

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。

はじめに

本 Out-of-Box (OOB) プログラムは、ポテンシオメータ、光センサ、温度センサで計測したアナログ値を読み取り、それらの値を同梱の PMOD LCD 画面に表示させるマルチスレッドアプリケーションデザイン例を示すものです。ボード上の 3 つの LED は、光センサかポテンシオメータから読み込んだ値に応じて同時、もしくは順繰りに点滅します。スイッチ 1 とスイッチ 2 は、”どちらのセンサを使用するのか?”、”LED を点滅させるか追従させるか?”を選択するのに使用されます。

このアプリケーションノートは、開発キット同梱のクイックスタートガイドより詳細に DK-S124 Out-of-Box プログラムを説明します。

前提条件

本アプリケーションノートは、Renesas e²studio ISDE と SSP の使用経験がいくらかあるユーザを想定しています。本アプリケーションノートの記載事項を実行する際は、事前に、所有しているボードの Blinky プロジェクトをビルドし実行するためのクイックスタートガイドの手順を経験しておくようにしてください。それを行うことで e²studio と SSP に慣れ、ボードへのデバッグ接続が正しく機能していることが確認できます。

必要環境

サンプルアプリケーションの対象は、Renesas Synergy S124 デバイスです。アプリケーションプログラムをビルドして実行するためには、以下が必要です：

- 以下に示すように、PMOD LCD が付属している Renesas Synergy DK-S124 ボード (v3.0)
- 以下の Renesas ソフトウェアをインストールした Microsoft® Windows® 7 で動作する PC：
 - e²studio ISDE v5.3.1.002 以降
 - IAR EW for Synergy v7.7.1.1 以降
 - Synergy™ Standalone Configurator 5.3.1.002
 - Synergy Software Package (SSP) 1.2.0.

必要な Renesas ソフトウェアは、Renesas Synergy Gallery (ギャラリー) からダウンロードできます。
<http://synergycastle.renesas.com>.

e²studio へのプロジェクトのインポート、およびプロジェクトのビルド / 実行については、
[\[r11an0023eu0116-synergy-ssp-import-guide.pdf\]](#) を参照してください。

所要時間と手順

サンプルアプリケーションのインストール、ビルド、実行にはおよそ 30 分かかります。所要時間に含まれる大まかな手順は下記の通りです。

1. DK-S124 ボードを、Out-of-Box (OOB) アプリケーション用に設定。
2. プロジェクトのインポート、ビルド、デバッグ。
3. LCD 画面で、アナログセンサ値の変化を観察。
4. 2つのモーメンタリ（押し続けているときだけ ON になる）スイッチを使い、モードを変更。
5. ポテンシオメータか光センサを使い、LED の点滅周期を変化させる。

目次

1. 概要	2
2. Synergy プラットフォームの機能	3
3. OOB アプリケーションのデザイン	4
3.1 入力スレッドロジック	5
3.2 加速度センサの初期化	6
3.3 LCD スレッドロジック	6
3.4 点滅スレッドロジック	7
3.5 スレッドとモジュール	7
3.6 PMOD LCD ライブラリ	8
3.7 ソースコードレイアウト	9
4. 設計事例	11
4.1 DK-S124 を Out-of-box プログラム用に設定	11
4.2 e ² studio 技法	12
4.3 プロジェクトのインポートとビルド	14
5. 次のステップ	14
6. 参考資料	14

1. 概要

このアプリケーションノートの目的は、ユーザに S124 プロセッサの能力と、廉価な SPI LCD を使用したアプリケーション設計時の問題解決方法に慣れていただくことです。

DK-S124 ボードを使用することで、Synergy 製品ラインナップ中の最もローコストなプロセッサに習熟することができます。S1 シリーズプロセッサは、128 KB コードフラッシュメモリ 内蔵の、32-MHz ARM Cortex-M0+ CPU です。

Synergy プラットフォームはまた、ハイエンドなグラフィックツール GUIX Studio を提供します。ハイエンドプロセッサを使用したポイントアンドクリック環境で、グラフィカルユーザインターフェースを短期間で開発することが可能です。豊富なアプリケーションノートも Renesas ウェブサイトに用意されており、ハイエンドボードとプロセッサの複雑なグラフィカルインターフェースの開発の詳細が解説されています。

S124 プロセッサは、重要な情報をユーザに伝えるシンプルな Human Machine Interface (HMI) が必要なアプリケーションに最適なプロセッサになっています。開発キットは 128x128-ピクセル PMOD LCD を同梱しています。DK-S124 ボードは、この LCD を搭載するためのシングル PMOD インターフェースを実装しています。この S124 プロセッサは、SPI インターフェースを介して LCD と通信を行います。このアプリケーションノートは、PMOD LCD に遅延なく文字を表示する小型のグラフィカルライブラリを提供することを目的に作成されました。

2. Synergy プラットフォームの機能

Synergy プラットフォームは、DK-S124 OOB プログラムでデモンストレーションするアプリケーションを迅速に試作するのに必要なリソースを備えています（PMOD LCD 駆動ライブラリを除く）。[図 2.1](#)は、OOB アプリケーションが使用する S124 ハードウェア周辺回路を示します。



図 2.1 S1 OOB アプリケーションが使う S124 ハードウェア周辺回路

表 2.1 に、DK-S124 ボードで周辺回路を外部コンポーネントに接続するために使用するプロセッサの端子接続を示します。詳細は、DK-S124 回路図を参照ください。

表 2.1 OOB アプリケーションが使う周辺回路接続ポートまとめ表

構成図部位	説明	S1 ポート接続	端子機能
S1	モーメンタリスイッチ	ポート 2、端子 6	I/O ポート端子
S2	モーメンタリスイッチ	ポート 0、端子 4	I/O ポート端子
POT1	10K ポテンシオメータ	ポート 0、端子 12	アナログチャンネル 7
U11	APDS-9005 光センサ	ポート 0、端子 0	アナログチャンネル 0
U14	TMP35 温度センサ	ポート 0、端子 1	アナログチャンネル 1
U13	BMA250 3 軸加速度センサ	ポート 2、端子 6	IRQ
U13	BMA250 3 軸加速度センサ	ポート 4、端子 1	SDA0 (I2C クロック)
U13	BMA250 3 軸加速度センサ	ポート 4、端子 0	SCL0 (I2C データ)
U5	ISL41387 RS-232/RS485 トランシーバ	ポート 4、端子 10	RXD0 (受信データ)
U5	ISL41387 RS-232/RS485 トランシーバ	ポート 4、端子 11	TXD0 (送信データ)

3. OOB アプリケーションのデザイン

SK-S124 ボードは、I²C バスに接続された 3 軸加速度センサを含む複数のアナログセンサを搭載しています（ポテンシオメータ、温度センサ、光センサ）。OOB アプリケーションは、センサからのアナログ値を連続的に読み込み、PMD LCD 画面に表示します。ボード上の 3 つの LED は、光センサもしくはポテンシオメータからの電圧値に比例して点滅します。LED は同時点滅、もしくは順繰りに点滅するようにプログラム可能です。どのセンサで LED の点滅頻度を変化させるか、LED を同時点滅するか順繰りに点滅するかは、DK-S124 ボードの右下の角部にある押しボタンスイッチ S1 と S2 で制御します。

本アプリケーションノートは、マルチスレッドアプリケーションのセットアップがどれだけ容易にできるかをデモすると同時に、センサ値を読み込みのための Synergy 周辺回路ドライバの使用方法についても説明します。このアプリケーションは、センサ値を読み込み、LCD に値を表示し、そして押しボタンとセンサからの入力に基づいて LED を点滅することに対し、別々のスレッドを有しています。

3 つのユーザ定義のスレッドは、表 3.1 にまとめてあります。

表 3.1 アプリケーションスレッド名と機能まとめ表

スレッド名	スレッド機能
点滅スレッド	3 つのボード上の LED の同時点滅/順繰り点滅を制御
LCD スレッド	3 つのアナログセンサからの新しいデータが来る毎に、LCD の表示を更新します。
入力スレッド	スイッチ入力とアナログセンサからの読み取りを定期的にポーリングします。

図 3.1 に、OOB アプリケーションの概念図を示します。入力スレッドは、押しボタンの状態とアナログセンサの電圧値を定期的にサンプリングします。そして、2 つのアクセス機能を通してそのデータを他のスレッドが利用できるようにすることで、グローバル変数を不要にしています。

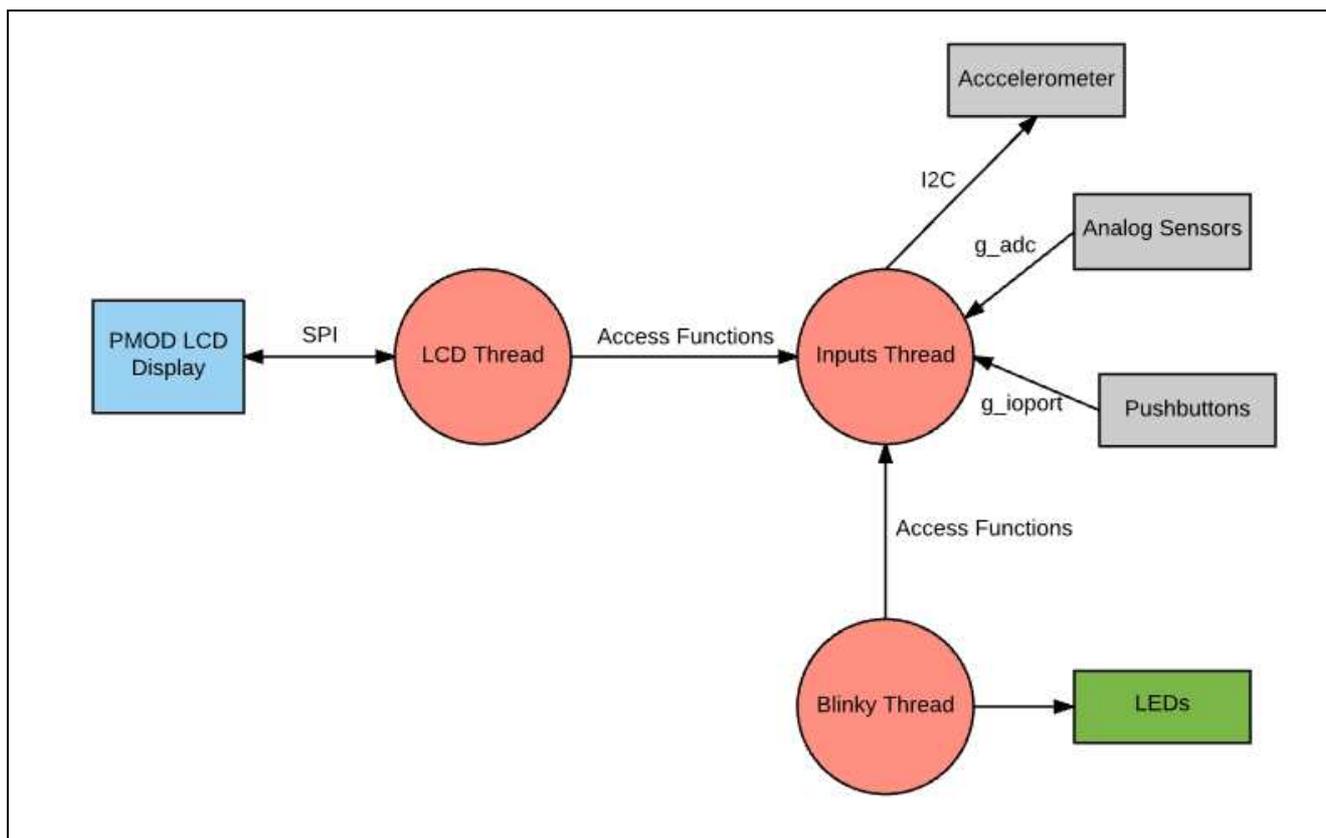


図 3.1 OOB アプリケーションの概念図

両方の押しボタンは、割込み機能を備えたハードウェア端子に接続されています。また、このアプリケーションに使用するスレッドは、スイッチに接続した I/O ポート端子を読むことで、スイッチの状態を定期的にポーリングします。このシンプルなテクニックで基本的なスイッチ・デバウシングを実現しますが、これは典型的な機械式スイッチのバウンス時間（50ms）よりも長い間隔でスイッチの状態を読み取ります。他のより複雑なデバウシング技術を使うことも可能です（例：Jack Ganssle (<http://www.ganssle.com>)が発表したデバウシングアルゴリズム）。

他のアプリケーションノート（シンプルオーディオサンプル）では、押しボタンが押されたことを検出する Synergy IRQ フレームワークの使用方法が、より重点的に説明されています。機械式スイッチを割り込みに接続することの妥当性については見解が分かれています。Synergy システムはどちらの方法も容易に利用することが可能です。

3.1 入力スレッドロジック

図 3.2 は、入力スレッドに実装されているロジックを示します。応答性を速めるため、入力スレッドはセンサとスイッチ入力を 50ms 毎にサンプリングします。このスレッドの利点の 1 つは、共有メモリスペースです。LCD スレッドは、押しボタンで選択されたカレントモードで、最新のセンサ値で定期的に PMOD LCD 画面を更新します。このアプリケーションノートの後続のバージョンでは、それらの値が変化したときに、LCD スレッドにメッセージを送る Synergy メッセージングフレームワークの使用方法をデモンストレーションします。これは LCD スレッドをイベントドリブンにします。

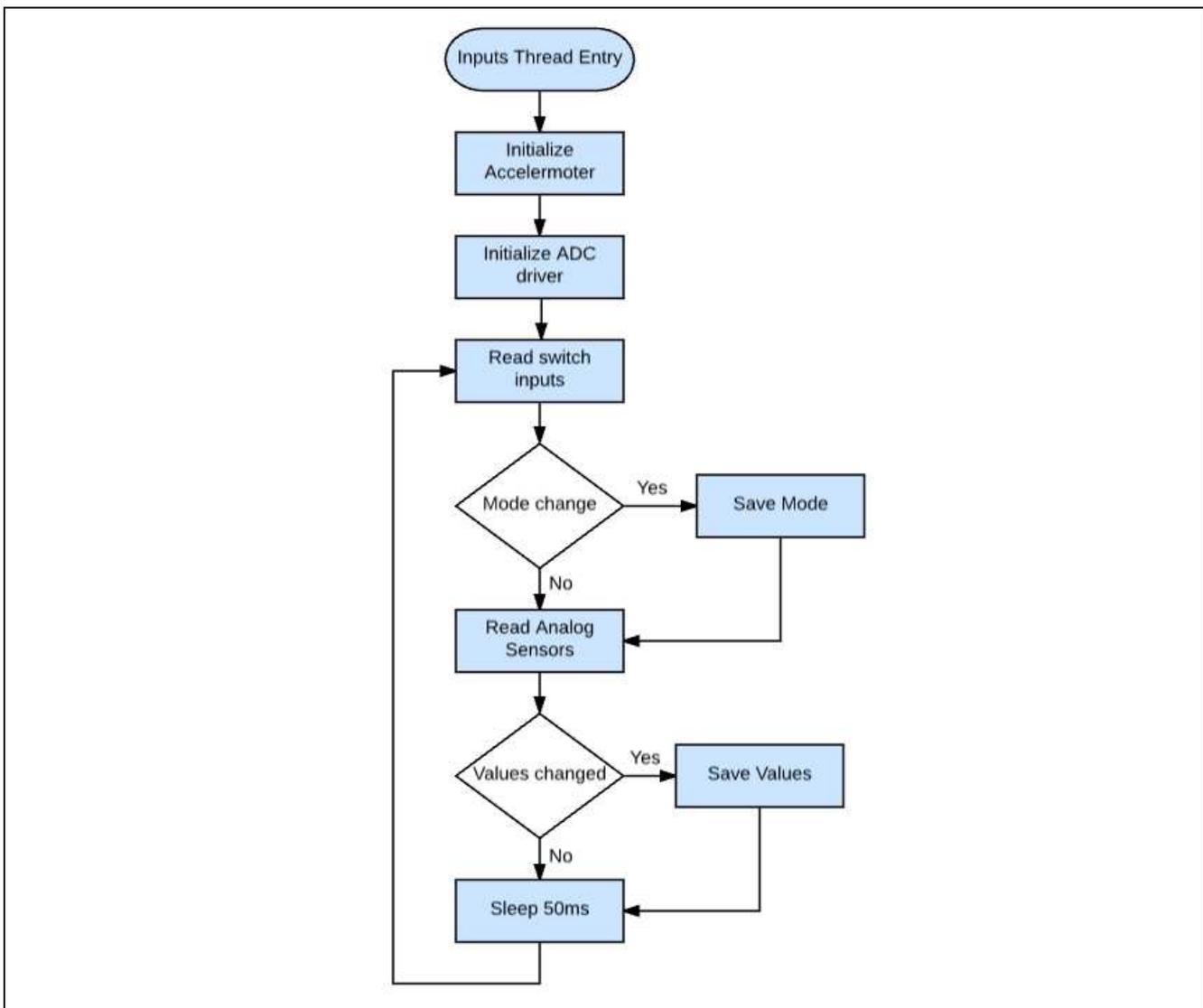


図 3.2 入力スレッドに実装されたロジックのフローチャート概要

3.2 加速度センサの初期化

SK-S124 v2.0 ボードで、加速度センサ IRQ ラインは S1 と共有されています。加速度センサ IRQ 端子はデフォルトでプッシュ/プル構成であり、アクティブハイの IRQ 状態として動作するためです。これは、S1 に接続するラインは常時プルダウンされることを意味し、このため、このラインを S1 の状態を読み取るために使用することができません。ポテンシオメータと光センサ間を選択する目的で S1 を使用するためには、加速度センサの割り込み設定を変更する必要があります。それを行うには、I²C フレームワークを入力スレッドに追加し、I²C コマンドをいくつか加速度センサに発行します。

DK-S124 v3.0 では、ジャンパ線 J20 を加速度センサ IRQ 端子と S1 を切り離すために使用しますので、上記の加速度センサ初期化を省略することができます。

3.3 LCD スレッドロジック

LCD スレッドは定期的に起動し、現状のモード、センサ値を読み込み、それらの値を PMOD LCD 画面に表示します。図 3.3 に LCD スレッドに実装されたロジックを示します。モードとセンサ値は、入力スレッドで提供される 2 つのアクセス機能呼び出すことで取り込みます。

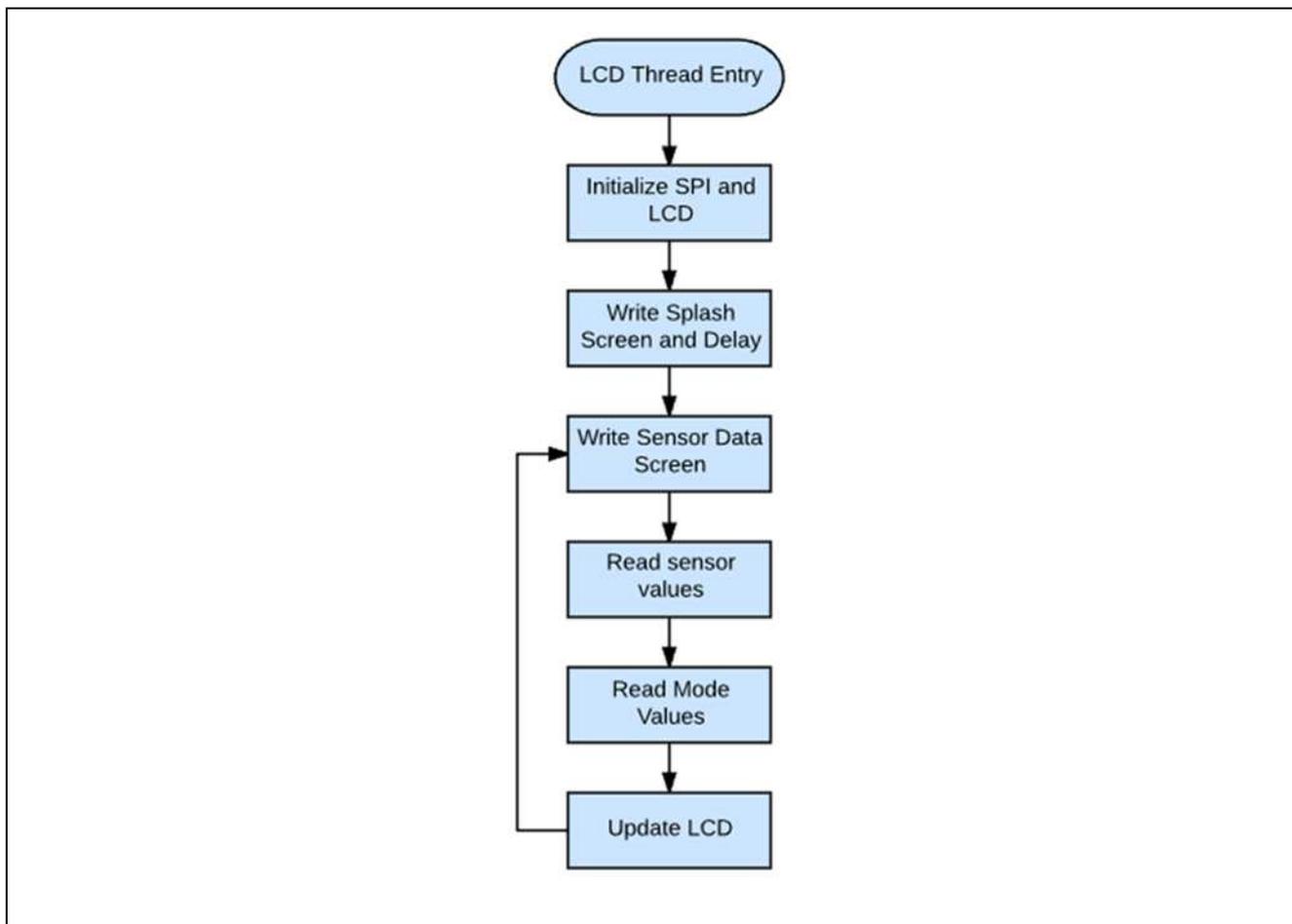


図 3.3 LCD スレッドに実装されたロジックのフローチャート概要

3.4 点滅スレッドロジック

点滅スレッドは、オペレータが選択した現状のモードに従い、光センサかポテンシオメータが示す電圧により決まる速さでLEDを定期的に取りフレッシュします。スタンダードな `tx_thread_sleep()` コールは定期的な遅延に使用されます。

点滅スレッドが起動すると、現在のセンサ値を読むために、異なる2つのアクセス機能 `[inputs_get_sensor_values()]` と `[inputs_get_mode()]` をコールします。これは、メモリスペースを共有するスレッドのメリットの一つを示します。点滅スレッドに対してそれらの変数へのアクセスをする間、入力スレッドがそれらの変数のローカルコピーを保持できるようにします。アクセス機能により、グローバル変数を使わないように、またセンサ変数が入力スレッドの内部のローカルスコープのみを持つようにできます。

3.5 スレッドとモジュール

全ての Synergy アプリケーションは、モジュールドライバがスレッドに追加され、そのプロパティは ISDE コンフィグレータを使って設定されます。e²studio のプロジェクトエクスプローラ内部から `[configuration.xml]` をダブルクリックして、コンフィグレータを起動します。このアプリケーションノートは、アプリケーションに追加されたドライバモジュールのすべては網羅していませんが、例として、PMOD LCD との通信に必要な、SPI ドライバの追加について説明します。他のすべてのドライバについても、ISDE を介して同様に確認できます。Synergy モジュールとドライバに関するより詳細な情報については、SSP ユーザーズマニュアルを参照してください。

前述の通り S124 プロセッサは、SPI バスを介して PMOD LCD と通信します。このため SPI ドライバを、ディスプレイと通信するスレッドに追加する必要があります。このケースでは LCD スレッドになります。図 3.4 に、ISDE コンフィグレータで LCD スレッドがハイライトされたスクリーンショットを示します。r_sci_spi ドライバが追加されているのが確認できます。

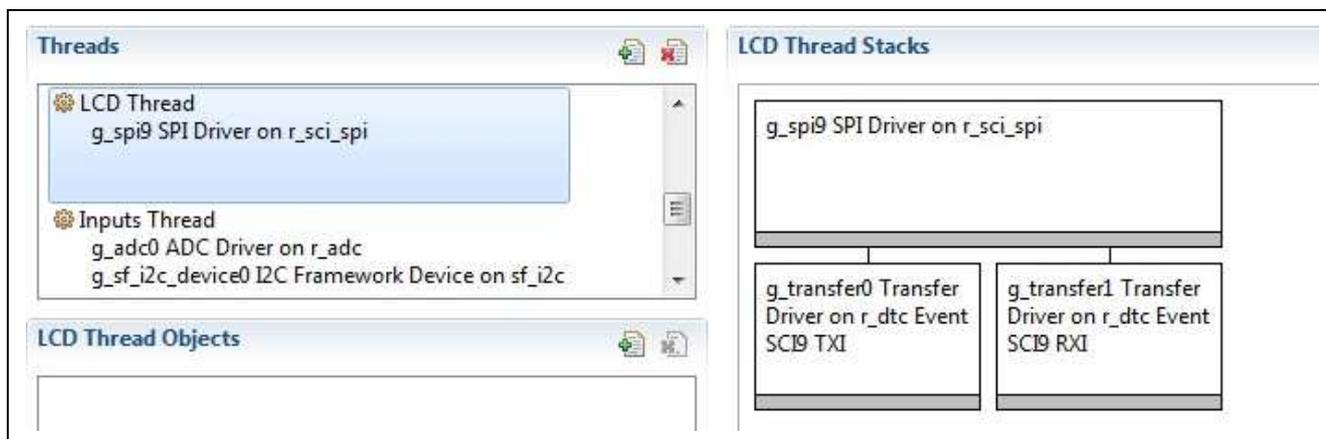


図 3.4 プログラムのスレッドタブ表示

Synergy プロセッサは、SPI 通信用に 2 つの周辺回路を備えています。SPI 専用回路と Serial Communication Interface (SCI) で、後者はシンプル I²C、シンプル SPI、もしくは UART (同期もしくは非同期) のいずれかとして機能します。SSP は、両方の周辺回路に対応しています。同様の考えは、I²C 周辺回路にも当てはまります。

どちらの周辺回路を使うかは、多くの場合システムの複雑さに基づいて選択されます。多くの場合、SPI や I²C 専用周辺回路の全ての機能は必要ありません。場合によっては、I²C 専用周辺回路に割り当てた端子を他の用途に使う必要があります。どちらの周辺回路を使うかは、多くの場合ハードウェア設計段階で決まります。プロセッサの端子が、実際にどちらの周辺回路に接続されるかに影響するためです。

システムを設計する際に考慮すべき事項の 1 つは、各タイプの周辺回路の数です。S124 プロセッサの SPI と I²C の専用周辺回路は 2 つですが、シンプル SPI とシンプル I²C に設定可能な SCI 周辺回路は 9 つあります。

本 OOB アプリケーションは、それぞれの方法の事例を示しています。PMOD LCD と通信するために使う SPI 周辺回路は、9 つの SCI チャンネルの一つに接続しています。他方、加速度センサは 2 つある専用 I²C バスの 1 つに接続しています。

3.6 PMOD LCD ライブラリ

PMOD LCD は 128x128 ピクセルの LCD で、業界標準の ST7735 グラフィックコントローラを使用しています。同じグラフィックコントローラを使用した他の同様なディスプレイと通信するため、各ベンダからオープンソースライブラリが入手可能です。SPI ベースの LCD を使う際の制約の 1 つが、SPI 書き込みの各ピクセルへのアドレッシング性能が遅いことです。多くのシステムで SPI 書き込み時にレイテンシが起きます。また、他のコントロールラインに書き込む際には、ディスプレイとのやり取りに幾分かのレイテンシが必要です。画面全体を描画するのに必要な全データを書き込むとき、もしくはテキストラインを出すとき、これらレイテンシははっきりわかるレベルまで積みあがります。

DK-S124 OOB に含まれる PMOD LCD ライブラリは、書き込みをバッファリングし、それらを一度にバースト出力することで性能を向上させています。これにより、SPI 書き込み動作起因のレイテンシの多くをなくしています。この種のバッファリングを行う際には、それを行うためのバッファスペースを割り当てる必要があります。現在のところ、PMOD LCD ライブラリは、128 バイトだけ消費する 2 次元アレーを文字バッファリング用に割り当てています。

```
static uint16_t lcd_pmod_buffer[2][64];
```

このアプリケーションノートの PMOD LCD ライブラリ用 API コールを [図 3.5](#) に示します (執筆時点)。このアプリケーションノートで使用する API コールは、R_SPILCD_Clear と R_SPILCD_DrawText のみです。PMOD LCD ライブラリは開発が続けられていますので、Renesas ウェブサイトで更新の確認をしてください。

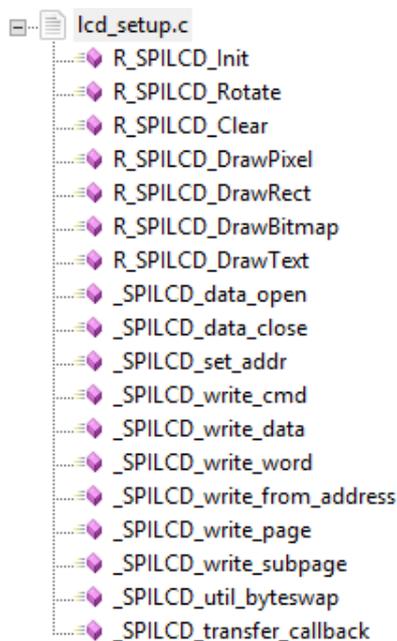


図 3.5 PMOD LCD ライブラリのための API コール

PMOD LCD ライブラリの使用は簡単です。以下のコードを使用するして、電源投入時になるシンプルフラッシュ画面を表示してください。

```
/** Initialize the PMOD LCD */  
R_SPILCD_Init((spi_instance_t * const) &g_spi9, LCD_ROTATION_90);  
R_SPILCD_Clear(LCD_COL_BLACK);  
  
R_SPILCD_DrawText("DK-S124", -1, TEXT_COLOR_FG, TEXT_COLOR_BG, CURSOR_X(7), CURSOR_Y(5));  
R_SPILCD_DrawText("V 1.0", -1, TEXT_COLOR_FG, TEXT_COLOR_BG, CURSOR_X(7), CURSOR_Y(8));
```

多くのアプリケーションでは、SPI ドライバにアクセスする前に、Synergy SPI ドライバに対して open コールをコールします。これは通常、スレッドループに入る前に、スレッドエントリコール内部で行われます。しかしながらこの場合は、LCD を初期化するために R_SPILCD_Init() ルーチンを呼び出したときに、PMOD LCD ライブラリがこれを行います。上記コードで分かるように、ポインタを R_SPILCD_Init() コールへの SPI ドライバインスタンスに渡すだけです。このインスタンスポインタは、SPI ドライバを LCD スレッドに加えたときに自動的に作成されます。

3.7 ソースコードレイアウト

この章は、e²studio の元でプロジェクトをビルドする際に想定されるソースコードレイアウトの概要を示します。

Renesas Synergy アプリケーションプログラムのコードをすでに書いた経験のあるユーザは、e²studio が通常プロジェクトをどのようにまとめるかを理解することができます。その場合、次の章を省略することが可能です。

DK-S124 のプロジェクトを e²studio にインポートしてビルドした後、ディレクトリ構造は [図 3.6](#) のようになります。プロジェクトを初めてインポートしたときは、[synergy_gen]、[synergy]、[debug]、

[synergy]、および [synergy_cfg] フォルダは存在しません。これらのフォルダはプロジェクトをコンパイルしたときに、フレームワークにより自動的にビルドされます。このため、プログラムを配布する際はサイズを小さくするため、これらのサブフォルダは含めません。通常、プロジェクト内で使う必要のあるファイルを緑色でハイライトして示します。

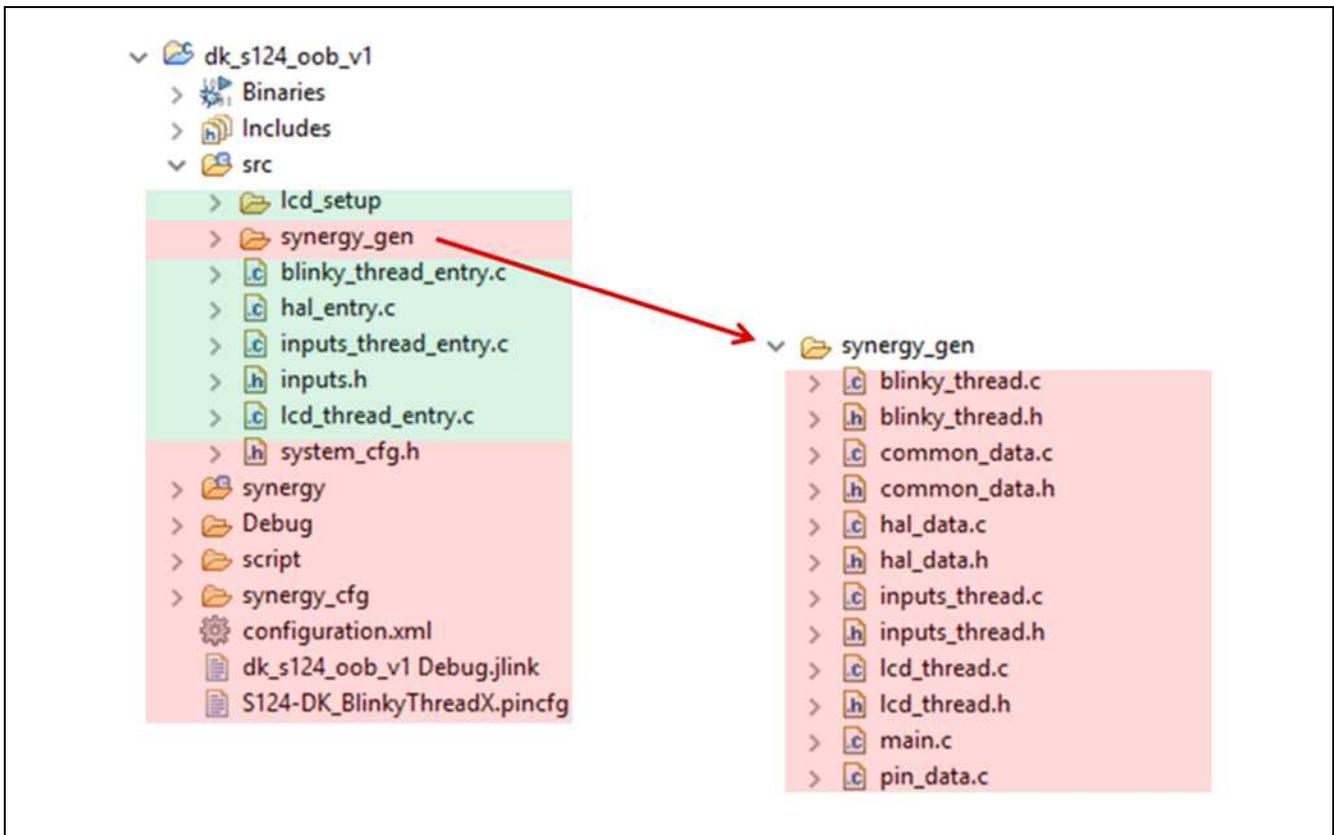


図 3.6 DK-S124 OOB v1.0 プログラムのソースコード配置

4. 設計事例

DK-S124 には、out-of-box アプリケーションソフトウェアがあらかじめプリインストールされています。プログラムを動かしたり、e²studio ISDE からボードに書き込みをする前に、以下の章に従いボードをまず設定する必要があります。

4.1 DK-S124 を Out-of-box プログラム用に設定

DK-S124 を設定する手順を以下に示します。

1. 図 4.1 に示すように、バッテリーのそばのジャンパ J3 が 2 つのジャンパ線で 1-3 と 2-4 を接続するようになっていることを確認してください。
2. マイクロ USB ケーブルを使い、DK-S124 の J18 にある JLink-OB を PC と接続してください。(図 4.1)

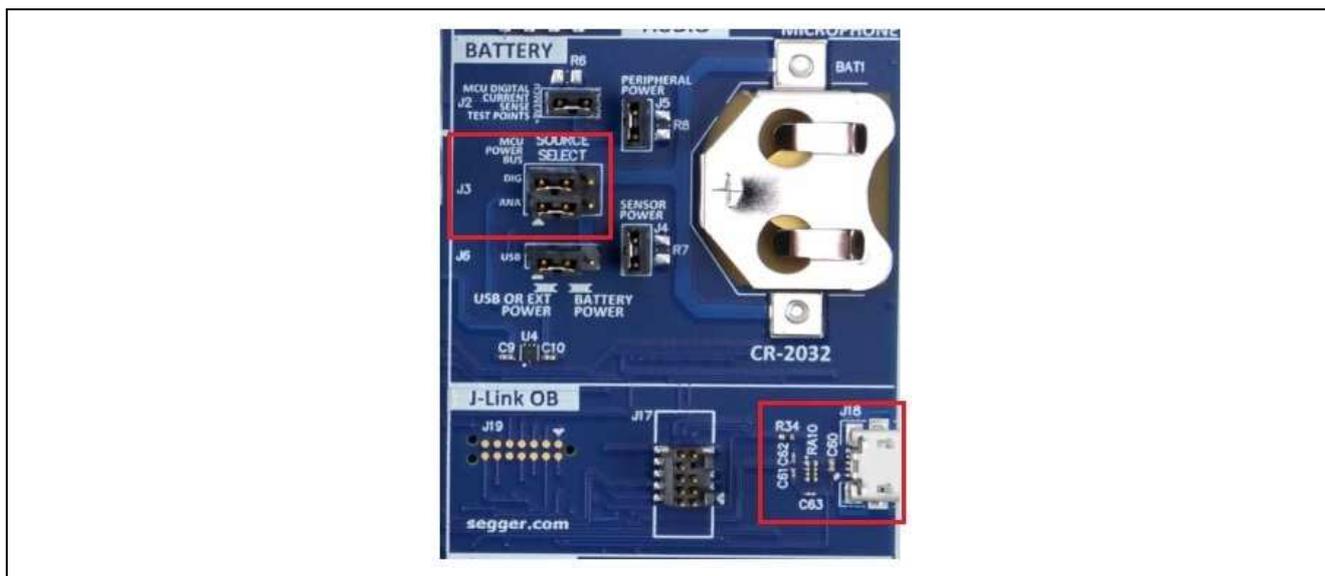


図 4.1 電源設定（上側）と-Link OB 接続（下側）

DK-S124 が接続されると、電源が入り、直ちに 3 つの LED（LED1、LED2、LED3）が点滅し始めます。PMOD LCD のシンプルフラッシュ画面を 2 秒間表示します。このフラッシュ画面は、ボード番号と、ボードと一緒に出荷された out-of-box ソフトウェアのバージョンを表示します。(図 4.2)



図 4.2 OOB フラッシュ画面

2 秒後、画面はアナログ測定画面に替わります(図 4.3)。SSP Out-Of-Box アプリケーションは、ポテンシオメータ (POT1)、光センサ U11 (APDS-9005)、温度センサ U14 (TMP 35) に現れた電圧値を読み込むために A/D コンバータを使います。プログラムは、光センサとポテンシオメータから読み込んだ 16 進数データと、温度センサから読み込んだアナログ値を華氏温度相当に変換した値を表示します。



図 4.3 OOB アナログセンサ画面

Out-of-Box プログラムは、以下のような方法で使用することができます。

1. 3つのLED (LED1、LED2、LED3)を同時点滅、もしくは順繰りに点滅するようにプログラムできます。モーメンタリ押しボタンスイッチ (押ししている間のみ有効) S2を押して、LEDの点滅方法を同時点滅、順繰り点滅の間で切り替えることができます。画面の左下でLEDがどちらのモード (同時点滅/順繰り点滅) で動作しているかを表示します。
2. LEDの点滅速度は、光センサかポテンシオメータからの読み取り値で決まります。モーメンタリ押しボタンスイッチ (押ししている間のみ有効) S1を押して、それら2つを切り替えることができます。画面の右下に、LEDの点滅速度がどちらのセンサ (ポテンシオメータ POT/光センサ Light) によって変化しているかが表示されます。
3. POTモードを選択して (S1を押す)、POT1を時計回りと反時計回りに回してみてください。LEDの点滅は適宜増減します。画面は、A/Dコンバータからの直接データを表示します。
4. Lightモードを選択して (S1を押す)、光センサ U11に懐中電灯のような光源を近づけたり離したり動かしてみてください。LEDの点滅間隔が変化するのを観察してください。画面に表示される16進の直接データ値も変化します。

4.2 e²studio 技法

アプリケーションにスレッドを追加すると、デフォルトで1KBのスタックスペースを取ります。未使用のスタックスペースを減らすと、より多くのメモリ領域をアプリケーションプログラムに使用できますが、どの程度のスタックスペースがあれば充分かを確かめるのは容易ではありません。

ThreadXは全部のスタックを、既知の値0xEFで初期化します。またこの値は、e²studioにあるメモリモニターを使って、各スレッドスタック検査を容易にします。まず初めに、物理メモリのどこにスタックがあるかを確認する必要がありますので、以下の2つの方法のうちいずれかで行ってください。1つ目は、[図 4.4](#)に示すように、アプリケーションをコンパイルしたときに作成されるマップファイルを見ることです。

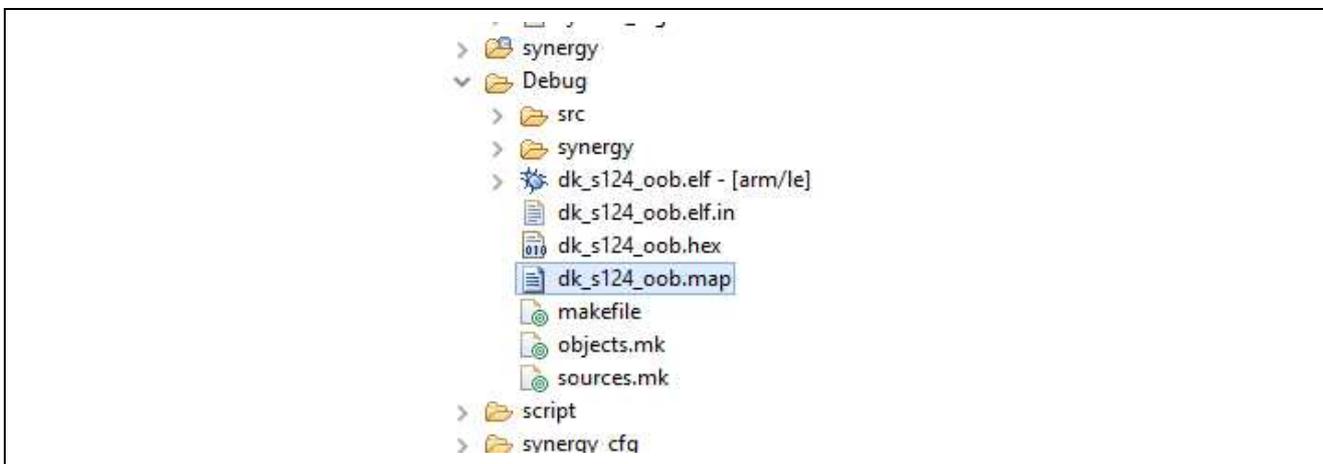


図 4.4 マップファイルの位置

システム設定によっては、e²studio はデフォルトで、他のエディタでそのファイルを開けてしまう可能性があります。例えば、Visual Studio がインストールされたシステムでは、このファイルをダブルクリックしたときに Visual Studio で開いてしまいます。それが起こる場合は、常にファイルを右クリックし、[Open With] > [Text Editor] と操作して、e²studio 内のファイルを見てください。

ファイルが開き、目指す情報を短時間で探し出すには、stack_dummy を検索し、スレッドスタックアドレスが定義されている map file のセクションに移動します。

図 4.5 に示すように、入力スレッドスタックは 0x20002160 で、サイズが 0x400 (1024 バイト) となります。スタックメモリを確認するため、メモリモニタを開くこともできます。e²studio の一番下にあるメモリタブを探してください。そのタブが見えない場合、上部のリボンツールバーを使い [Window] > [Show View] > [Memory] と操作して開くことができます。小さな緑色の[+]シンボルをクリックし、新規メモリモニタを追加し、このアドレスをダイアログボックスにタイプしてください。

```

.stack_dummy 0x20000f60 0x1a00 load address 0x0000bf60
              0x20000f60 . = ALIGN (0x8)
              0x20000f60 __StackLimit = .

*(.stack*)
.stack 0x20000f60 0x1000 ./synergy/ssp/src/bsp/cmsis/Device/RENESAS/S124/Source/startup_S124.o
.stack.blinky_thread 0x20001f60 0x200 ./src/synergy_gen/blinky_thread.o
.stack.inputs_thread 0x20002160 0x400 ./src/synergy_gen/inputs_thread.o
.stack.lcd_thread 0x20002560 0x400 ./src/synergy_gen/lcd_thread.o
              0x20002960 __StackTop = .
              0x20002960 PROVIDE (__stack, __StackTop)
              0x20002960 RAM segment used end = ALIGN ( __StackTop, 0x4)
    
```

図 4.5 マップファイルを使うスレッドスタックアドレスの配置

プロセスが何らかの変数のアドレスを探すために動作している間は、メモリモニタのメモリアドレスを使う際に注意が必要です。進行中に変更をすると、プログラムアドレスは変わるかもしれません。そのため、可能であれば常にシンボルを使う方がよいです。

このプロセスへのショートカットは、ユーザがスレッドに与えた現状シンボル名への [appending_stack] を含んでいます。この場合スレッド名は [inputs_thread] となり、この名前に対する [appending_stack] は名前 [inputs_thread_stack] を与えます。このシンボル名を、図 4.6 に示すメモリモニタダイアログボックスに入力することができます。このシンボルを使う利点は、スレッド名を明示的に変更しない限り、プロジェクトにコードを加えても差し引いてもこのシンボルは変わらないということです。

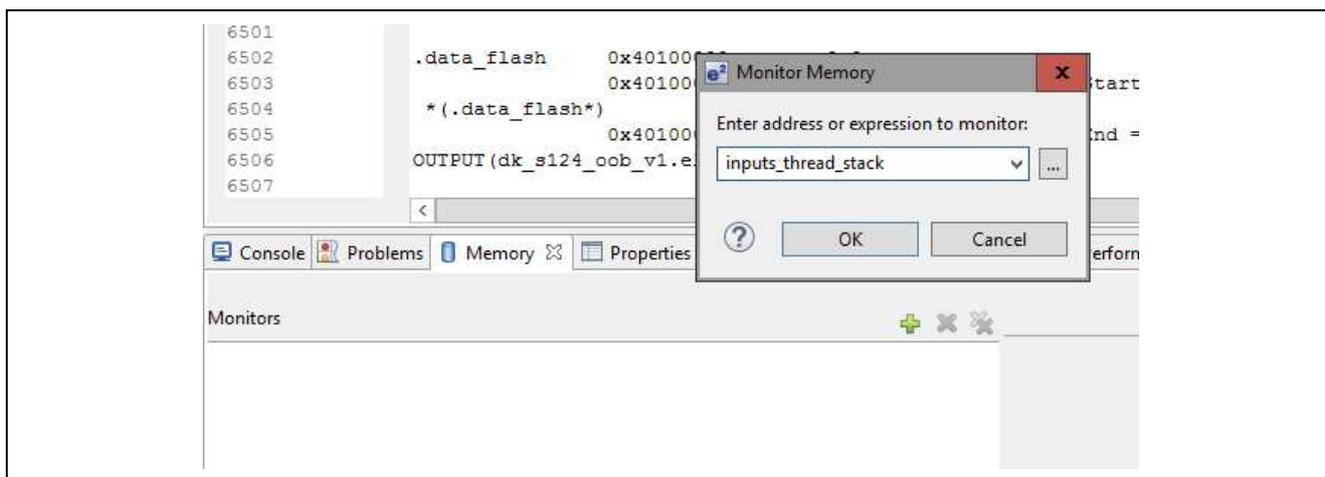
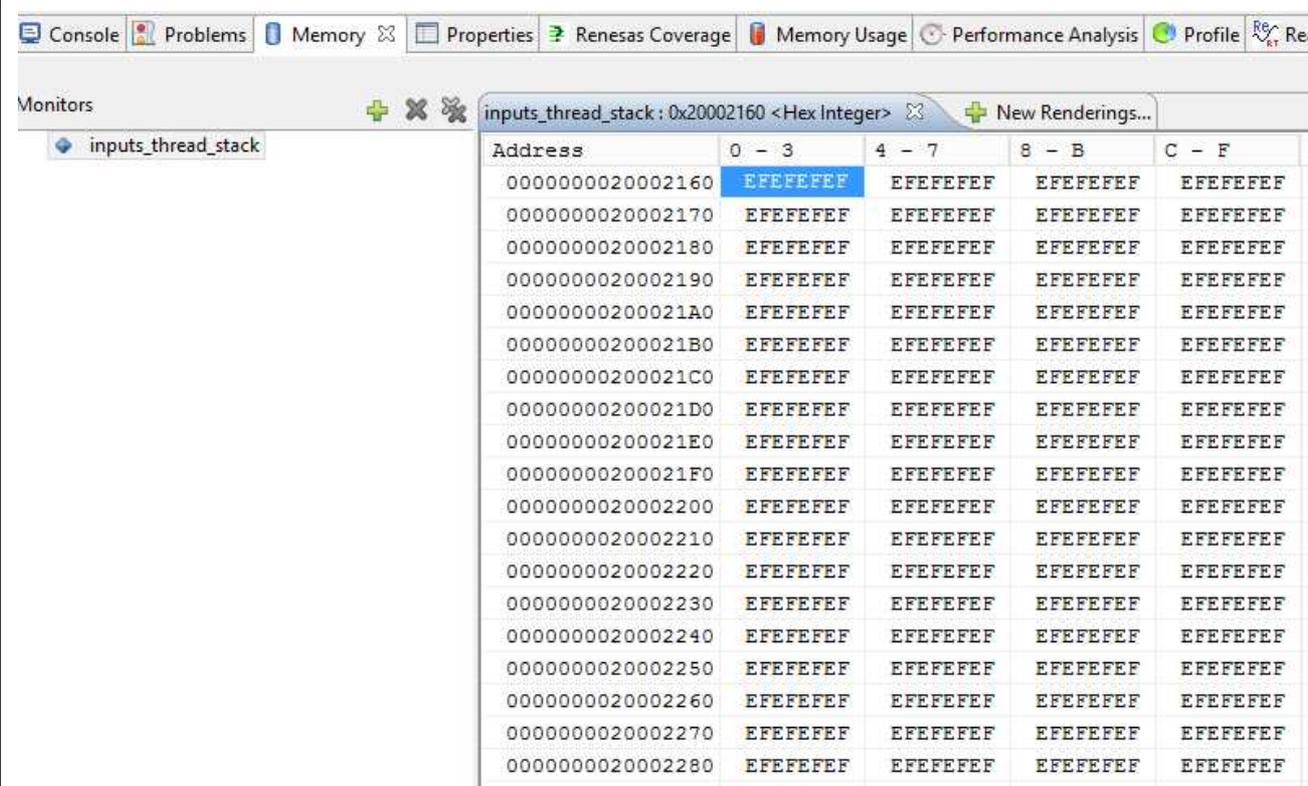


図 4.6、メモリモニタの設定

図 4.7 で、加速度センサスレッドスタックにかなりの量の未使用メモリがあることを示します。各行は 16 バイトを示します。EFEFEFEF と表示されている行の数に 16 を掛けて、どれだけ未使用のバイト数があるかを見てください。その量の半分までは、スタックサイズをほぼ何の問題もなく減らすことができるはずで

す。プログラムを実行した後しばらくの間、スタックの空きスペースを監視するためにスタックの使い方を監視することができます。

スタックスペースの数百バイトを節約してもあまり多いとは思えないかもしれませんが、プログラム内の多くのスレッドで積み重なると、その差はアプリケーションにもう1つの機能を追加できるほどになる場合があります。



Address	0 - 3	4 - 7	8 - B	C - F
0000000020002160	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002170	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002180	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002190	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
00000000200021A0	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
00000000200021B0	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
00000000200021C0	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
00000000200021D0	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
00000000200021E0	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
00000000200021F0	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002200	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002210	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002220	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002230	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002240	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002250	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002260	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002270	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F
0000000020002280	EF0F0F0F	EF0F0F0F	EF0F0F0F	EF0F0F0F

図 4.7 メモリモニタでのスタックメモリの表示

4.3 プロジェクトのインポートとビルド

[r11an0023eu0116-synergy-ssp-import-guide.pdf] に示した手順に従って、e²studio ISDE にプロジェクトをインポートし、ビルドし、デバッグしてください。デバッグの設定については、[dk_s124_ooob Debug] ([Renesas GDB Hardware Debugging]の中) を選択してください。

5. 次のステップ

サンプルアプリケーションを実行した後、アプリケーションがどのように動作したか、どのように API コールを行ったかをアプリケーションのソースコードを調べることでより多く知ることができます。

追加の Synergy サンプルアプリケーションプログラムは以下の URL からダウンロードすることができます。

https://www.renesas.com/en-us/products/embedded_systems_platform/synergy/sample-code.html

6. 参考資料

Renesas Synergy プロジェクト インポートガイド : r11an0023eu0116-synergy-ssp-import-guide.pdf

ホームページとサポート窓口

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

技術的な質問の窓口の詳細

- 米国: 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
1.00	2016/09/23	-	初版
1.10	2016/10/20	-	SSP 1.2.0-b.1 バージョンに更新
1.11	2016.11.30	-	IAR EW のサポートを追加
1.12	2017.02.21	-	1.2.0 のサポートを追加

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/>