロセッサに対する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ブを立ち上げます。e<sup>2</sup>studio 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。

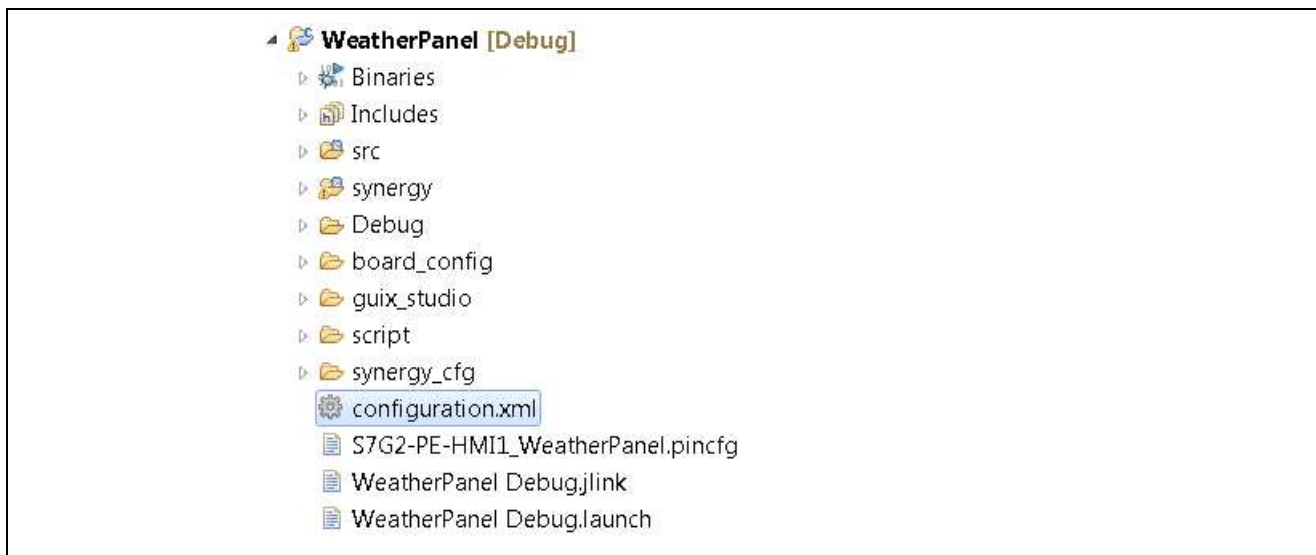


図 18 プロジェクトペイン上の[configuration.xml]

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ださい。

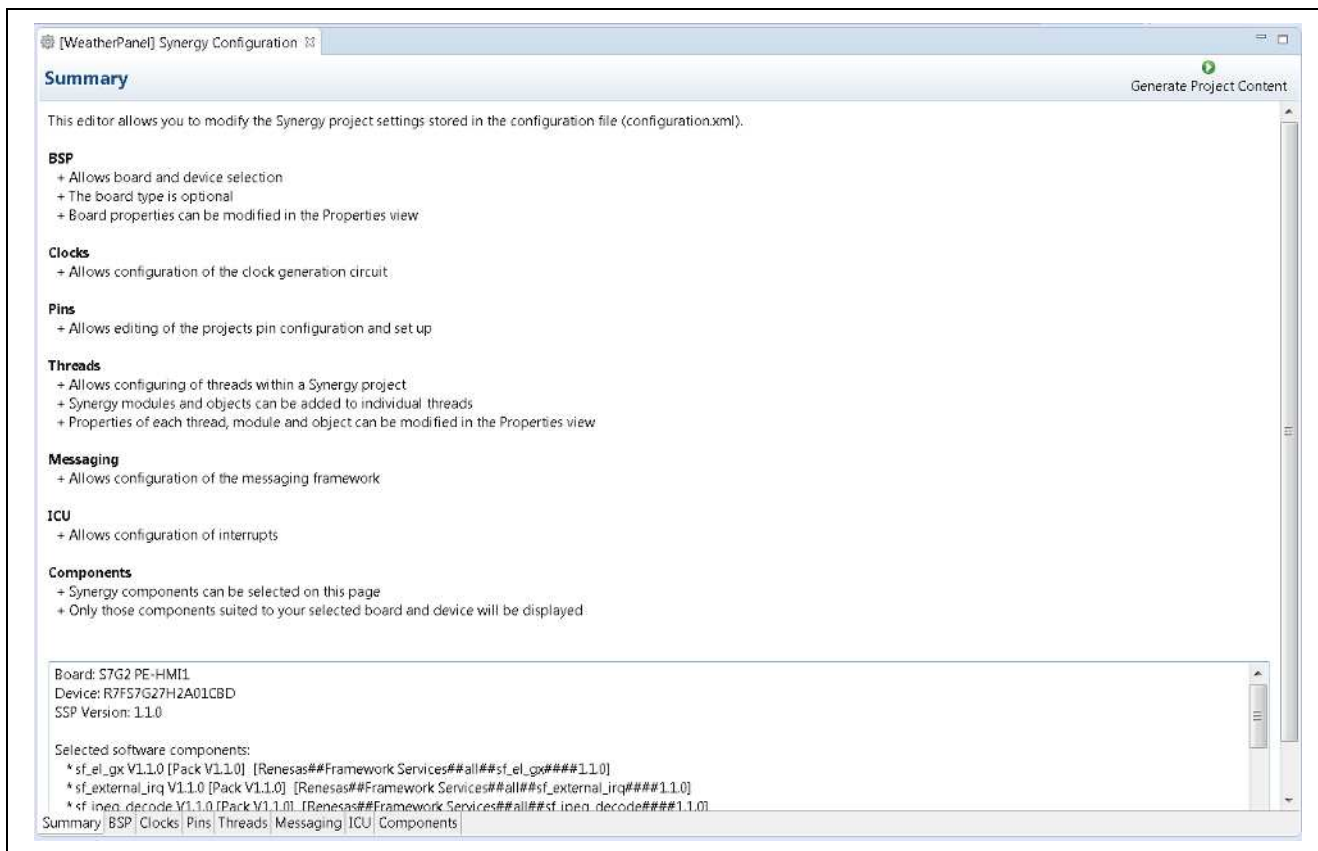


図 19 ウェザーパネルアプリケーションのスクリーンショット

## 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。

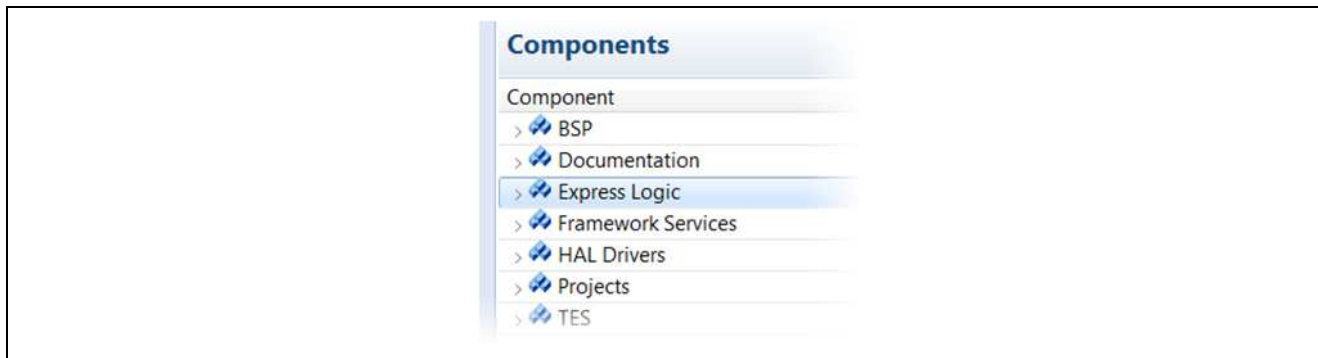


図 20 コンポーネントタブ分類


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。

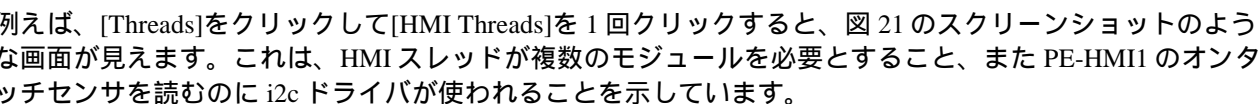
表 2 は、ウェザーパネルアプリケーションで使用している構成要素のリストです。[Components]タブの特長で、構成要素の機能および各構成要素の依存性の詳細を記述しています。例えば、sf\_message 構成要素は ThreadX を必要とします。構成要素を選択する時に、この依存性のリストにより、適切な依存する構成要素を選択しなかった場合に生じるコンパイルタイムエラーを除外することができます。

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

分類	構成要素	バージョン	改訂内容
BSP Express Logic	s7g2_pe_hmi1	1.1.0	S7G2_PE_HMI1 ボードサポートパッケージ ( B S P )
	gx	1.1.0	Express Logic GUIX:Provides=[GUIX], Requires=[ThreadX]
フレームワークサービス	tx	1.1.0	Express Logic ThreadX:Provides=[ThreadX]
	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 LCDProvides=[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]
TES	r_jpeg_decode	1.1.0	JPEG デコード: Provides=[Key Matrix]
	dave2d	1.1.0	TES Dave/2d:Provides=[Dave/2d]

## 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、 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。

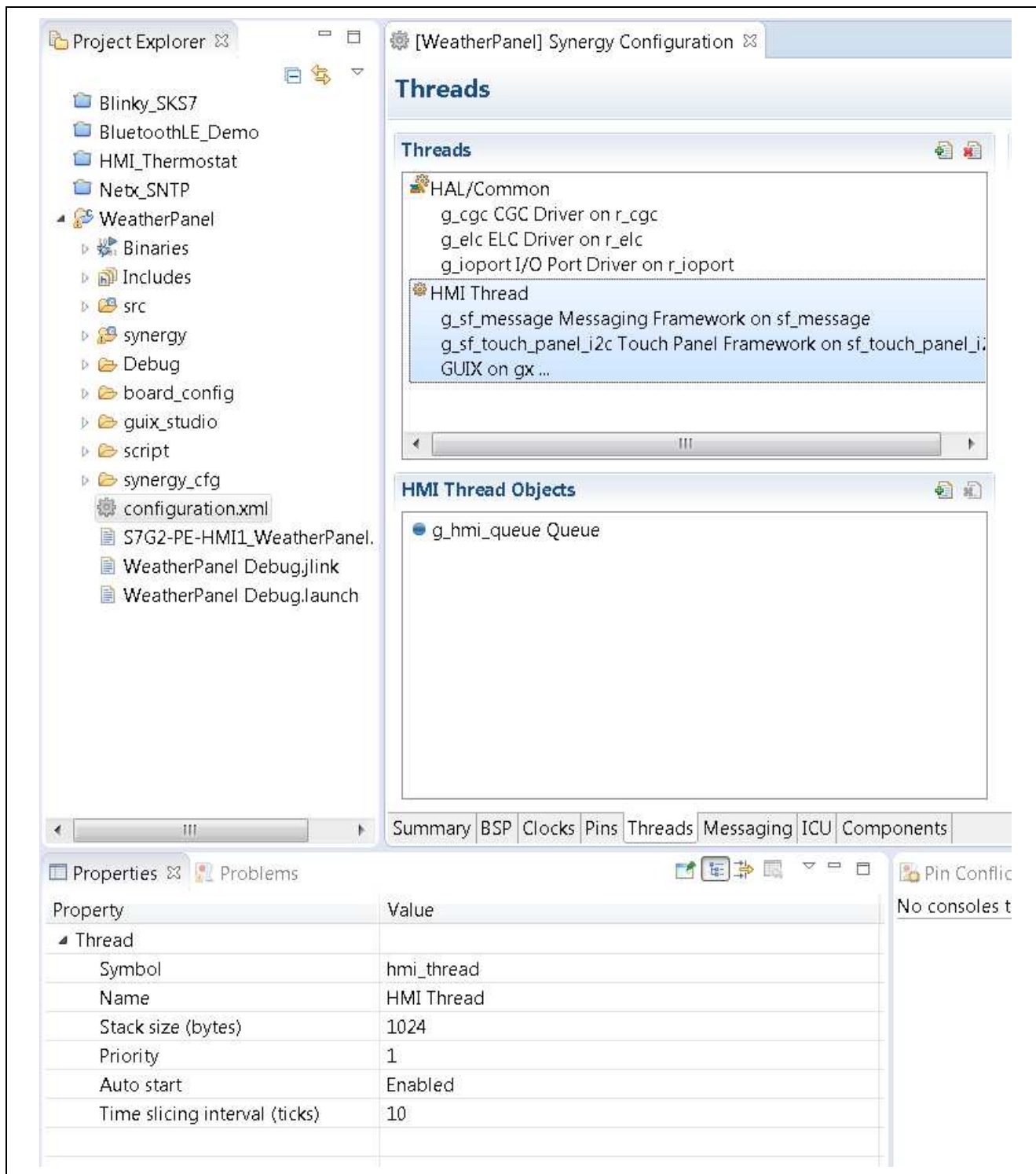


図 21 HMI スレッド プロパティ



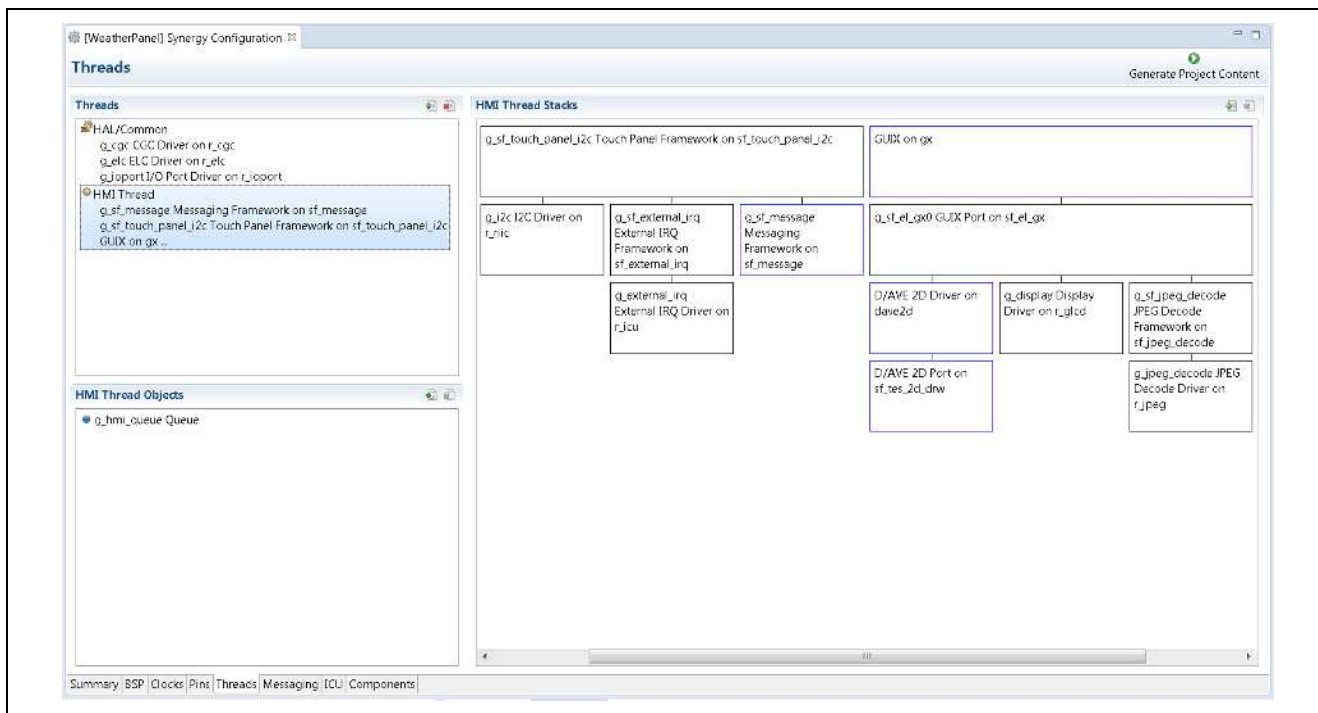


図 22 ウェザーパネルアプリケーション用 HMI スレッドモジュール

ボタンをクリックすることでモジュールがスレッドに追加されます。スレッドにモジュールを追加する前に構成要素を適切に選択している場合、エラーは発生しません。例えば、図 23 に示すとおり、**Driver Timers r\_gpt** を選択することでタイマを HMI スレッドに追加できます。

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感知した場合、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。

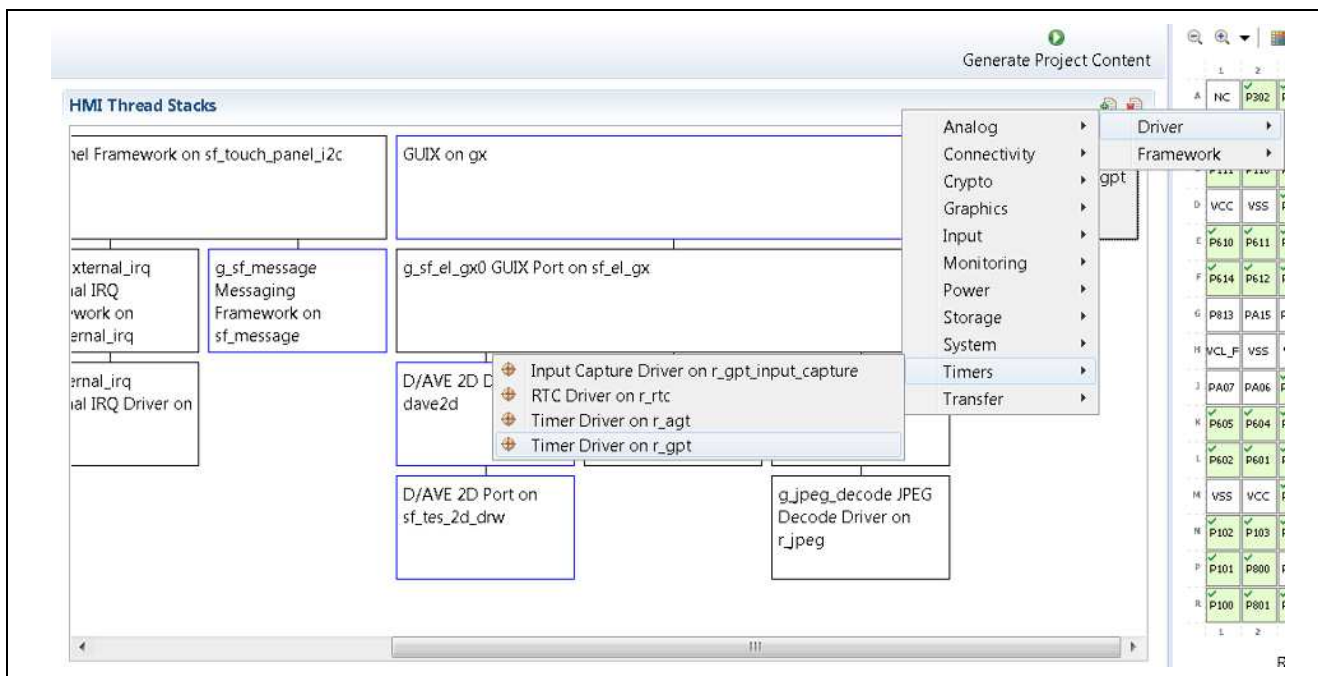




図 23 HMI スレッドへの PWM タイマの追加

### 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。

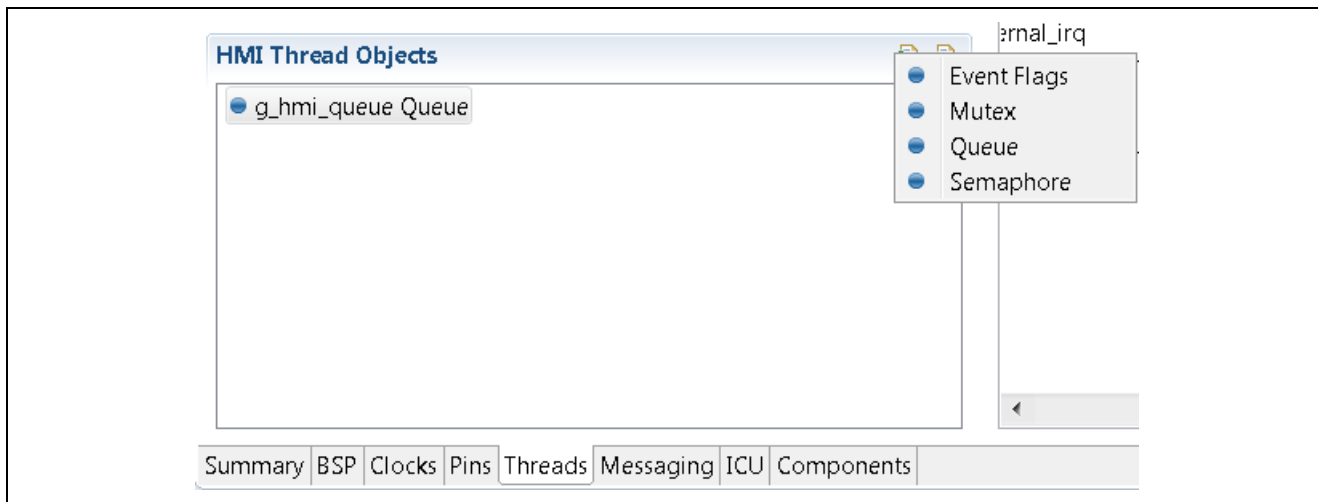


図 24 スレッドオブジェクトの追加および ThreadX が対応するオブジェクト

[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。

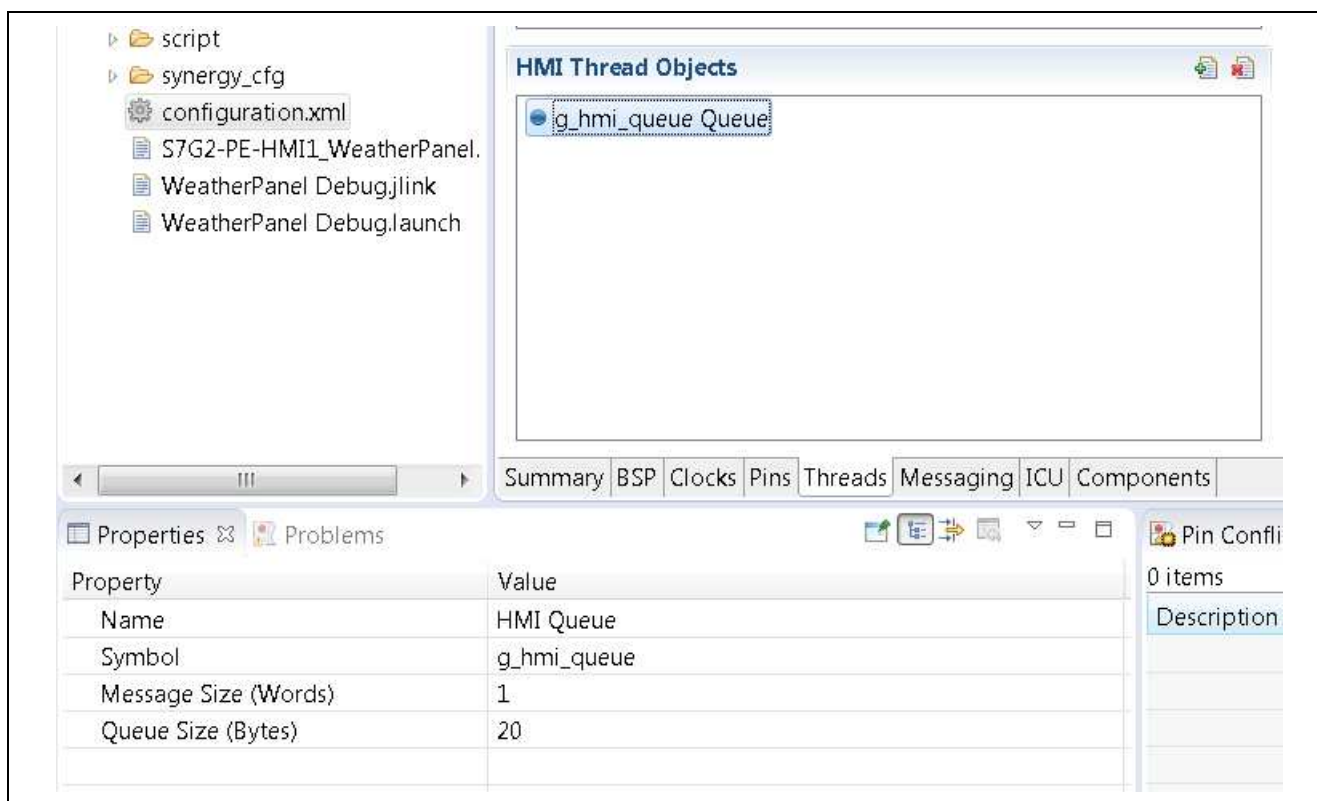


図 25 HMI スレッドオブジェクト g\_hmi キューのプロパティ

## 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きな区分があります。

- 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まれます。
- 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、GLCD 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から行います。

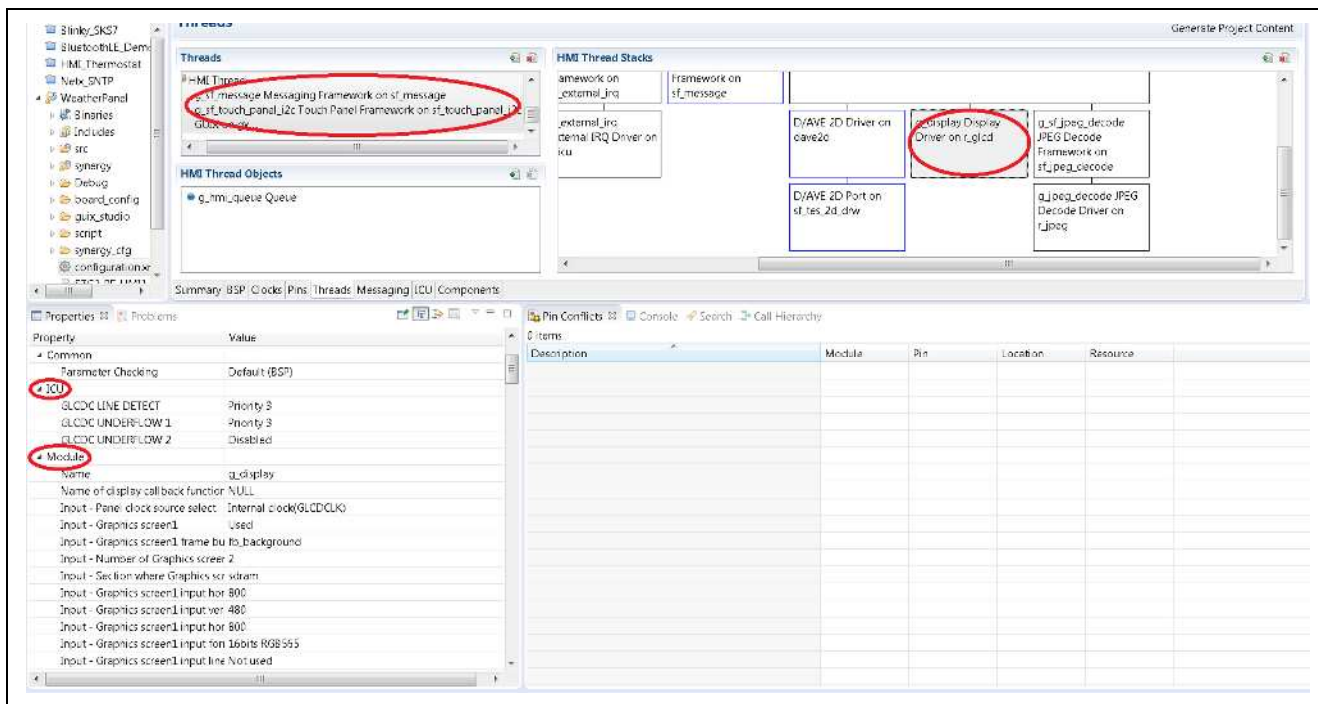


図 26 Properties を使用した GLCD プロパティコンフィギュレーション

### 5.4.2 TCON コンフィギュレーション

[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 ブ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。



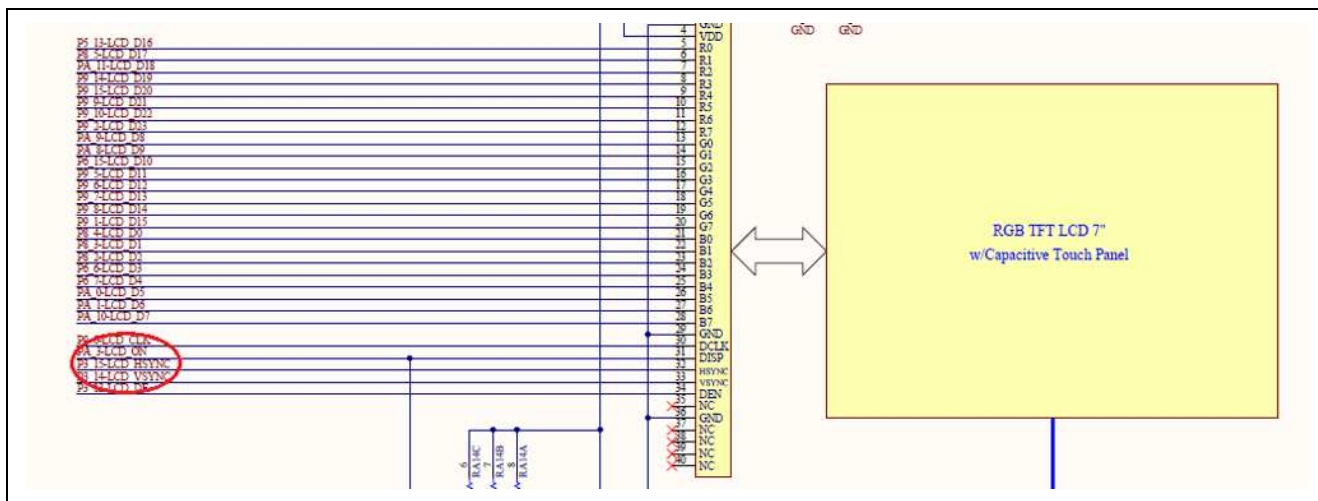


図 28 回路図からの PE-HMI1 LCD 特定信号

[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ライン端子の 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 が選択されデータライン信号を動作させることがわかります。



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。

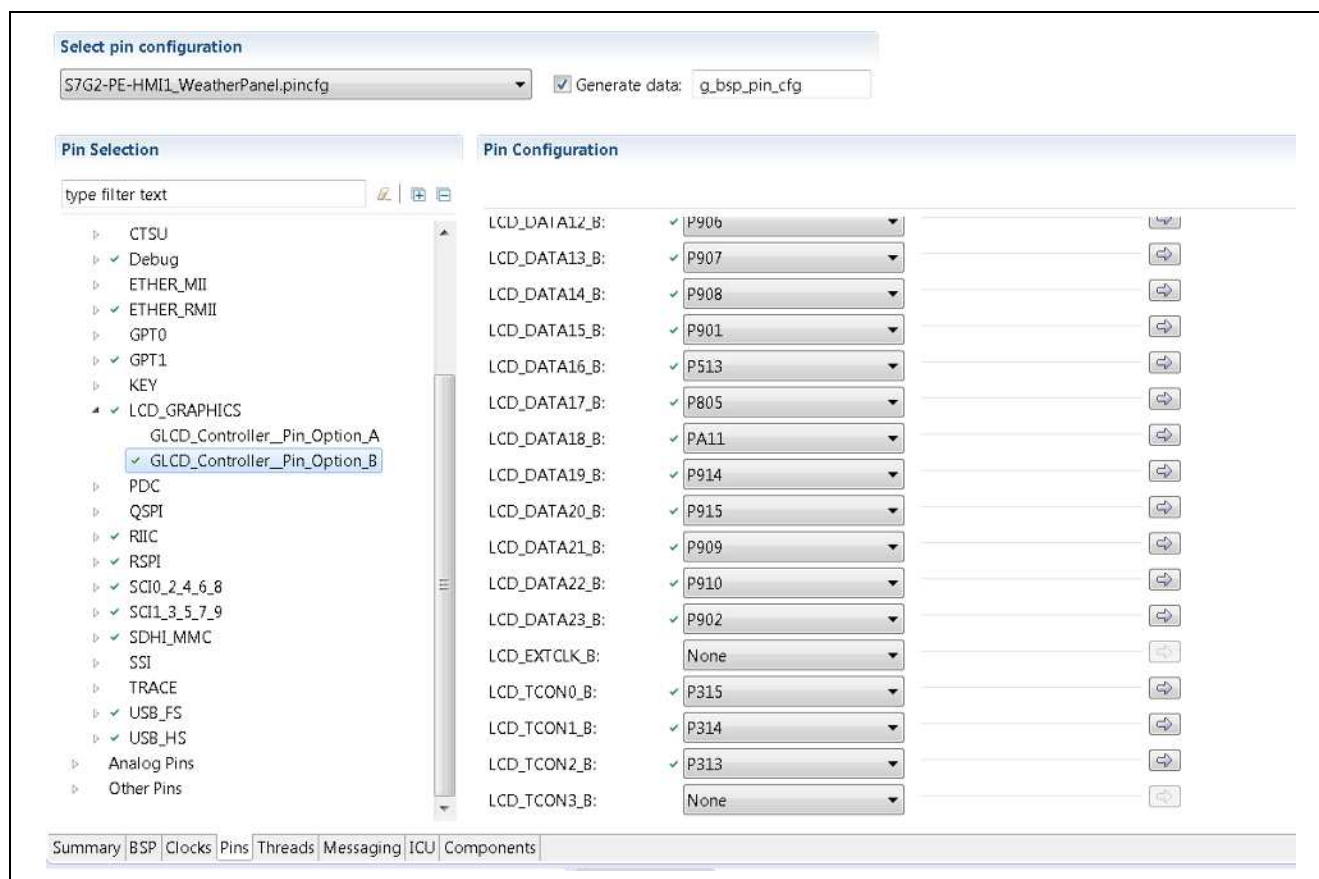


図 30 コンフィグレータを使用した場合の LCD 端子構成

例えば、LCD\_TCON0\_B 端子の横の矢印をクリックすると、端子選択画面は図 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。

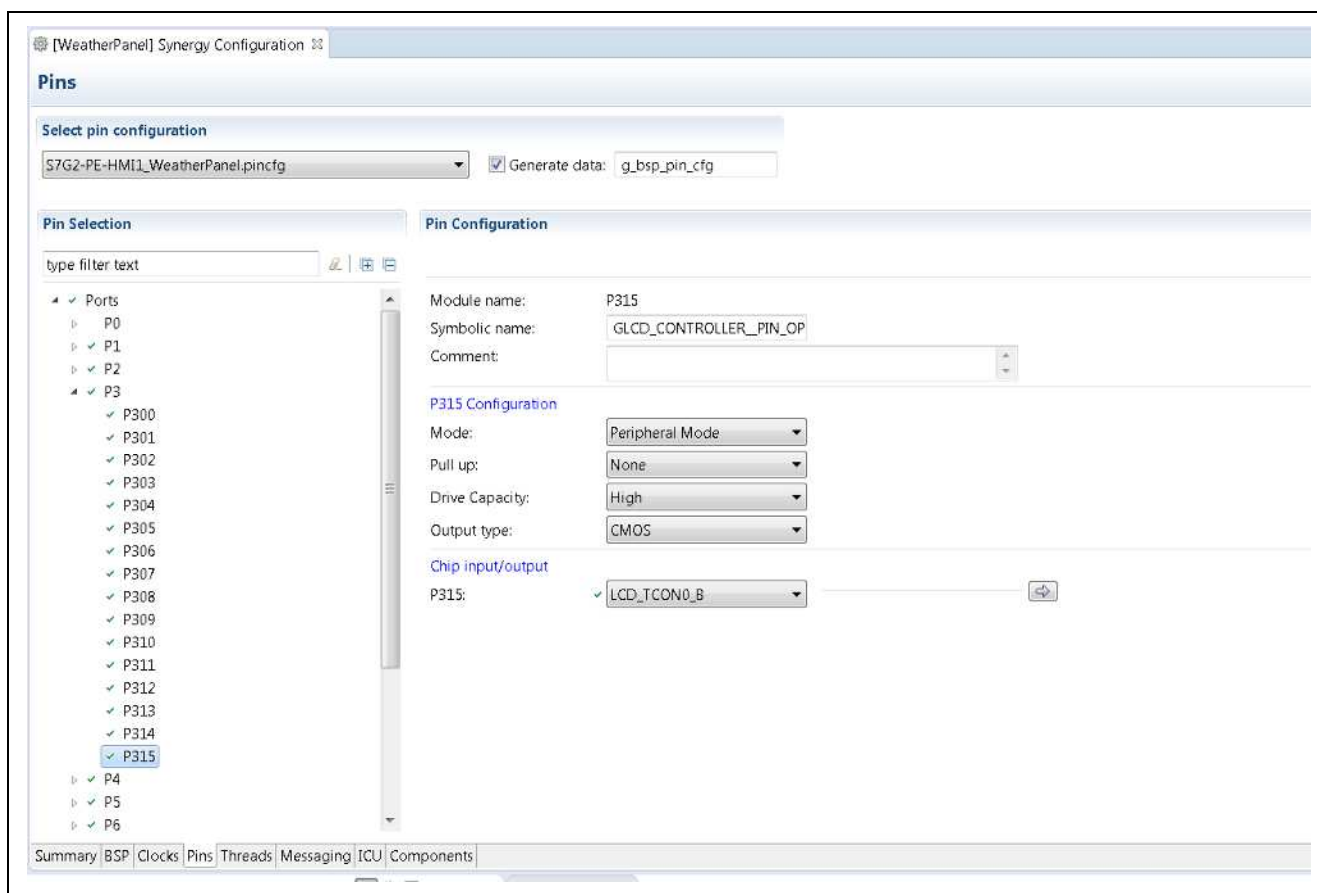


図 31 Pin Configuration 構成タブ

### 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ポートの相違点の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。このような場合には、通常、外部メモリデバイスをポートに追加します。

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。

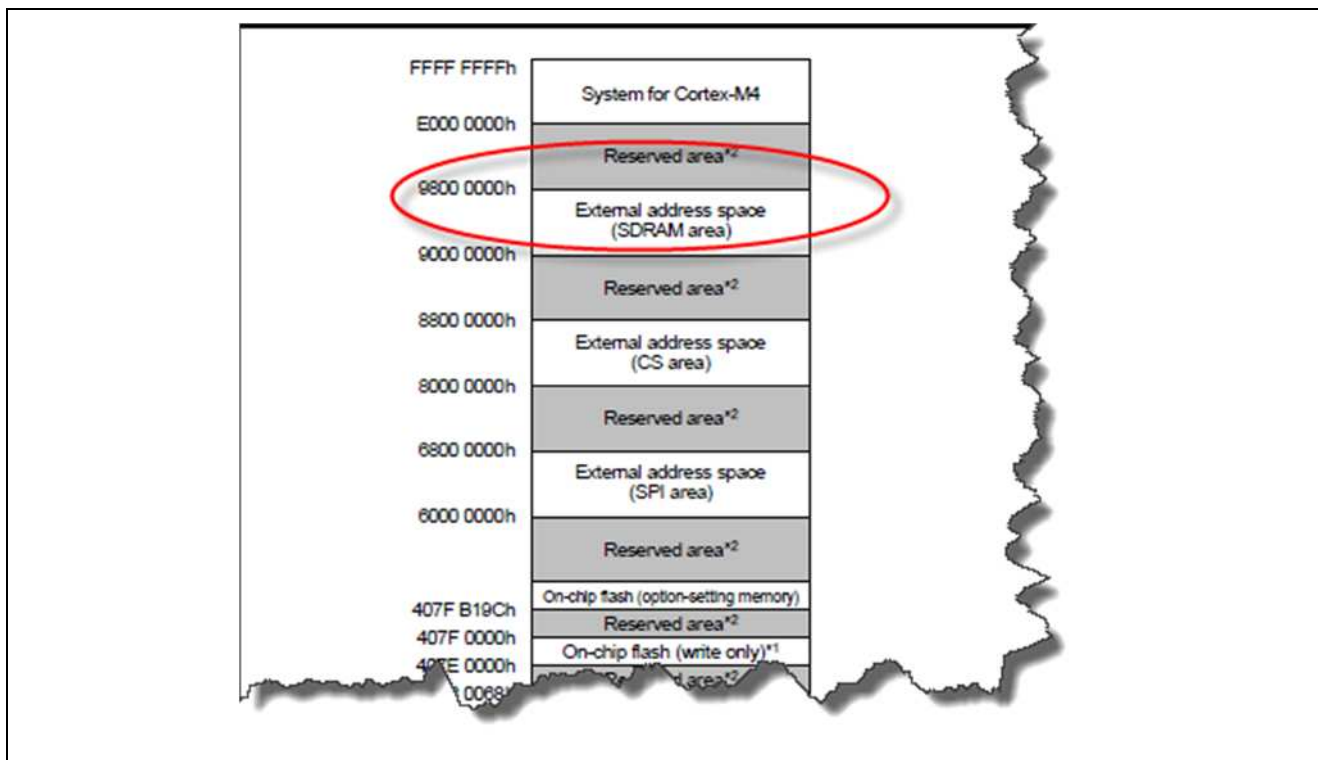


図 32 PE-HMI1 ポートの SDRAM メモリ領域

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。

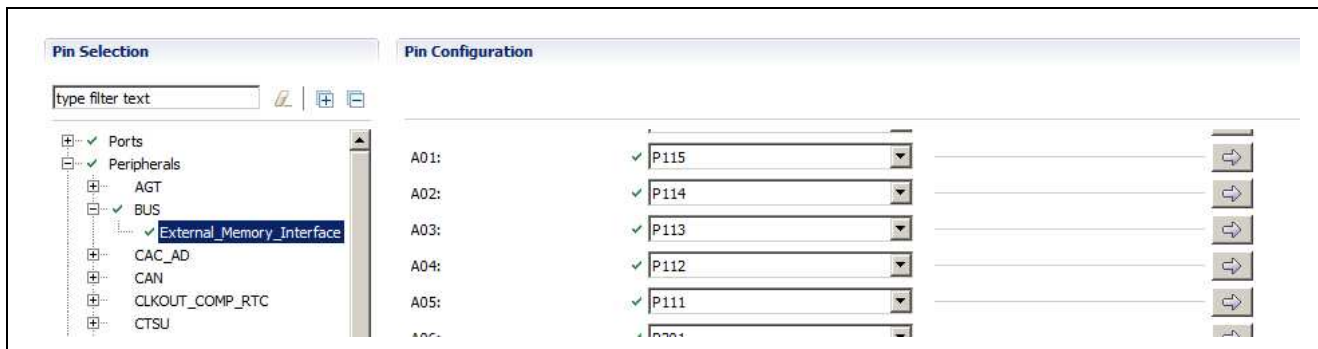


図 34 外部メモリインターフェースの設定

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。e<sup>2</sup>studio の便利な機能

e<sup>2</sup>studio 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、e<sup>2</sup>studio 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(0x90000000)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。

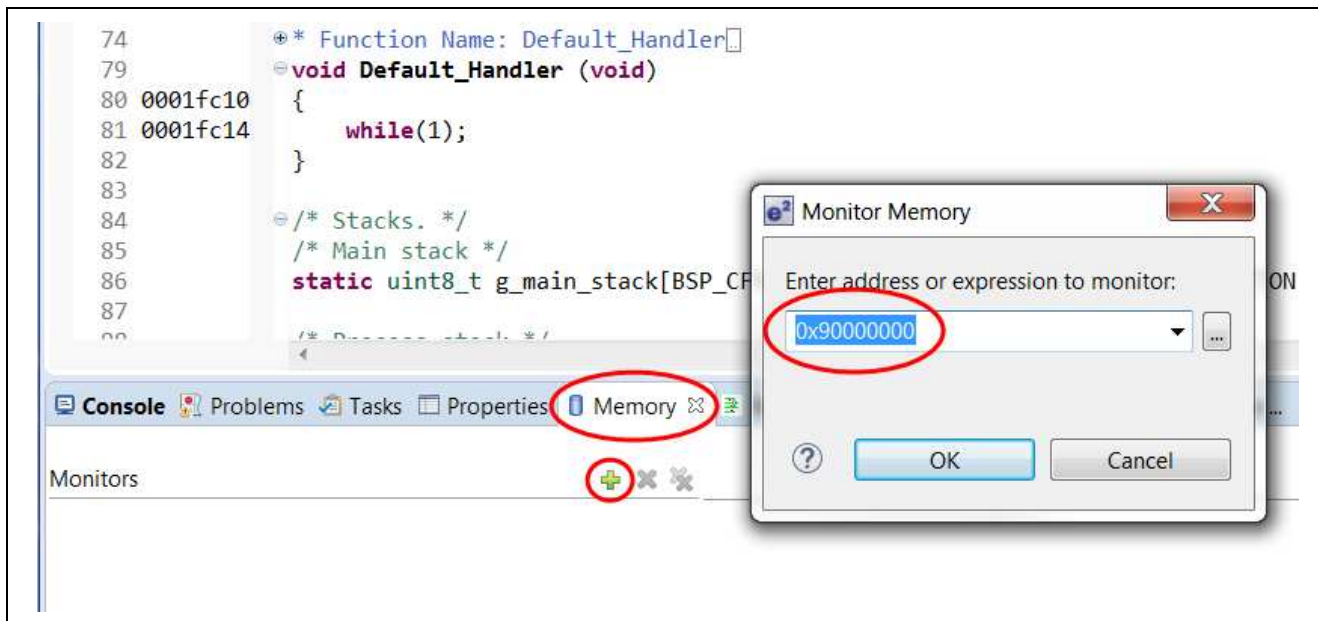


図 35 SDRAM コンテンツを表示するためのメモリモニタ

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。もちろんほと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。

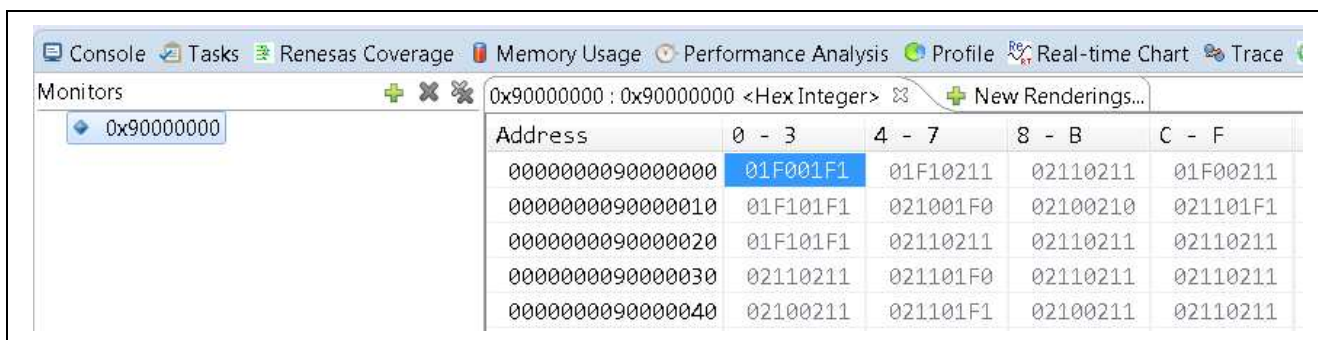


図 36 SDRAM コンテンツビュー

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。



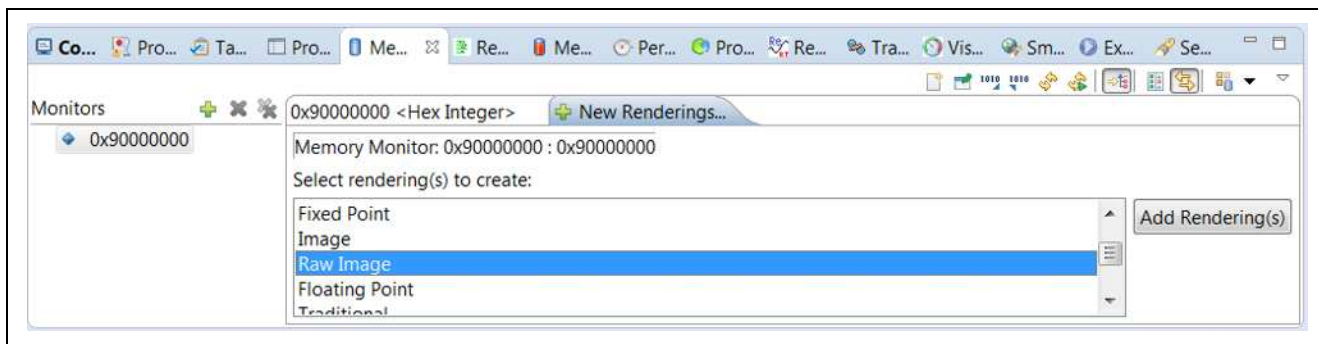


図 37 GUIX レンダリング形式選択ビュー

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。

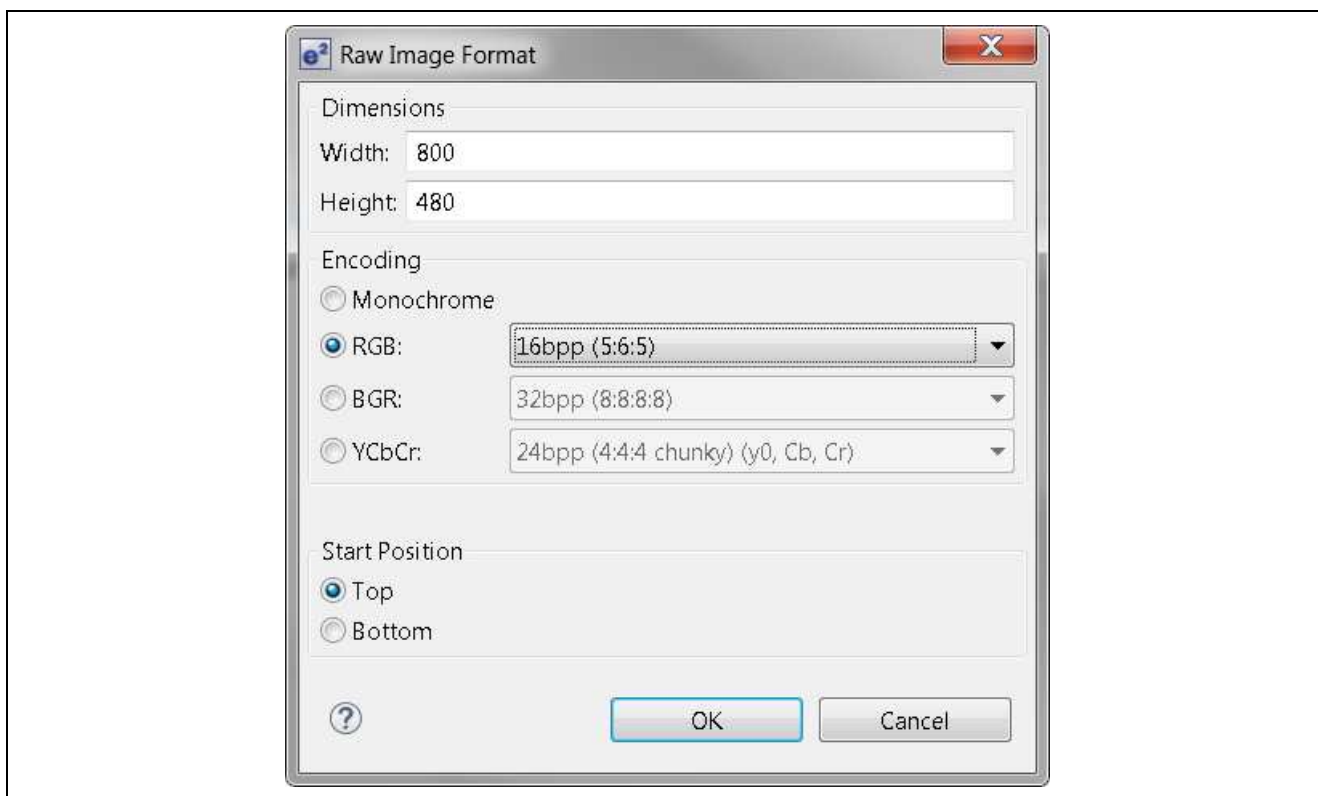


図 38 PE-HMI1 におけるウェザーアプリの Raw Image フォーマット

[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。

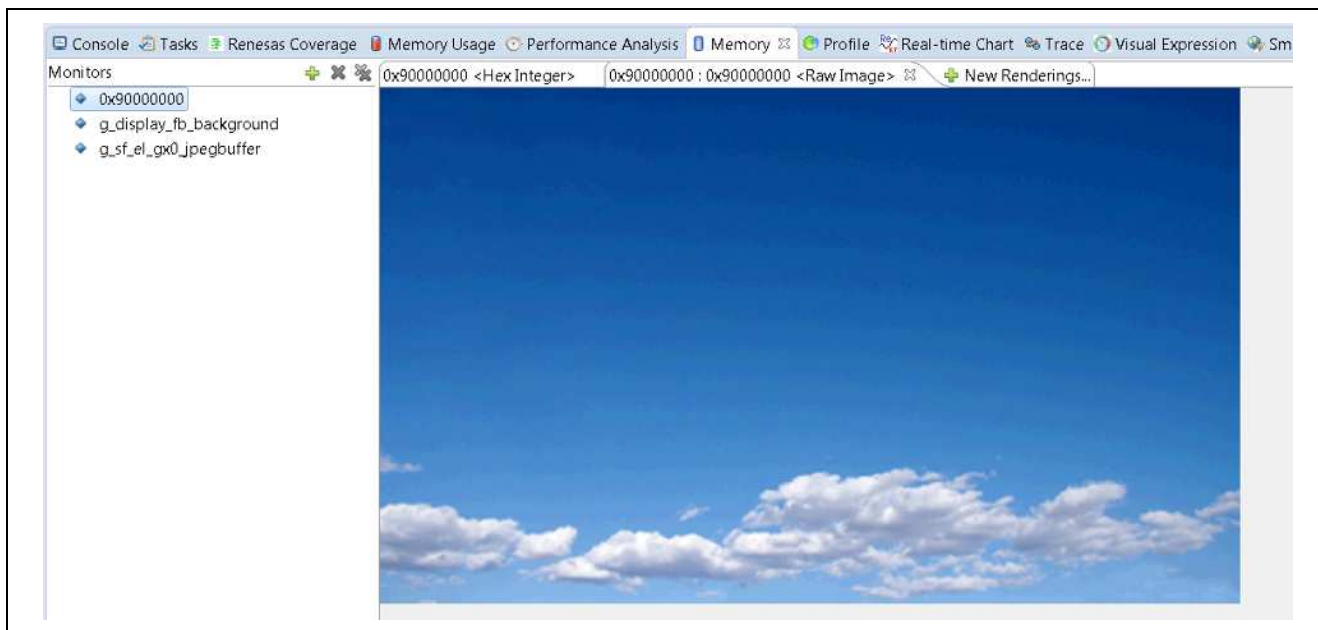


図 39 e2studio メモリモニタを使用した GUIX 実行



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。



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 = gx_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_TERMINATE 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_GAINED 7
#define GX_EVENT_FOCUS_LOST 8
#define GX_EVENT_HORIZONTAL_SCROLL 9
#define GX_EVENT_VERTICAL_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_SLIDER_VALUE 19
#define GX_EVENT_TOGGLE_ON 20
#define GX_EVENT_TOGGLE_OFF 21
#define GX_EVENT_RADIO_SELECT 22
#define GX_EVENT_RADIO_DESELECT 23
#define GX_EVENT_CLICKED 24
#define GX_EVENT_LIST_SELECT 25
#define GX_EVENT_VERTICAL_FLICK 26
#define GX_EVENT_HORIZONTAL_FLICK 28
#define GX_EVENT_MOVE 29
#define GX_EVENT_PARENT_SIZED 30
#define GX_EVENT_CLOSE_POPUP 31
#define GX_EVENT_ZOOM_IN 32
#define GX_EVENT_ZOOM_OUT 33
#define GX_EVENT_LANGUAGE_CHANGE 34
#define GX_EVENT_RESOURCE_CHANGE 35
#define GX_EVENT_ANIMATION_COMPLETE 36
#define GX_EVENT_SPRITE_COMPLETE 37
#define GX_EVENT_TEXT_EDITED 40
#define GX_EVENT_TX_TIMER 41
#define GX_EVENT_FOCUS_NEXT 42
#define GX_EVENT_FOCUS_PREVIOUS 43
#define GX_EVENT_FOCUS_GAIN_NOTIFY 44
#define GX_EVENT_SELECT 45
#define 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 */
    /*
    USHORT   gx_event_target;     /* ID of event destination */
    /*
    ULONG    gx_event_display_handle;
    union
    {
        UINT    gx_event_timer_id;
        GX_POINT gx_event_pointdata;
        GX_UBYTE gx_event_uchardata[4];
        USHORT   gx_event_ushortdata[2];
        ULONG    gx_event_ulongdata;
        GX_BYTE  gx_event_chardata[4];
        SHORT    gx_event_shortdata[2];
        INT      gx_event_intdata[2];
        LONG     gx_event_longdata;
    } gx_event_payload;
} 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。

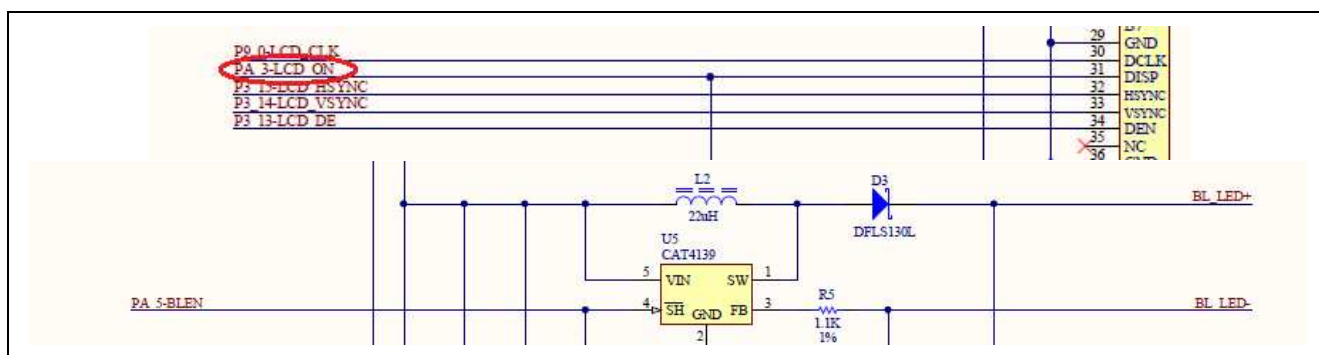


図 42 PE-HMI1 の LCD On and バックライト端子の詳細

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。

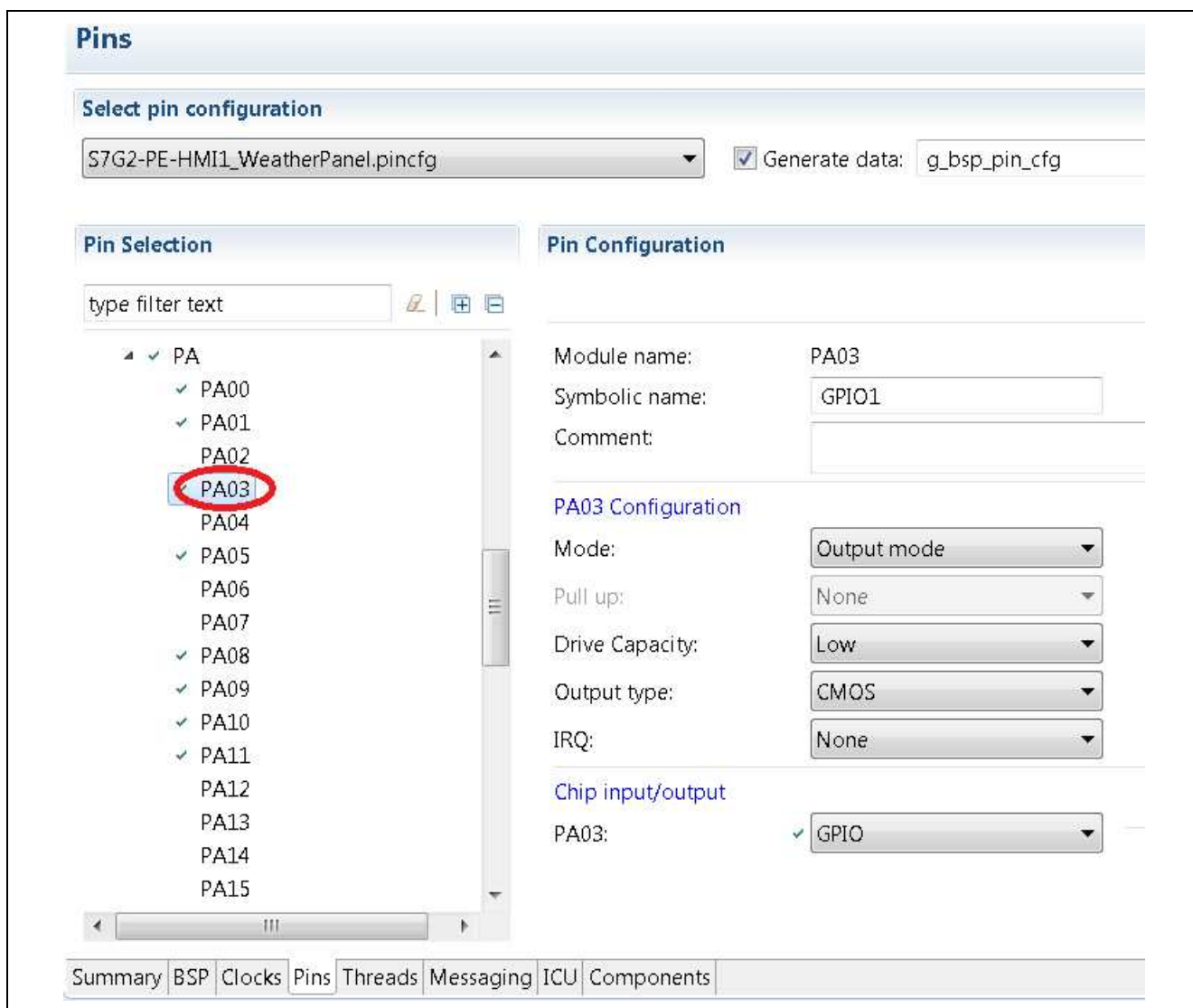


図 43 PE-HMI1 の LCD ON 端子設定

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、端子ヘ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_ioport.p_api->pinWrite()`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

e<sup>2</sup>studio 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. e<sup>2</sup>studio 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. 対象ボード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ugg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 ドキュメント(GUIX、ThreadX)
- Synergy™ Software Package (SSP) v1.1.0 ユーザーズマニュアル
- e<sup>2</sup>studio へ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\_ju0100\_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: [https://renesas.zendesk.com/anonymous\\_requests/new](https://renesas.zendesk.com/anonymous_requests/new)
- 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>

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

Rev.	発行日	改訂内容	
		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



## ご注意書き

- 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まれます。以下同じです。）に関し、当社は、一切その責任を負いません。
- 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。
- 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。
- 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ださい。かかる改造、改変、複製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。
- 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り、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。
- 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。
- 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。
- 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。
- 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は、(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。
- 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。
- 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。
- 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. 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間接に保有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.3.0-1 2016.11)



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/>