Renesas Synergy™プラットフォーム

IAR C-SPY によるアプリケーションデバッグの実行

R20AN0418JU0100
Rev.1.00
2016.09.23

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、SK-S7G2 Blinky with ThreadX **プロジェクトテンプレートを使用しています。**Renesas Synergy™ **の** IAR Embedded Workbench **入門を参照し、**Synergy **スタンドアロンコンフィグレータ**(SSC)**を使用して、プロジェクトを作成してください。**

目次

1. 初期設定

1. ワークスペースのプロジェクトの上で右クリックして、[S7G2-SK Blinky with ThreadX]のプロジェクトオプションを開きます。
2. Synergy ボードに実装されている J-Link ドライバが選択されていることを確認します。手順 3 および 4 で、最終的なデバッグプローブ設定を行います。Options ウィンドウを開けたままにしておいてください。

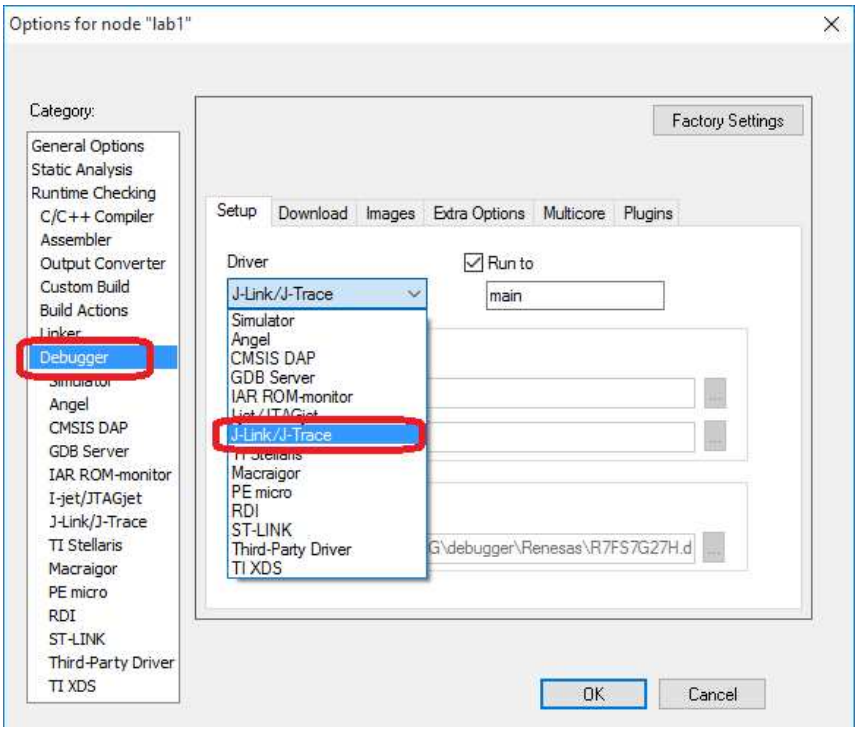


図 1 J-Link ドライバ

3. 全てのプロジェクトのオプションへアクセスできるようにプロジェクトツリーの中で太字のプロジェクトを必ず選択します。

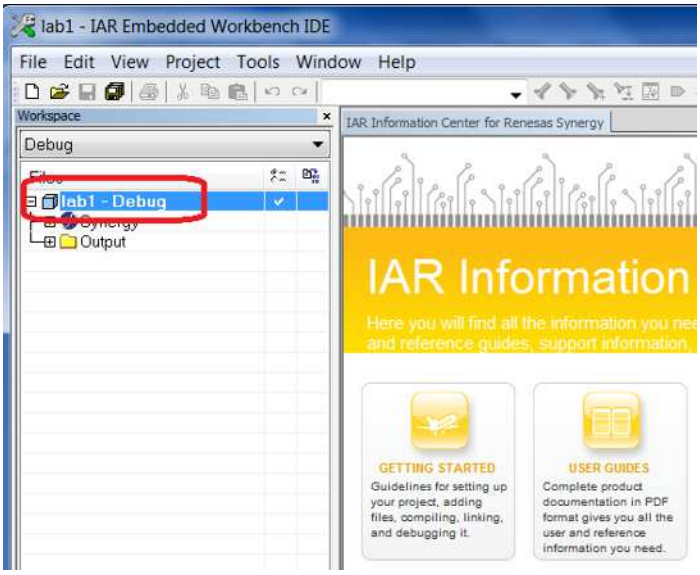


図 2 Debug プロジェクト

4. [**Debugger**]の下の[**J-Link/J-Trace**]を選択して、[**Setup**]のタブで、[CPU clock]を SWO クロックに必要とされる 120MHz に設定します。

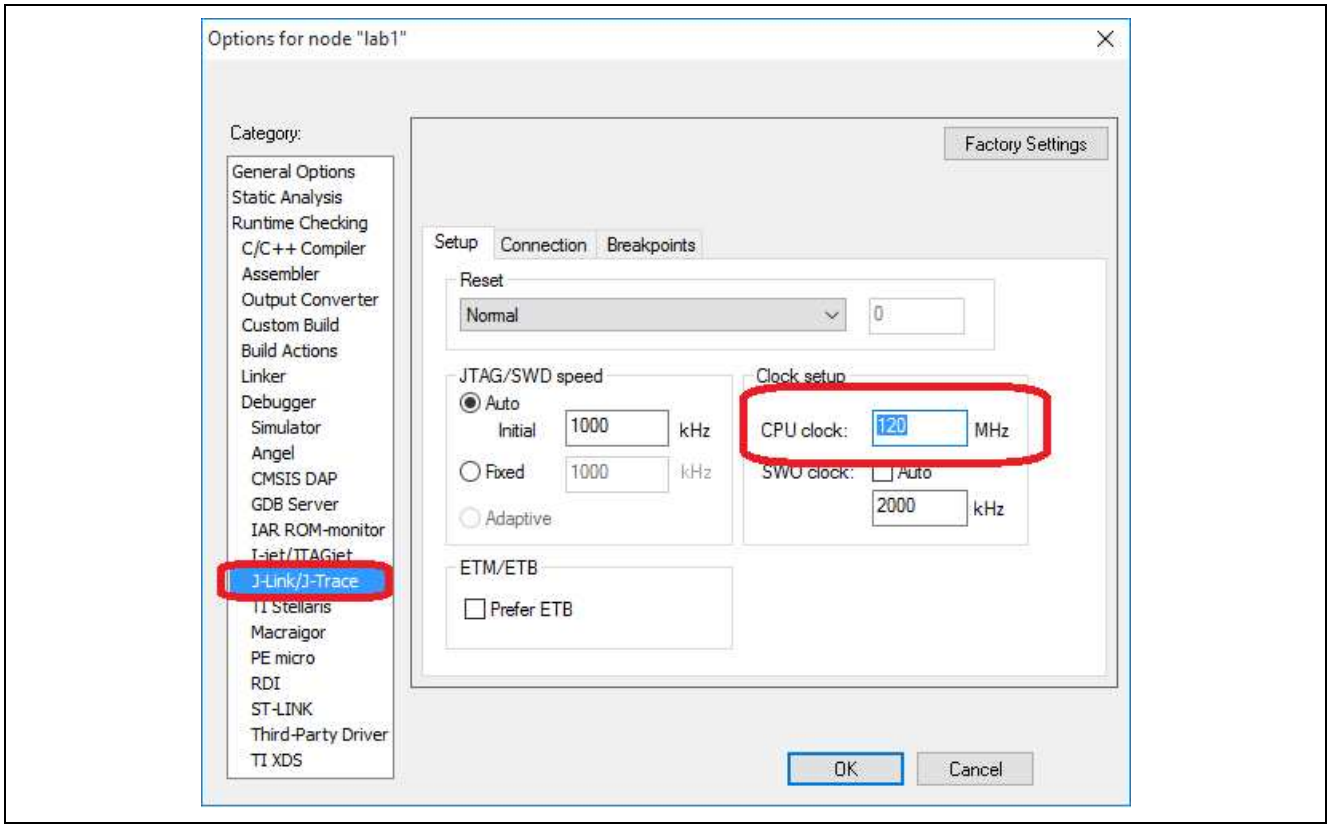


図 3 CPU クロックの設定

5. **J-Link/J-Trace の[Connection]タブから、本書で全デバックに使用する SWD インターフェースを選択します。**

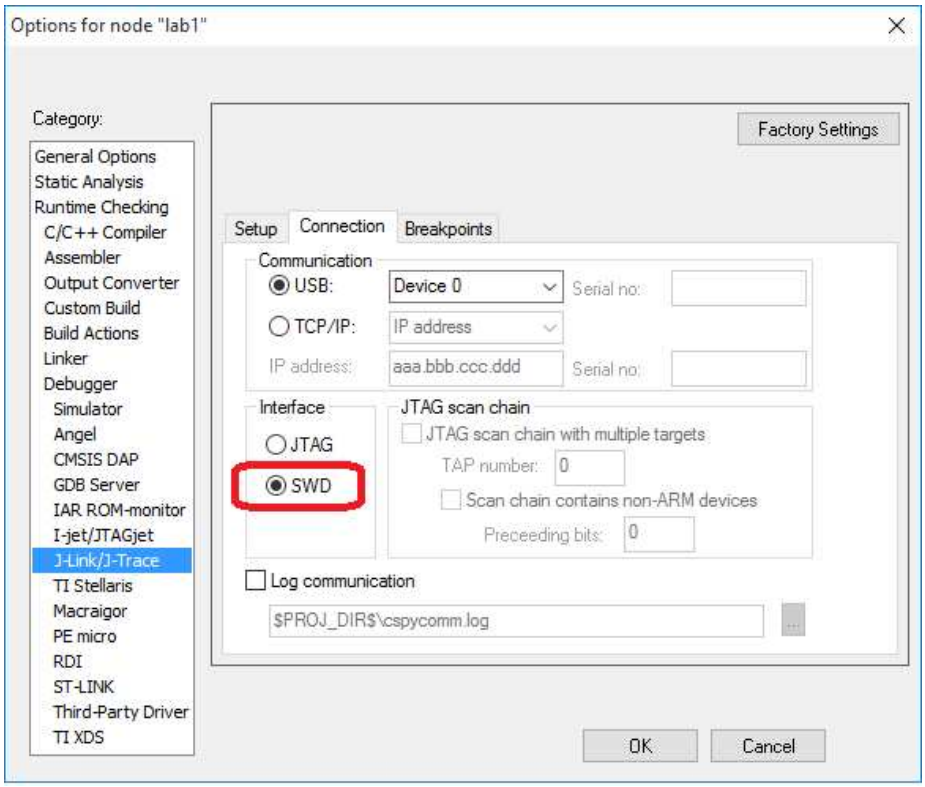


図 4 SWD インターフェースの選択

6. **全ての変更を確認して[OK]をクリックします。**
7. **[Project]メニューから[Download and Debug](ショートカットキー：**CTRL+D)**を選択してアプリケーションをダウンロードしてデバッグします。**

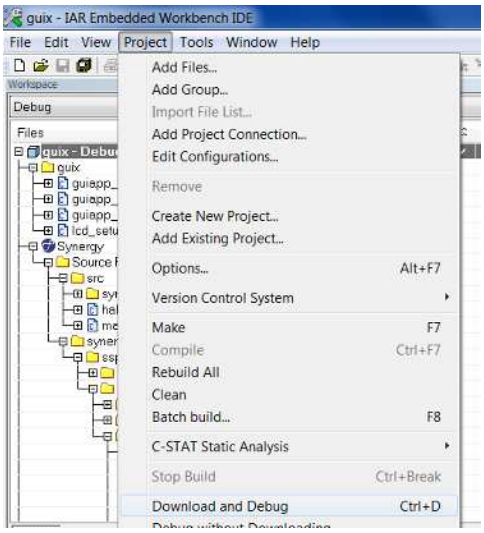


図 5 Download and Debug の選択

8. **アプリケーションプログラムは** `main` **関数のところで止まります。**

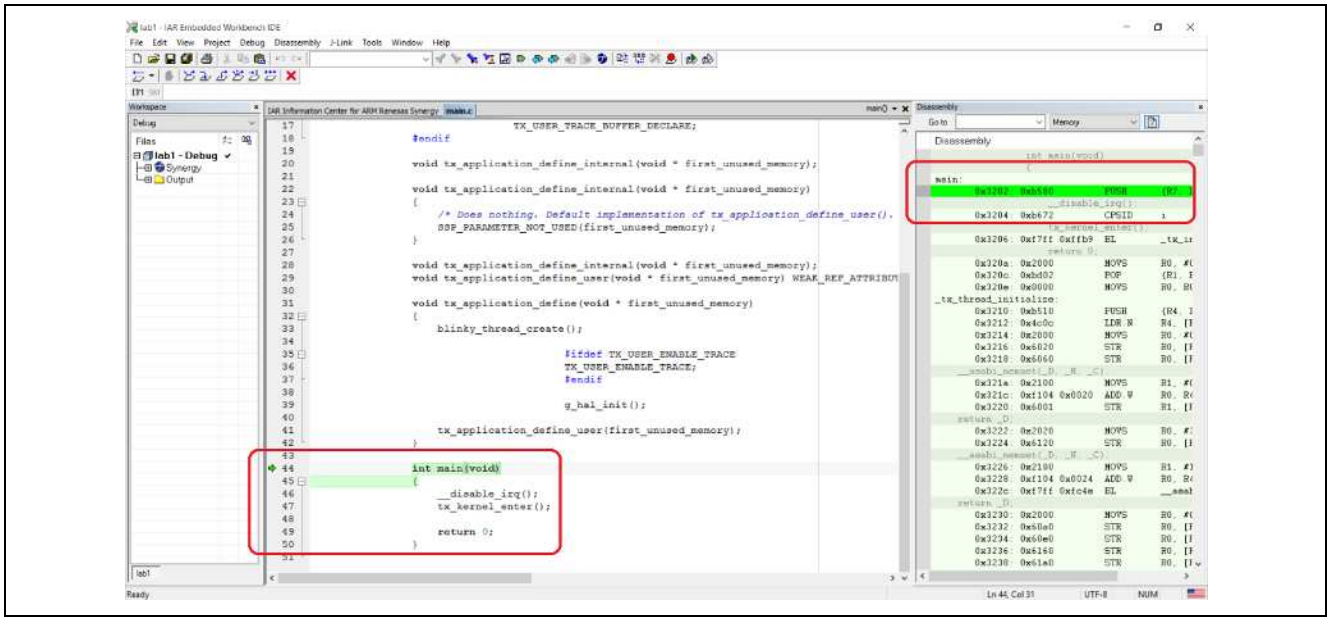


図 6　main()関数での停止

9. **ソースコード内でライン番号を有効にします。[Tools]メニューから[Options]＞[Editor]＞[Show line numbers]を選択します。デバッグ中でもいつでもライン番号を有効にできます。**

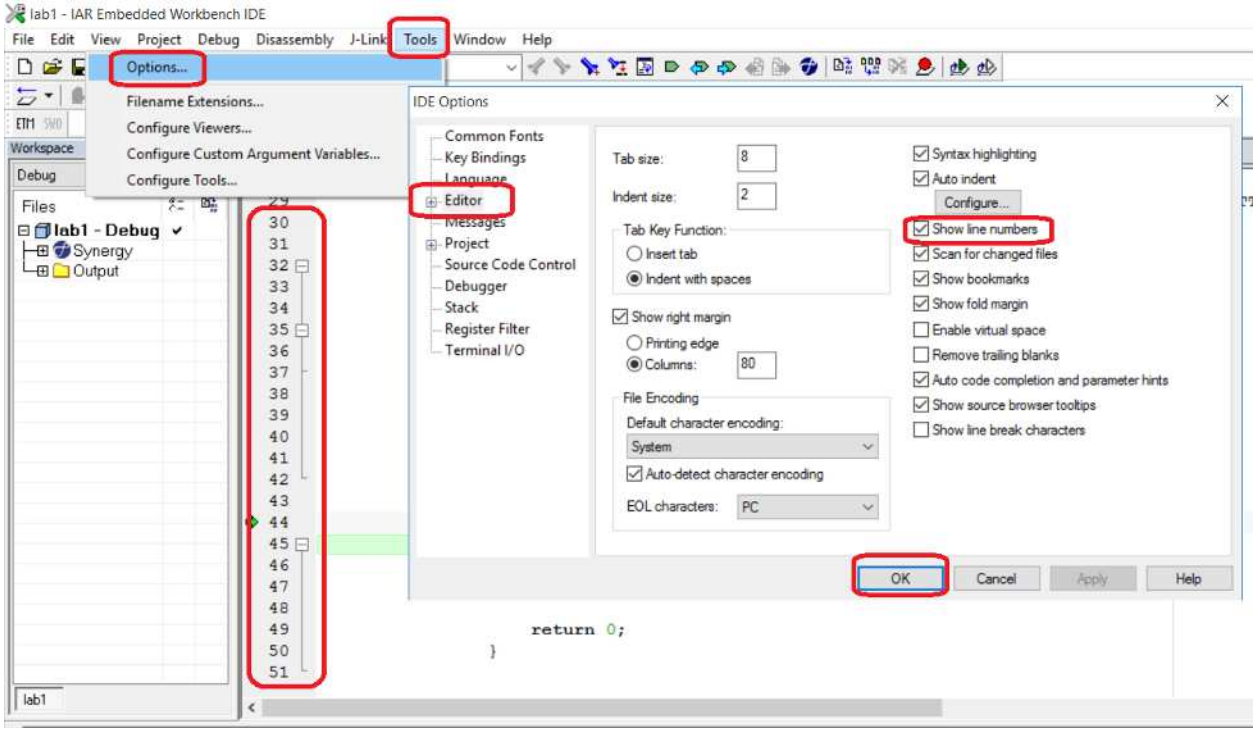


図 7 ライン番号

2.　デバッグベイシック＆ライブウォッチ

1. **[Disassembly]ウィンドウが開いていない場合、[View]メニューから開きます。C レベルおよびアセンブラーレベルの両方で、コードを使用すればひと手順でできます。**

2. [Step into]、[Step over]、[Step out]、[Restart debugger]、[Go]、[Break] **の機能を使います。**

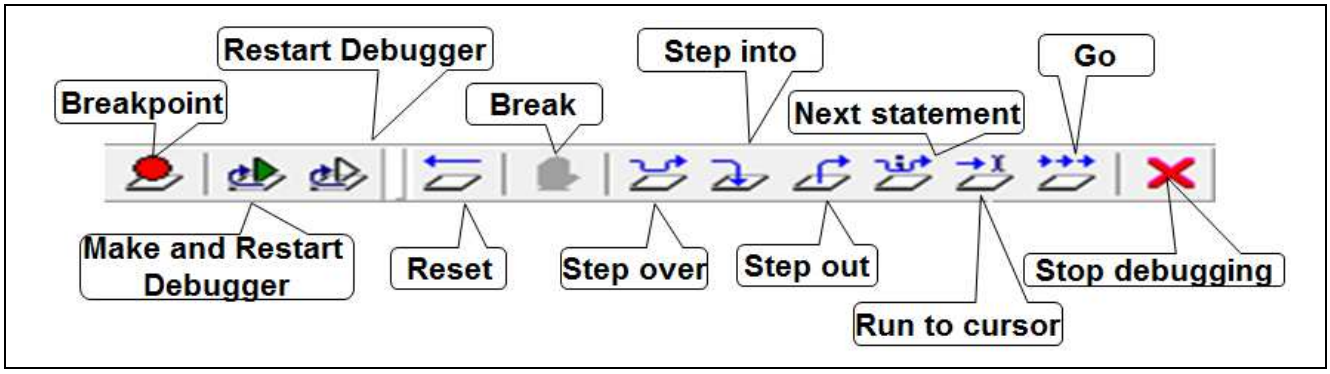


図 8 Debug 機能

3. **[View]メニューから[Register]ウィンドウを開き、コードを追いながら、いかにレジスタが値を変更していくかを見ます。**
4. **[View]メニューから[Call Stack]ウィンドウを開き、1 つの手順内の** call**（呼出し）を見ます。**
5. **アプリケーションを中断または停止して、レジスタを見てスタック情報を呼び出します。**
6. **プロジェクトエクスプローラにある** blinky_thread_entry.c **ファイルをダブルクリックします。図 9 のスクリーンショットのように、**while **ループ内でシンプルカウンタがインクリメントされます。**
   a. `int counter = 0;` > 15 **行目**
   b. `counter++;` > 40 **行目、**while **ループ内**

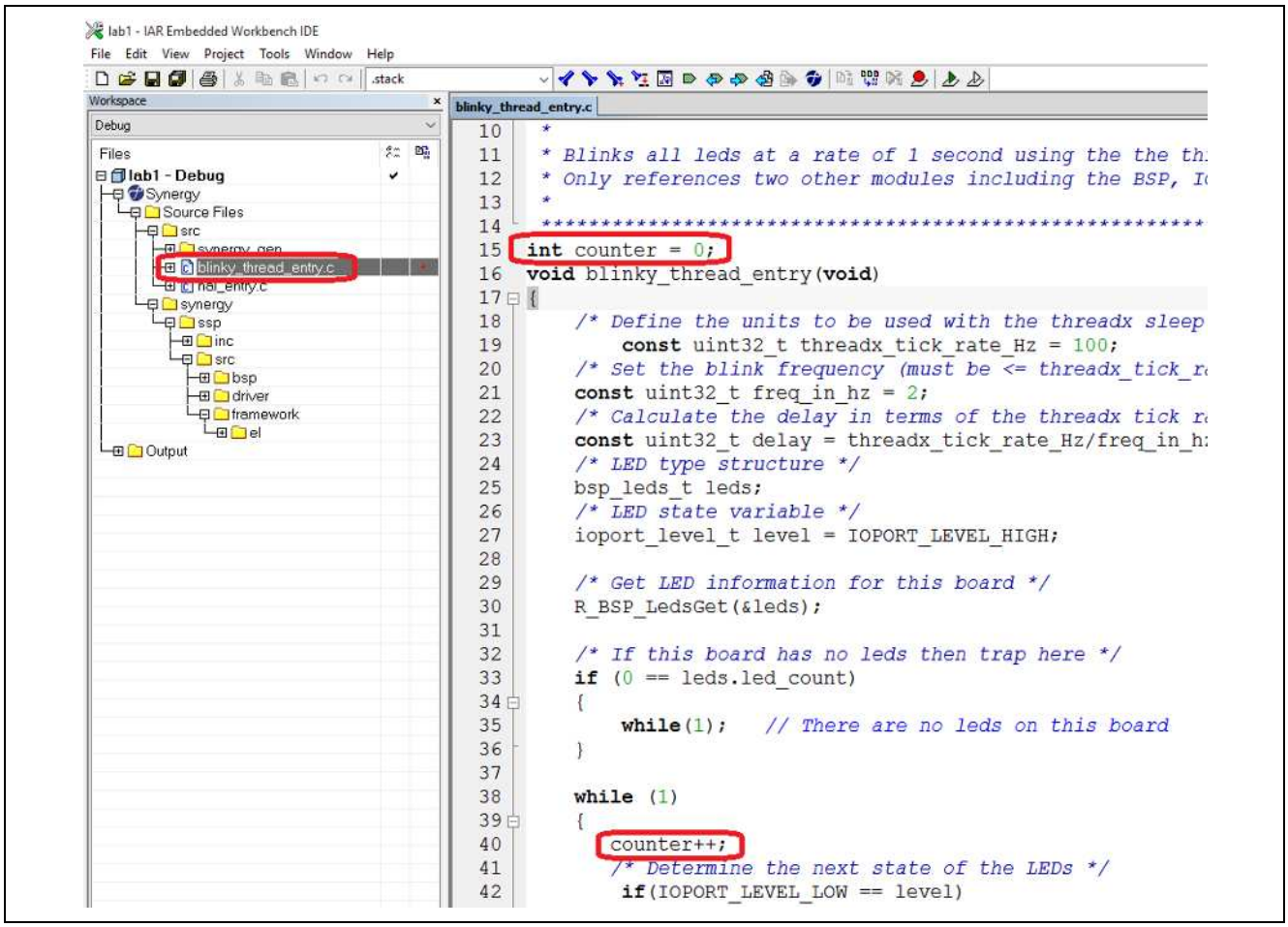


図 9 インクリメントカウンタ

10. ソースコードに変更を加えた後、[**Project**]メニューから、[**Make & Restart Debugger**] (ショートカットキー：Ctrl+R) または、上部の[**Make & Restart Debugger**]アイコンをクリックします。

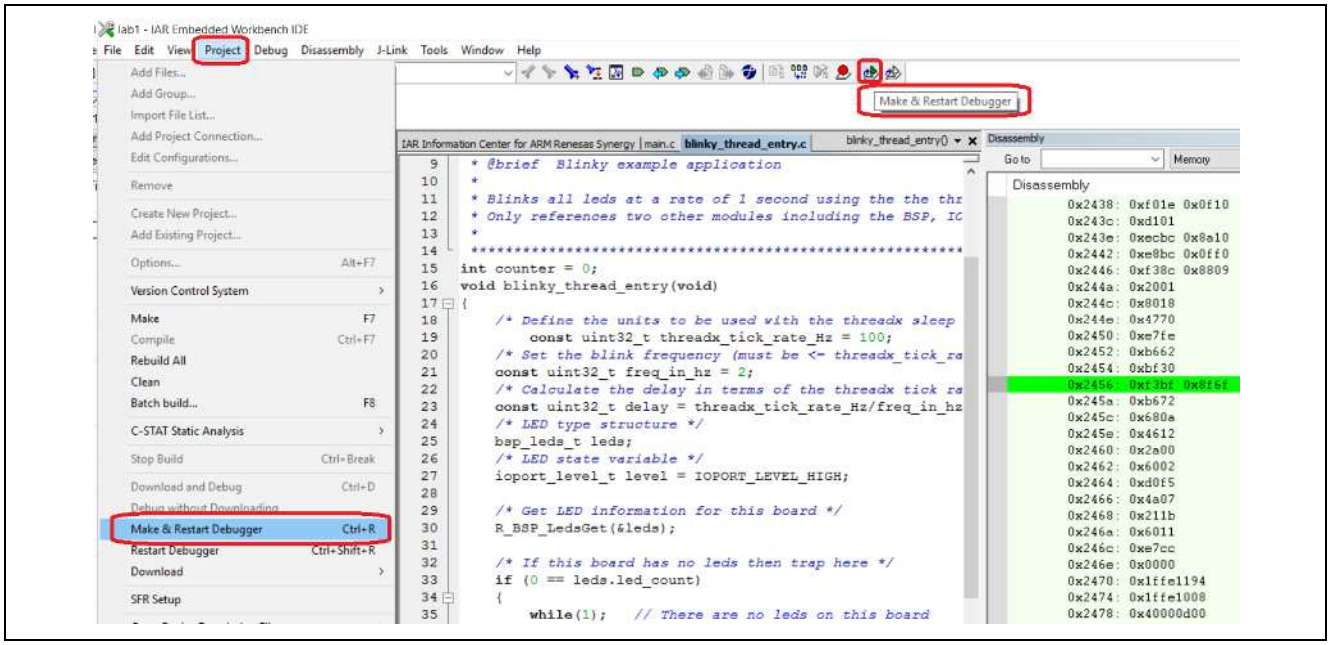


図 10　Make & Restart Debugger

11. [**View**]メニューから[**Live Watch**]ウィンドウを開き、”counter”と入力します。
12. アプリケーションをリセットして `main` 関数に戻ります。
13. [**Go**]をクリックして、動作中にカウンタ値がどう変化するかを見ます。アプリケーションを停止せずに変数値をモニタすることができます。
14. [**Live Watch**]では、短いアップデート間隔が必要になる場合が時々あります。例えば、デフォルトでは、1000ms で設定されていますが、1 ms まで短縮できます。アップデート間隔は、[**Live Watch**]ウィンドウで右クリックし、[**Options**]メニューから、 **Debugger** > **Update intervals (milliseconds)** > **Live watch** と選択して、変更可能です。

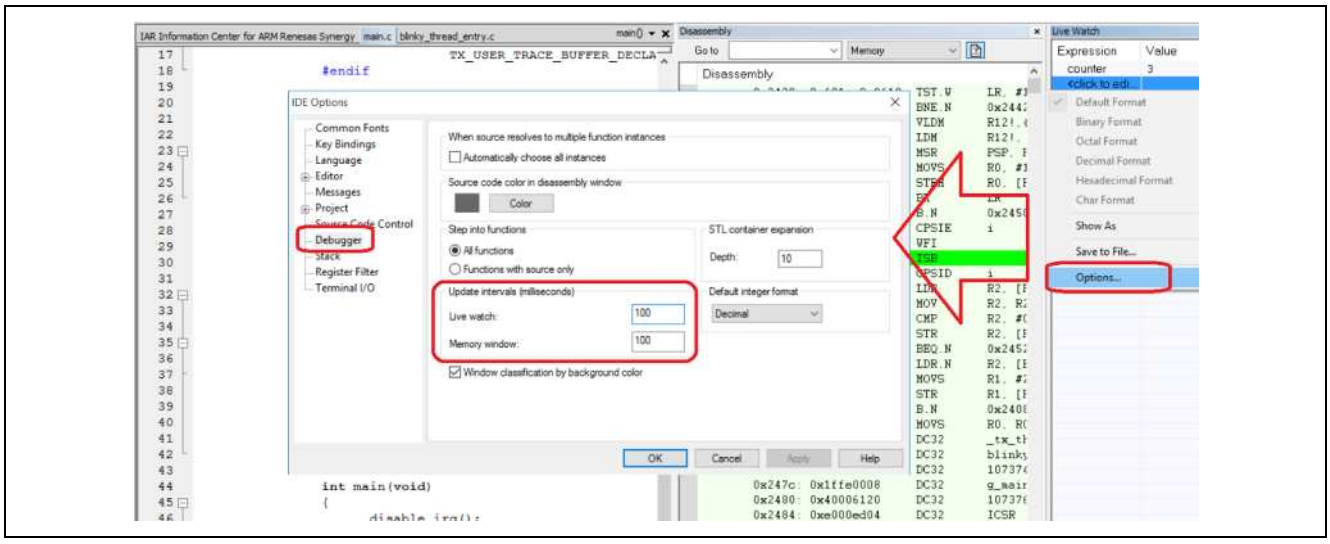


図 11　Live Watch のアップデート間隔

15. デバッガを中断させて、アプリケーションをリセットし、再度、強制的に”counter == 0”を設定します。

3. 条件付きのデータブレークポイント

特定の条件が合った場合に特定の時点で実行を停止するには、コンディショナルブレークポイントを挿入します。

1. blinky_thread_entry.c の 57、58 行目に以下の表現があります。

```
/* Delay */
 tx_thread_sleep (delay);
```

a. 右クリックをして[**Toggle Breakpoint (Code)**]を選択します。

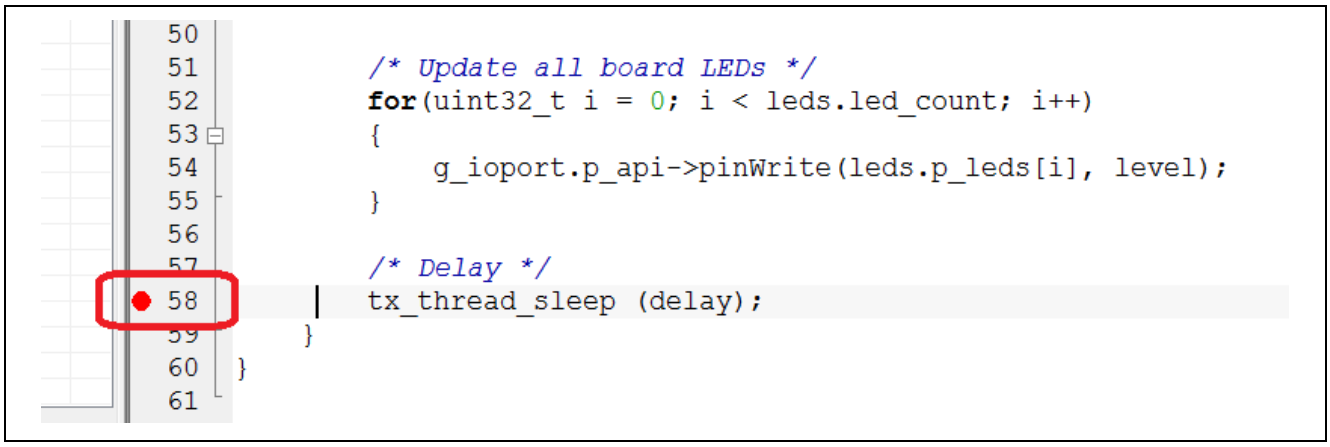


図 12 Break する位置

b. 再度右クリックをして **[Edit Breakpoint]を選択します。**

c. [**Conditions]の[Expression**]ボックス内に[counter>10]とタイプします。

d. この設定により、カウンタ値が 10 を超えた場合、実行が停止されます。

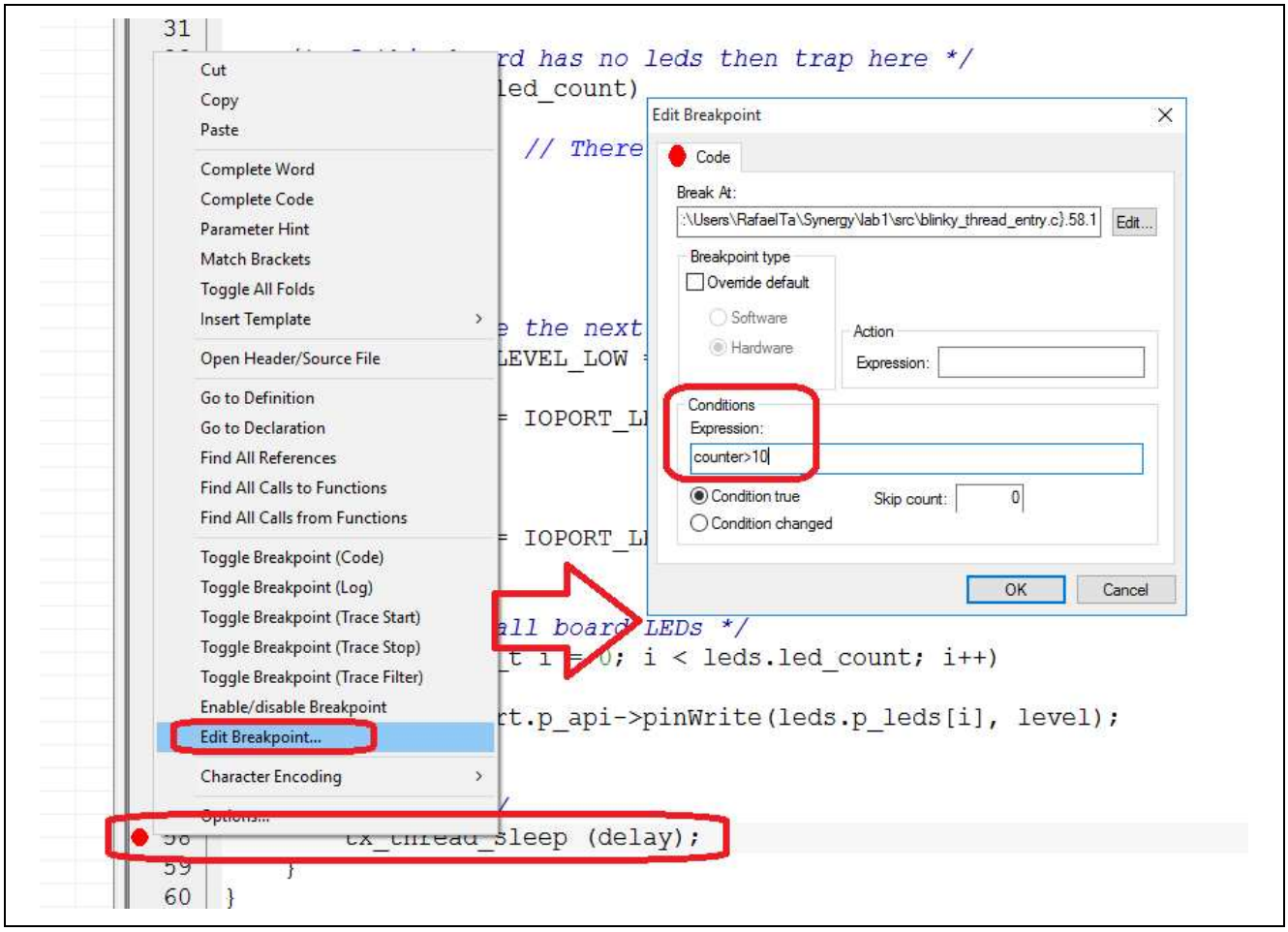


図 13 実行停止:　カウンタ値が 10 を超えた場合

2. アプリケーションをリセットして、main へ戻り、強制的に force counter == 0 を設定します。
3. プログラムを再度走らせます。**View** > **Live Watch** からカウンタが変わるのを見ます。条件が真の場合、ブレークポイントは有効になります。
4. ここでは表示されていませんが、実行が停止した場合、条件が判定され、その結果が偽の場合、実行は再開します。複雑なブレークポイントは負荷がかかりますが、とても効力を発揮します。
5. 下記種類のシンボルは式に使用できます。
   a. C/C++ シンボル
   b. アセンブラシンボル(レジスタ名およびアセンブララベル)
   c. C-SPY マクロ関数
   d. C-SPY マクロ変数
6. かなり複雑な式も書くことができます。しかし、これらの式はその式の真偽を判定しなくてはならないので、性能が損なわれます。
7. ブレークポイントがトリガされるときにアクションボックスが別の式を評価するために使用されます。
8. 58 行にある条件付きのブレークポイントを削除します。削除する方法の 1 つは、[View]メニュー から [Breakpoints]ウィンドウを開き、ブレークポイントを全て削除します。このウィンドウから新規でブレークポイントを設定することもできます。
9. 15 行目のグローバル変数カウンタを右クリックします。メニューで、今回カウンタの[Set Data Breakpoint]を選択します。

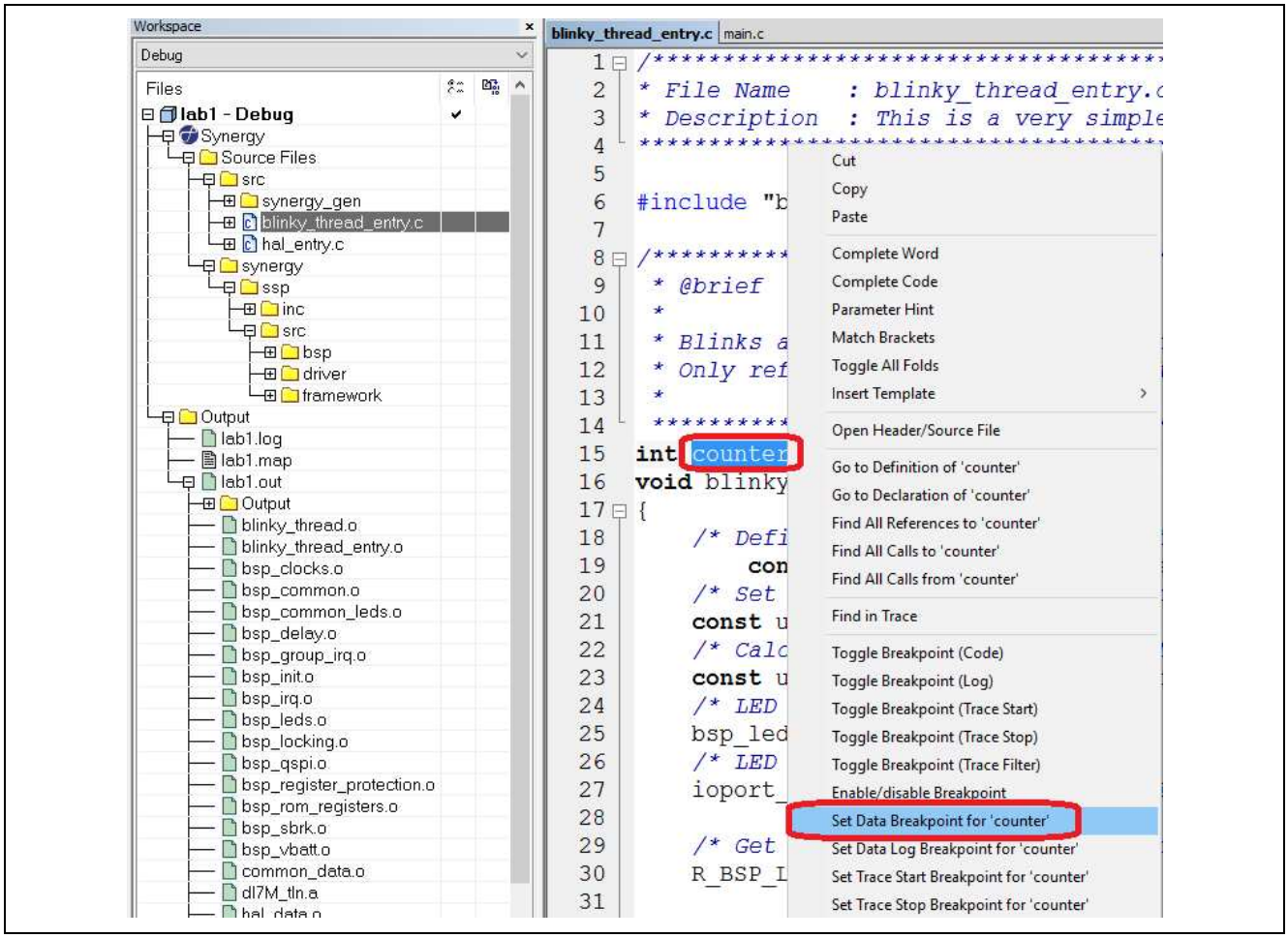


図 14 変数用ブレークポイントの設定

10. **[Breakpoint]ウィンドウでは、([View]メニューから[Breakpoints]), データブレークポイントを編集します。**

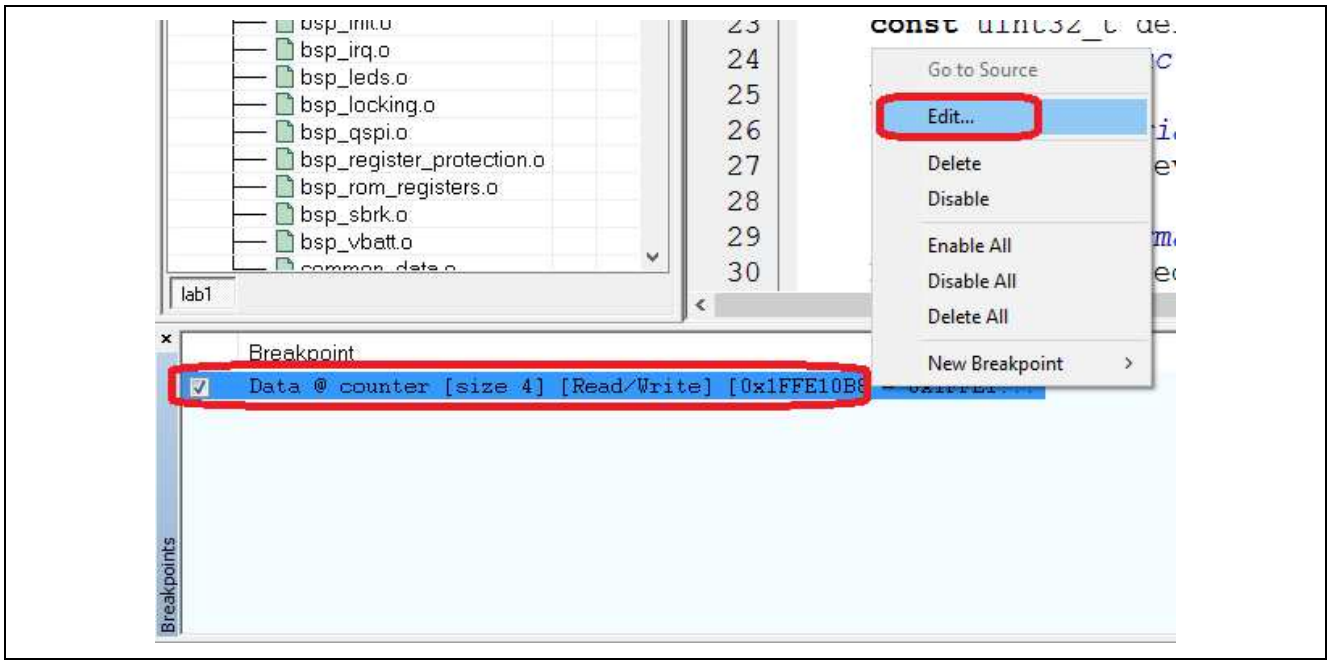


図 15 ブレークポイントの編集

11. **ブレークポイントの [Access type]を[Read]に設定し、[Match data]の値を 0x05** (5d)**に設定します。**

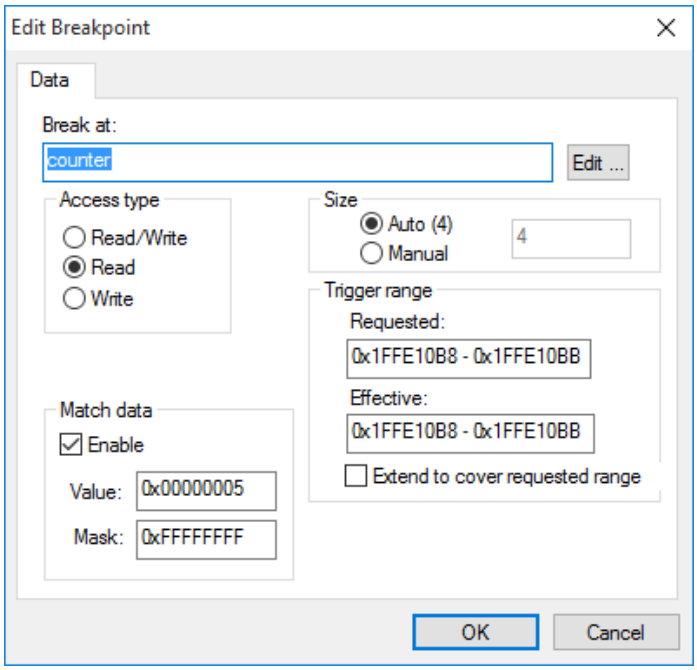


図 16 ブレークポイント範囲の編集

12. **[Go]をクリックします。**
13. **変数への読み出しがあり、データが一致した場合にアプリケーションは停止します。**
14. **ライトおよびリード/ライトのように追加のアクセスの設定を試してください。**

4. C-SPY Macros

1. **古いブレークポイントをすべて削除します。**
2. **blinky_thread_entry.c の 57、58 行目にある式に戻ります。**

```
    /* Delay */
tx_thread_sleep (delay);
```

3. **その行を右クリックして、[Toggle Breakpoint (Log)]を選択します。それから、もう一度右クリックして[Edit Breakpoint]を選択します。**
4. **C-Spy マクロメッセージを有効にして、**[Message]**ボックスに“** "data = ", counter”**を追加します。**

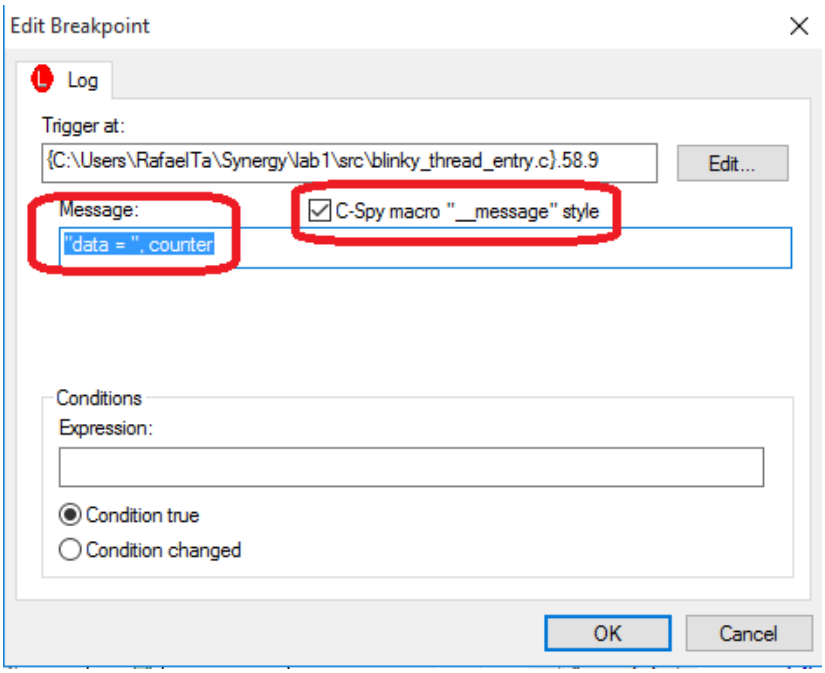


図 17 C-SPY メッセージ

5. プログラムを実行します。[Debug Log]ウィンドウ内にメッセージが見えます。このブレークポイントに条件を付けることができます。

5. SWO 経由 PRINTF およびセミホスティング

1. 単純な `printf()` 関数コールをアプリケーションの while 文の中に追加します。例えば：

```
printf("Hello Synergy!SSP 1.10\n");
```

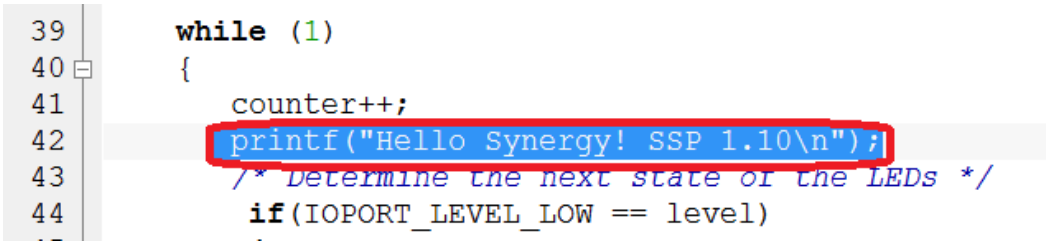


図 18 Printf 関数

2. ソースコードに変更を加えた後、[**Project**]メニューから、[**Make & Restart Debugger**] (ショートカットキー：Ctrl+R)または、上部の[**Make & Restart Debugger**]アイコンをクリックします。

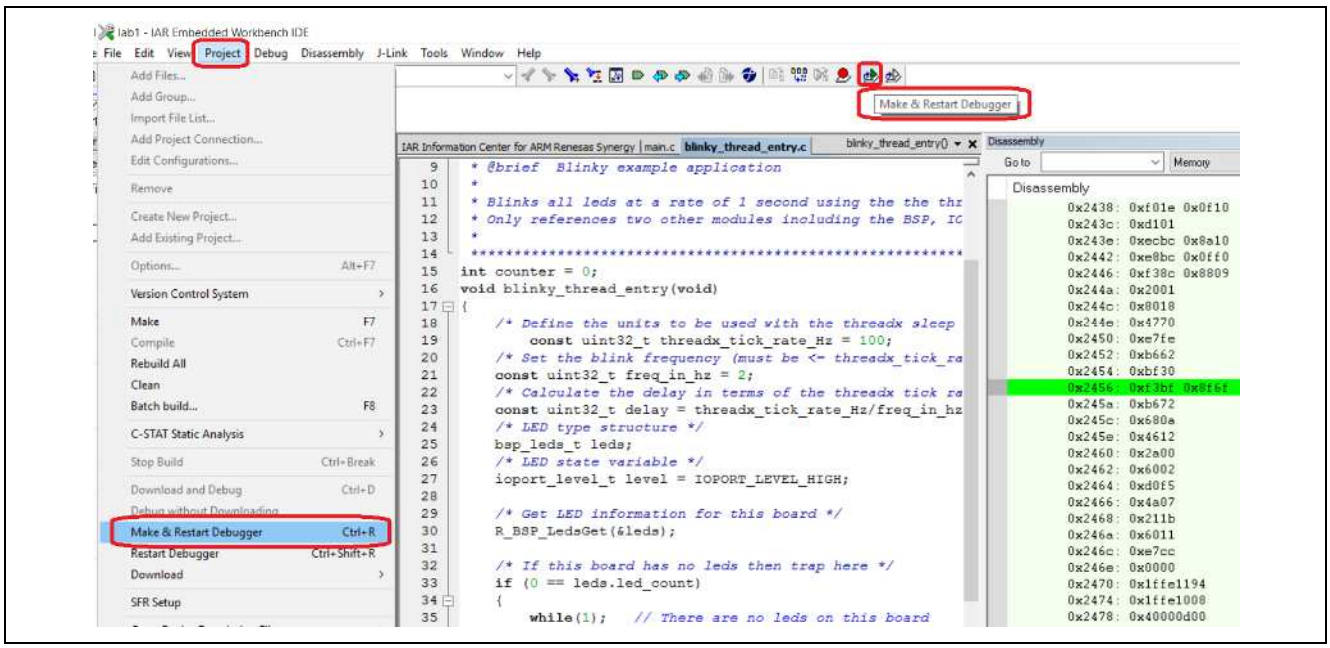
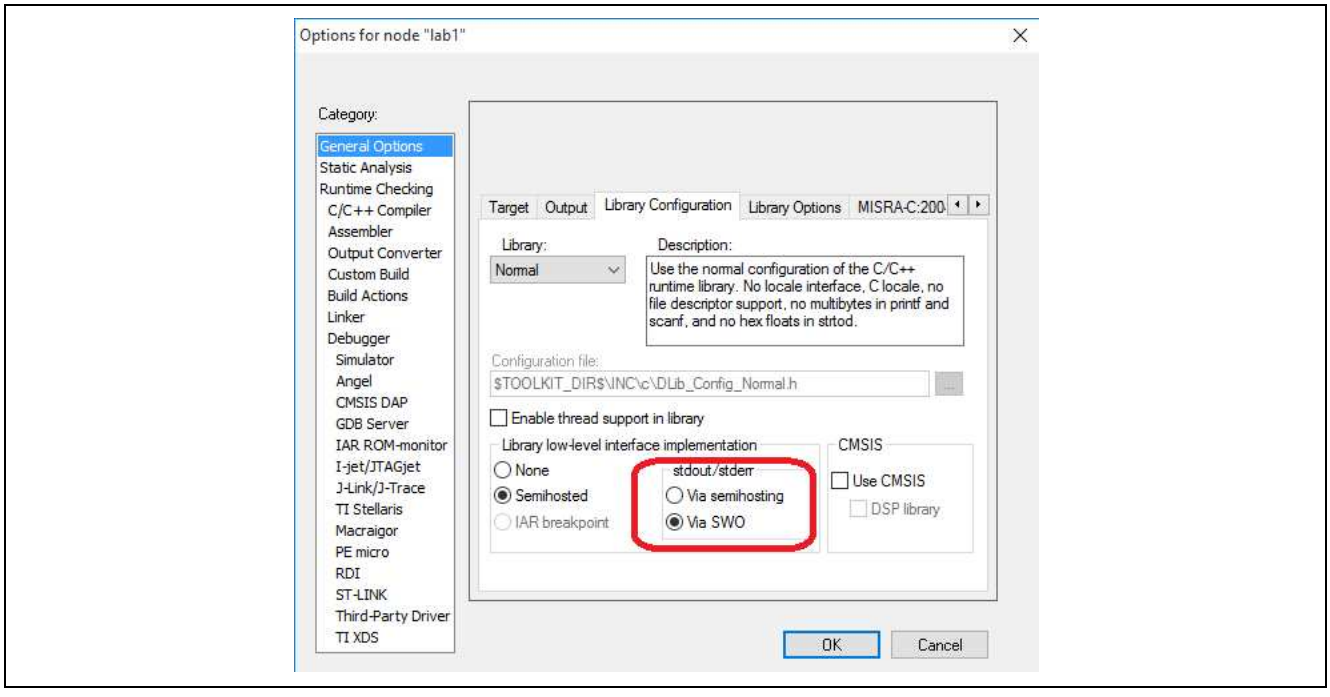


図 19 Make & Restart Debugger

3. **[View]メニューから[Terminal I/O]を選択して、[Terminal I/O]ウィンドウが開いていることを確認します。**
4. **プログラムを走らせ、終了するまでの時間を見ます。**
5. **デバッガを終了します。**
6. **プロジェクトオプションを開き、カテゴリから[General Options]を選択し、[Library Configuration]タブの[stdout/stderr]を[Via SWO]に変更します。**



7. **プログラムをビルドしてダウンロードします。**
8. **プログラムを実行し、終了するまでの時間を見ます。**
9. **何か違いがわかりましたか？**

ご覧の通り、セミホスティングがデータを送信するためにコアを停止する間に、SWOは最大速度で送信され、その後再び実行されます。

6. ITM イベント

時間を計測するためにITMイベントログを設定するには、下記の手順を完了してください。

1. [blinky_thread_entry.c]から、48 行目で IOPORT レベルが高い間に、while ループ内に ITM イベントを追加して、7 行目に必要なヘッダーファイル#include <arm_itm.h>を入れます。入れない場合はビルドエラーが発生します。大量のデータを転送する必要はないので、8 ビットバリアントを選択します。例）ITM_EVENT8_WITH_PC(1,1)
2. 53 行目のインポートレベルが低い場合、2 番目の ITM イベント[ITM_EVENT8_WITH_PC(1,0)]を追加します。LED がトグルされた場合、２つの ITM イベント間の時間を測定します。

```
40      while (1)
41      {
42          counter++;
43          printf("Hello Synergy! SSP 1.10\n");
44          /* Determine the next state of the LEDs */
45           if(IOPORT_LEVEL_LOW == level)
46           {
47               level = IOPORT_LEVEL_HIGH;
48               ITM_EVENT8_WITH_PC(1,1);
49           }
50           else
51           {
52               level = IOPORT_LEVEL_LOW;
53               ITM_EVENT8_WITH_PC(1,0);
54           }
55
```

図 20 ITM イベントの設定

3. ビルドしてからダウンロードします。
4. [J-Link]メニューから[Timeline]を選択します。
5. [Timeline]ウィンドウでは、[Event]ログフィールド内でポイントし、右クリックして[Enable]を選択します。

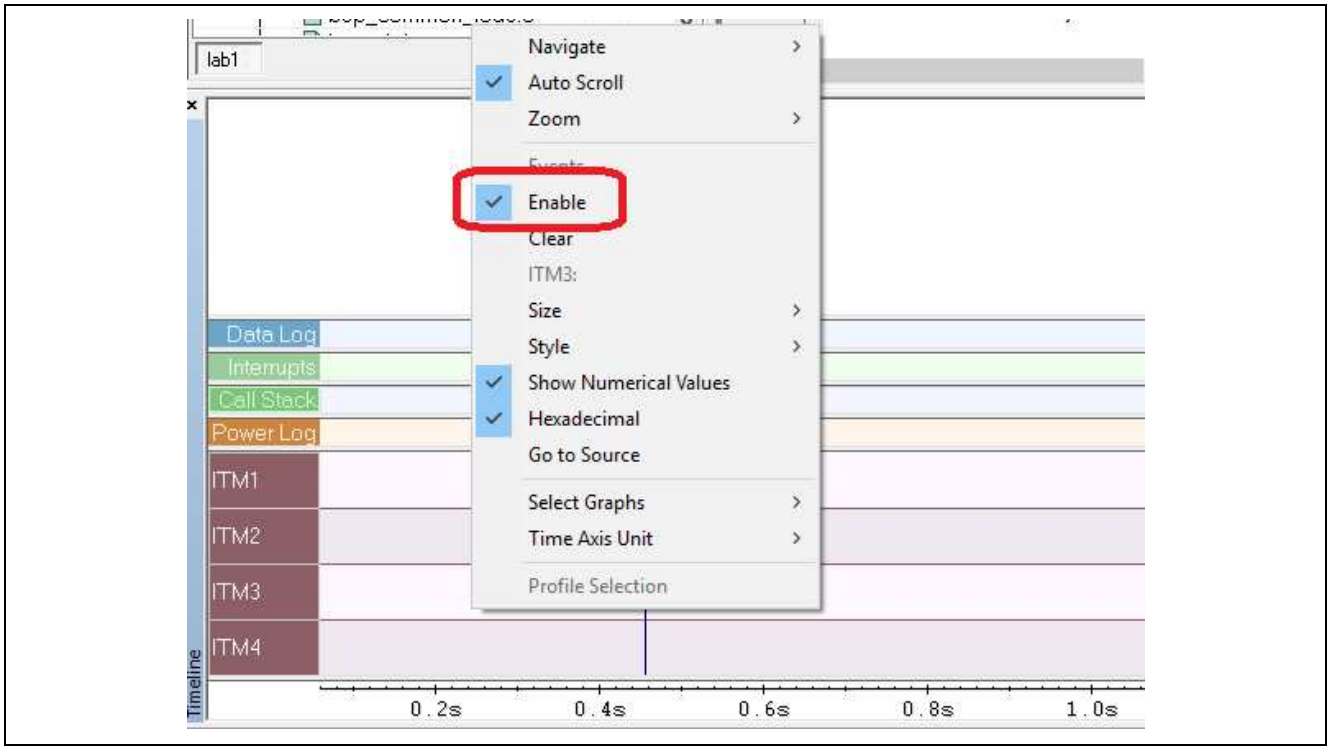


図 21 イベントログを有効にする

6. タイムラインにいくつかのイベントが出てくるまで、数秒間の間プログラムを走らせます。
7. [**Timeline**]ウィンドウにイベントを見つけた場合、重要な部分にどのくらいの時間をかけたか測定できます。拡大／縮小するとより簡単にイベントが探しやすくなります。ITM ログフィールドを右クリックするとタイムラインを拡大／縮小できますが、ショートカットキー[＋]と[－]を使ってもできます。

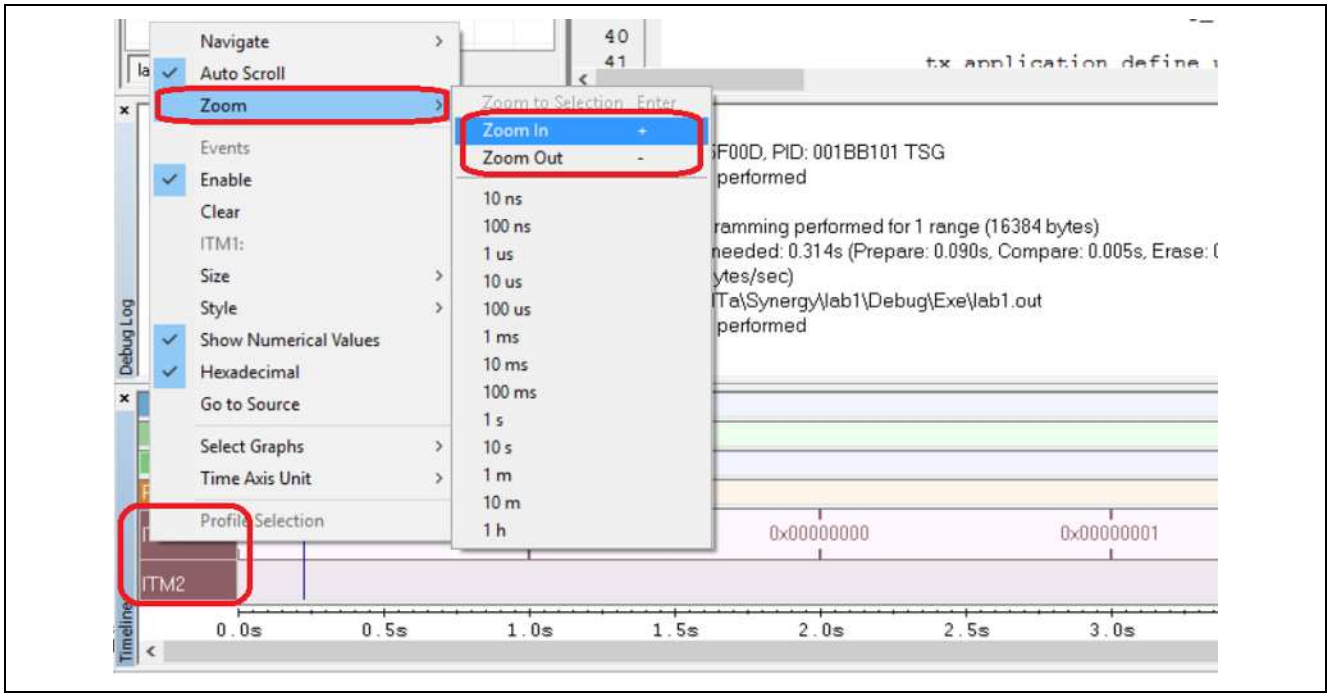


図 22 タイムラインのズーム

8. イベント間の時間を測定するには、タイムライン内で最初のイベントを見つけ、次のイベントへ移行するのに、[Shift]キーを押して右矢印を押します。印があるフィールドの上にカーソルを持ってきて、イベント間の時間を見ます。

9. ITM イベントについてもソースコードナビケーションはわかりやすくなっています。イベントをダブルクリックすると、ソースコードの正しい行に移動します。

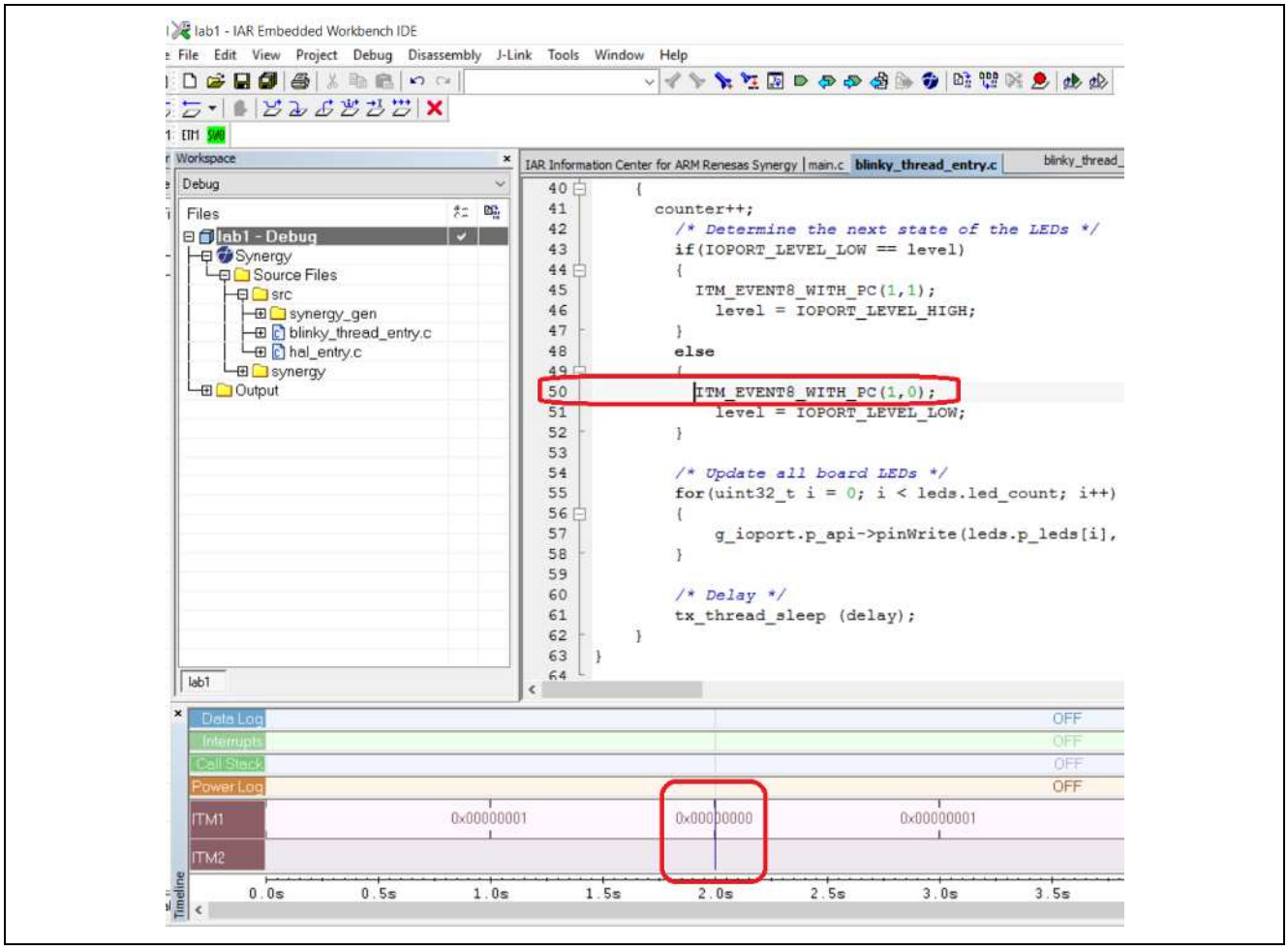


図 23 タイムラインからソースの検索

10. 同じ方法で tx_thread_sleep (delay) がどのくらいの時間がかかっているか測定してみてください。
注： 60 および 62 行に ITM イベントを追加します。

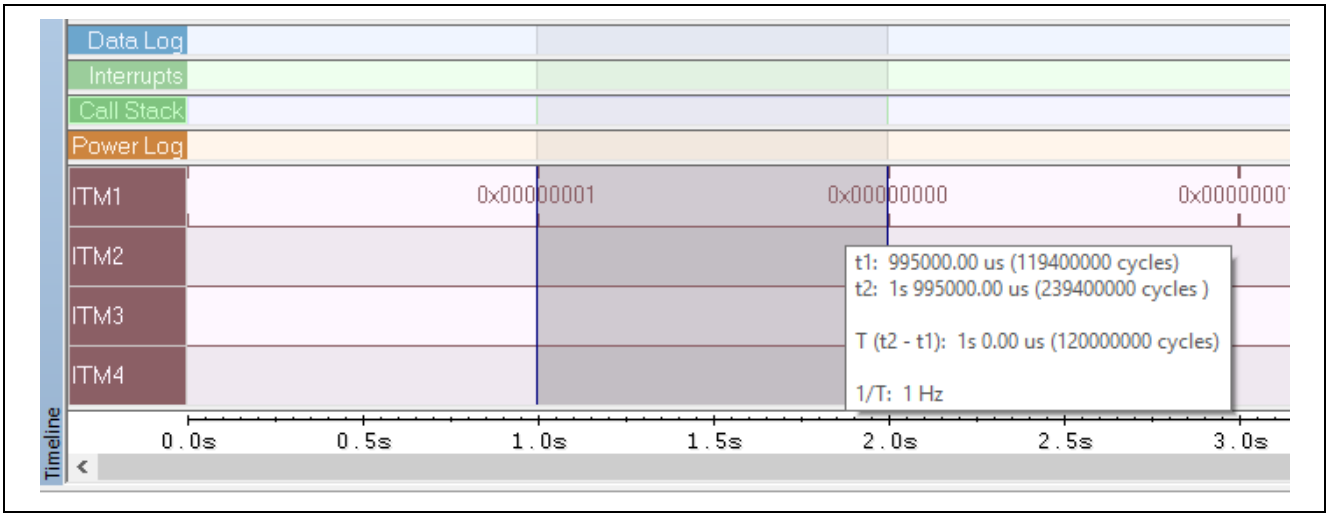
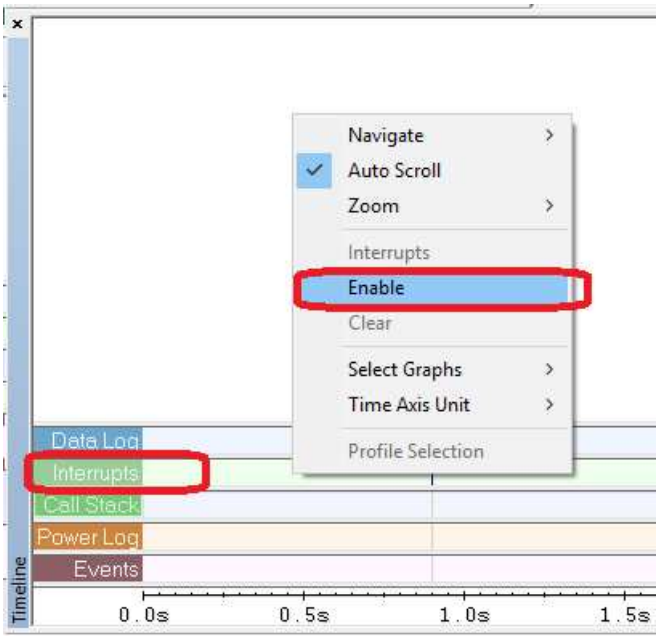


図 24 tx_thread_sleep delay のタイミング

7. 割り込みログ

割り込みログ機能を使うには、以下の手順を行います。

1. ブレークポイントを全て削除します。
2. ビルドしてダウンロードします。
3. [Timeline]を開き、J-Link メニューから[Interrupt log]および[Interrupt Log Summary]ウィンドウを開きます。
4. タイムラインウィンドウで、[Interrupt Log]を有効にします。



5. アプリケーションを実行[Go]します。
6. [Timeline]ウィンドウには全ての割り込みが表示されます。
7. 選択してズームしながら、SysTick にかかるサイクル数を確認します。

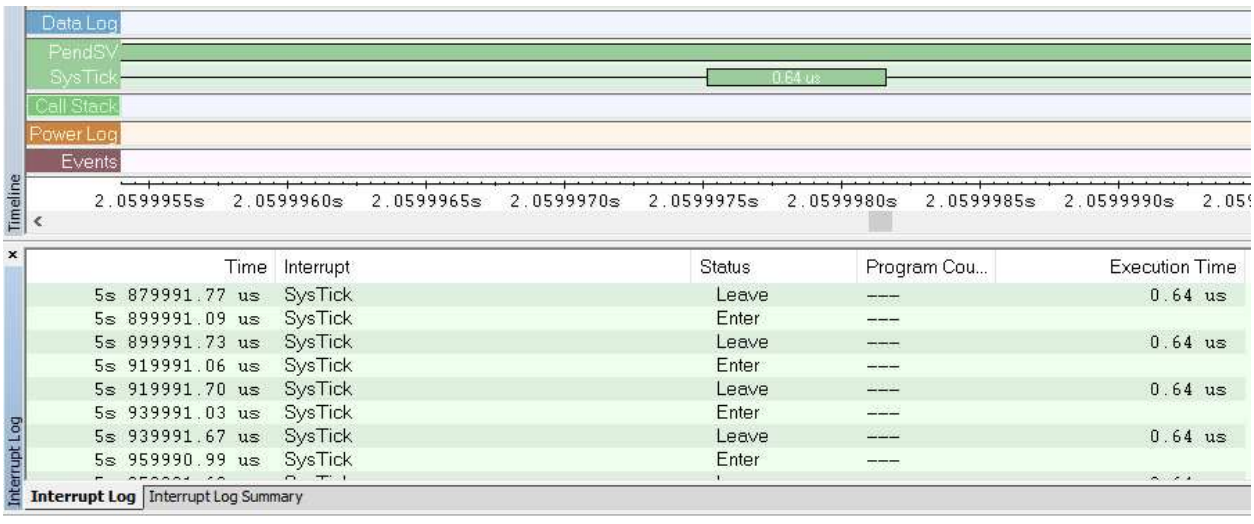


図 25 SysTick タイミング

8. Attach to running target オプションによる実行中プログラムの追跡

リセットせず、またシステム上で実行中のプログラムに影響することなく対象システムへ接続するには、以下の手順を行います。

1. アプリケーションプログラムのビルド、ダウンロード、実行[Go]をしてください。LED ランプが点滅している場合は、アプリケーションが実行中です。
2. 変化するカウンタを[View]メニューから[LiveWatch]ウィンドウより見ることができます。
3. 実行を停止せずにデバッガを終了してください。
4. LED ランプがまだ点滅しているのが確認できます。
5. [Options]メニューから [Debugger]を選択して、[Download]タブより[Attach to running target]オプションを設定します。

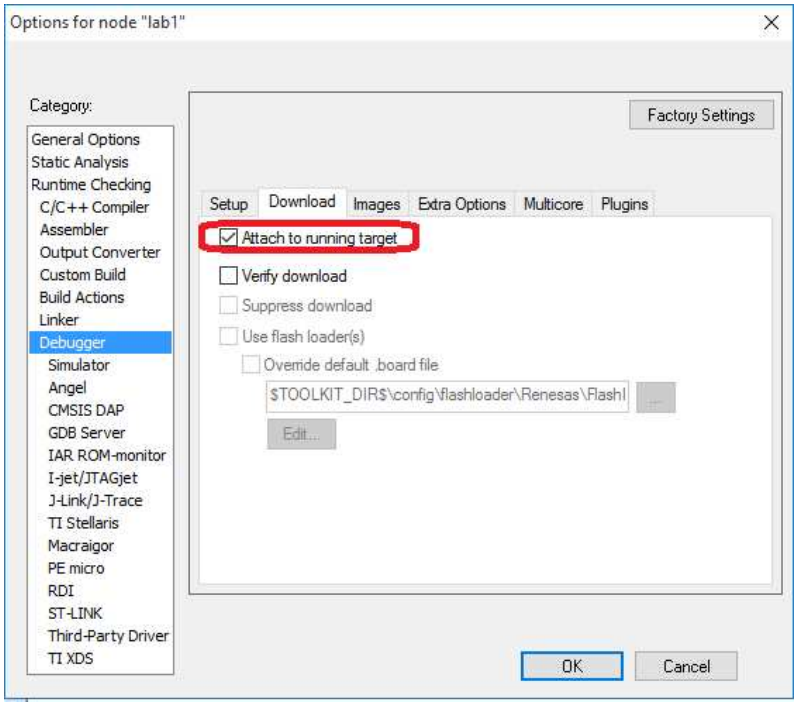


図 26　Attach to running target

6. [Ctrl+D]または[Download and Debug]ボタンを押してデバッガを再起動します。
7. ターゲットがリセットされていないためにカウンタ変数が進行しているのを見てください。ターゲットが実行中ですが、デバッガが完全にコントロールしているので、[Stop]を押すと実行中のプログラムは停止します。変数およびレジスタを全て可視化することもできます。この機能はアプリケーションのクラッシュを調べる場合にとても便利です。
8. 前の設定に戻るために、デバッガを停止して[Attach to running target]機能を無効にします。全ての機能は **Options** > **Debugger** > **Download** より無効にできます。

9.　ThreadX RTOS プラグイン

IDE の ThreadX で、組み込み式 RTOS awareness プラグインを使用するには、以下の手順を行います。

1. **Options** > **Debugger** > **Plugins** から、RTOS awareness プラグインを有効にします。

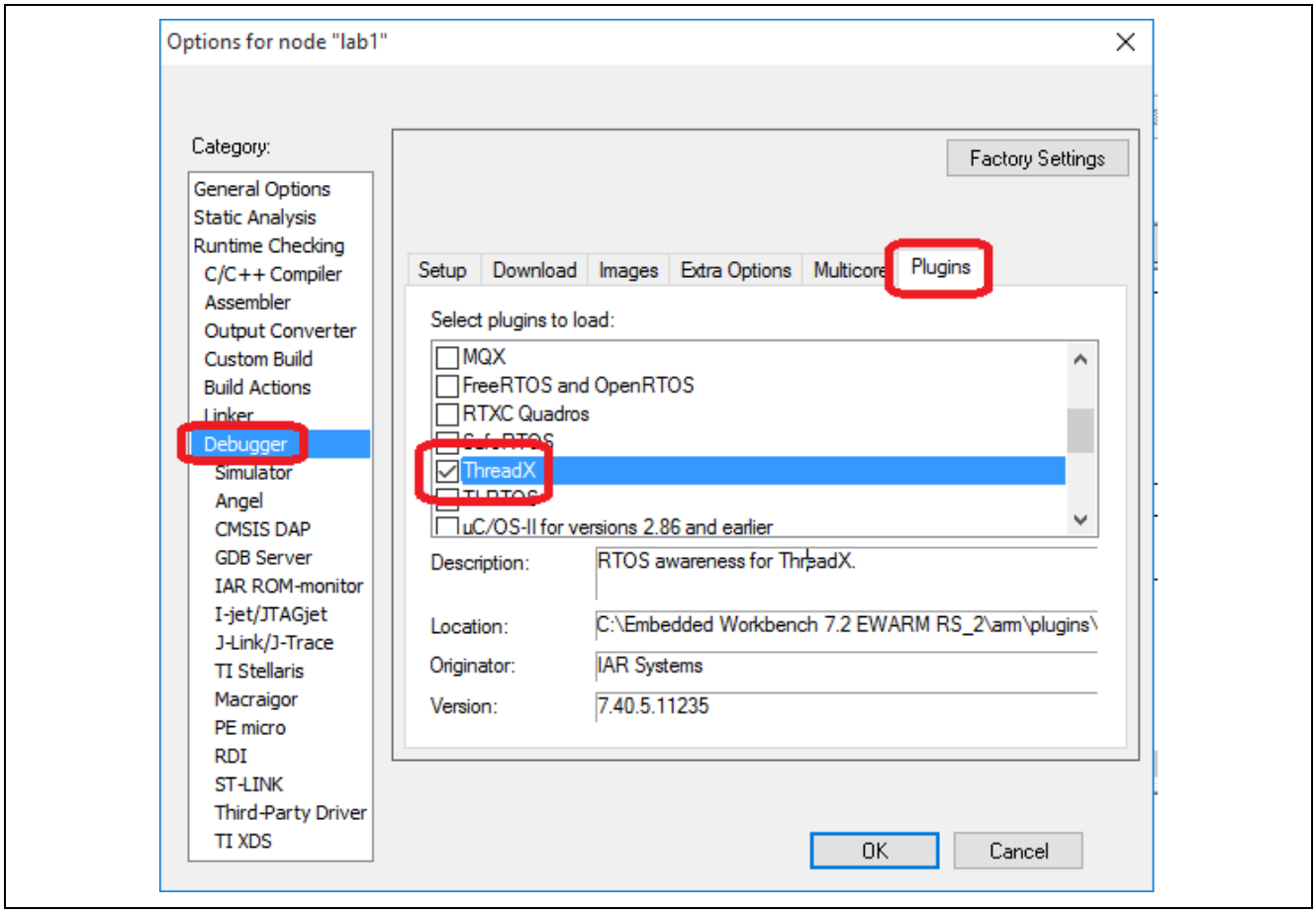


図 27　ThreadX awareness プラグイン

2. [Ctrl+D]または[**Download and Debug**]ボタンを押してデバッガを再起動します。
3. **デバッグ中に ThreadX に新しいメニューが追加されているのがわかります。**

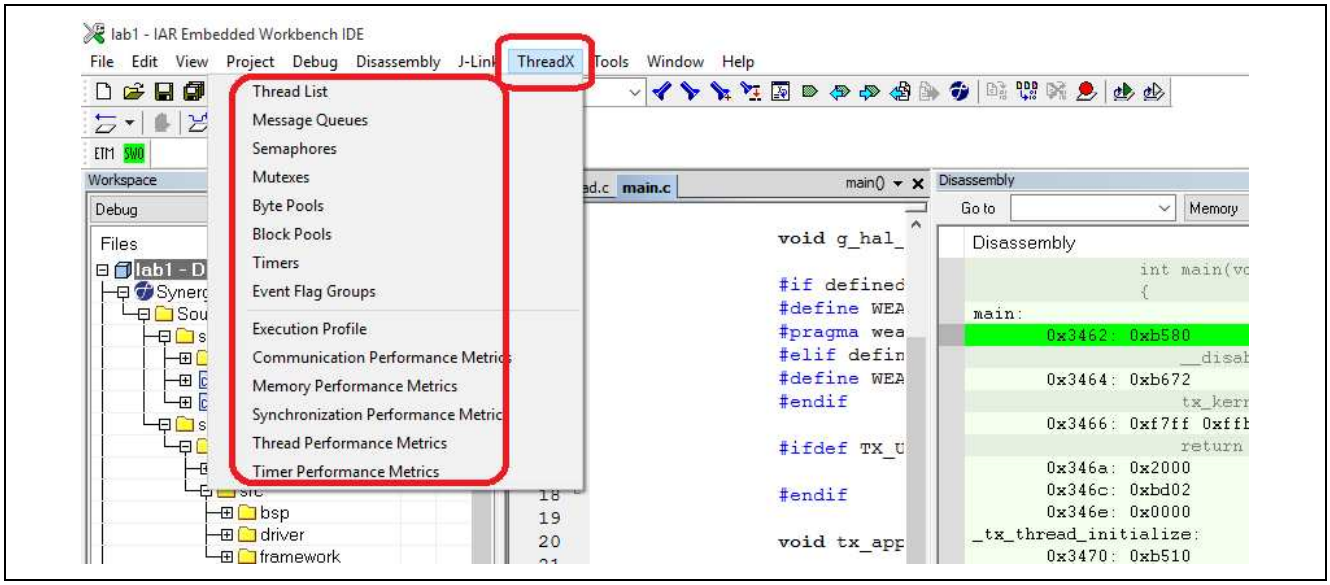


図 28　ThreadX メニュー

4. **[ThreadX] メニューから[ThreadX List]ウィンドウを開きます。**

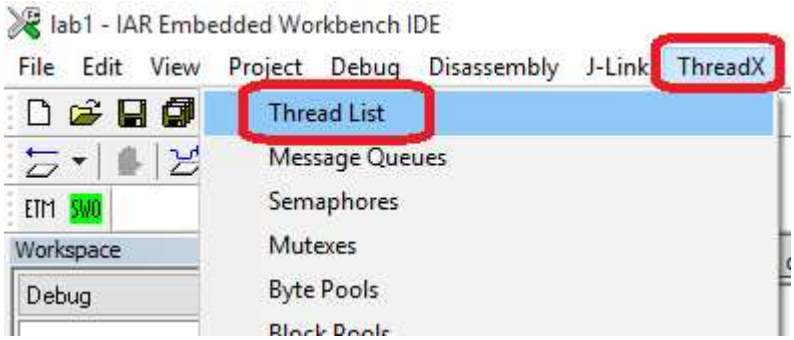


図 29 ThreadX List ウィンドウの表示

5. [ThreadX List]ウィンドウの active タスクの一覧で、スタック、最大スタック使用、スレッド状態、およびスレッドの呼び出し回数等の追加情報も表示されます。

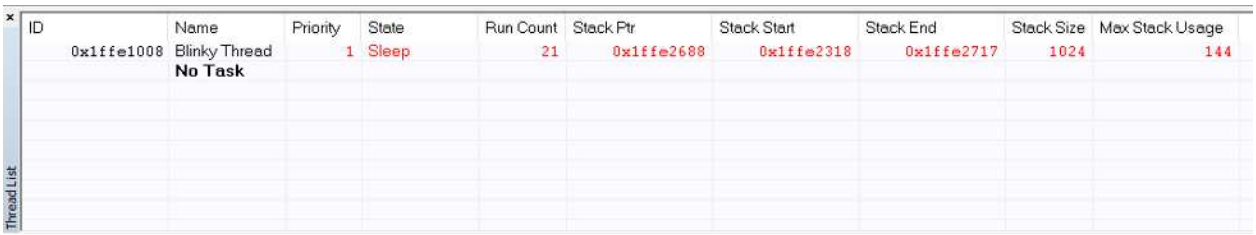


図 30 ThreadX List ウィンドウ

6. タスクがない場合は、しばらくアプリケーションを動作させるか、もしくは、少なくともマルチタスクが始まるまでアプリケーションを実行させておかなければなりません。
7. スレッド用の最大使用スタックや使用可能なスタックも見てください。アプリケーションプログラムにスレッドのサイズを設定する時に役に立ちます。
8. デバッガを停止して終了してください。

10. デバッグ中のスタック分析 (View > Stack Usage)

1. EWSYN の SSP パッケージでデバッグ中の画像スタック分析を有効にするには、下記よりデバッガの追加オプションを有効にする必要があります。**Project** > **Options** > **Debugger** > **Extra Options** > **Use command line options**:

```
--proc_stack_main=g_main_stack,g_main_stack+sizeof(g_main_stack)
```

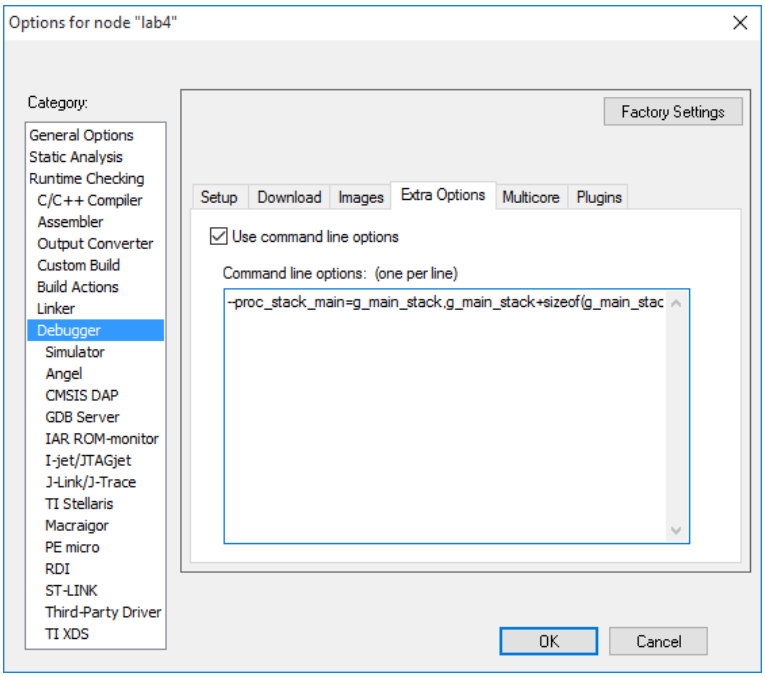


図 31 スタック分析用の Command line options

2. **Tools** > **Options** >**Stack** から**[Enable Graphical Stack display]**を選択して、追加で画像スタックディスプレイを有効にします。

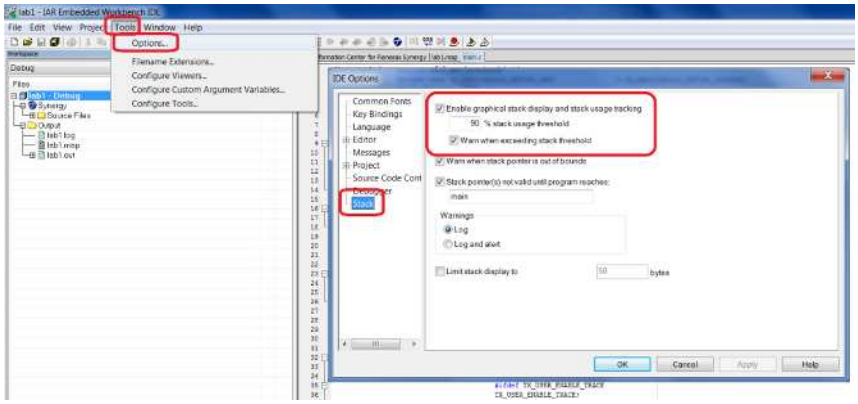


図 32 画像スタックディスプレイの有効化

3. [Ctrl+D]または**[Download and Debug]**ボタンを押してデバッガを再起動します。
4. `main()`に来たら、 **View** > **Stack** > **Stack 1** から **Stack** ウィンドウを開いてください。

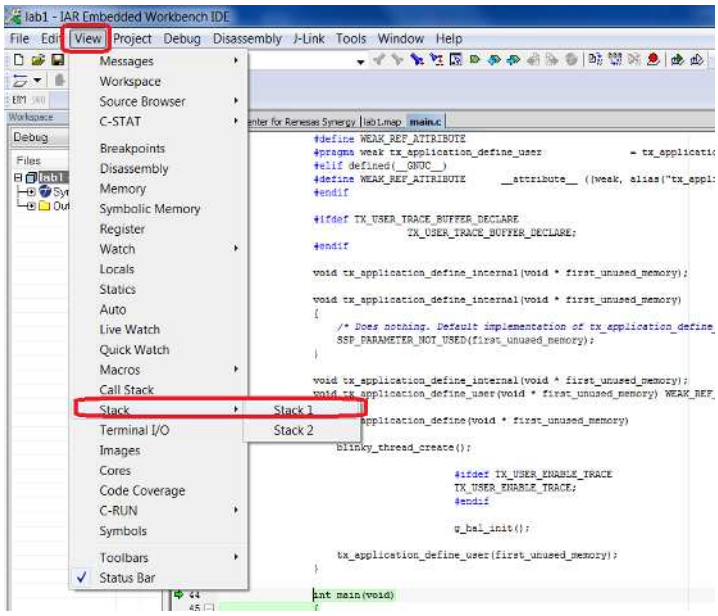


図 33 Stack ウィンドウを開く

5. アプリケーションを実行したままにしてしばらくの間(5～10 秒)おきます。
6. デバッガを停止して、[Stack]ウィンドウでスタック使用情報を見てください。スタックウィンドウの上部にグラフで最大の使用量のログが表示されます。

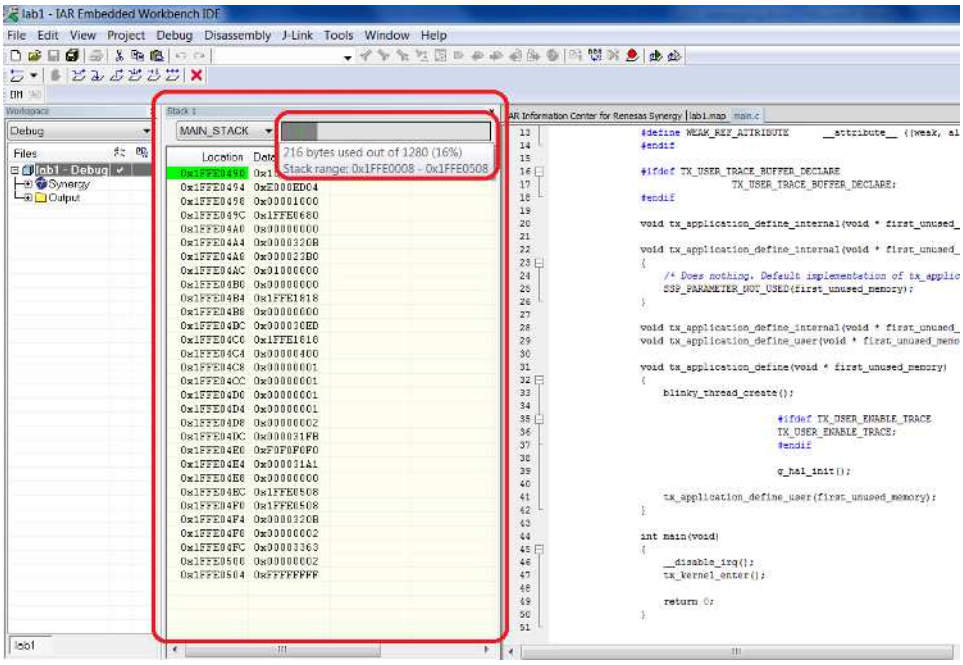


図 34 Stack ウィンドウ

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
Rev.1.00	2016.09.23	-	初版

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子
   （または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。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。
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.　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/