

Technical Note

TecStar

Silicon Labs 社 EFM8/C8051 クイックスタートガイド

2018 年 3 月

株式会社 **マクニカ**
テクスター カンパニー

クイックスタートガイド

目次

1 はじめに	4
2 開発環境のご紹介	5
2-1 ハードウェア	5
2-1-1 EFM8 Starter Kit	5
2-1-2 C8051 Development Kit	7
2-1-3 C8051 ToolStick	7
2-2 ソフトウェア	8
2-2-1 Simplicity Studio	8
3 各種ドキュメントの入手方法	10
3-1 ドキュメントの入手方法 (Simplicity Studio から)	10
3-1-1 情報が表示されない場合には?	12
3-1-2 欲しい情報が見つからない場合には?	12
3-1-3 表示される情報を制限したい場合には?	13
3-1-4 いつも使うドキュメントに素早くアクセスしたい場合には?	13
3-2 ドキュメントの入手方法 (Web から)	14
3-3 EFM8 の API 情報	15
4 ソフトウェア・インストール	16
4-1 Simplicity Studio のインストール	16
4-2 インストールがうまくいかない場合	17
4-2-1 シリコンラボ社アカウントの取得方法	17
4-2-2 企業プロキシサーバーを介して接続している場合	18
4-2-3 オフライン・インストーラ	20
4-3 KEIL コンパイラのライセンス設定	21
5 ハードウェア・セットアップ	22
5-1 EFM8 Starter Kit のセットアップ	22
5-2 C8051 Development Kit のセットアップ	22
5-3 C8051 ToolStick のセットアップ	23
6 使用方法	24
6-1 サンプルコードを動かしてみる	24
6-2 デバッグ機能を使ってみる (Debug)	29
6-3 消費電流を測定してみる (Energy Profiler)	31
6-4 ピン設定やペリフェラル設定を試してみる (Hardware Configurator)	32
6-5 ピン設定やペリフェラル設定を試してみる (Configuration Wizard 2)	35

6-6 Simplicity Studio ver.3 から ver.4 への移行	37
7 ソフトウェア設計	38
7-1 ソースコードの追いか方	38
7-2 サンプルコードにペリフェラルを実装してみる（外部割込み）	39
7-2-1 サンプルコードを理解する（EFM8BB3_Blinky）	40
7-2-2 サンプルコードを理解する（EFM8BB3_ExternalInterrupts）	42
7-2-3 ペリフェラル設定を移植する（EFM8BB3_ExternalInterrupts の設定を読み取る）	44
7-2-4 ペリフェラル設定を移植する（EFM8BB3_Blinky に設定を移植する）	47
7-2-5 アプリを実装する	50
改版履歴	52
参考文献	52

1 はじめに

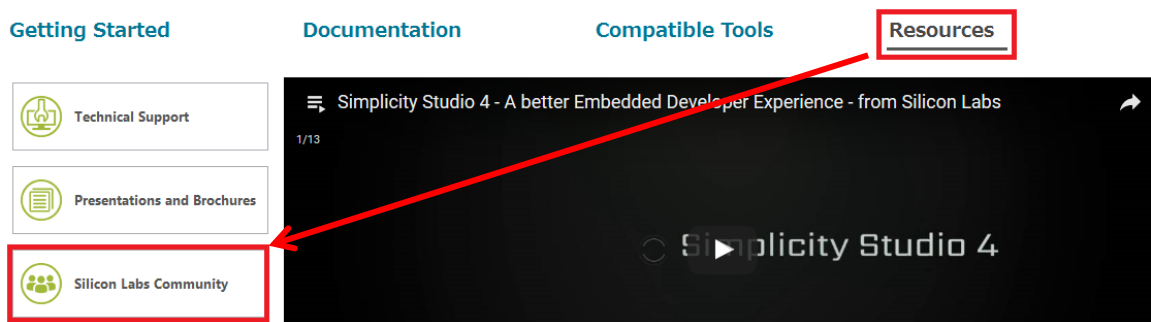
この資料は、Silicon Laboratories(以下、Silicon Labs)社製 MCU EFM8/C8051 ファミリの開発環境について簡易にまとめたものです。内容に誤りがないよう注意は払っておりますが、もし Silicon Labs 社が提供するドキュメント等と差異がございましたら、メーカー提供のものを優先してご参照ください。

また、Silicon Labs 社の ナレッジベース(FAQ)やコミュニティフォーラム(ユーザ同士で問題解決。Silicon Labs のエンジニアも頻繁にコメントしています)には、本資料で取り上げていない様々な情報が記載されております。

製品をご使用頂く過程で疑問や課題が生じることもあると思いますが、他のユーザが既に解決方法を見つけている場合も多々ございます。非常に有益ですので、ぜひご活用下さい。

◆ アクセス方法

Simplicity Studio から



Web Site から

<https://www.silabs.com/community> (Silicon Labs 社製品全般)

<https://www.silabs.com/community/mcu/8-bit/forum> (8-bit MCU に特化)

◆ 使用方法



2 開発環境のご紹介

EFM8/C8051 の開発環境について、ハードウェアとソフトウェアに分けてご紹介します。

2-1 ハードウェア

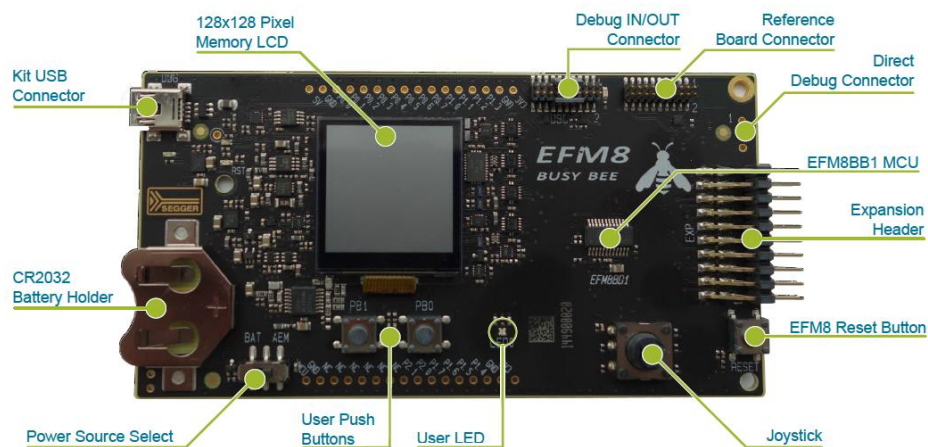
開発環境としては、EFM8 用に Starter Kit を、C8051 用に Development Kit と ToolStick を用意しています。Starter Kit と ToolStick は小型サイズで、お手軽にご評価頂けます。

2-1-1 EFM8 Starter Kit

Starter Kit は、各ファミリに 1 種ずつ用意されています。同一ファミリであっても、ROM/RAM サイズやペリフェラルの数に差異がありますが、Starter Kit にはフルセットの MCU が実装されていますので、これを用いて設計を進めて頂くことが可能です。

ファミリ名	形名	Starter Kit	実装されている型番
Busy Bee	EFM8BB1	SLSTK2020A	EFM8BB10F8G
Busy Bee	EFM8BB2	SLSTK2021A	EFM8BB22F16G
Busy Bee	EFM8BB3	SLSTK2022A	EFM8BB31F64G
Sleepy Bee	EFM8SB1	SLSTK2010A	EFM8SB10F8G
Sleepy Bee	EFM8SB2	SLSTK2011A	EFM8SB20F64G
Universal Bee	EFM8UB1	SLSTK2000A	EFM8UB10F16G
Universal Bee	EFM8UB2	SLSTK2001A	EFM8UB20F64G
Universal Bee	EFM8UB3	SLTB005A	EFM8UB31F40G
Laser Bee	EFM8LB1	SLSTK2030A	EFM8LB12F64E

◆ EFM8BB1 : SLSTK2020A



◆ EFM8UB3 : SLTB005A



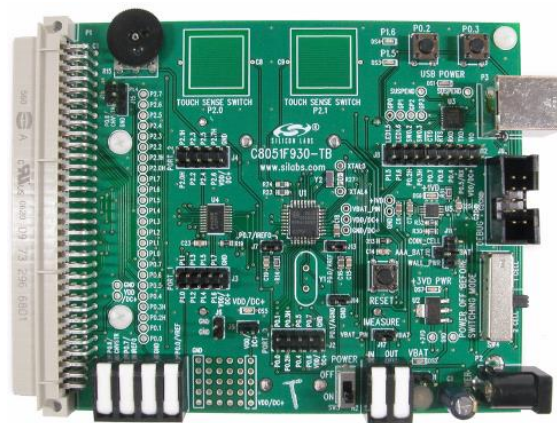
2-1-2 C8051 Development Kit

ベーシックな評価基板が入った開発キットです。ターゲットボード、USB Debug Adaptor(PC とターゲットボードとをつなぐ機材)、AC/DC アダプタが同梱されています。MCU ファミリごとに Development Kit が用意されており、ターゲットボードに実装されている MCU が異なります。

ターゲットボードには LED やスイッチが実装されている他、アナログ入力やタッチボタンなど、各 MCU ファミリが持つ特徴的な機能を評価できる作りになっています。また、全ての I/O ピンが引き出されていますので、拡張性にも富んでいます。



Development Kit

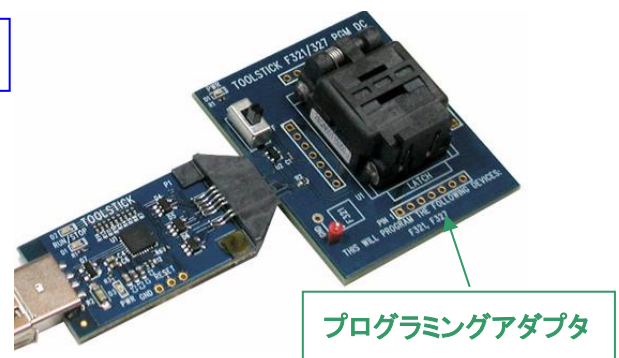
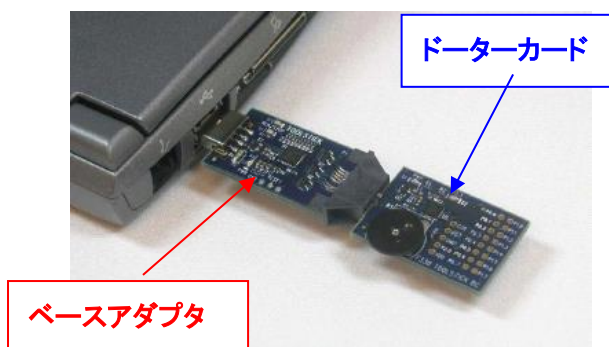


ターゲットボード

2-1-3 C8051 ToolStick

USB 給電で動作する、非常にコンパクトな評価基板です。PC に繋がるベースアダプタ、評価対象を搭載したドーターカード、の 2 つで構成されています。

ドーターカードは、MCU ファミリごと、パッケージごとに多種用意しており、評価したい MCU に最適なドーターカードをお選び頂くことができます。またソケットが載ったプログラミングアダプタも用意しており、プログラミングにご使用頂けます。



2-2 ソフトウェア

EFM8/C8051 の開発環境である Simplicity Studio を使用して設計を行うこととなります。C・アセンブラのコンパイラについては、KEIL 社のコンパイラ(フルライセンス)を無償提供しています。

2-2-1 Simplicity Studio

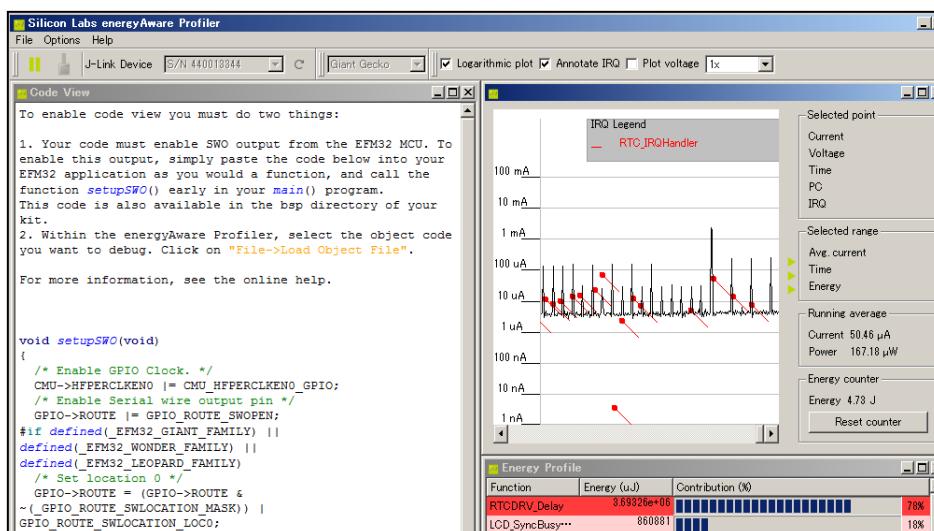
Simplicity Studio は、EFM8/C8051 をターゲットとしたコンパイル・デバッグ・プログラミングを1つのプラットフォームで提供することができるソフトウェアです。統合開発環境 (IDE)を中心に、非常に便利なツール群が充実しています。同社製の 32bit MCU や無線 MCU も同一プラットフォームで開発が可能です。



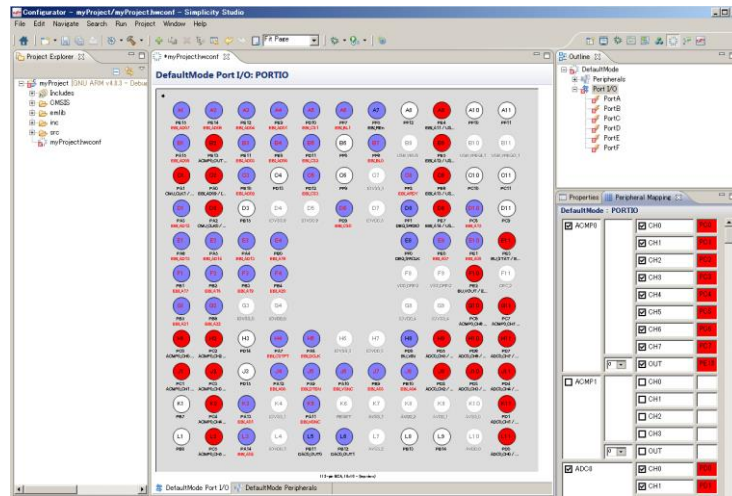
注) 画像は Simplicity Studio v3 のものです

ツール名	機能の概要
Simplicity IDE	統合開発環境 (IDE)。無償の KEIL 社コンパイラを搭載
Energy Profiler	実機の消費電流値を測定することが可能。EFM8 に対応
Hardware Configurator	ピン設定やペリフェラル設定を簡単に行うことができる。新しい製品 (EFM8、C8051 の一部) に対応
Configuration Wizard 2	ピン設定やペリフェラル設定を簡単に行うことができる。レガシー製品 (C8051) に対応
Flash Programmer	フラッシュ ROM のライト/イレース

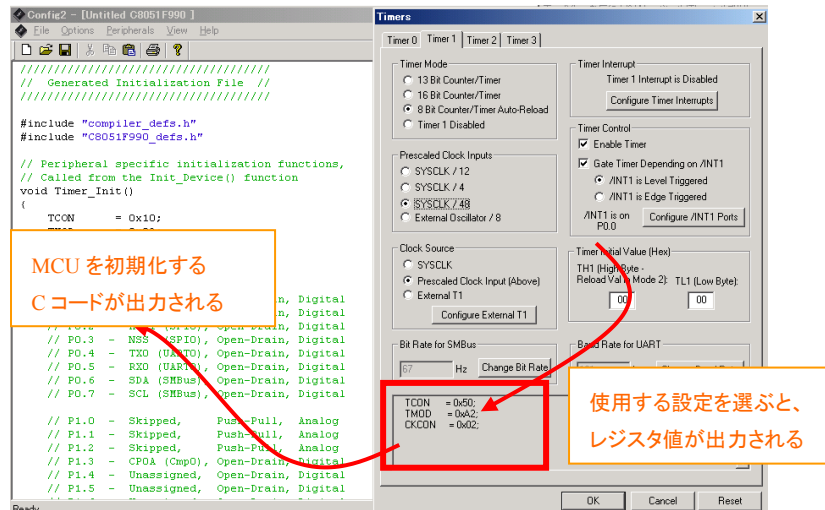
◆ 消費電流が実測できます (Energy Profiler)



- ◆ ピン設定やペリフェラル設定を簡単に行えます (Hardware Configurator)



- ◆ ピン設定やペリフェラル設定を簡単に行えます (Configuration Wizard 2)



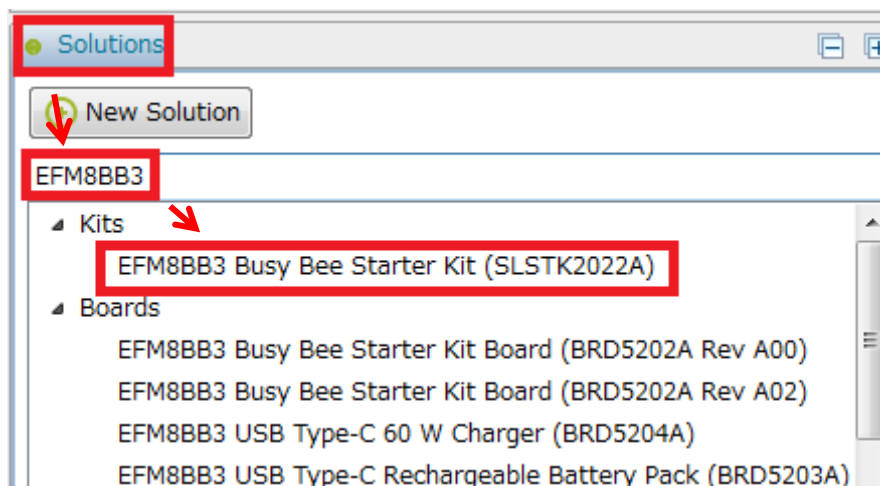
3 各種ドキュメントの入手方法

EFM8/C8051 のドキュメントの入手方法について紹介します。

3-1 ドキュメントの入手方法（Simplicity Studio から）

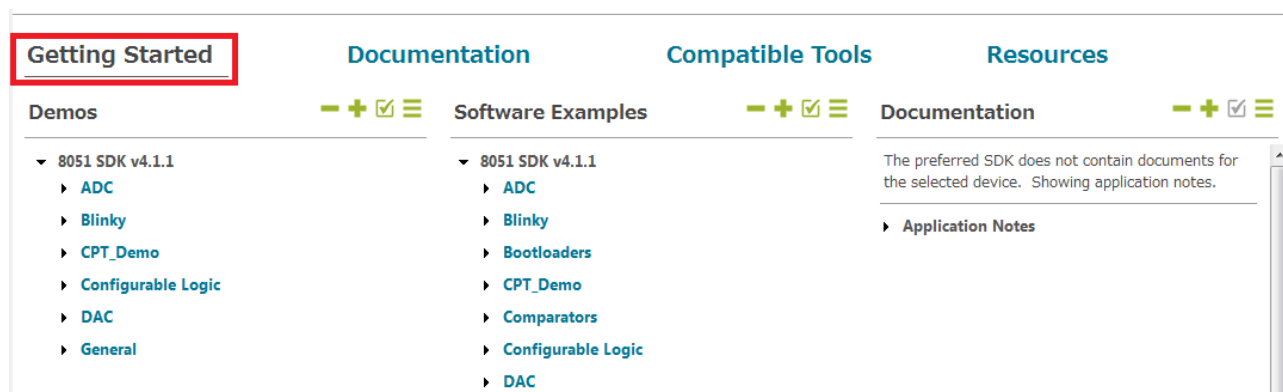
EFM8/C8051 のデータシート、リファレンス・マニュアル、エラッタ、アプリケーションノート および 評価基板 (starter kit) の回路情報などは、Simplicity Studio からご入手頂くことが可能です。

Simplicity Studio を起動し、Solutions タブ ⇒ 空欄に使用する製品型番を入力 ⇒ 候補の中から該当する型番を選択します。



製品型番を指定すると、関連するドキュメントやサンプルコードが自動でリストアップされます。情報の種別に応じて、Getting Started、Documentation、Compatible Tools、Resources というタブに分類されています。

◆ Getting Started タブ



Demos:

評価基板上で動作するデモンストレーション用のソフトです。Build することなくモジュールに書き込んで、動作を確認することができます。

Software Example:

評価ボード上で動作するサンプルコードです。ソフトの実装方法について学んだり、機能について理解したりするのに役立ちます。ペリフェラルごとにサンプルコードが用意されています。

Documentation:

Application Notes が入手できますが、後述する Documentation タブからも入手できますので、そちらで説明します。

◆ Documentation タブ

Getting Started

Documentation

Compatible Tools

Resources

My Favorite Documents - + ☑

No documents have been favorited. Click the 'Favorite' icon to add a document here.

All Documents - + ☑

8051 SDK v4.1.1
EFM8BB3 Busy Bee Starter Kit (SLSTK2022A)

- ▶ Application Notes
- ▶ Data Sheets
- ▶ Errata
- ▶ Reference Manuals
- ▶ Schematic and Layout Files
- ▶ Uncategorized Documents
- ▶ User's Guides

My Favorite Documents:

お気に入り登録したドキュメントがリストアップされます。

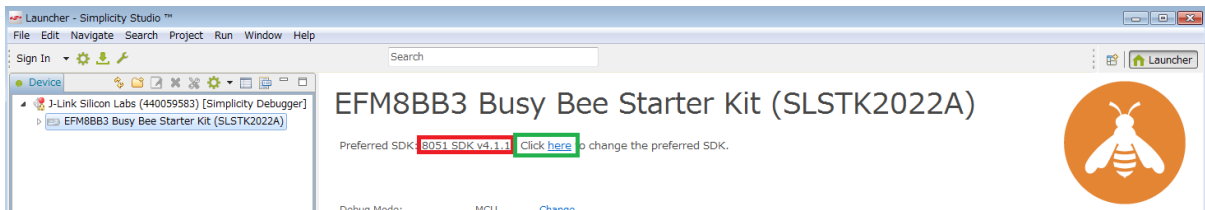
All Documents:

各種ドキュメントがまとめてあります。

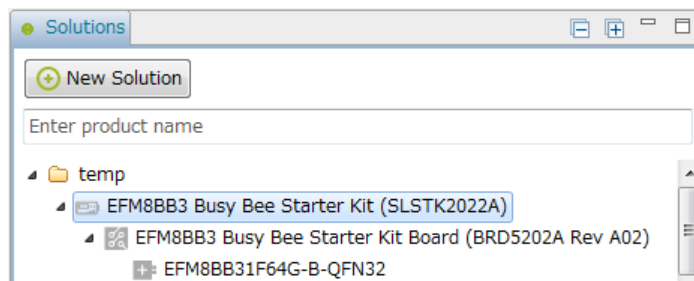
- Application Notes ... 特定の用例について記しています。各ペリフェラル(ADC やシリアルインタフェースなど)の使用方法に関する情報も用意されています。
- Data Sheets ... EFM8/C8051 のデータシート。
- Errata ... EFM8/C8051 のバグ情報。
- Reference Manual ... EFM8 の動作仕様書。C8051 ではデータシート内に記載されています。
- Schematic and Layout Files ... ラジオボードの回路図・部品表・レイアウト情報。
- Uncategorized Documents ... EFM8 など特定型番向けのライブラリ情報。
- User's Guide ... 評価ボードの取説。

3-1-1 情報が表示されない場合には？


Demos, Documentation などに情報が表示されない場合には、SDK が適正に選択されていない可能性があります。下図を参考に、Bluetooth SDK が選択されているか確認してみてください。SDK が選択されていない場合には、[Click here](#) から SDK を選択してください。

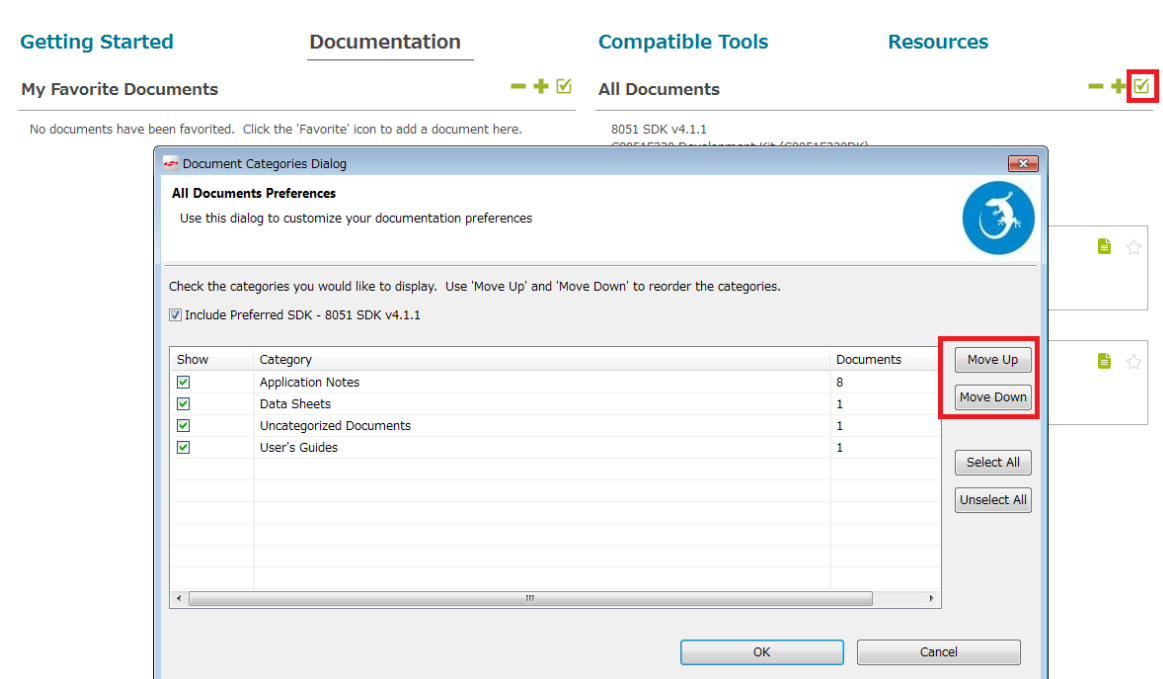


また、Solutions タブで何を選ぶかで、表示される情報も変わりますので、その点も確認ください。







3-1-2 欲しい情報が見つからない場合には？

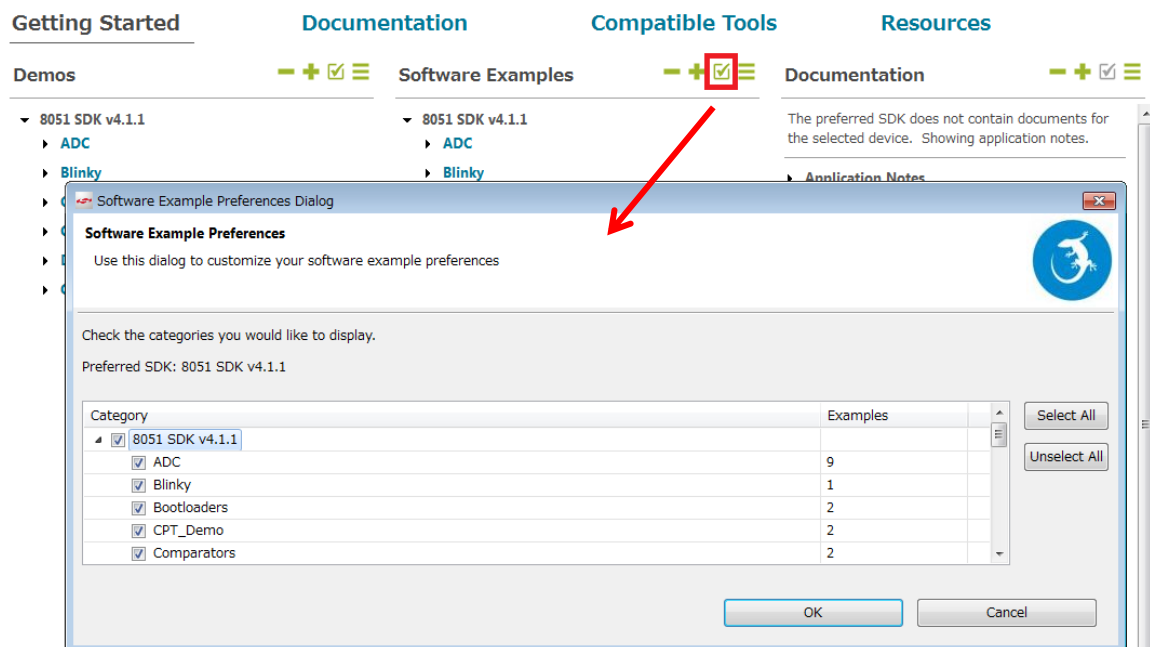
All Documents の右横にある  アイコンで、表示項目を選択したり、表示項目の並び替え (Move Up / Move Down) を行うことができます。



3-1-3 表示される情報を制限したい場合には？

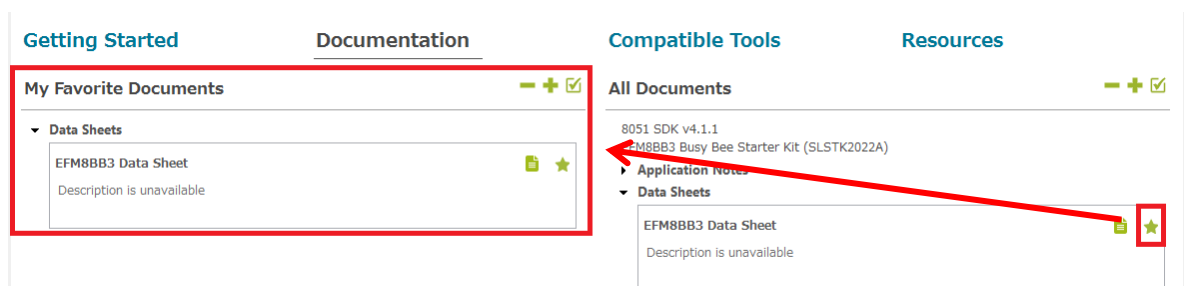
Demos, Software Examples, SDK Documentation の右横に、4つのアイコンが並んでいます。このアイコンを使うことで、表示される情報を制限することができます。

-  リストを折りたたむ
-  リストを展開する
-  表示する大項目を選択する
-  別の一覧表を表示する (Demos, Software Examples のみ)



3-1-4 いつも使うドキュメントに素早くアクセスしたい場合には？

各ドキュメントの右横にある☆印をクリックすると、☆の色が変わり、My Favorite Documents に追加されます。良く使うドキュメントを追加しておく便利です。



3-2 ドキュメントの入手方法 (Web から)

EFM8/C8051 のデータシート、リファレンス・マニュアル、エラッタ、アプリケーションノート および 評価基板 (starter kit) の回路情報などは、Silicon Labs 社の Web Site からのご入手可能です。

<http://www.silabs.com/support/pages/document-library.aspx>

Products や Resource Type で、リストアップする対象を絞り込むこともできます。

EFM8/C8051 は、Products -> Microcontrollers -> 8-bit MCUs の下に分類されています。



Technical Resource Search

Expand All / Collapse All **Showing 50 of 956 Results**

Narrow by:

Products: 8-bit MCUs

Clear All

Products

- Analog
- Audio and Radio
- Interface
- Isolation
- Microcontrollers
 - 32-bit MCUs
 - 8-bit MCUs
 - C8051F00x-01x
 - C8051F02x
 - C8051F04x

Apply text filter

Title	Version	Resource Type
1002-TCB1 D 434 Schematics and Layout		Schematic and Layout Files
1106221 F52x-F53x EOL and LTB std		Product Change Notifications (PCN)
...

Resource Type

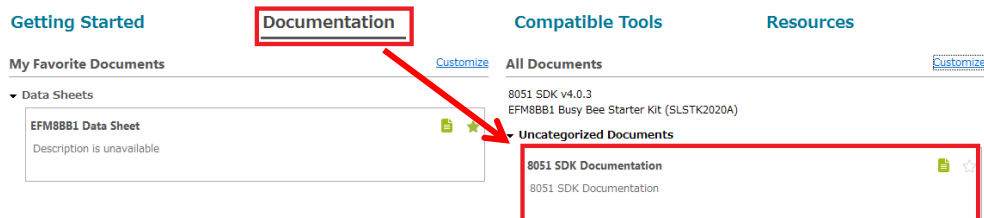
- Application Notes
- Data Sheet Addendums
- Data Sheets
- Errata
- Example Code
- Getting Started
- Manuals
- Miscellaneous
- Product Change Notifications (PCN)
- Reference Designs
- Release Notes
- Schematic and Layout Files
- Software

3-3 EFM8 の API 情報

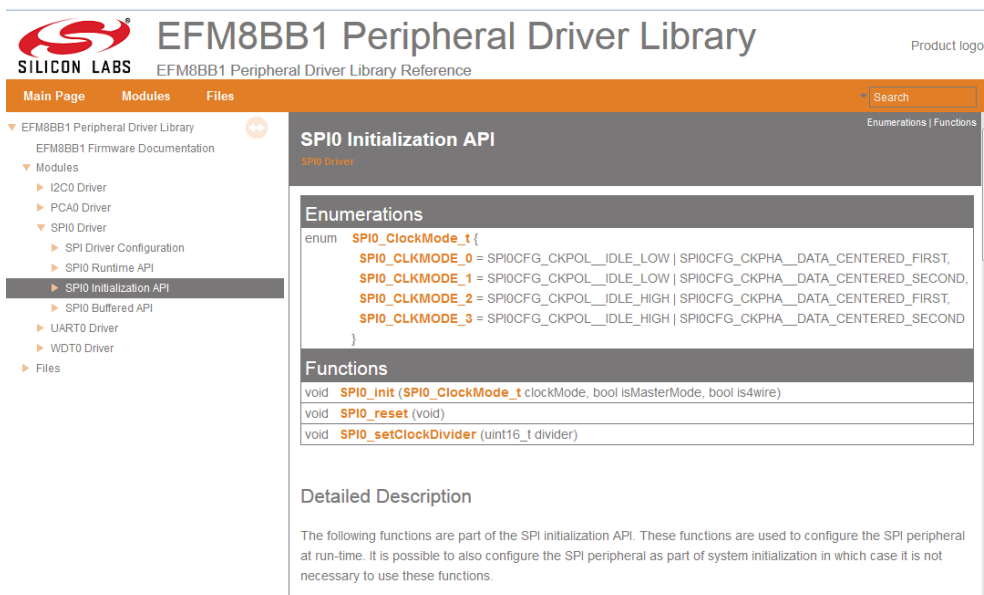
EFM8 には、ペリフェラルを制御するためライブラリ(API)が用意されており、それを使用することでソフト設計を円滑に進めて頂くことが可能です。一部 C8051 についても用意されています。

「3-1 EFM8/C8051 のドキュメント」の手順に従って、使用する型番を選択してください。

Documentation タブ ⇒ 8051 SDK documentation を選択します。



ブラウザが起動し、API 情報が表示されます。



4 ソフトウェア・インストール

EFM8/C8051 の評価に必要なソフトウェアをインストールします。

4-1 Simplicity Studio のインストール

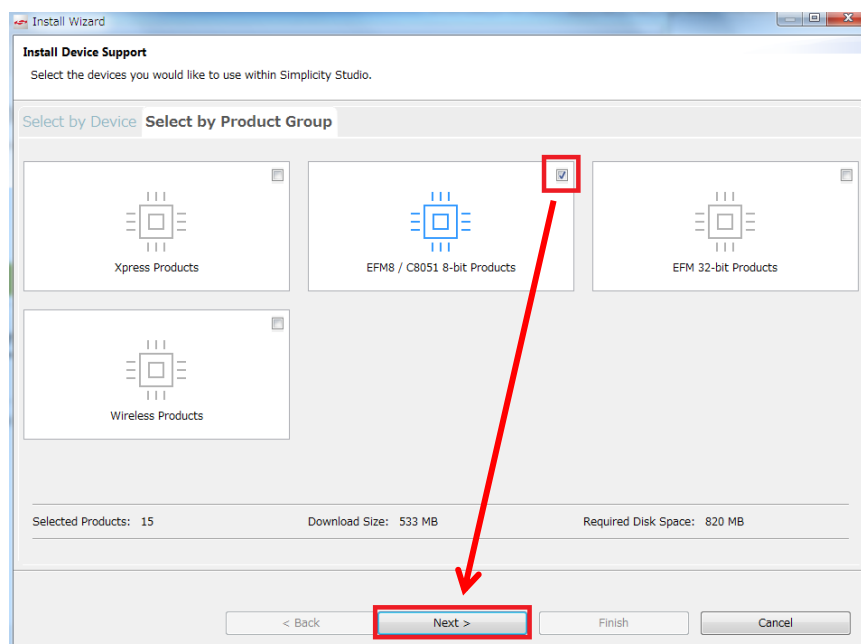
下記 URL よりインストーラをダウンロードしてください。

<http://www.silabs.com/products/mcu/Pages/simplicity-studio.aspx>

ダウンロード完了後 ”install-studio-v4_xx.exe” を起動し、インストールを開始してください。

Simplicity Studio はインターネット回線に接続した上でのインストールを想定しています。オフラインでのインストールは行えませんのでご注意ください。

インストールが進むと、Log in (サインイン) 画面が表示されますが、サインインしなくてもインストールできます。更にインストールが進むと、Install Wizard が表示されます。Select by Product Group タブに切り替え、EFM8 / C8051 8-bit Products にチェックして、Next をクリックします。



インストールを行うモジュールが自動でリストアップされます。Finish をクリックすると、残りのインストールが実行されます。以上でセットアップは完了です。

Install Wizard が起動しない場合には、Update Software アイコンをクリックしてください。



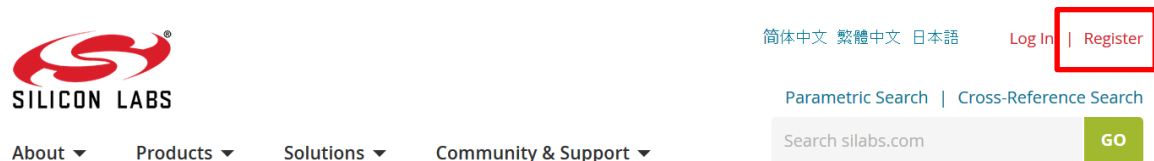
4-2 インストールがうまくいかない場合

4-2-1 シリコンラボ社アカウントの取得方法

EFM8/C8051 SDK の入手には、シリコンラボ社 WEB サイトのアカウントは不要ですが、Bluetooth SDK などのインストールを行う際には必要となります。アカウントの作成は無料です。

- ① 下記 URL にアクセスし、右上の Register からアカウント作成に進んでください。

<https://www.silabs.com/>



- ② 必要事項を入力し、Create an Account でアカウントを作成してください。

- ③ アカウントが生成できたら、念のため発行されたアカウントでログインできることを確認してください

い。下記 URL にアクセスし、右上の Log In からログインを行ってください。

<https://www.silabs.com/>



About ▾ Products ▾ Solutions ▾ Community & Support ▾

简体中文 繁體中文 日本語

Log In | Register

Parametric Search | Cross-Reference Search

Search silabs.com

GO

④ ログインに成功すると、画面右上に「Welcome, 名前」が表示されます。



About ▾ Products ▾ Solutions ▾ Community & Support ▾

简体中文 繁體中文 日本語

Welcome, [Redacted] ▾

Parametric Search | Cross-Reference Search

Search silabs.com

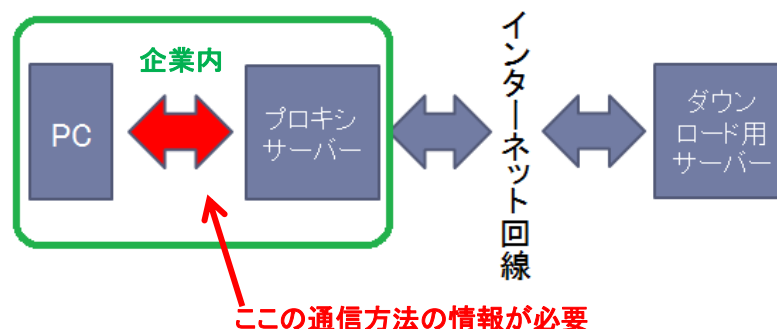
GO

4-2-2 企業プロキシサーバーを介して接続している場合

インストールにはインターネット接続が必要になりますが、プロキシサーバーを導入している企業ユーザ様の場合にはプロキシ設定が必要になる場合があります。設定内容については、自社のネットワーク管理者にご相談下さい。プロキシを介さずにインターネット回線に接続できる環境が構築できる場合には、そちらをご利用頂くのが簡単です。(WiFi ルータや自宅など)

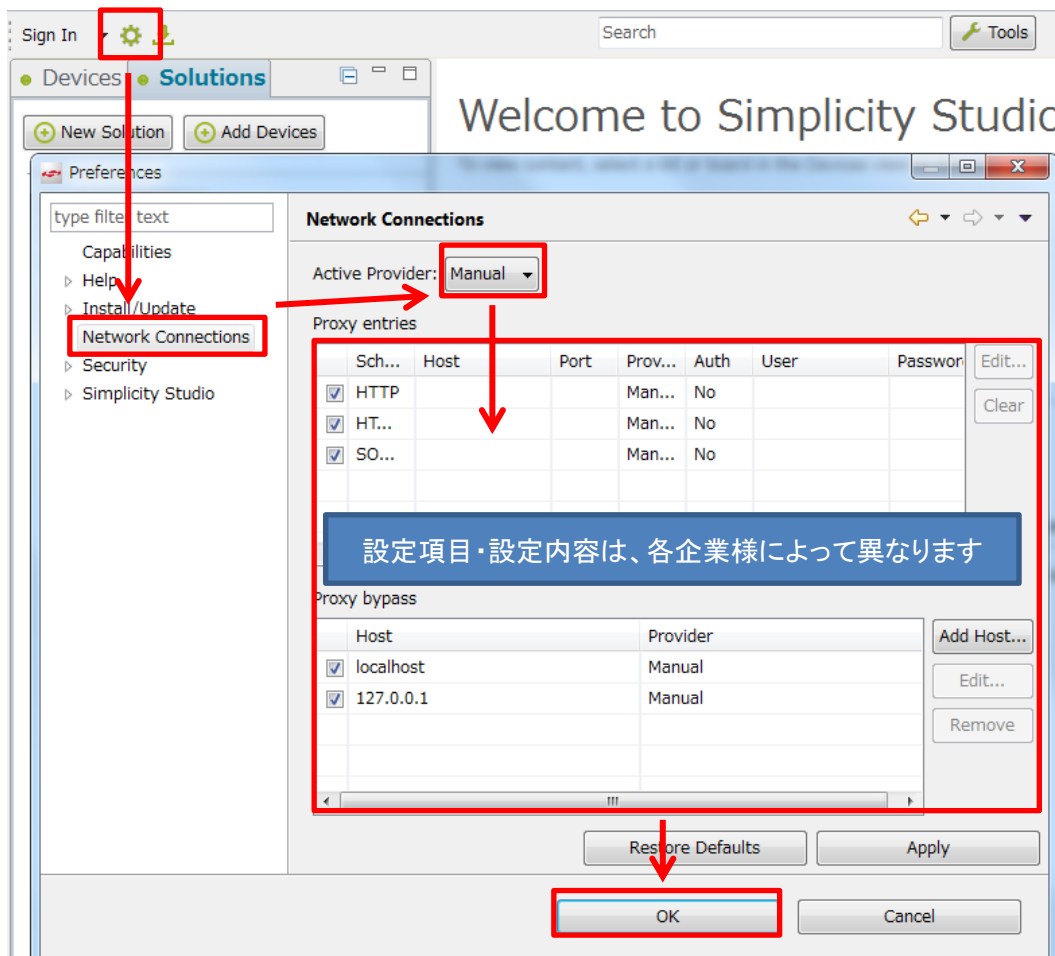
Simplicity Studio がアクセスする先については、シリコンラボ社のコミュニティフォーラムに関連情報があります。企業プロキシサーバーのセキュリティオプション(ホワイトリスト)で回避するような場合にご利用ください。

<http://community.silabs.com/t5/Simplicity-Studio-and-Software/Simplicity-Studio-v4-installation-error-download-error/thread/181331>

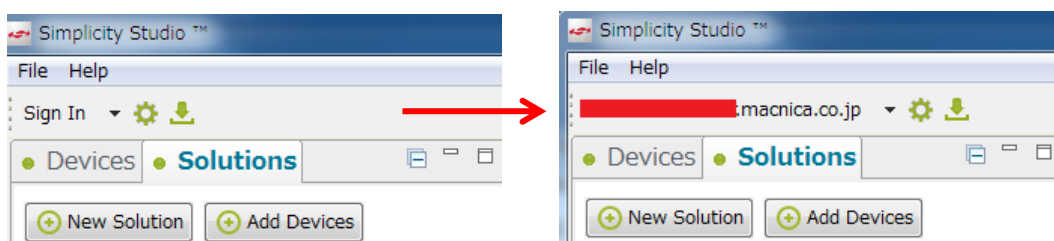


プロキシサーバーの設定は、以下の手順で行います。

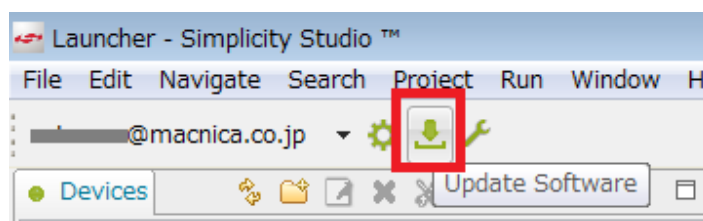
Simplicity Studio の Settings アイコンを選択し、Network Connections を選択します。プロキシ設定の画面が表示されますので、Active Provider を Manual に設定変更し、Proxy entries に必要な設定を入力してください。

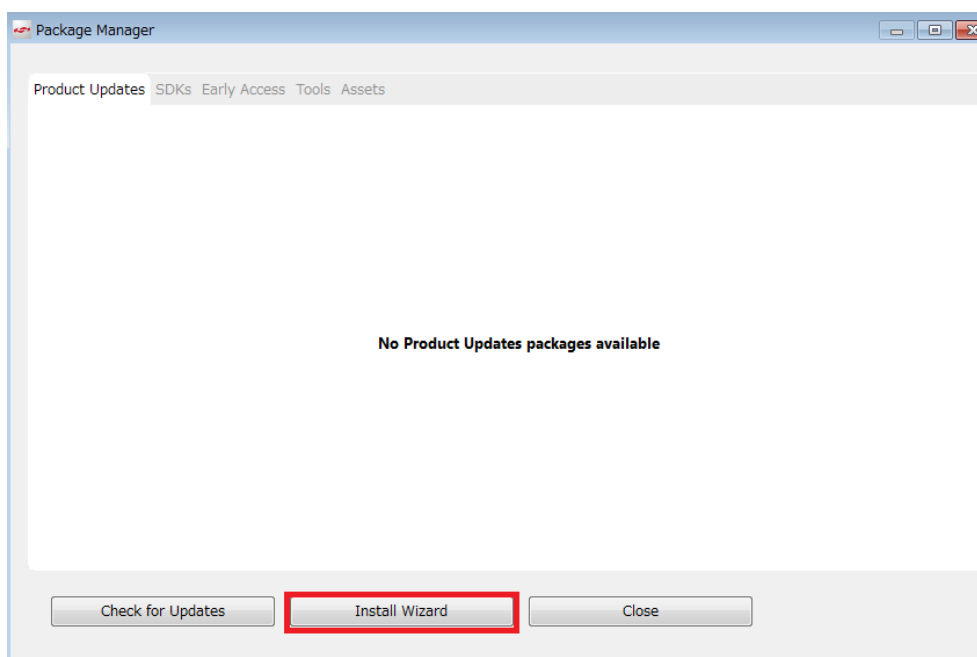


設定が終わったらログイン(Sign In)を行います。画面左上の Sign In をクリックし、シリコンラボ社 WEB サイトのアカウントを入力します。ログインに成功すると、画面左上にメールアドレスが表示されます。

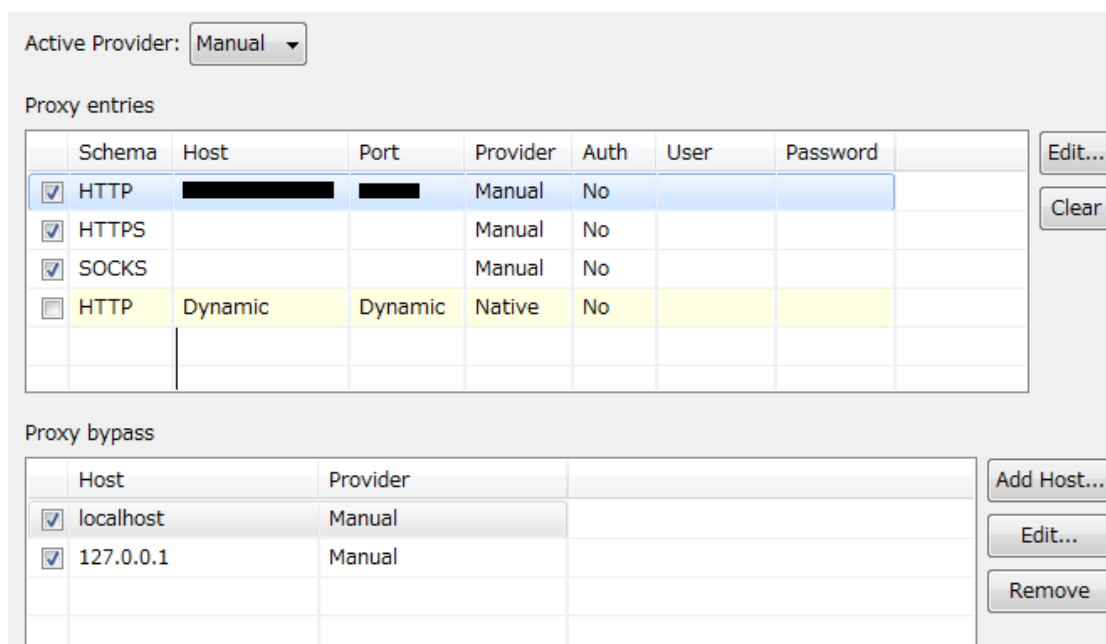


ログインに成功したら、Update Software アイコンをクリックし、Package Manager の Install Wizard からインストールが継続できます。





設定例： PC とプロキシサーバー間の通信に HTTP のみを使用している場合



4-2-3 オフライン・インストーラ

オンラインでインストールすることが望ましいですが、どうしてもプロキシの設定がうまくいかない場合には、オフライン・インストーラも活用頂けます。

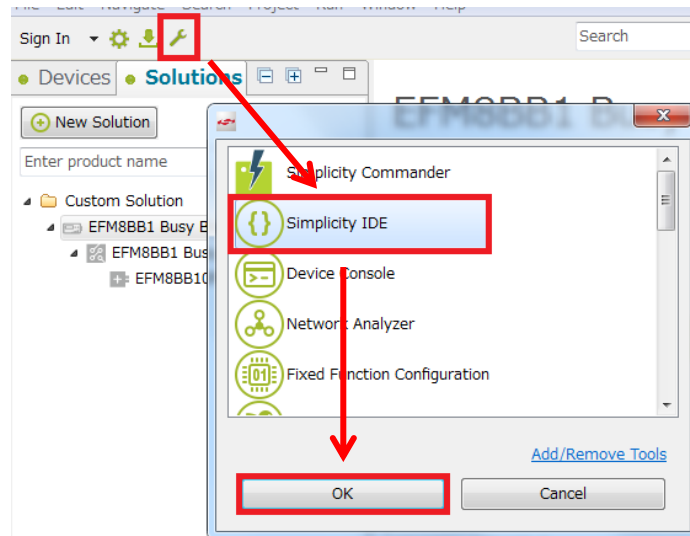
入手については、マクニカオンラインサービスの FAQ をご参照ください。

<https://service.macnica.co.jp/support/faq/125501>

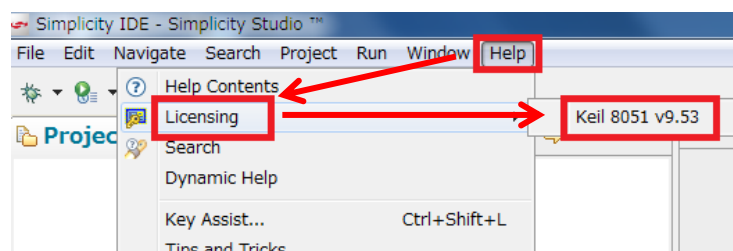
4-3 KEIL コンパイラのライセンス設定

KEIL コンパイラを利用するために、ライセンス登録を行います。

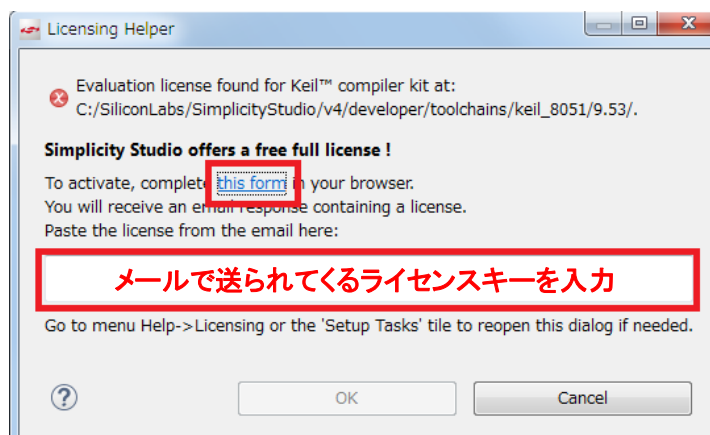
Simplicity Studio を起動し、Tools ⇒ Simplicity IDE ⇒ OK を選択します。



Help メニュー ⇒ Licensing ⇒ Keil を選択します。



Licensing Helper ウィンドウの中央にある「this form」をクリックし、必要事項を入力します。メールでライセンスキーが送られてきますので、それを Licensing Helper ウィンドウに入力してください。



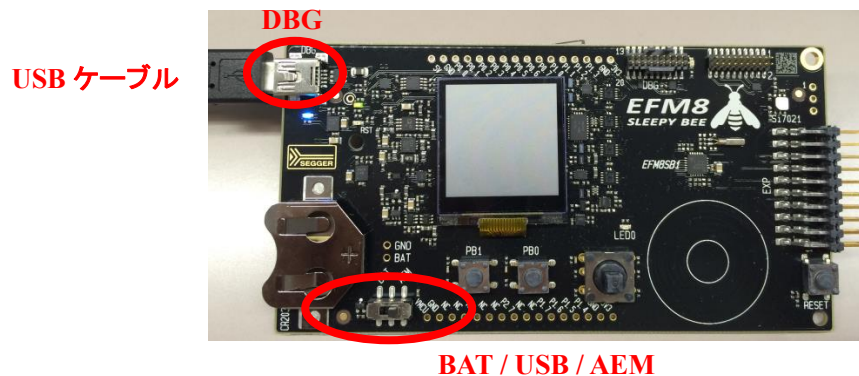
5 ハードウェア・セットアップ

EFM8/C8051 の評価に必要なハードウェアの設定を行います。

5-1 EFM8 Starter Kit のセットアップ

以下の手順で設定していきます。

1. BAT, USB, AEM の中から、基板に給電する方法を選びます。スイッチを AEM に切り替えます。
2. DBG と PC を USB ケーブルで接続します



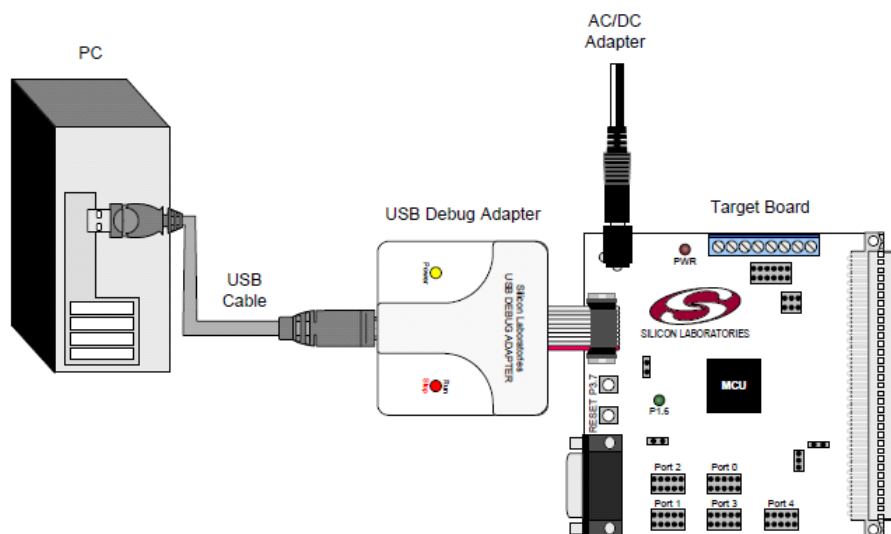
5-2 C8051 Development Kit のセットアップ

以下の手順で設定していきます。

1. USB デバッグアダプタを、ターゲットボードの DEBUG ポートに接続します。
2. USB ケーブルを、USB デバッグアダプタに接続します。
3. ターゲットボードにショートブロックがある場合には、ユーザガイドの指示に従って、正しく結線されているか確認してください。

例) F912DK, F930DK, F996DK の場合: J17 をショート、SW5 を ON に設定。

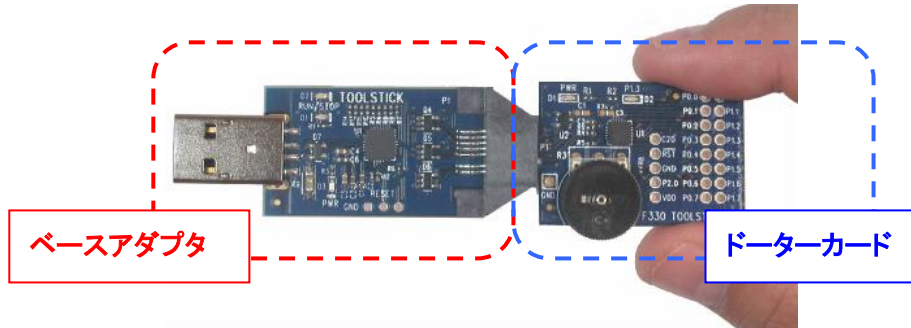
4. USB ケーブルのもう一方を PC に接続します。
5. AD/DC アダプタをターゲットボード、コンセントに接続します。



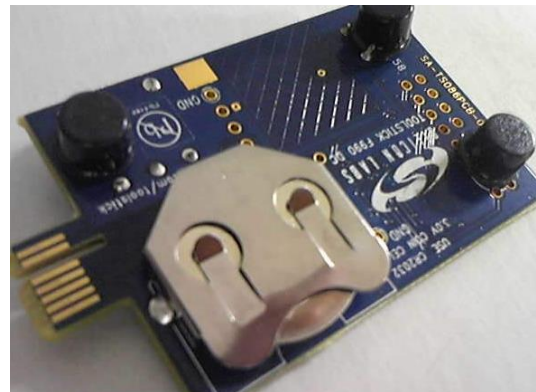
5-3 C8051 ToolStick のセットアップ

以下の手順で設定していきます。

1. ベースアダプタとドーターカードとを、下の写真のように接続します。



2. F912DC や F990DC など、いくつかのドーターカードでは電池でも動作するようになっています。USB 給電で動作するか、電池で動作するかを切り変えるスイッチがありますので、USB 給電 (TS PWR) を選択して下さい。

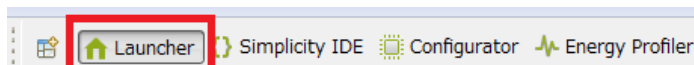


給電用スイッチ

3. USB コネクタを PC に接続します

6 使用方法

評価キットとSimplicity Studioを使用した評価手順をご紹介します。ここではSLSTK2010A (Sleepy Bee) を使用しておりますが、他の評価キットでも手順は同じです。なお、各ツールからSimplicity Studio のトップ画面に戻るには、画面右上の Launcher アイコンを使用します。

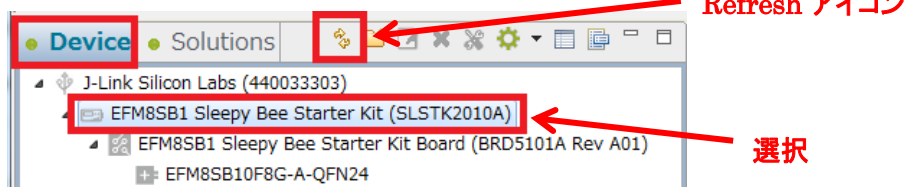


6-1 サンプルコードを動かしてみる

Starter Kit 上の LED を点滅させるサンプルコードを、ダウンロードして動作を見てみます。

Starter Kit を PC に接続すると、Simplicity Studio が Starter Kit を自動認識します。Device タブに接続した Starter Kit および MCU の名称が表示されますので、Starter Kit を選択してください。

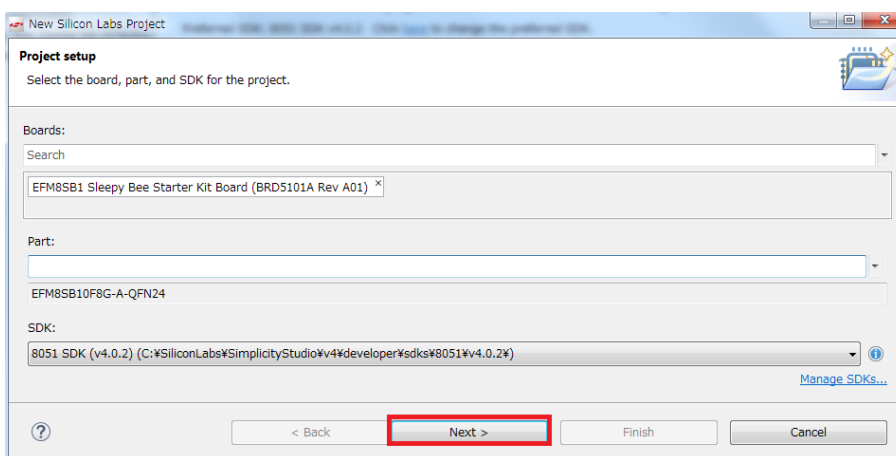
うまく認識してくれない場合には、Refresh アイコンを押してみてください。



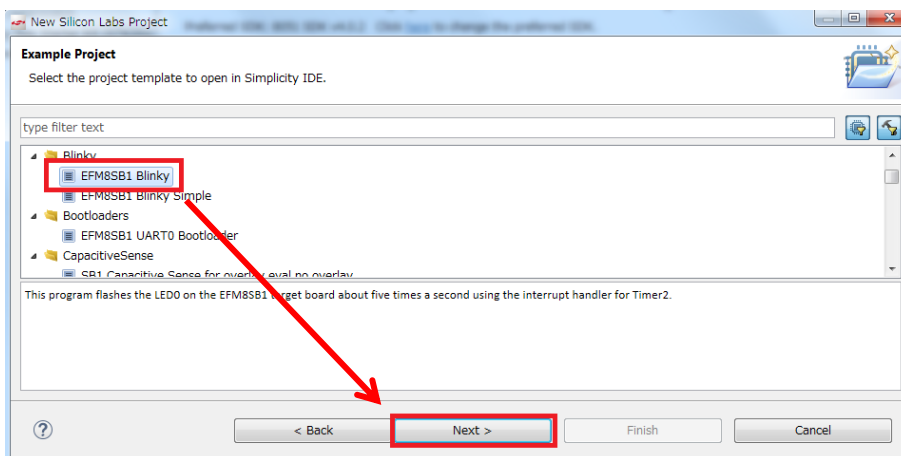
Getting Started タブ ⇒ Software Examples 横の View All Software Examples を選択します。



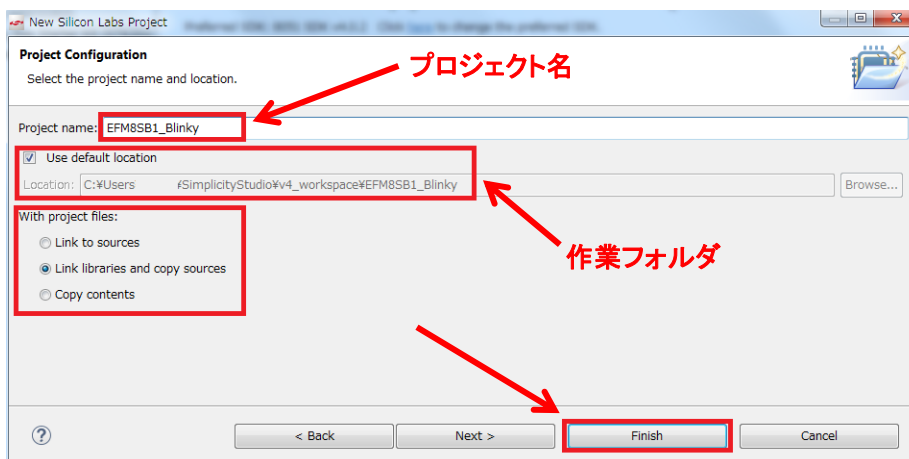
接続した Starter Kit に合わせて、Boards, Part, SDK が自動で選ばれますので、Next をクリックします。



Example Project で EFM8SB1 Blinky を選択し、Next をクリックします。

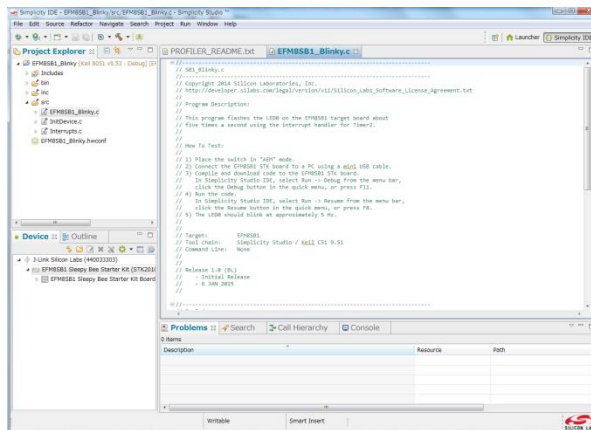


プロジェクト名を入力し、作業フォルダを指定します。With project files では、サンプルコードをローカルにコピーして使うかどうかを指定します。指定が終わったら、Finish をクリックします。

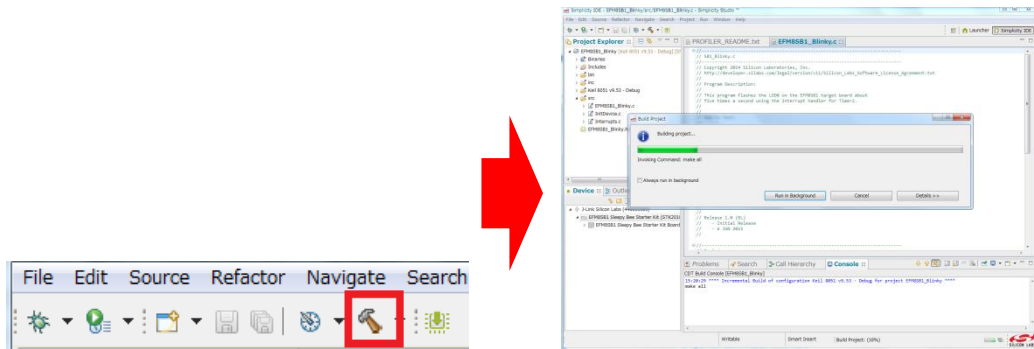


With project files	内容
Link to sources	ライブラリもソースも、オリジナルのものを使う。ライブラリもソースも修正しない人向け。
Link libraries and copy sources	ライブラリはオリジナルのものを参照し、ソースコードはローカルにコピーして使う。
Copy contents	ライブラリもソースも、ローカルにコピーして使う。ライブラリを修正する可能性がある人向け。

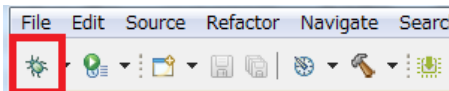
サンプルコードの準備が整うと、Simplicity IDE が起動します。Simplicity IDE の使い方については「6-2 デバッグ機能を使ってみる」で紹介しています。



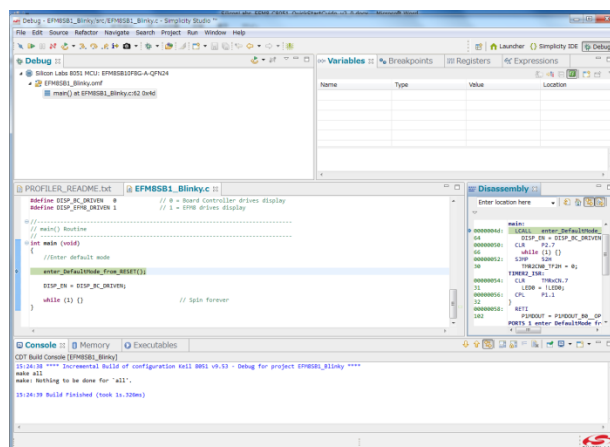
この Simplicity IDE を使用して、サンプルコードをビルドし、Starter Kit にダウンロードします。まずはトナチのアイコン (Build) をクリックします。コンパイラが走り、サンプルコードがビルドされます。



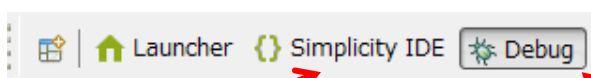
ビルドが完了したら、次に虫のアイコン (Debug) をクリックし、Starter Kit にダウンロードします。



ダウンロードが完了すると、デバッグ用の画面に切り替わります。



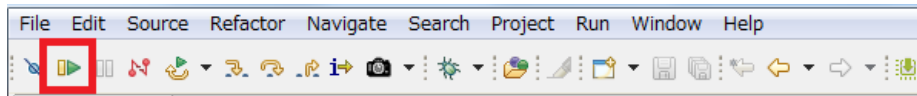
なお、ビルド用の画面と、デバッグ用の画面の切り替えは、ウィンドウ右上のアイコンで行います。



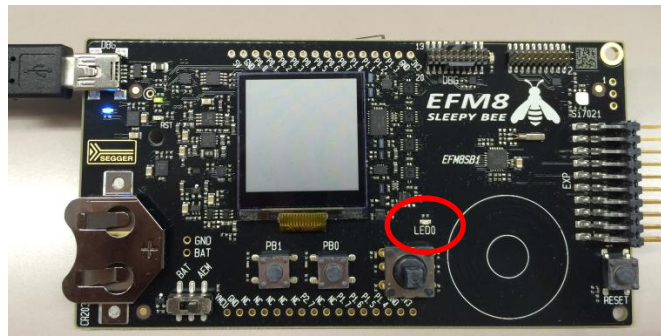
Simplicity IDE (ビルド用)

Debug (デバッグ用)

サンプルコードを実行します。下図の実行のアイコン (Resume) をクリックしてください。

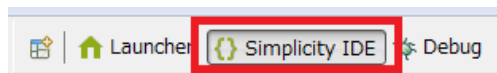


Starter Kit 上の LED が、ゆっくりと点滅しているのが確認できます。SLSTK2010A の場合には、LCD 下の LED0 が点滅します。

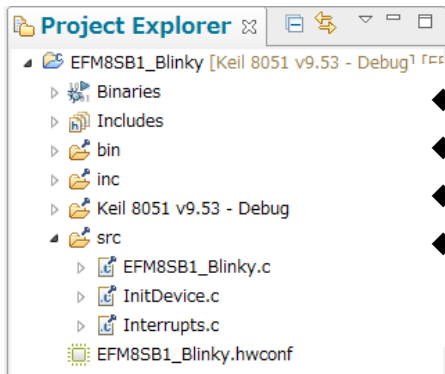


◆ LED の点滅スピードを変更してみましょう。

Simplicity IDE アイコンをクリックして、ビルド用の画面に切り替えます。



画面左に Project Explorer があり、ソースコードの階層が表示されています。



- ◆ EFM8SB1_Blinky.c: メインルーチン
- ◆ InitDevice.c: 主にペリフェラルの初期化
- ◆ Interrupts.c: 割り込み処理
- ◆ EFM8SB1_Blinky.hwconf: Hardware Configurator のプロジェクトファイル

main()を見ると LED 点滅に関する処理は行われておらず、Interrupts.c を見ると Timer2 割り込みで LED への制御ピンを反転させています。Timer2 のオーバフロー周期を変更すれば点滅スピードを変えられそうです。

```

//-----
// main() Routine
//-----
int main (void)
{
    //Enter default mode
    enter_DefaultMode_from_RESET();

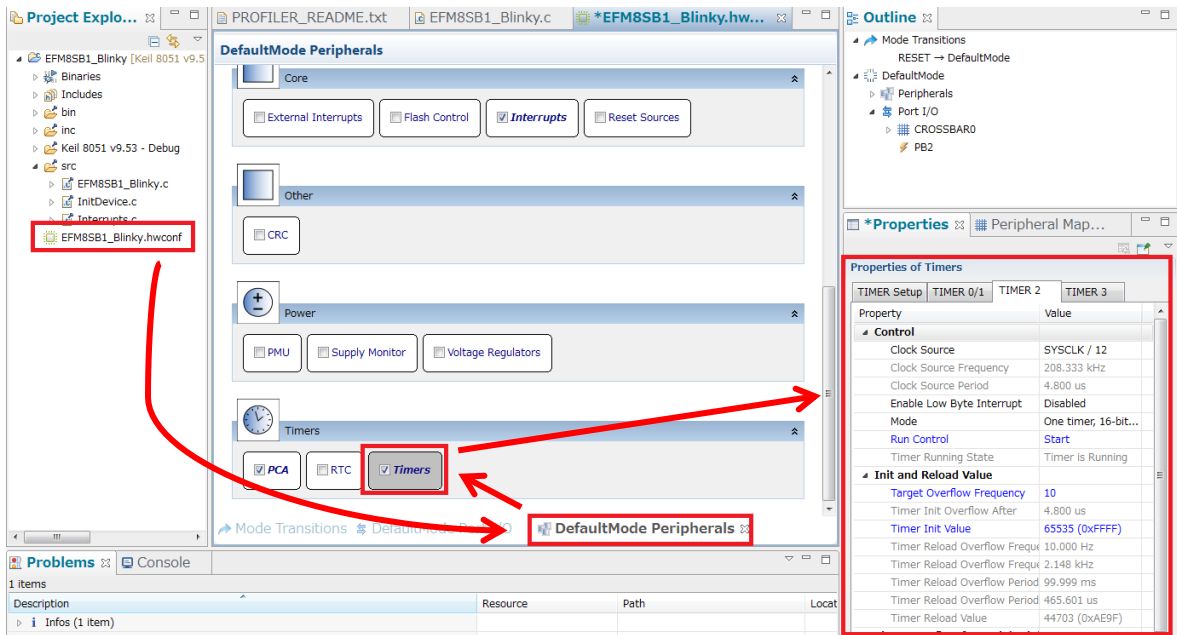
    DISP_EN = DISP_BC_DRIVEN;

    while (1) {} // Spin forever
}

//-----
// TIMER2_ISR
//-----
// TIMER2 ISR Content goes here. Remember to clear flag bits:
// TMR2CN0::TF2H (Timer # High Byte Overflow Flag)
// TMR2CN0::TF2L (Timer # Low Byte Overflow Flag)
//-----
SI_INTERRUPT (TIMER2_ISR, TIMER2_IRQn)
{
    TMR2CN0_TF2H = 0; // clear Timer2 interrupt flag
    LED0 = !LED0; // change state of LEDs
}
    
```

もちろんソースコードを追って、設定変更する方法もありますが、ここでは Hardware Configurator を使ってみます。FM8SB1_Blinky.hwconf をダブルクリックすると Hardware Configurator が起動します。

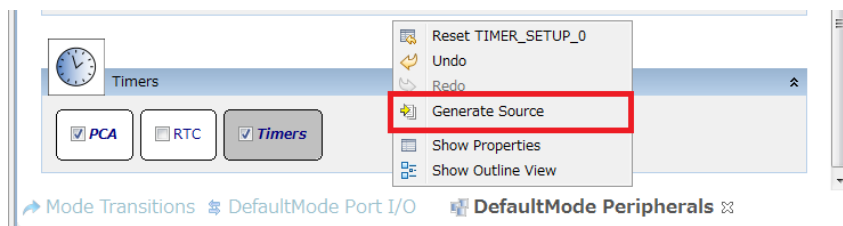
DefaultMode Peripherals タブを選択し、Timers を選択、画面右に Timer 設定が表示されます。



Timer2 の初期値設定を見てみると、Timer2 のオーバフロー周期が 10Hz に設定されていました。これを 50Hz に変更してみます。

Init and Reload Value		Init and Reload Value	
Target Overflow Frequency	10 (0xA)	Target Overflow Frequency	50 (0x32)
Timer Init Overflow After	4.800 us	Timer Init Overflow After	4.800 us
Timer Init Value	65535 (0xFFFF)	Timer Init Value	65535 (0xFFFF)
Timer Reload Overflow Frequency	10.000 Hz	Timer Reload Overflow Frequency	49.996 Hz
Timer Reload Overflow Frequency(Low Byte)	2.148 kHz	Timer Reload Overflow Frequency(Low Byte)	2.934 kHz
Timer Reload Overflow Period	99.999 ms	Timer Reload Overflow Period	20.002 ms
Timer Reload Overflow Period(Low Byte)	465.601 us	Timer Reload Overflow Period(Low Byte)	340.801 us
Timer Reload Value	44703 (0xAE9F)	Timer Reload Value	61369 (0xEF89)

設定変更をソースコードに反映させます。DefaultMode Peripherals タブに戻り、画面上で右クリックし、Generate Source を選択します。

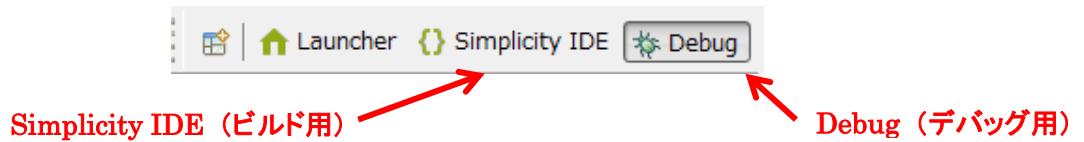


あとは、ビルドして、ダウンロードして、実行します。先ほどと同じ手順で、トンカチのアイコン (Build) ⇒ 虫のアイコン (Debug) ⇒ 実行のアイコン (Resume) の順にクリックします。Starter Kit の LED の点滅が、先ほどよりも早くなったことを確認できるかと思ます。

使用する MCU ファミリーによってサンプルコードの内容は異なりますが、Blinky.c のようにシンプルなサンプルコードは、制御方法を理解するのに最適です。

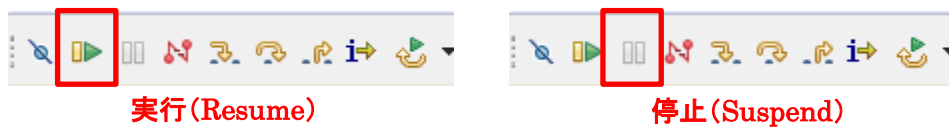
6-2 デバッグ機能を使ってみる (Debug)

Debug では、ブレークポイント、ステップ実行などのソフトウェア・デバッグの機能がご使用になれます。「6-1 サンプルコードを動かしてみる」でも紹介しましたが、ビルド用の画面 (Simplicity IDE) と、デバッグ用の画面 (Debug) は右上のアイコンで切り替えます。



◆ コードの実行・停止

コードの実行には Resume ボタン、停止には Suspend ボタンを使用します。



◆ ハードウェア・リセット

MCU にハードウェア・リセットをかけます。



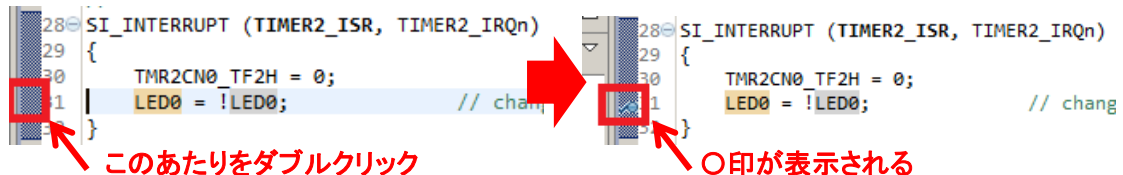
◆ デバッグ経路の切断

デバッグモード経路を切断して、デバッグ用の画面を終了します。ビルド用の画面に切り替わります。



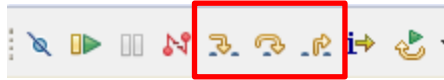
◆ ブレークポイント

ブレークポイントを設定するには、停止させたい行の左横をダブルクリックします。設定されると、水色の小さな○印が表示されます。再度ダブルクリックすれば解除されます。



◆ ステップ実行

各種ステップ実行に対応しています。



実機で実際に動作を見て頂くのが、判りやすいです。



◆ レジスタ値の閲覧・変更

レジスタ・変数の閲覧や変更は、下のウィンドウ (Register ウィンドウなど) で行うことができます。前回の停止から、値が変化した場合には黄色で表示されます。

Name	Value	Description
▶ SFR Paging		SFR Paging Registers
▶ SMBus 0		SMBus 0 Registers
▶ SPI 0		SPI 0 Registers
▶ Temperature Sensor		Temperature Sensor Registers
▶ TIMER 0/1		TIMER 0/1 Registers
▶ TIMER 2		TIMER 2 Registers
▶ TMR2	0x0	Timer 2 Word
▶ TMR2CN0	0x0	Timer 2 Control 0
▶ TMR2RL	0x0	Timer 2 Reload Word
▶ TIMER 3		TIMER 3 Registers
▶ TIMER Setup		TIMER Setup Registers
▶ UART 0		UART 0 Registers
▶ Supply Monitor		Supply Monitor Registers
▶ Voltage Reference		Voltage Reference Registers

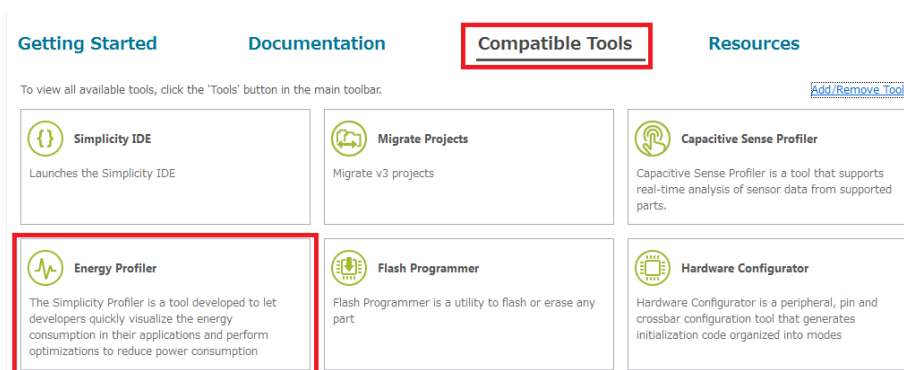
▶ TIMER 2		TIMER 2 Registers
▶ TMR2	0xFE10	Timer 2 Word
▶ TMR2CN0	0x44	Timer 2 Control 0
▶ TMR2RL	0xEFB9	Timer 2 Reload Word

6-3 消費電流を測定してみる (Energy Profiler)

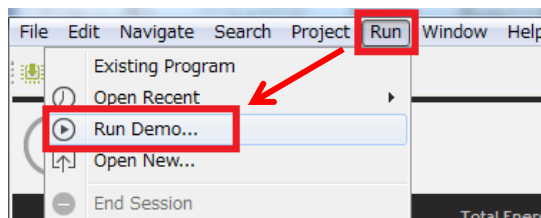
EFM8 Starter Kit には電流センサが搭載されており、消費電流測定ツール (Energy Profiler) と組み合わせることで nA レベルでの電流測定が可能です。Starter Kit には LCD など外部部品も実装されていますが、MCU 単体の消費電流が測定できるように配慮されています。ただし C8051 の評価キットは対応していません。

ここではサンプルコードを使用して、消費電流測定ツール (Energy Profiler) の使用方法をご紹介します。

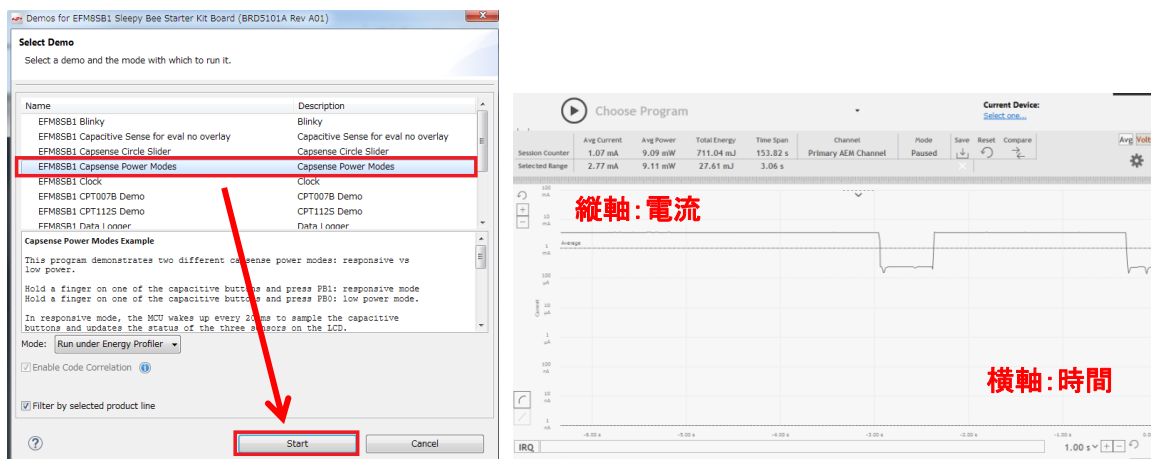
Compatible Tools の中にある Energy Profiler を起動します。



Run メニューから、Run Demo を選択します。



EFM8SB1 Capsense Power Modes を選択します。そして Start をクリックします。

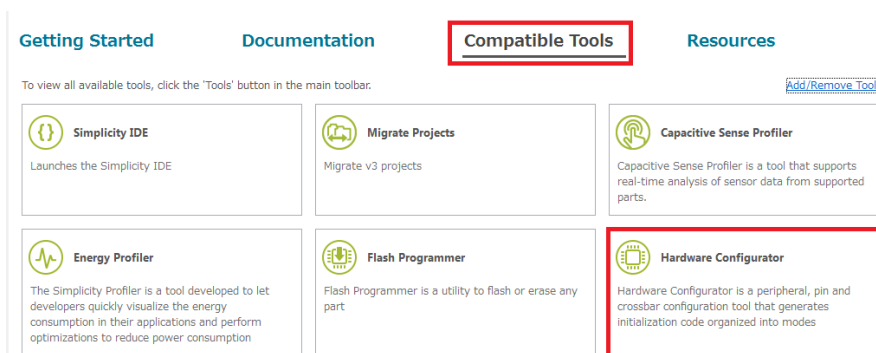


このサンプルコードは、ボタンで EFM8 のパワーモードを切り替えて、その際の消費電流をモニタできる機能です。EFM8 の Starter Kit には、消費電力のモニタ機能がついています。

6-4 ピン設定やペリフェラル設定をしてみる (Hardware Configurator)

レジスタ設定を補助するツールとして Configurator(新しい製品向け)および Configuration Wizard2 (レガシー製品向け)が用意されています。ここでは、Configurator の使用方法を簡単にご紹介します。

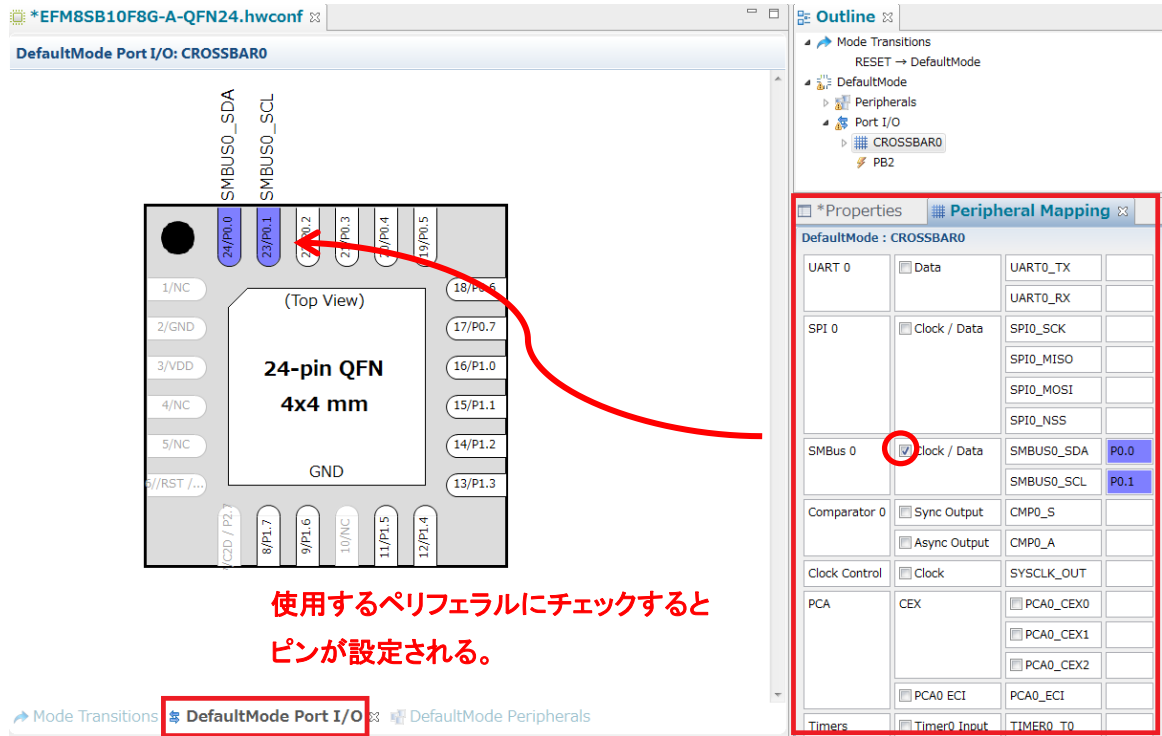
Compatible Tools の中にある Hardware Configurator を起動します。



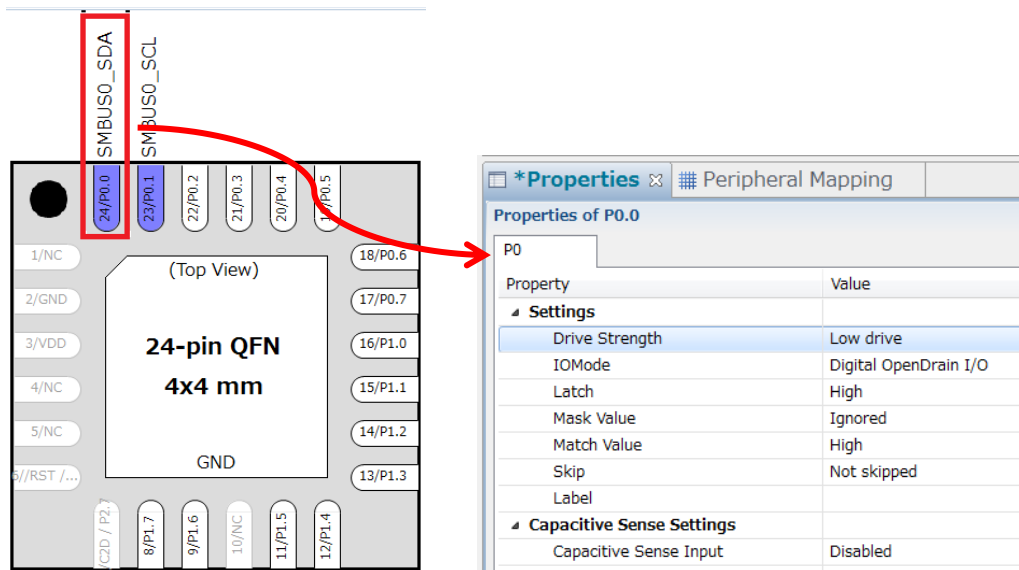
使用する Boards, Part, SDK や作業フォルダを指定してプロジェクトを作成します。プロジェクトを作成済みであれば、使用するプロジェクトを選択します。完了すると Hardware Configurator が起動します。

◆ ピン設定

ピンの設定は、DefaultMode Port I/O タブで行います。ピンを使用するペリフェラルにチェックを入れると、占有されるピンが紫色に変わります。

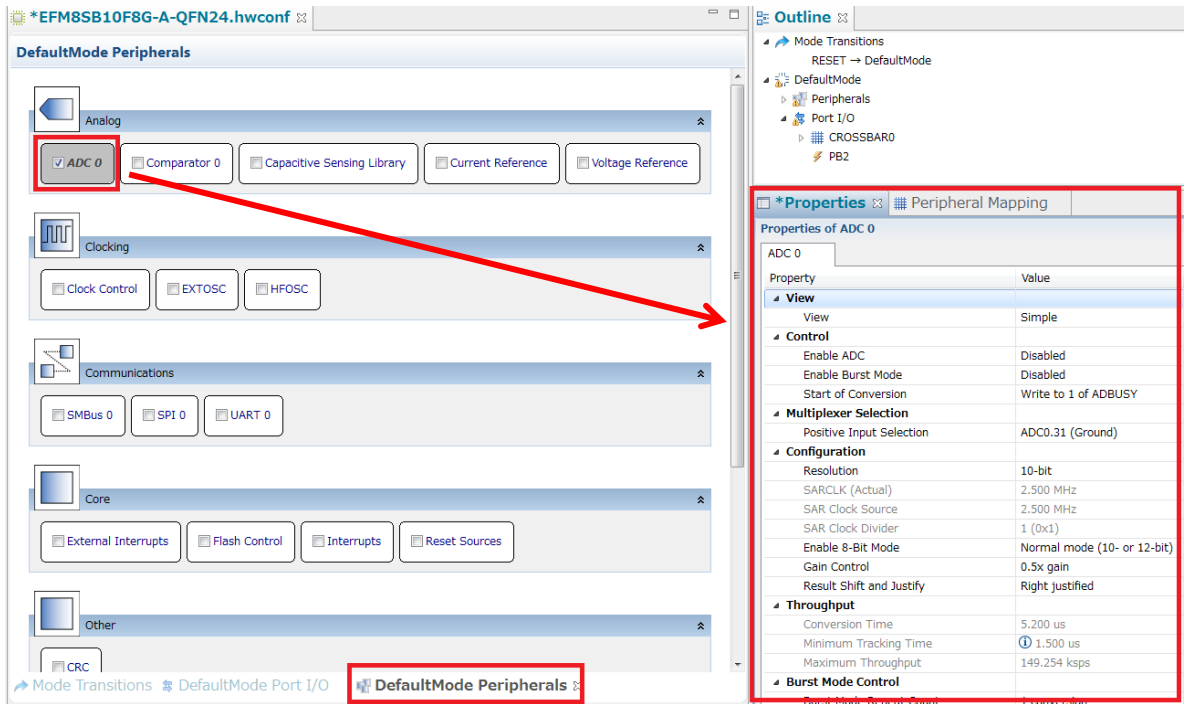


ピンの上でクリックすると、IO モード (Open Drain、Push-pull)、ドライブ・ストレンクス、Pin Skip 設定などを変更できます。



◆ ペリフェラル設定

ペリフェラルの設定は、DefaultMode Peripherals タブで行います。使用するペリフェラルを選択し、Properties ウィンドウで詳細設定を行います。



設定が完了したら、画面上で右クリックして Generate Source を実行し、ソースコードに反映させます。

6-5 ピン設定やペリフェラル設定をしてみる (Configuration Wizard 2)

Configuration Wizard 2 の使用方法を簡単にご紹介します。

Compatible Tools の中にある Configuration Wizard 2 を選択します。

[Getting Started](#)

[Documentation](#)

Compatible Tools

[Resources](#)

To view all available tools, click the 'Tools' button in the main toolbar.

Simplicity IDE
Launches the Simplicity IDE

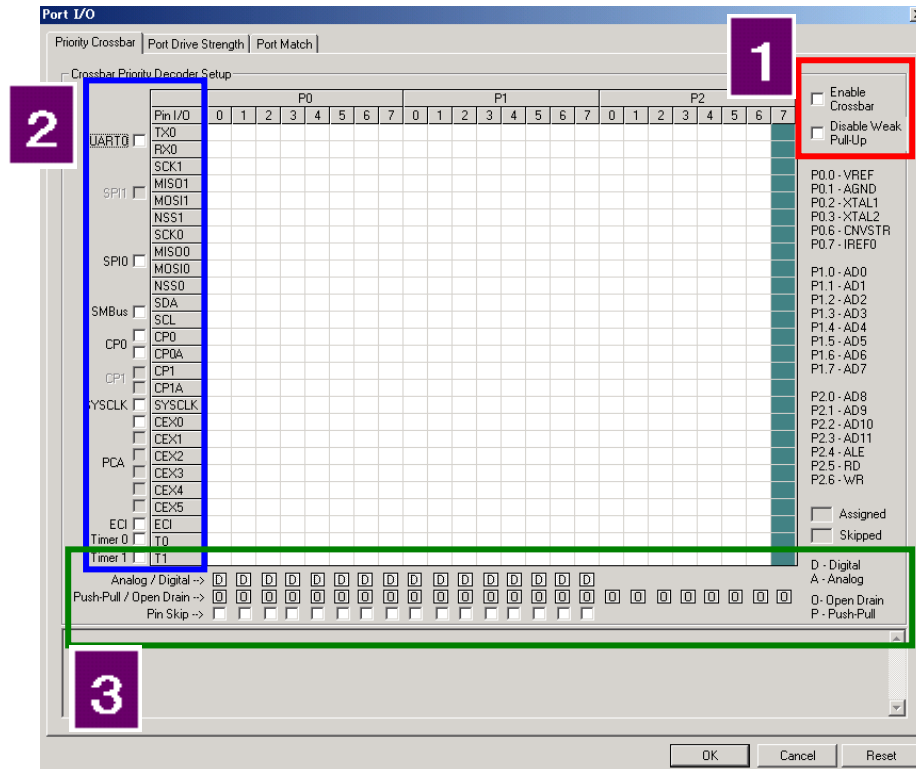
Migrate Projects
Migrate v3 projects

Configuration Wizard 2
Configuration Wizard 2 is the legacy C8051 configuration tool.

New Project ウィンドウで MCU の型番を選択すると、Configuration Wizard 2 が起動します。

◆ ピン設定

Peripherals メニューから Port I/O を選択すると、下図のようなウィンドウが表示されます。使用する MCU の機能やピン数によって、画面が異なります。



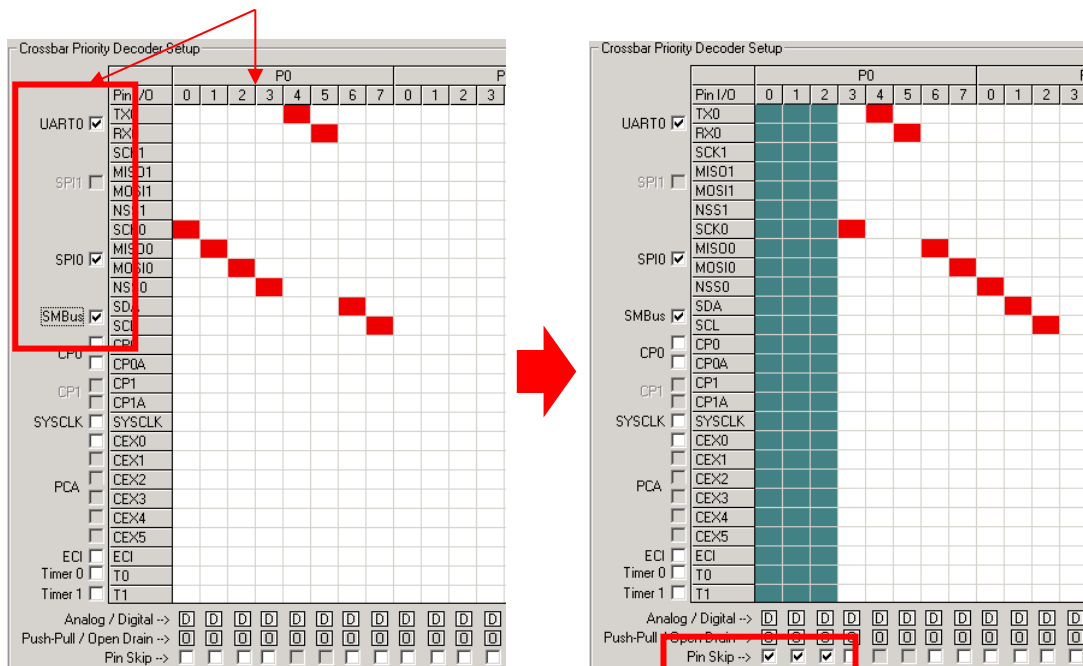
番号	概要
1	Crossbar の有効・無効 (必ず有効にしてください) 内蔵 pull-up の有効・無効
2	使用するペリフェラルの選択
3	アナログピン、デジタルピンの切り替え 出力モード(push-pull, open-drain)の切り替え ピンスキップの設定

I/O を使用するペリフェラル(UART, SPI など)を使用する場合、ポートはそれらペリフェラルに優先的に割り振られます。特定のポートを GPIO として使用したい場合には、ピンスキップの設定を行うことで優先的に確保できます。

下図(左)は、UART, SPI0, SMBus を使用する場合のポート設定です。P0.0~P0.7 が、これらペリフェラルに割り当てられています。

それに対し、下図(右)は、P0.0~P0.2 をピンスキップ設定した場合です。ペリフェラルのポート割り当てが P0.3~P1.2 へ変更されたことが判ります。

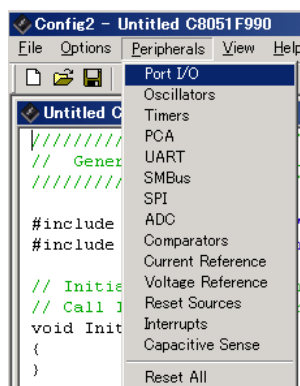
UART, SPI0, SMBus を選択。 P0.0~P0.7 が優先的に占有された。



ピンスキップ設定

◆ ペリフェラル設定

各ペリフェラルの設定は、Peripherals メニューを使用して行っていきます。一つ設定が終るたびに、C コードのヘッダーファイルが更新されます。このヘッダーファイルには、設定に準じたペリフェラル初期化ルーチンが記述されていますので、ソフトウェア設計の際にご使用頂けます。



6-6 Simplicity Studio ver.3 から ver.4 への移行

Simplicity Studio ver.3 から ver.4 へ簡単にプロジェクトを移行できるように、専用ツールが用意されています。

Compatible Tools 中にある Migrate Projects2 を選択し、ツールの指示に従って移行ください。

[Getting Started](#)

[Documentation](#)

[Compatible Tools](#)

[Resources](#)

To view all available tools, click the 'Tools' button in the main toolbar.

[Add/Remove Tools](#)



7 ソフトウェア設計

ソフトウェア設計に役立つ情報をご紹介します。

7-1 ソースコードの追い方

Simplicity IDE でソースコードを追うための方法を紹介합니다。

- ◆ 変数や関数を定義している記述を探す

```
while (1)
{
  if(i2c_rxInProgress){
    /* Receiving data */
    receiveI2CData();
  }else if (i2c_startTx){
    /* Transmitting data */
    performI2CTransfer();
    /* Transmission complete */
    i2c_startTx = false;
  }
}
```

①調べたい変数
や関数をクリック
(色がグレーに変わる)

```
while (1)
{
  if(i2c_rxInProgress){
    /* Receiving data */
    receiveI2CData();
  }else if (i2c_startTx){
    /* Transmitting data */
    performI2CTransfer();
    /* Transmission complete */
    i2c_startTx = false;
  }
}
```

```
void performI2CTransfer(void)
{
  /* Transfer structure */
  I2C_TransferSeq_TypeDef i2cTransfer;

  /* Setting pin to indicate transfer */
  GPIO_PinOutSet(gpioPortC, 0);
}
```

②右クリックして、
Open Declaration を選択
(F3 キーでも OK)

Open Declaration	F3
Open Type Hierarchy	F4

③変数や関数の定義にジャンプ

* 上記の説明では、EFM32 向けのコードを使用しています。EFM8/C8051 でも手順は同じです。

7-2 サンプルコードにペリフェラルを実装してみる（外部割込み）

EFM8/C8051 には、非常に多くのサンプルコードが用意されています。ADC のサンプルコード、UART のサンプルコード、外部割込みのサンプルコード…といった具合に 1 つのペリフェラルにスポットを当てたサンプルコードが多く、ペリフェラルの機能・動作を学ぶのに非常に役立ちます。

実際のアプリケーション設計では、複数のペリフェラルを使用することがほとんどかと思えますので、サンプルコードに別のペリフェラルを追加する手順を学ぶことは非常に有益です。

この章では、2 つのサンプルコードを migration していく手順について紹介します。

題材として、「EFM8BB3 Blinky」「EFM8BB3 External Interrupts」という 2 つのサンプルコードを使用します。どちらも EFM8BB3 STK 向けのサンプルコードです。

「EFM8BB3 Blinky」はタイマに連動して LED の色が次々と変っていくサンプルコードです。

「EFM8BB3 External Interrupts」は、ボタンを押すと外部割込みが生じ、LED の色を変えるサンプルコードです。

「EFM8BB3 Blinky」に外部割込み (External Interrupts) を実装し、LED の色が次々変わるのをスタート/ストップする機能を実装してみます。具体的には、ボタン 0 を押すと Timer2 をストップ (つまり LED の色変更も停止)、ボタン 1 を押すと Timer2 をスタート (つまり LED の色変更を再開)、という機能を実装します。

大まかな流れとしては、

- サンプルコードを理解する (7-2-1、7-2-2)
 - 「EFM8BB3 External Interrupts」のペリフェラル設定を、「EFM8BB3 Blinky」に移植する (7-2-3、7-2-4)
 - アプリを実装する (7-2-5)
- です。

7-2-1 サンプルコードを理解する (EFM8BB3_Blinky)

EFM8BB3_Blinkyは、初期化を行ったあと、while ループの中で延々とTimer 割り込みを待ち続ける、非常にシンプルなサンプルコードです。

初期化には、enter_DefaultMode_from_RESET() 関数を使っています。この関数は InitDevice.c の中で定義されており、更に各ペリフェラルの初期化関数を呼び出しています。例えば、WDT_0_enter…は Watchdog Timer の初期化関数で、TIMER16_2_enter…は Timer2 の初期化関数です。

EFM8BB3_Blinky.c

```
67 //-----
68 // Main Routine
69 //-----
70 void main (void)
71 {
72     enter_DefaultMode_from_RESET();
73
74     DISP_EN = DISP_BC_DRIVEN;           // Display not driven by EFM8
75
76     IE_EA = 1;                          // Enable global interrupts
77
78     while (1) {}                         // Spin forever
79 }
```

InitDevice.c

```
19 //-----
20 // enter_DefaultMode_from_RESET
21 //-----
22 extern void enter_DefaultMode_from_RESET(void) {
23     // [Config Calls]
24     // Save the SFRPAGE
25     uint8_t SFRPAGE_save = SFRPAGE;
26     WDT_0_enter_DefaultMode_from_RESET();
27     PORTS_1_enter_DefaultMode_from_RESET();
28     PBCFG_0_enter_DefaultMode_from_RESET();
29     TIMER16_2_enter_DefaultMode_from_RESET();
30     INTERRUPT_0_enter_DefaultMode_from_RESET();
31     // Restore the SFRPAGE
32     SFRPAGE = SFRPAGE_save;
33     // [Config Calls]
34
35 }
```

割り込み処理は Interrupts.c で行っています。Timer2 のオーバフローが発生する度に、LED の色を次々と変えています。色は全 7 種で、割り込み ⇒ case0 の色 ⇒ 割り込み ⇒ case1 の色 ⇒ 割り込み ⇒ case2 の色…といった具合に動きます。

Interrupt.c

```

24 //-----
25 // TIMER2_ISR
26 //-----
27 //
28 // TIMER2_ISR Content goes here. Remember to clear flag bits:
29 // TMR2CN::TF2H (Timer # High Byte Overflow Flag)
30 // TMR2CN::TF2L (Timer # Low Byte Overflow Flag)
31 //
32 // This routine changes the state of the LED whenever Timer2 overflows.
33 //
34 //-----
35 SI_INTERRUPT (TIMER2_ISR, TIMER2_IRQn)
36 {
37     TMR2CN0_TF2H = 0;           // Clear Timer2 interrupt flag
38
39     switch (LEDCOUNT)
40     {
41         case 0:
42             LED0 = 1;
43             LED1 = 0;
44             LED2 = 0;
45             break;
46         case 1:
47             LED0 = 0;
48             LED1 = 1;
49             LED2 = 0;
50             break;
51         case 2:
52             LED0 = 0;
53             LED1 = 0;
54             LED2 = 1;
55             break;
56         case 3:
57             LED0 = 1;
58             LED1 = 1;
59             LED2 = 0;
60             break;
61         case 4:
62             LED0 = 0;
63             LED1 = 1;
64             LED2 = 1;
65             break;
66         case 5:
67             LED0 = 1;
68             LED1 = 0;
69             LED2 = 1;
70             break;
71         case 6:
72             LED0 = 1;
73             LED1 = 1;
74             LED2 = 1;
75             break;
76     }
77
78     if (LEDCOUNT <= 6) LEDCOUNT++;
79     else LEDCOUNT = 0;
80 }

```

7-2-2 サンプルコードを理解する (EFM8BB3_ExternalInterrupts)

Blinky と同様に、初期化を行ったあと、while ループの中で延々と外部割り込みを待ち続ける、非常にシンプルなサンプルコードです。

初期化には、やはり Blinky と同様に `enter_DefaultMode_from_RESET()` 関数を使っています。

各ペリフェラルの初期化関数の中に `EXTINT_0_enter_DefaultMode_from_RESET()` という関数がありますが、これが外部割り込みの初期化関数です。`EXTINT_0_enter_DefaultMode_from_RESET()`は、`InitDevice.c` の後半で定義されています。

EFM8BB3_ExternalInterrupts.c

```
73 //-----
74 // Main Routine
75 //-----
76 void main (void)
77 {
78     enter_DefaultMode_from_RESET();
79
80     DISP_EN = DISP_BC_DRIVEN;           // EFM8 does not drive display
81
82     IE_EA = 1;
83
84     while(1);                           // Infinite while loop waiting for
85                                         // an interrupt from /INT0 or /INT1
86 }
```

InitDevice.c

```
19 //-----
20 // enter_DefaultMode_from_RESET
21 //-----
22 extern void enter_DefaultMode_from_RESET(void) {
23     // [Config Calls]
24     // Save the SFRPAGE
25     uint8_t SFRPAGE_save = SFRPAGE;
26     WDT_0_enter_DefaultMode_from_RESET();
27     PORTS_0_enter_DefaultMode_from_RESET();
28     PORTS_1_enter_DefaultMode_from_RESET();
29     PORTS_2_enter_DefaultMode_from_RESET();
30     PBCFG_0_enter_DefaultMode_from_RESET();
31     RSTSRC_0_enter_DefaultMode_from_RESET();
32     CLOCK_0_enter_DefaultMode_from_RESET();
33     TIMER01_0_enter_DefaultMode_from_RESET();
34     TIMER16_2_enter_DefaultMode_from_RESET();
35     TIMER_SETUP_0_enter_DefaultMode_from_RESET();
36     EXTINT_0_enter_DefaultMode_from_RESET();
37     INTERRUPT_0_enter_DefaultMode_from_RESET();
38     // Restore the SFRPAGE
39     SFRPAGE = SFRPAGE_save;
40     // [Config Calls]
41
42 }
```

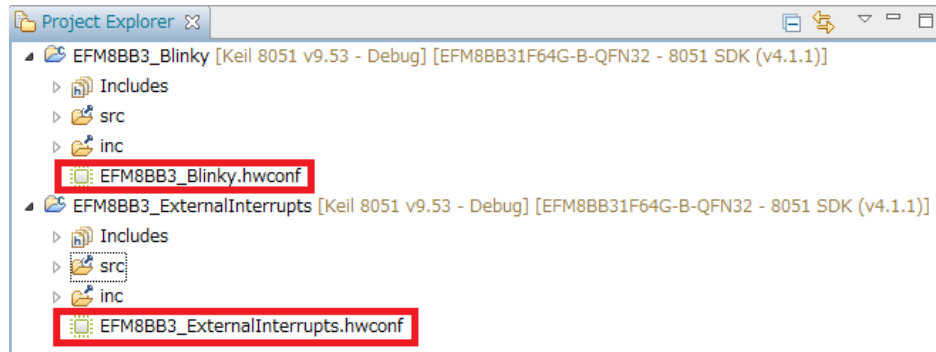
割り込み処理は Interrupts.c で行っています。INT0(外部割り込み 0)が発生したら、LED_GREEN(という名称を付けた P1_4 ピン)を反転し、INT1(外部割り込み 1)が発生したら、LED_BLUE(という名称を付けた P1_5 ピン)を反転する、という動作を行います。

Interrupt.c

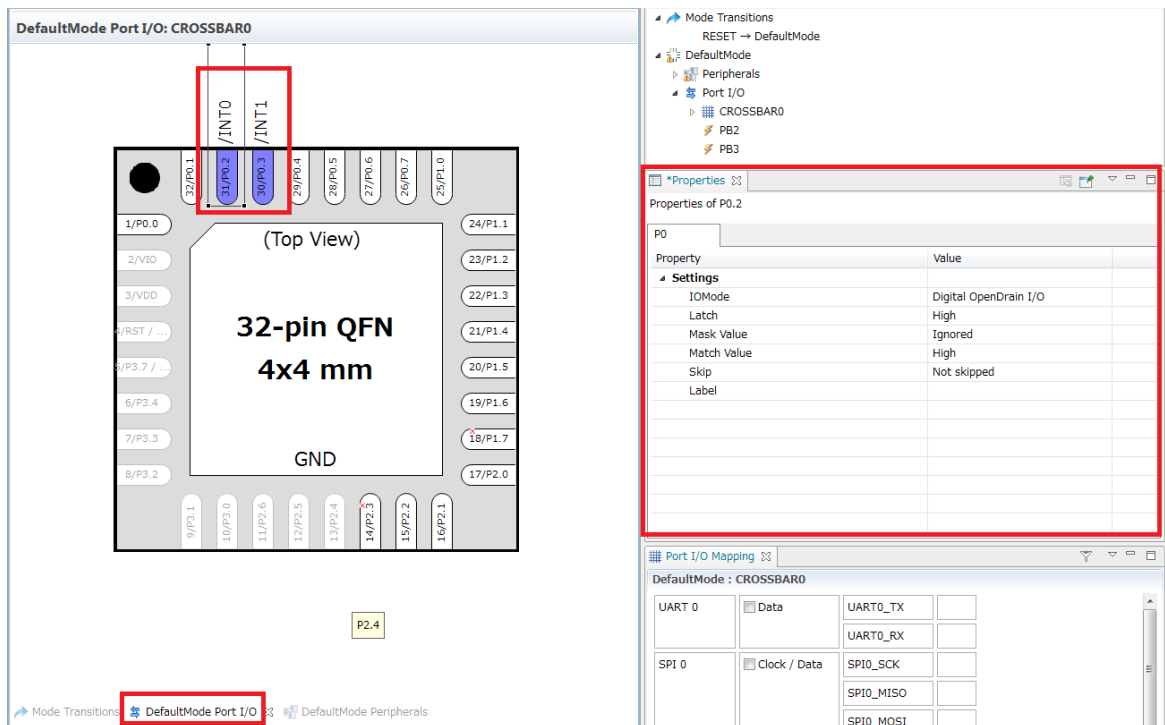
```
10 // USER INCLUDES
11 #include <SI_EFM88B3_Register_Enums.h>
12
13 //-----
14 // Pin Declarations
15 //-----
16 SI_SBIT (LED_GREEN, SFR_P1, 4);          // green LED
17 SI_SBIT (LED_BLUE, SFR_P1, 5);          // blue LED
18
19 //-----
20 // INT0_ISR
21 //-----
22 //
23 // INT0_ISR Content goes here. Remember to clear flag bits:
24 // TCON::IE0 (External Interrupt 0)
25 //
26 // Whenever a negative edge appears on P0.2, toggle LED_GREEN.
27 // The interrupt pending flag is automatically cleared by vectoring to the ISR
28 //
29 //-----
30 SI_INTERRUPT (INT0_ISR, INT0_IRQn)
31 {
32     LED_GREEN = !LED_GREEN;
33 }
34
35 //-----
36 // INT1_ISR
37 //-----
38 //
39 // INT1_ISR Content goes here. Remember to clear flag bits:
40 // TCON::IE1 (External Interrupt 1)
41 //
42 // Whenever a negative edge appears on P0.3, toggle LED_BLUE.
43 // The interrupt pending flag is automatically cleared by vectoring to the ISR
44 //
45 //-----
46 SI_INTERRUPT (INT1_ISR, INT1_IRQn)
47 {
48     LED_BLUE = !LED_BLUE;
49 }
50
```

7-2-3 ペリフェラル設定を移植する (EFM8BB3_ExternalInterrupts の設定を読み取る)

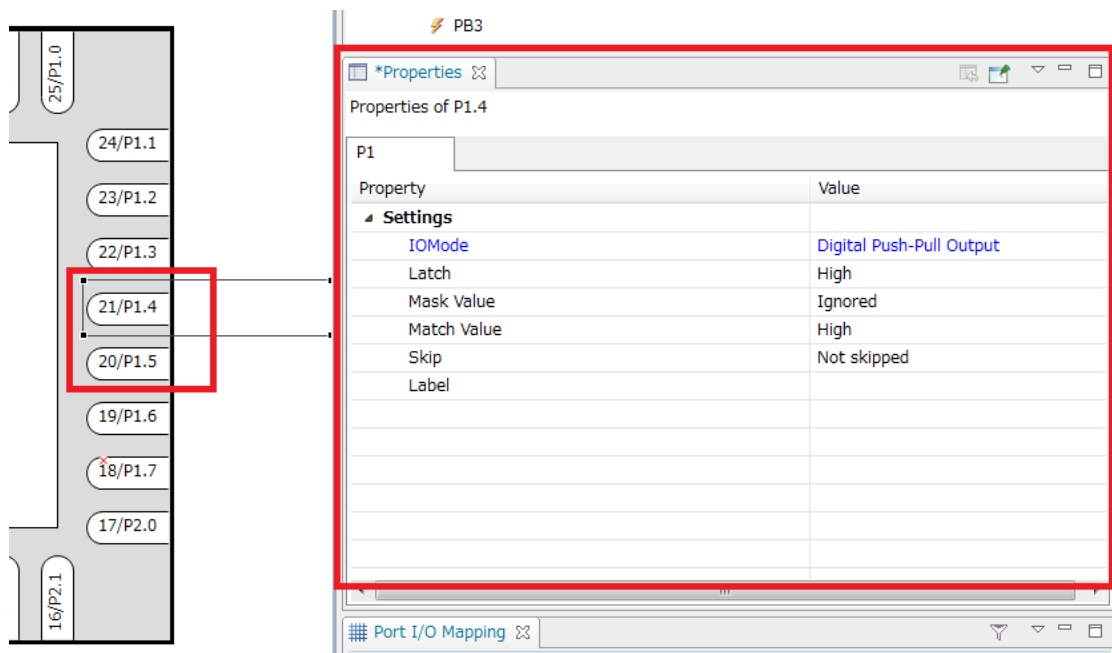
2つのサンプルコードをロードします。それぞれに.hwconf ファイルが生成されますが、これが Hardware Configurator のプロジェクトファイルです。まずは EFM8BB3_ExternalInterrupts の方から見ていきます。



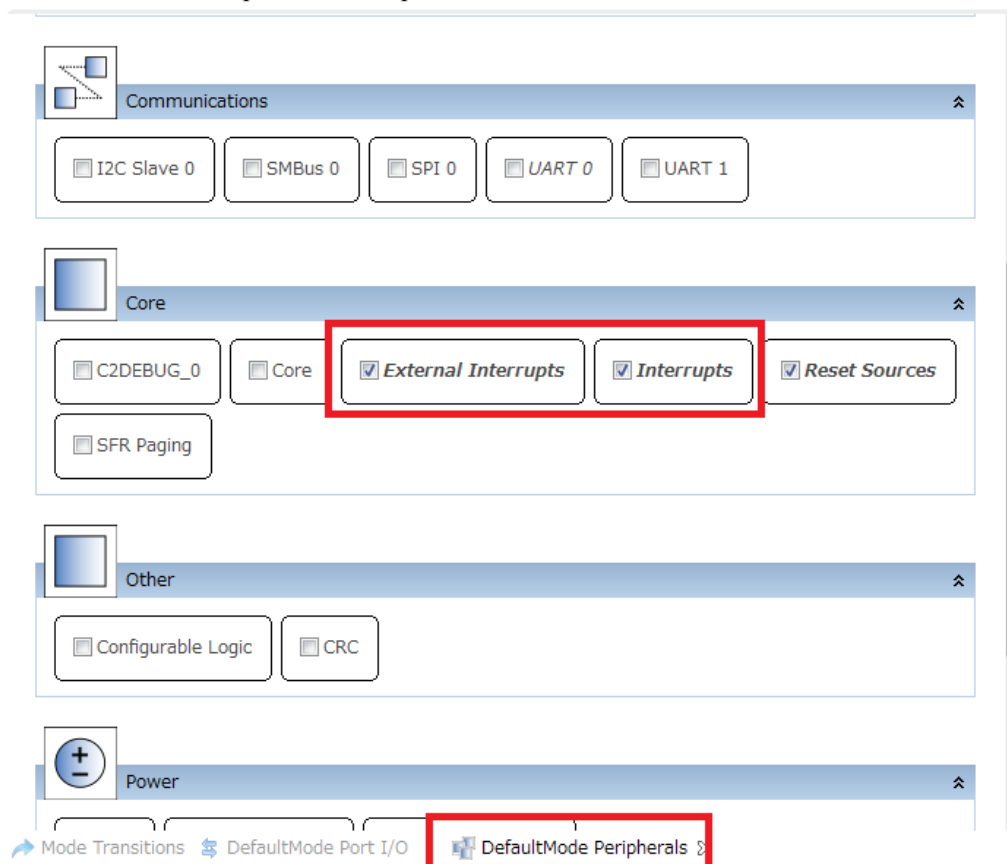
Default Mode Port I/O の設定から見ていきます。外部割込み用に、P0.2(INT0)と P0.3(INT1)の2ピンを使用しています。Properties を見てみると、特に変更点はなくデフォルトのままです。(変更点があれば色が変わっています)



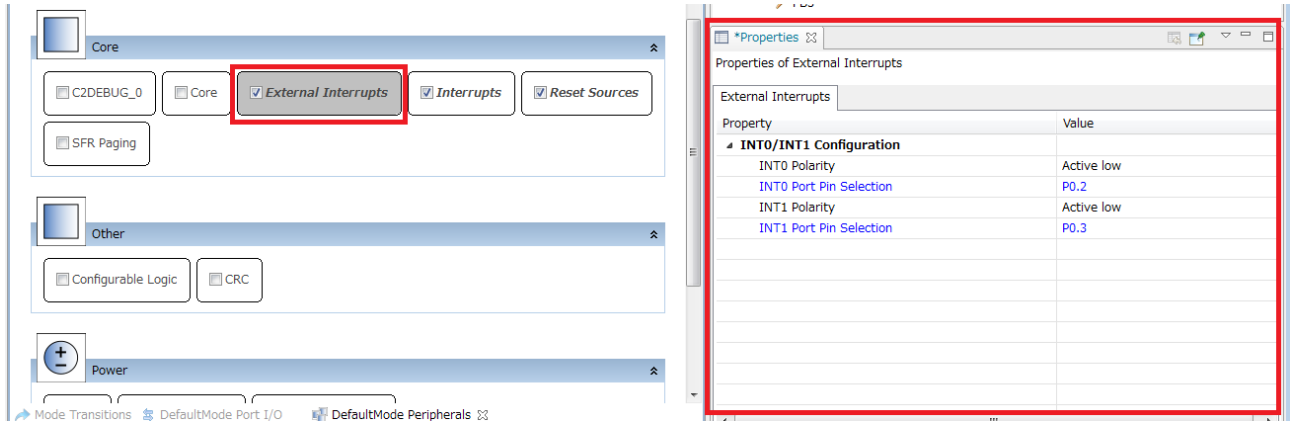
なお、LED を制御するために使っている P1.4、P1.5 は、IOMode を Digital Push-Pull Output に変更してあります。今回は EFM8BB3_Blinky の IO 設定を流用しますので、下記情報は使用しません。



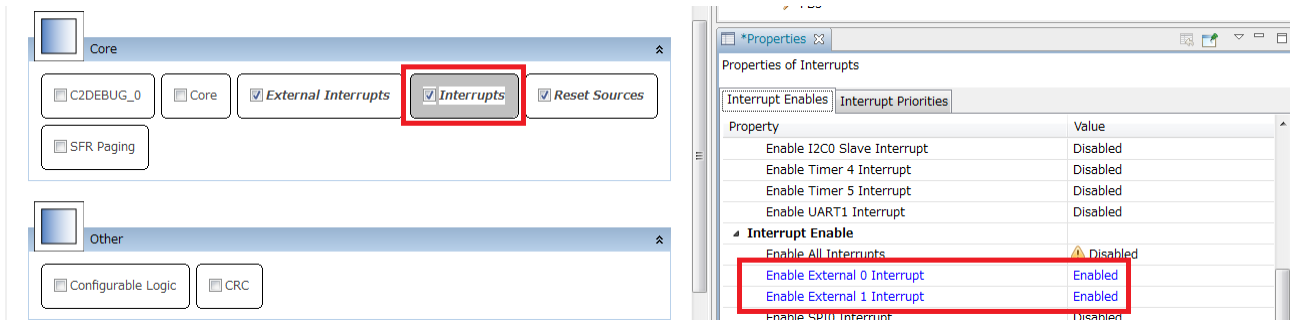
次に DefaultMode Peripherals の設定を見ます。External Interrupts と Interrupts にチェックが入っています。その他のペリフェラルにもチェックが入っていますが、External Interrupt と直接関係しないので割愛します。 External Interrupts と Interrupt の 2 つを掘り下げてみていきます。



External Interrupt では、INT0 を P0.2 に、INT1 を P0.3 に割り当てています。

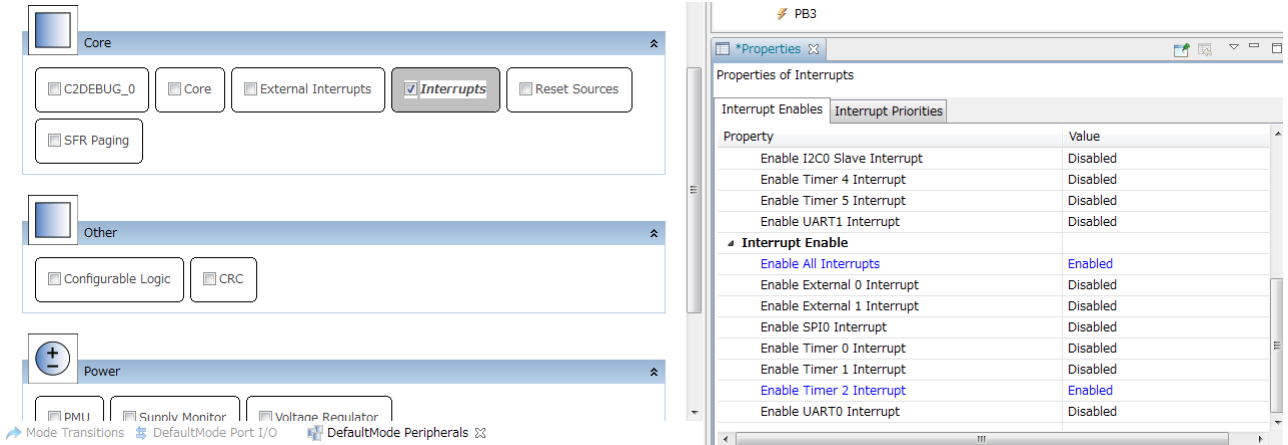


Interrupts では、Interrupt Enable(割込みの有効・無効設定)で、2つの外部割込みを有効(Enabled)に設定しています。

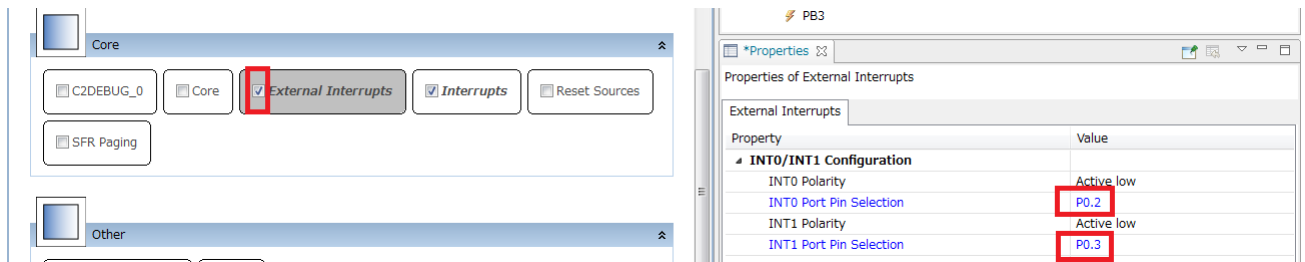


7-2-4 ペリフェラル設定を移植する (EFM8BB3_Blinky に設定を移植する)

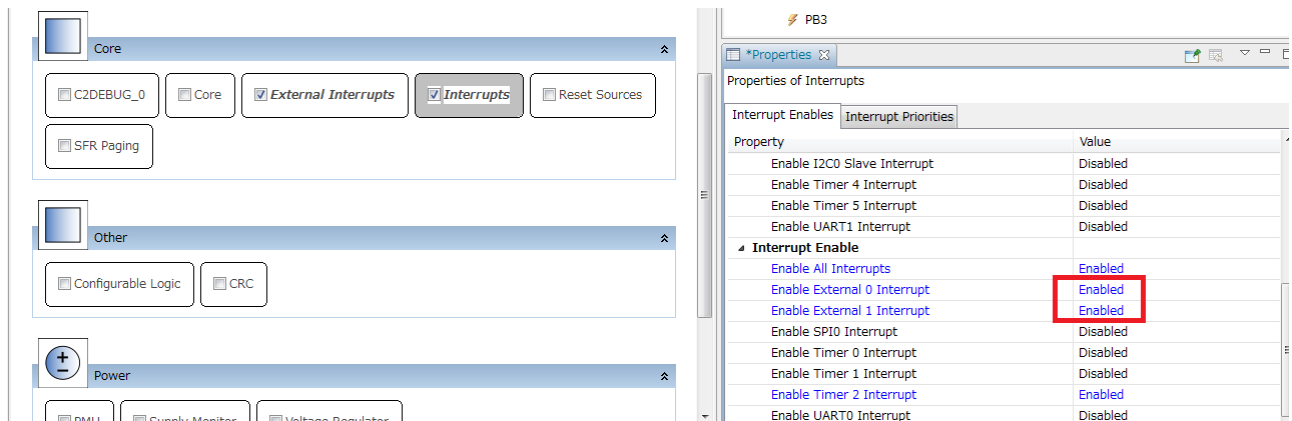
7-2-3 で読み取った設定を、今度は EFM8BB3_Blinky に反映していきます。Blinky の.hwconf ファイルを開きます。DefaultMode Peripherals の設定を見ると、External Interrupts は未チェックです。Interrupts にはチェックが入っていますが、外部割込みは無効(Disabled)のままです。



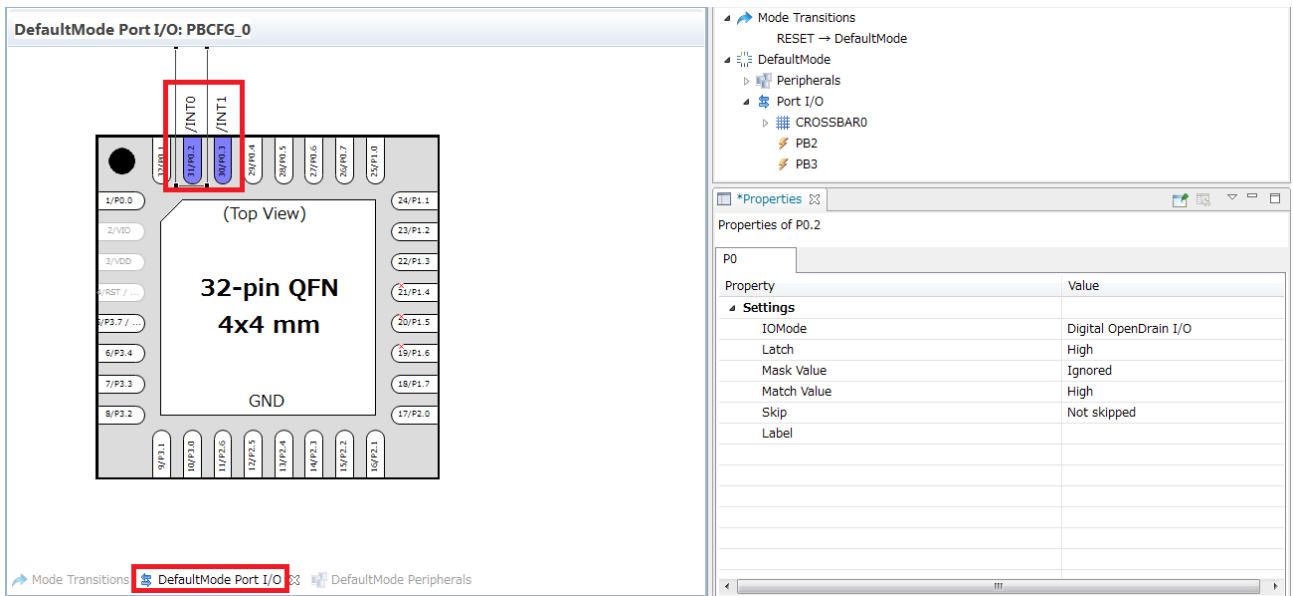
External Interrupts にチェックを入れ、INT0 Port Pin Selection を P0.2 に、INT0 Port Pin Selection を P0.3 に変更します。



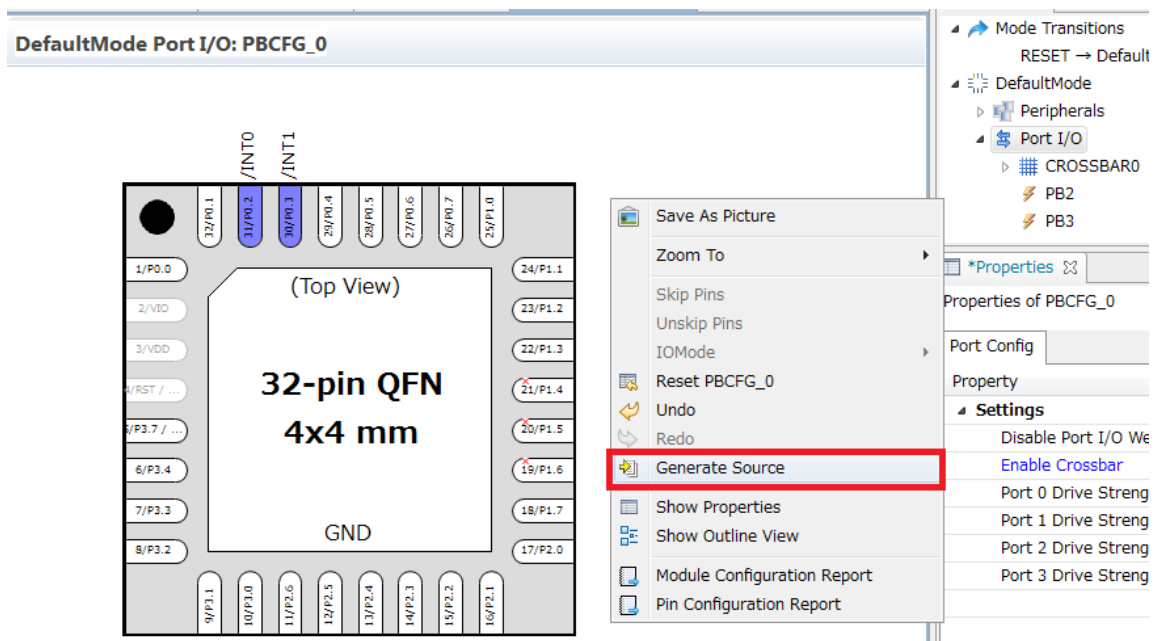
次に Interrupts で、Enable External 0 Interrupt と Enable External 1 Interrupt を Enabled に変更します。



次に Default Mode Port I/O です。今回は特に変更箇所はありませんが、INT0 と INT1 用にピンがリザーブされたことが確認できます。



設定が完了したら、行った設定をソースコードに反映させます。Hardware Configurator 上で適当な場所で右クリックし、Generate Source を選択します。



この Generate Source で、InitDevice.c と Interrupts.c が更新されています。その更新内容を見ていきましょう。

InitDevice.c の初期化関数 (enter_DefaultMode_from_RESET) を、変更前 (7-2-1) のものと比較すると、EXTINT_0_enter_DefaultMode_from_RESET() が追加されていることが確認できます。

InitDevice.c

```
19 //-----
20 // enter_DefaultMode_from_RESET
21 //-----
22 extern void enter_DefaultMode_from_RESET(void) {
23     // $[Config Calls]
24     // Save the SFRPAGE
25     uint8_t SFRPAGE_save = SFRPAGE;
26     WDT_0_enter_DefaultMode_from_RESET();
27     PORTS_1_enter_DefaultMode_from_RESET();
28     PBCFG_0_enter_DefaultMode_from_RESET();
29     TIMER16_2_enter_DefaultMode_from_RESET();
30     EXTINT_0_enter_DefaultMode_from_RESET();
31     INTERRUPT_0_enter_DefaultMode_from_RESET();
32     // Restore the SFRPAGE
33     SFRPAGE = SFRPAGE_save;
34     // [Config Calls]$
35
36 }
```

また、Interrupt.cでは、外部割込み(INT0、INT1)の割込みハンドラが追加されています。処理内容が何もありませんので、ここにユーザコードを追記していきます。

Interrupts.c

```
81 //-----
82 // INT0_ISR
83 //-----
84 //
85 // INT0_ISR Content goes here. Remember to clear flag bits:
86 // TCON::IE0 (External Interrupt 0)
87 //
88 //-----
89 SI_INTERRUPT (INT0_ISR, INT0_IRQn)
90 {
91
92 }
93
94 //-----
95 // INT1_ISR
96 //-----
97 //
98 // INT1_ISR Content goes here. Remember to clear flag bits:
99 // TCON::IE1 (External Interrupt 1)
100 //
101 //-----
102 SI_INTERRUPT (INT1_ISR, INT1_IRQn)
103 {
104
105 }
```

7-2-5 アプリを実装する

新たに追加された割込みハンドラ(7-2-4 参照)の中では、タイマ2のスタート/ストップを制御したいです。タイマ2の仕様はリファレンス・マニュアルを参照します。

<https://www.silabs.com/documents/public/reference-manuals/efm8bb3-rm.pdf>

21.4.13 TMR2CN0: Timer 2 Control 0

Bit	7	6	5	4	3	2	1	0
Name	TF2H	TF2L	TF2LEN	TF2CEN	T2SPLIT	TR2	T2XCLK	
Access	RW	RW	RW	RW	RW	RW	RW	
Reset	0	0	0	0	0	0	0x0	
SFR Page = 0x0, 0x10; SFR Address: 0xC8 (bit-addressable)								

Bit	Name	Reset	Access	Description
7	TF2H	0	RW	Timer 2 High Byte Overflow Flag. Set by hardware when the Timer 2 high byte overflows from 0xFF to 0x00. In 16-bit mode, this will occur when Timer 2 overflows from 0xFFFF to 0x0000. When the Timer 2 interrupt is enabled, setting this bit causes the CPU to vector to the <code>USER_INTERRUPT</code> . Timer 2 operates as the CPU starts releasing timers.
2	TR2	0	RW	Timer 2 Run Control. Timer 2 is enabled by setting this bit to 1. In 8-bit mode, this bit enables/disables TMR2H only; TMR2L is always enabled in split mode.

Timer2の制御レジスタ(TMR2CN0)の説明を読むと、bit2にTR2というビットがあり、これを使ってスタート/ストップの制御が行えます。TR2を1に設定するとタイマ2が有効(enabled)になります。つまり、0に設定するとタイマ2が無効になります。

ここで注目したいのは、”bit-addressable”の文字です。バイト単位ではなく、ビット単位でアクセスができますので、わざわざリードモディファイライトする必要がありません。

EFM8/C8051が持つレジスタは、全て定義ファイル内で宣言されています。EFM8BB3の場合は、SI_EFM8BB3_Register_Enums.h および SI_EFM8BB3_Defs.h が定義ファイルになります。

Interrupt.c (EFM8BB3_Blinky.c、InitDevice.cでも使われています)

```

9 // USER INCLUDES
10 #include <SI_EFM8BB3_Register_Enums.h>

```

SI_EFM8BB3_Register_Enums.h

```

49
50 //Standard device includes
51 #include "SI_EFM8BB3_Defs.h"

```

使用したい TR2 ビットですが、SI_EFM8BB3_Defs.h の中で定義されており、TMR2CN0_TR2 という名称になっています。

```
---
518 // TMR2CN0 (Timer 2 Control 0)
519 #define SFR_TMR2CN0 0xC8
520 SI_SBIT (TMR2CN0_T2XCLK0, SFR_TMR2CN0, 0); ///< Timer 2 External Clock Select Bit 0
521 SI_SBIT (TMR2CN0_T2XCLK1, SFR_TMR2CN0, 1); ///< Timer 2 External Clock Select Bit 1
522 SI_SBIT (TMR2CN0_TR2, SFR_TMR2CN0, 2); ///< Timer 2 Run Control
523 SI_SBIT (TMR2CN0_T2SPLIT1, SFR_TMR2CN0, 3); ///< Timer 2 Split Mode Enable
524 SI_SBIT (TMR2CN0_TF2CEN, SFR_TMR2CN0, 4); ///< Timer 2 Capture Enable
525 SI_SBIT (TMR2CN0_TF2LEN, SFR_TMR2CN0, 5); ///< Timer 2 Low Byte Interrupt Enable
526 SI_SBIT (TMR2CN0_TF2L, SFR_TMR2CN0, 6); ///< Timer 2 Low Byte Overflow Flag
527 SI_SBIT (TMR2CN0_TF2H, SFR_TMR2CN0, 7); ///< Timer 2 High Byte Overflow Flag
```

Interrupt.c の外部割込みの割込みハンドラに、TMR2CN0_TR2 の制御を盛り込みます。値を 0 にするとストップ、1 にするとスタートです。

Interrupt.c

```
81 //-----
82 // INT0_ISR
83 //-----
84 //
85 // INT0_ISR Content goes here. Remember to clear flag bits:
86 // TCON::IE0 (External Interrupt 0)
87 //
88 //-----
89 SI_INTERRUPT (INT0_ISR, INT0_IRQn)
90 {
91     TMR2CN0_TR2 = 0; //ボタン0を押すとストップ
92 }
93
94 //-----
95 // INT1_ISR
96 //-----
97 //
98 // INT1_ISR Content goes here. Remember to clear flag bits:
99 // TCON::IE1 (External Interrupt 1)
100 //
101 //-----
102 SI_INTERRUPT (INT1_ISR, INT1_IRQn)
103 {
104     TMR2CN0_TR2 = 1; //ボタン1を押すとスタート
105 }
```

その後、ビルドして、ダウンロードすると、LED の点灯スタート/ストップをボタンで制御できるようになったことが確認できます。

改版履歴

Version	改定日	改定内容
1.0	2011年07月	・新規作成
1.3	2015年06月	・EFM8に対応。マクニカオンラインで公開
2.0	2016年12月	・Simplicity Studio ver.4に対応
2.1	2017年03月	最新のSimplicity Studioに合わせて説明を一部変更
2.2	2018年03月	最新のSimplicity Studioに合わせて説明を一部変更。7章(ソフトウェア設計)追加

参考文献

- Silicon Labs 社 各種ドキュメント
- Silicon Labs 社 ナレッジベース、コミュニティフォーラム

免責、及び、ご利用上の注意

弊社より資料を入手されましたお客様におかれましては、下記の使用上の注意を一読いただいた上でご使用ください。

1. 本資料は非売品です。許可無く転売することや無断複製することを禁じます。
2. 本資料は予告なく変更することがあります。
3. 本資料の作成には万全を期していますが、万一ご不審な点や誤り、記載漏れなどお気づきの点がありましたら、弊社までご一報いただければ幸いです。
4. 本資料で取り扱っている回路、技術、プログラムに関して運用した結果の影響については、責任を負いかねますのであらかじめご了承ください。
5. 本資料は製品を利用する際の補助的なものとしてかかれたものです。製品をご使用になる場合は、メーカーリリースの資料もあわせてご利用ください。

本社

〒222-8561 横浜市港北区新横浜 1-6-3 TEL 045-470-9841 FAX 045-470-9844