

PM-EUB02

ファーストステップガイド

1 PM-EUB02 の仕様概略

PM-EUB02 は、シリコンラボ社の USB コントローラ内蔵の 8 ビットマイコン、EFM8UB31F40G-A-QSOP24 を使用したマイコンボードです。主な仕様を表 1 に示します。

EFM8UB3 シリーズはこのクラスの 8 ビットマイコンでは最大 48MHz で動作する 8 ビット CPU です。内部動作はパイプライン化されていて大部分（109 個中 79 個）の命令を 1 ～2 クロックサイクルで実行します。

また、EFM8UB3 は USB ブートロー

ダを内蔵しているため、PC の USB ポートと直結してプログラムを書き込むことができます。USB ブートローダは通常の使用では消去されませんので、何度でも USB 経由で書き換えができます。

基板サイズは 300mil 幅の 28 ピンの DIP パッケージとほぼ同じ大きさです。

I/O ピンは 16 ピン（8 ピン×2 列）あり、ここに 12 本の I/O ピンと、リセット、電源（+5V、+3.3V 出力、GND）を引き出しています。ピンの間隔は 300mil ですので、通常の 16 ピン IC ソケットも利用できます。

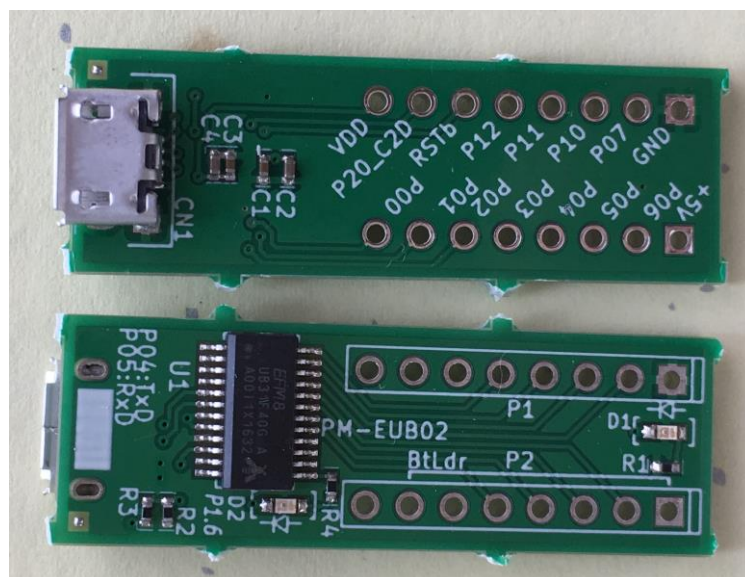


図 1 : PM-EUB02 基板外観

分類	項目	仕様	備考
CPU	CPU	EFM8UB31F40G-A-QSOP24	Silicon Labs.製
	コア	8051互換	最高48MHz動作
	ROM	40KByte	
	RAM	3328Byte	
	入力電源	+5V	USBのVBUS、または外部供給
	出力電源	+3.3V	EFM8UB3内蔵レギュレータ出力
	ブートローダ	USBブートローダ	P2.0をGNDと接続して起動
内蔵I/O	USB	USBスレーブ	フルスピード／ロウスピード
	UART	4バイトFIFO付き	CTS/RTSあり
	SPI	最大12Mbps	
	SMBus	100kbps/400kbps/1Mbps可	7ビットアドレスモードをH/Wサポート
	CLU	4ユニット	3to1のLUT+DFF
	タイマ・PWM	タイマ0～5、PCA	16ビットタイマ
	ADC	12bit	
	コンパレータ	2ユニット	
	CRC	16bitCRC	入力データ: 8bit単位
GPIO	ドライブ能力	2段階から選択	
	割り込み	直接(INT0/1)	割り込み極性選択可
		ポートマッチ	PnMaskした値がPnMATCHと一致
	ピン割付	クロスバーによる優先度順	
	出力電圧	+3.3V	CPU動作電圧と一致
外部I/O	入力電圧	VIO+3.3V	+5V入力可
	LED	P1.6に接続	H'で点灯
	USB	マイクロUSBコネクタ	電源供給兼用
	I/Oピン	16ピンDIP(300mil幅)	電源x3、リセット、GPIO×12

表 1 : PM-EUB02 の主な仕様

2 基板レイアウト

PM-EUB02 の基板レイアウトは図 2 のようになっています。図の上部の LED (D1) は電源 LED で電源供給中であることを示します。下部右側の LED(D2)は P1.6 端子に接続されていて、'H'レベル出力 ('1'出力) で点灯します。

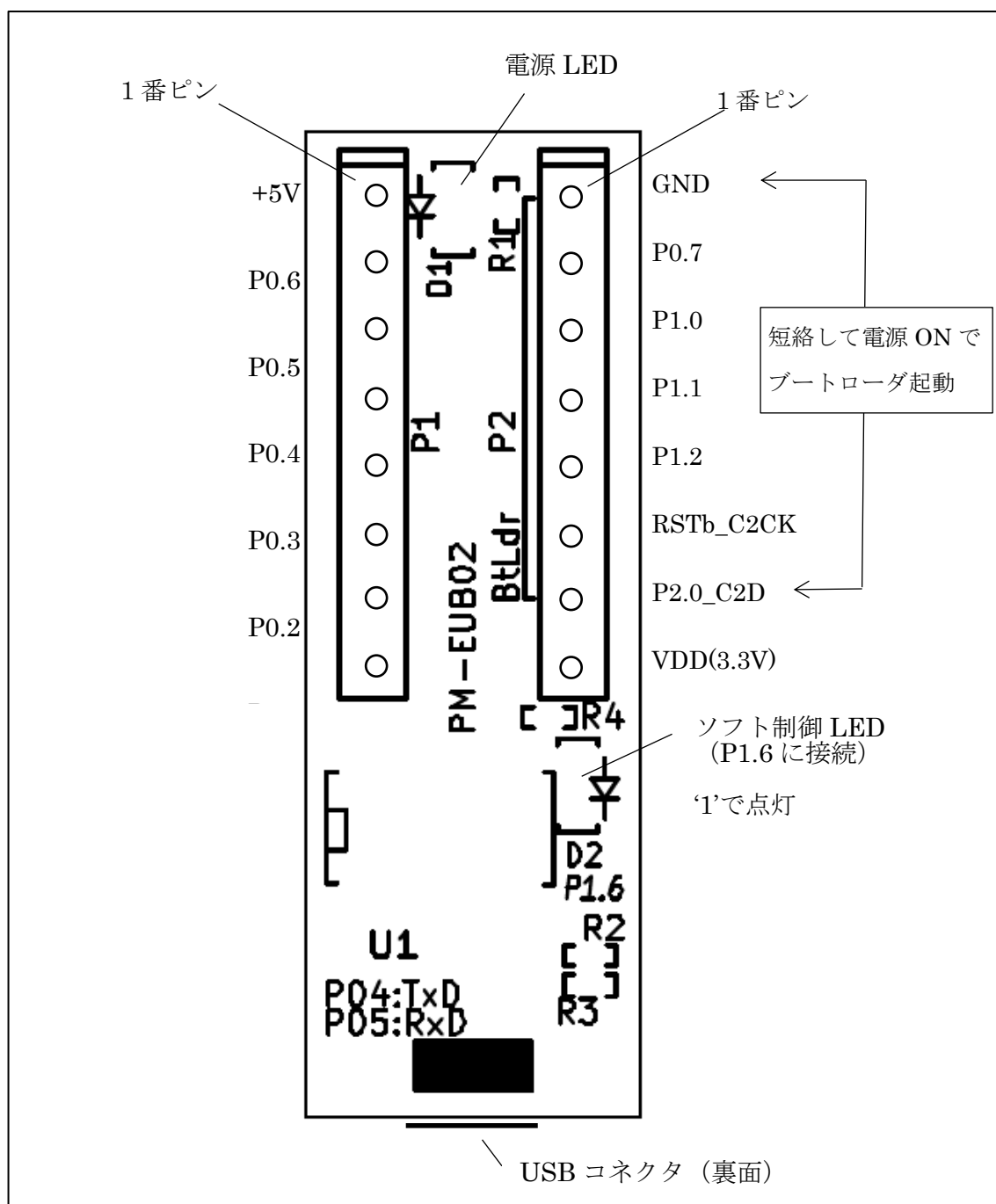


図 2 : PM-EUB02 基板レイアウト

3 チュートリアル

ここでは、Timer2 からの割り込み（10Hz にします）を使って、PM-EUB02 の LED（P1.6 に接続）を点滅させるサンプルを作成してみます。作成手順の大まかな流れは、次のようになります。

- 1)開発ツールのインストール
- 2)プロジェクトの作成
- 3)GPIO ピンや内部 I/O のコンフィギュレーション
- 4)ソースコード生成
- 5)プログラム記述
- 6)ビルド
- 7)書き込み&動作確認

3.1 開発ツールのダウンロード

開発ツールの Simplicity Studio は SiliconLabs 社のサイトからダウンロードできます。

<https://jp.silabs.com/products/development-tools/software/simplicity-studio>

適宜ダウンロードしてインストールしてください。なお、本書では Windows 版を使用しています。

このほか、ブートローダを使った書き込みのためにファイル変換ツール（hex2boot.exe）とダウンロードプログラム（efm8load.exe）が必要です。

<https://www.silabs.com/documents/public/example-code/AN945SW.zip>

にありますので、ダウンロードして適当なディレクトリに展開しておいてください。後で作成したプロジェクトの中にコピーします。ツールのマニュアル（ユーザズガイド）は

<https://www.silabs.com/documents/public/application-notes/an945-efm8-factory-bootloader-user-guide.pdf>

にあります。

※リンク先の URL はいずれも 2018 年 9 月時点です。

3.2 Simplicity Studio の起動

画面上の **Simplicity Studio** アイコンをダブルクリックして起動します。図のようなランチャー画面になっていない場合は、“Launcher”タブをクリックしてください。

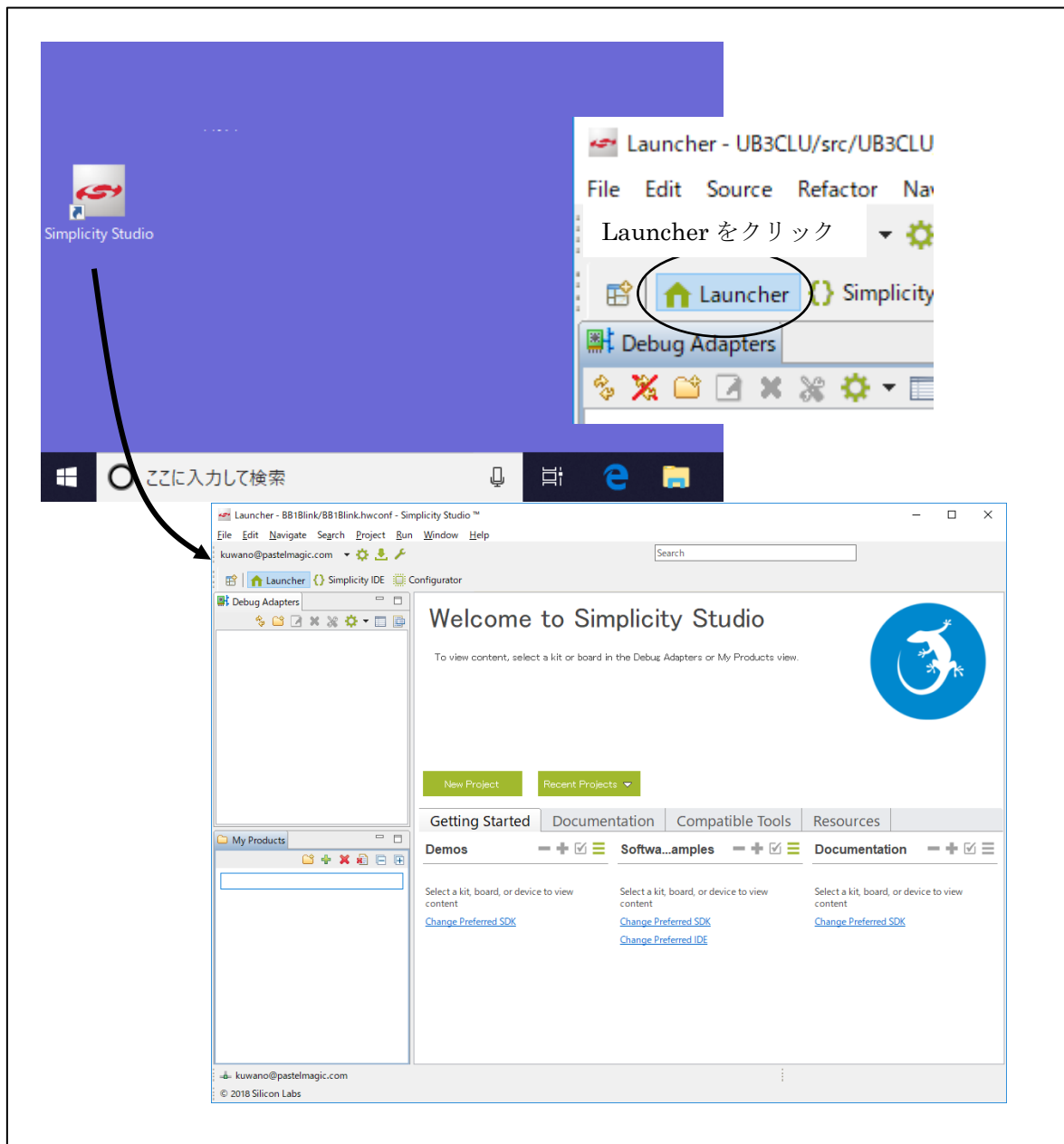


図 3 : Simplicity Studio の起動

3.3 MyProject へのデバイスの登録

自分で使用するデバイスを登録しておきます図 4 のように画面左下の「MyProject」の部分で、EFM8UB31F と入力していくと、候補が絞られていきますので、PM-EUB02 で使用している、EFM8UB31F40G-A-QSOP24 を選択します。

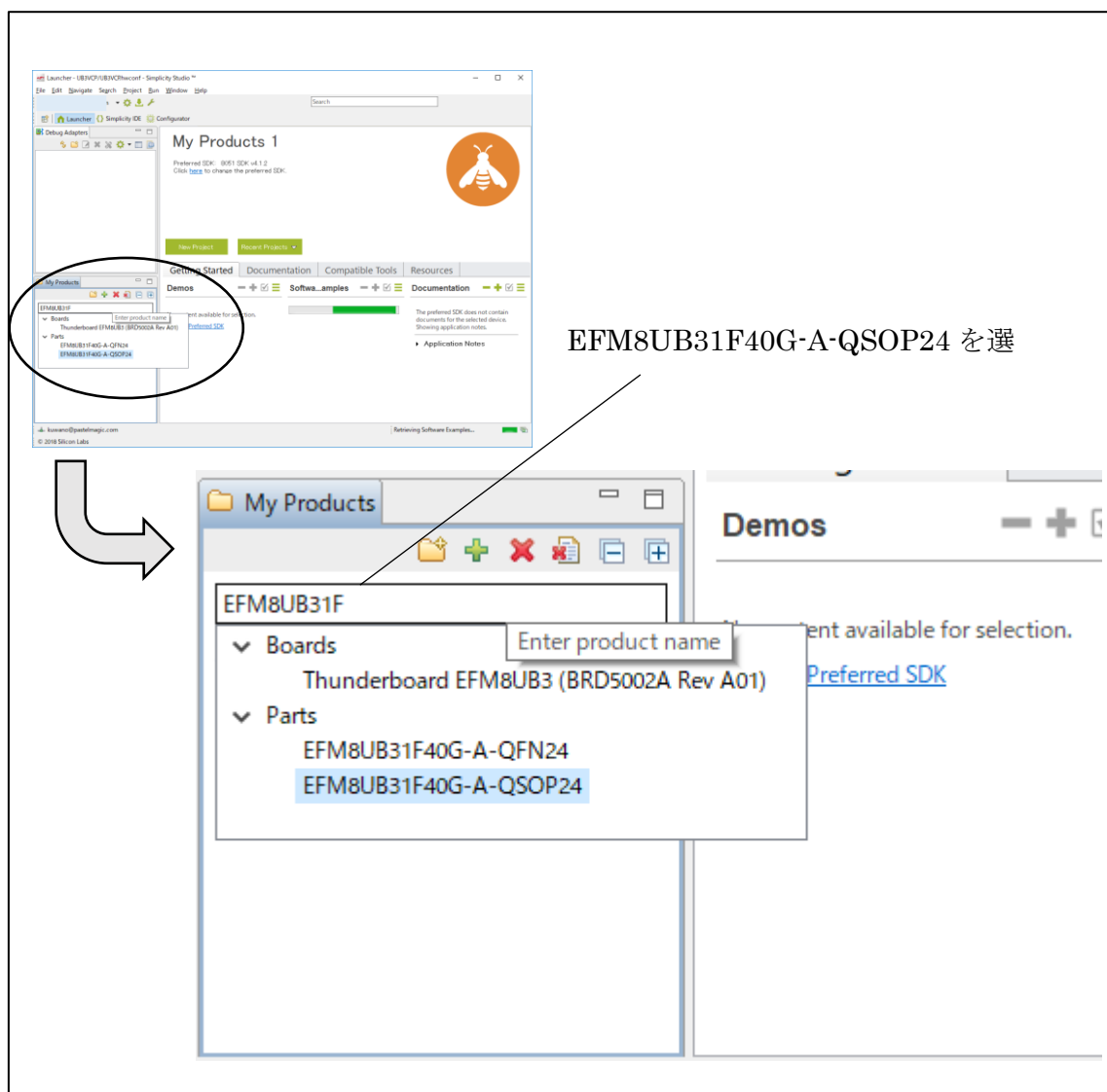


図 4 : MyProjects へのデバイス登録

3.4 新規プロジェクトの作成

SimplicityStudio のツールバーから File→New→Project を選択して新規プロジェクトを作成します。（図 5 参照）

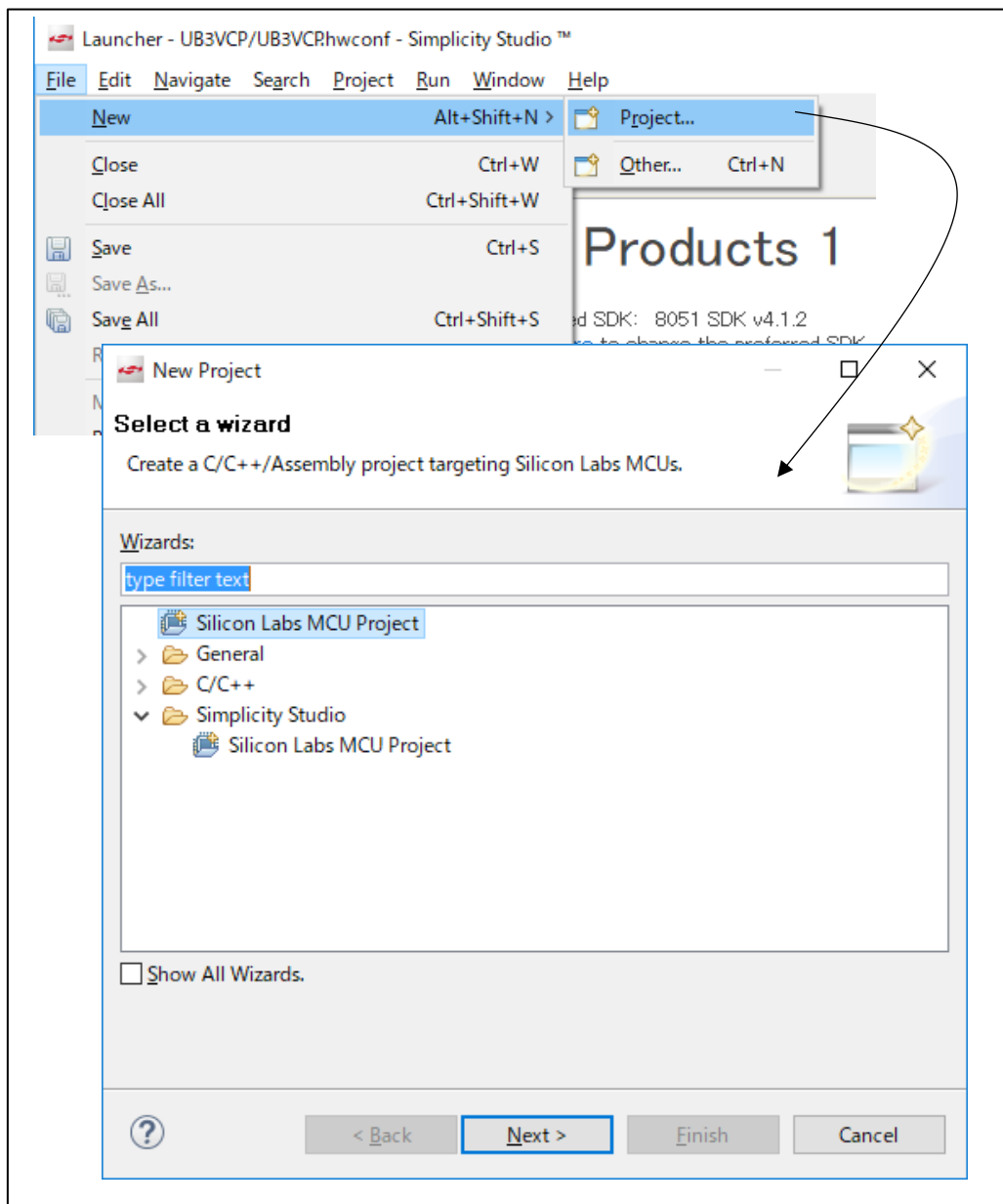


図 5：プロジェクトの作成

※：デフォルトのワークスペースの作成先は

C:\ユーザー¥(ユーザー名)¥SimplicityStudio¥v4_workspace

です。（変更することもできます）

3.5 デバイス選択

図 6 のように、プロジェクトで使用するデバイスの選択画面になります。
EFM8UB3140G-A-QSOP24 を選択します。先ほどはあくまでも自分でよく使うデバイスや開発キットの登録でしたが、ここでは、実際のプロジェクトで使用するデバイスを設定します。

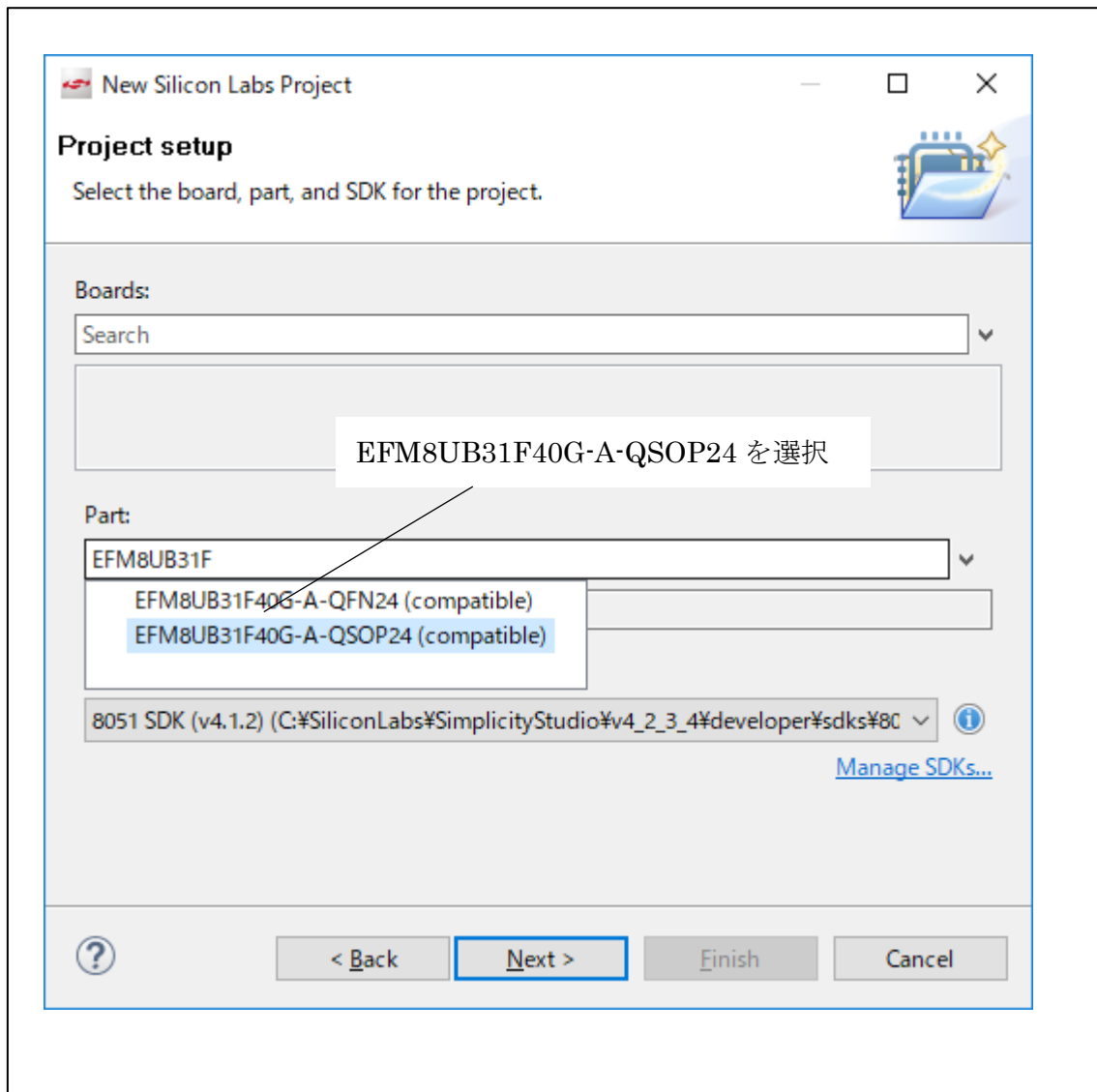


図 6 : デバイス選択

3.6 プロジェクト名、ワークスペースの指定

プロジェクト名を指定します。ここでは図 7 のように UB3BLINK としました。ワークスペースの作成先を変更したいときは、“Use default location” のチェックを外します。

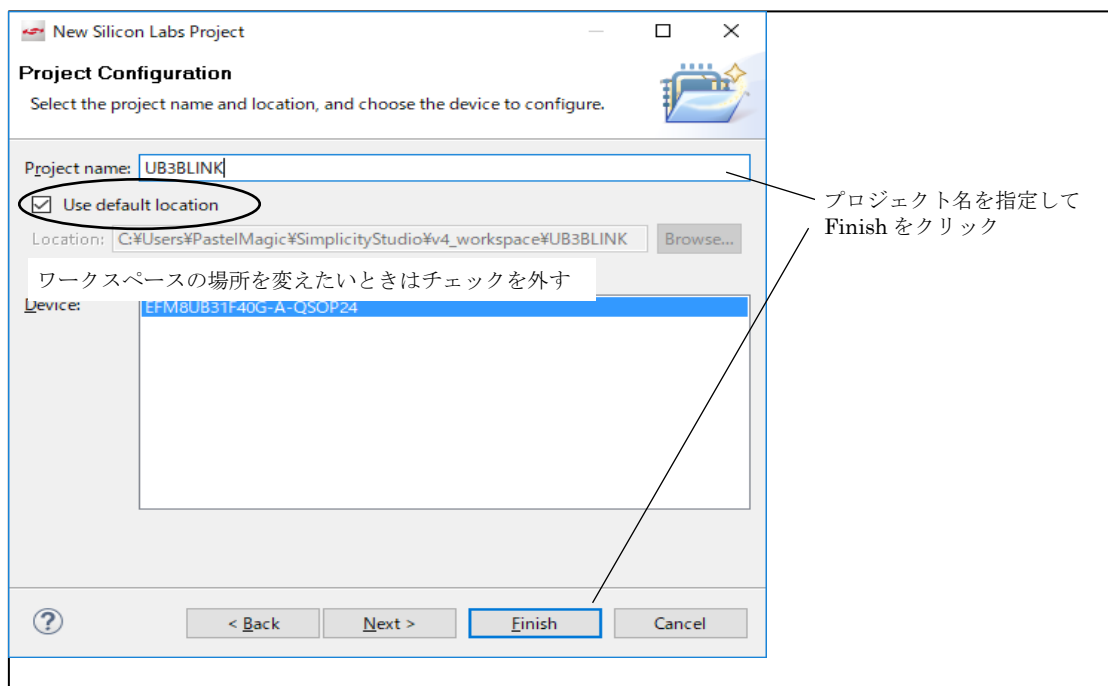


図 7: プロジェクト名とワークスペースの指定

3.7 デバイスのコンフィギュレーション

3.7.1 GPIO ピンのモード設定

PM-EUB02 では GPIO ピンの P1.6 に LED が繋がっています。'H'レベルで点灯しますので、ピンのモードを Digital Push-Pull Output に変更します。画面がピン配置図になっていない場合には.hwconf ファイルをダブルクリックして、DefaultMode Port I/O を選択します。

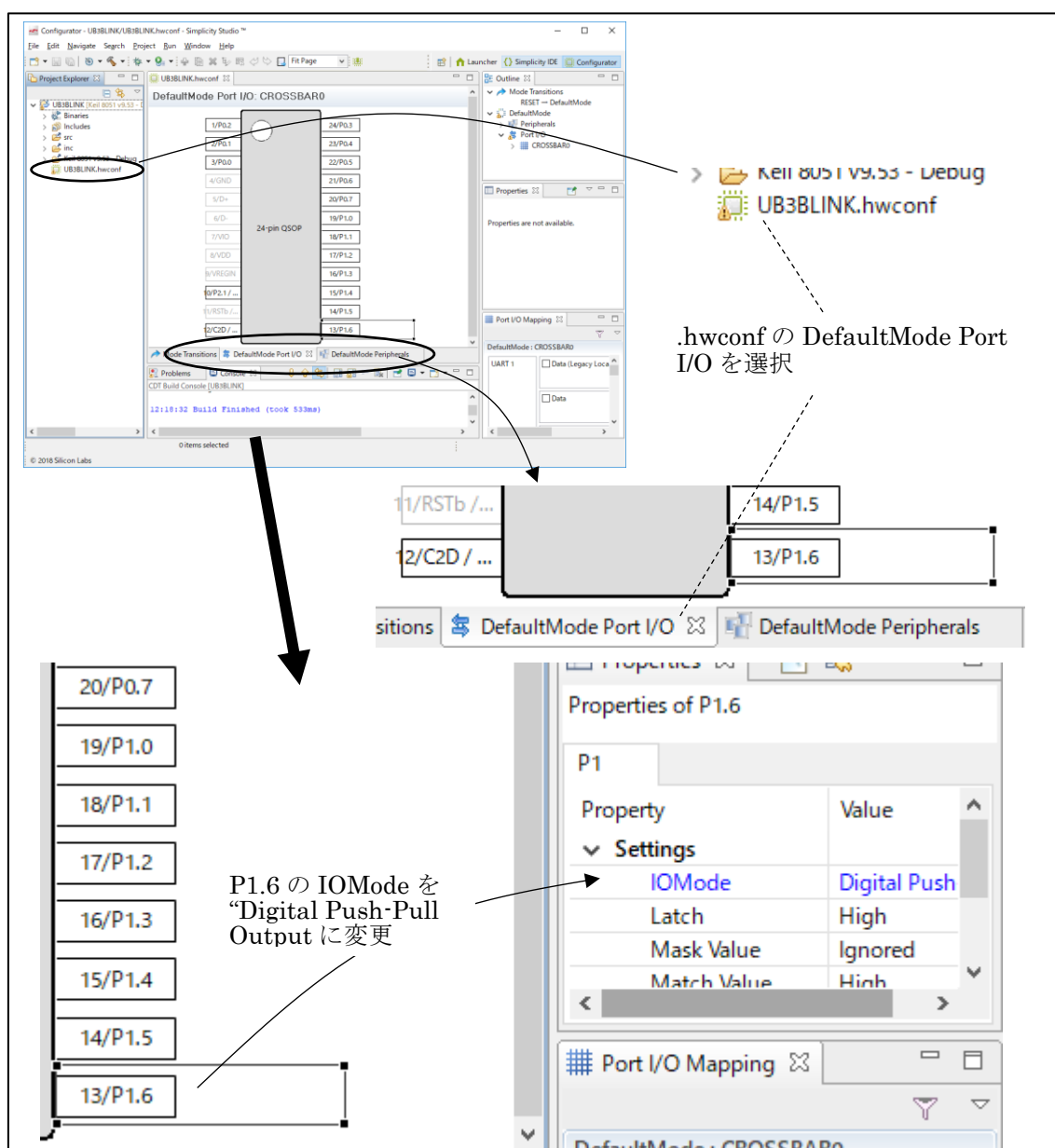
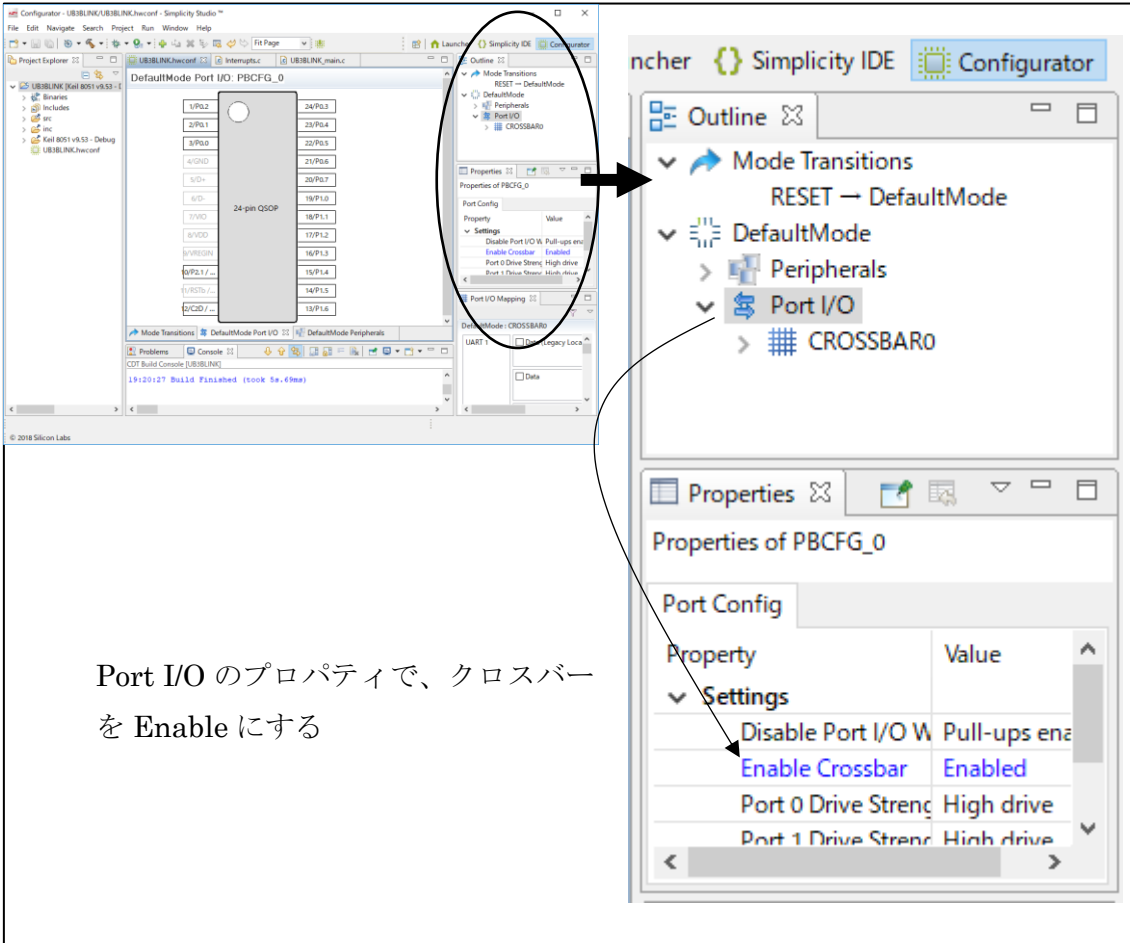


図 8 : P1.6 の I/O モードの変更・設定

3.7.2 CROSSBAR の設定

EFM8 の GPIO ピンと内部 I/O の間にあるクロスバー（CROSSBAR）を Enable にします。（これを忘れると CPU によるポート出力が行えません）



The image shows the Simplicity Studio Configurator interface. On the left, a 24-pin QSOPT package is shown with pins 1 through 24 labeled. The 'Port I/O' property is selected in the 'Properties' pane. The 'Enable Crossbar' property is highlighted in blue and set to 'Enabled'. The 'Port I/O' property is also highlighted in blue. The 'Port I/O' property is also highlighted in blue.

Port I/O のプロパティで、クロスバーを Enable にする

Property	Value
Settings	
Disable Port I/O W. Pull-ups ena	Enabled
Enable Crossbar	Enabled
Port 0 Drive Streng	High drive
Port 1 Drive Streng	High drive

図 9: クロスバーをイネーブルにする

3.7.3 ペリフェラル（内部 I/O）の設定

「DefaultMode Peripherals」タブをクリックしてペリフェラル設定画面を開きます。

3.7.3.1 Watchdog Timer

WDT によるリセットがかからないようにするため、WDT のプロパティで WDT Enable を Disable にします。

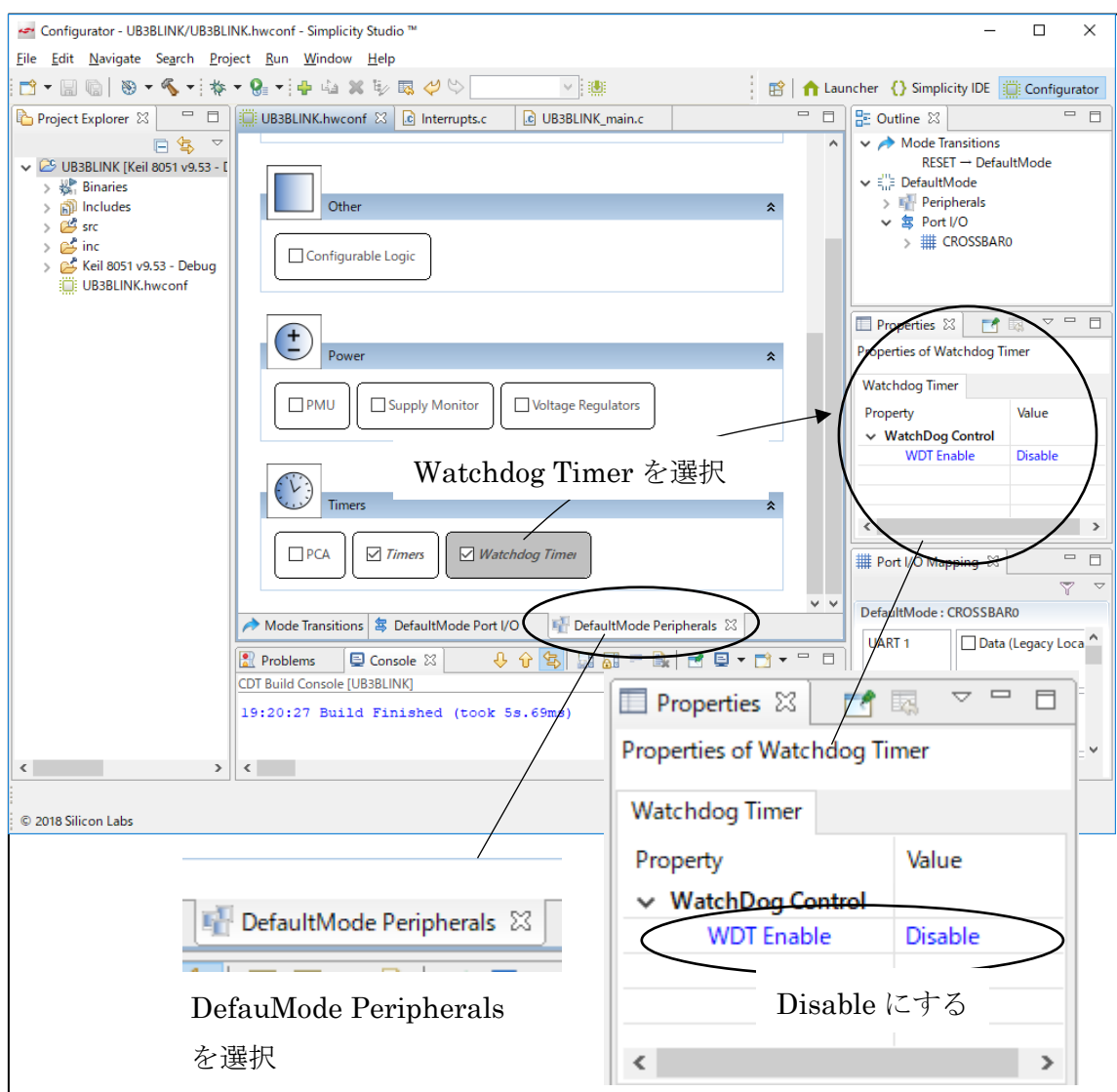


図 10 : WDT の設

3.7.3.2 タイマの設定

タイマから 10Hz 周期の割り込みをかけます。EFM8UB3 には Timer0 から Timer5 までの 6 つのタイマがありますが、ここでは Timer2 を使用します。Timer2 のプロパティで、

- Run Control : Start
- Target Overflow Frequency:10 (10Hz の意味)

に設定します。

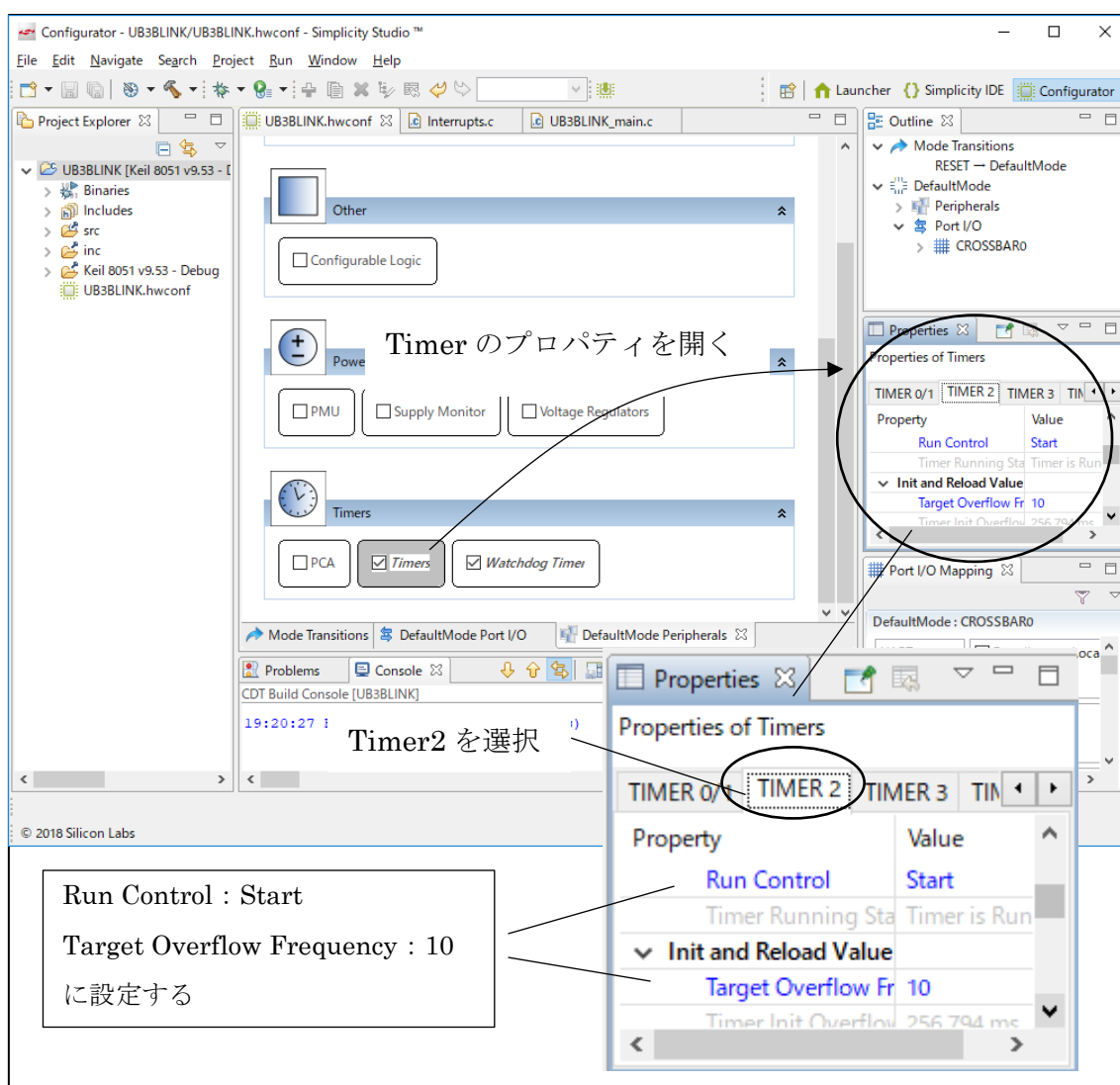


図 11 : Timer の設定

3.7.3.3 割込みコントローラの設定

タイマからの割り込みが受け付けられるよう、割込みの設定を行います。**Enable All Interrupts** を **Enable** にすると、個別に **Enable** された割り込みソースからの割り込みが入ようになります。更に **Timer2** の割り込みを **Enable** にすることで **Timer2** からの割り込みが入ようになります。

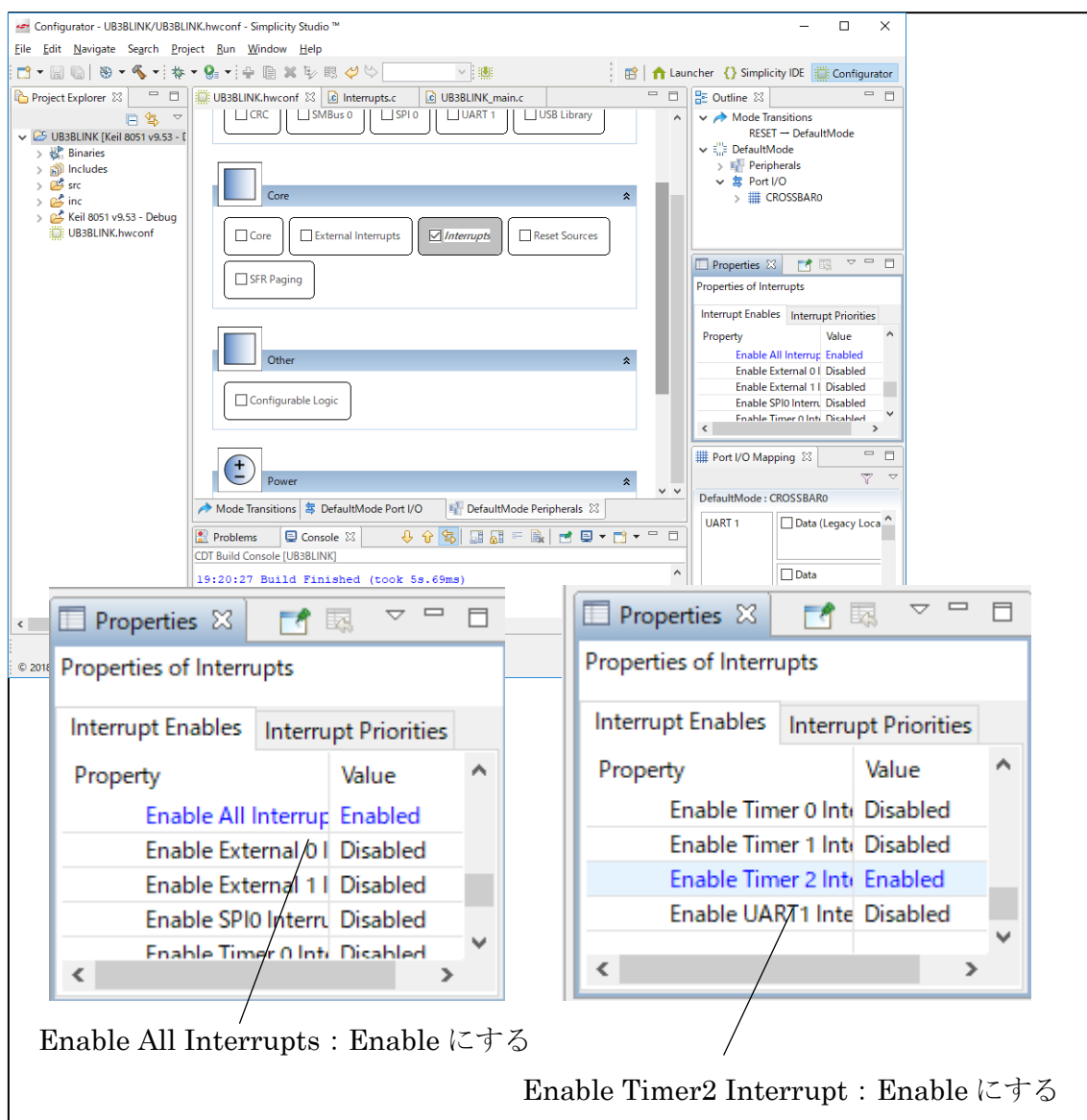


図 12 : 割込みの設定

3.8 ソースコード生成

画面上で右クリックすると、図 13 のようなポップアップメニューが現れますので、「Generate Source」を選択します。

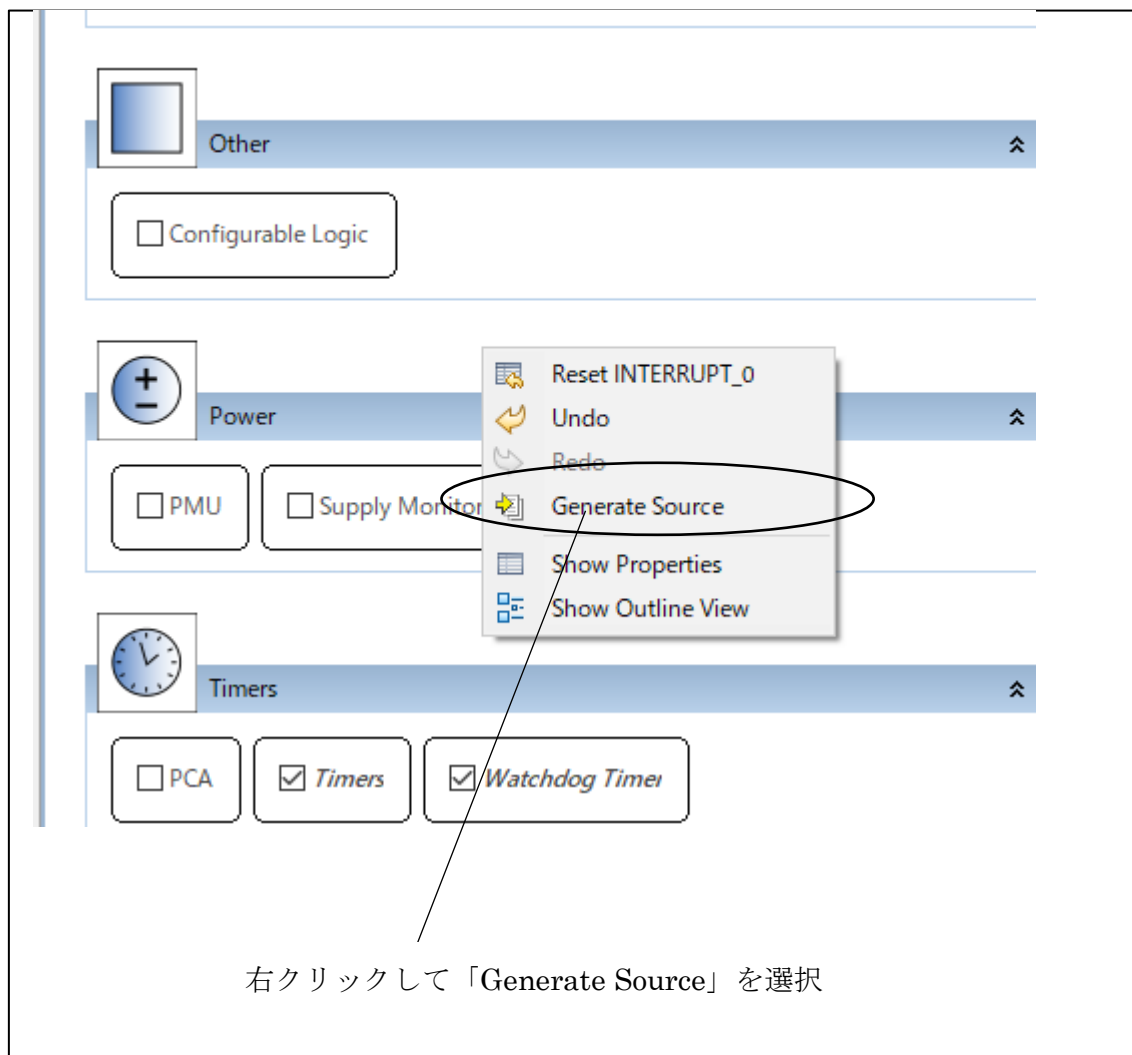


図 13 : ソースコード生成

3.9 ソースコード記述

サンプルでは割り込み処理の中で LED の点滅を行います。割り込み処理のエンタリは、interrupt.c に生成されています。図 14 のように interrupt.cni に LED の点滅コードを追加します。ここでは割り込みが入るたびに P1.6 (LED に接続されています) の状態を反転しています。

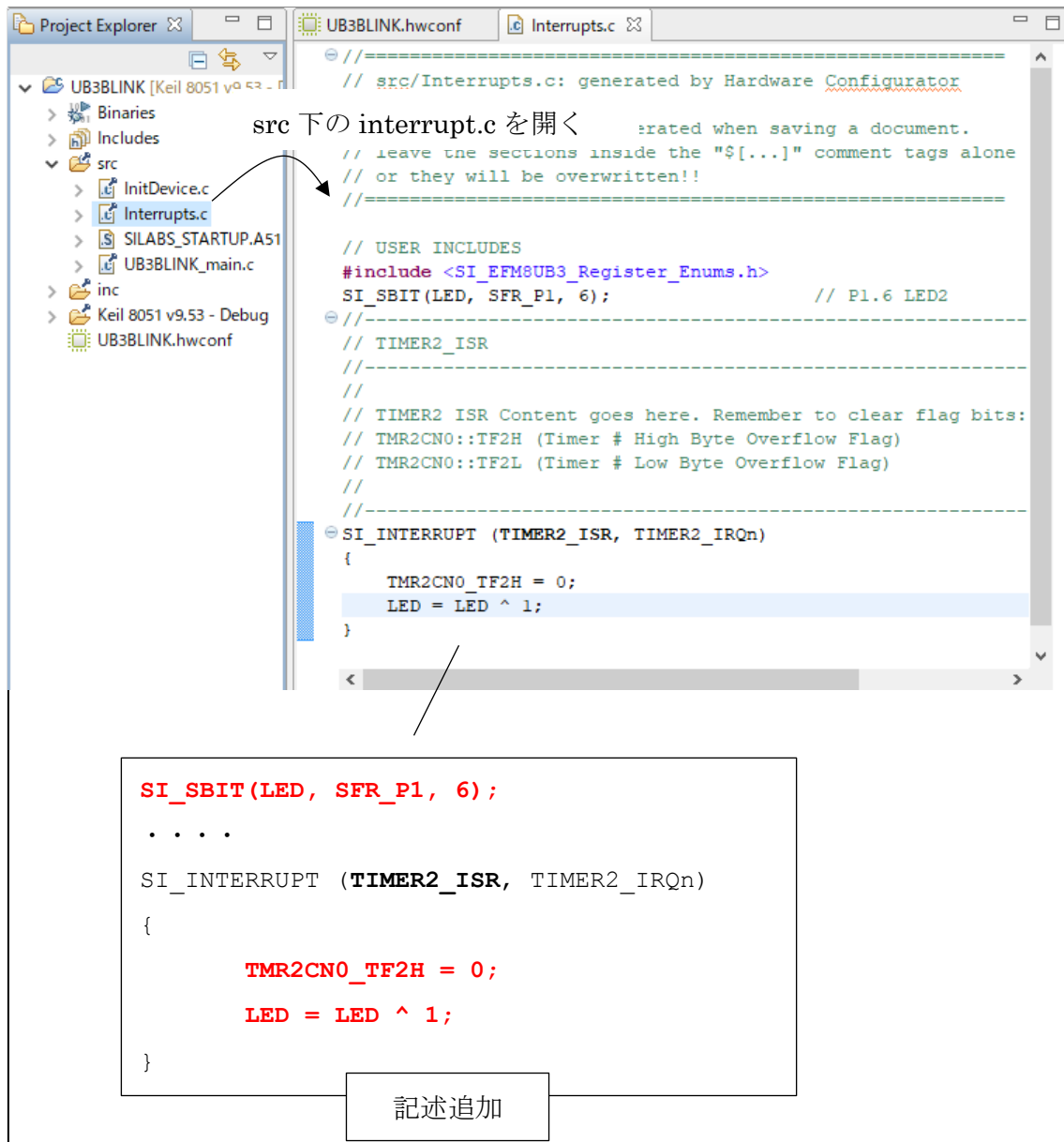


図 14 : 割り込み処理の記述

3.10 ビルド

ソースコードの記述ができたら、ビルドします。画面左端のプロジェクトエクスプローラで、プロジェクト名のところを右クリックします。

ポップアップメニューから **Build Project** を選択すると、ビルドが始まります。

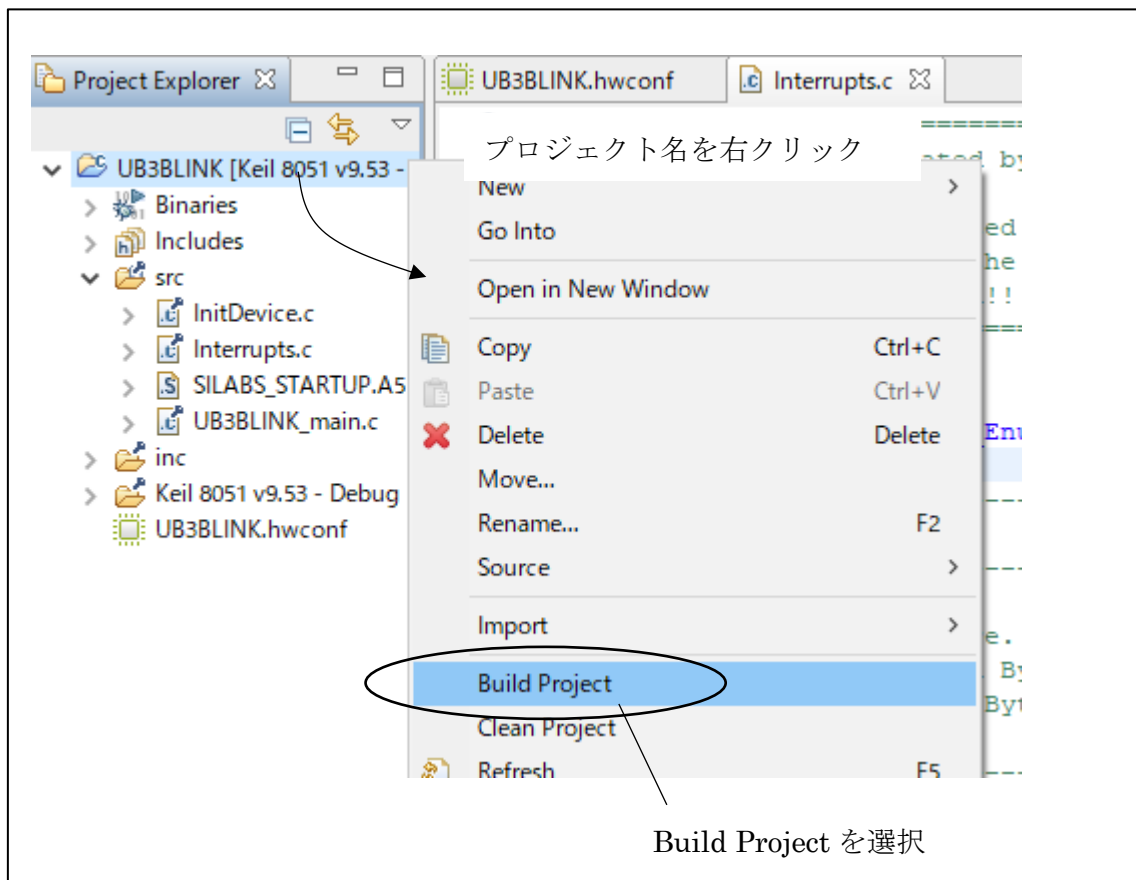


図 15 : ビルド実行

3.11 ダウンロードツールのコピー

ビルドすると HEX ファイルが生成されます。これを hex2boot.exe を使ってファイル変換し、efm8load.exe を使ってブートローダで書きこみます。AN945SW を解凍すると、この二つのファイルがありますので、プロジェクトの HEX ファイルと同じ場所にドラッグ&ドロップしてコピーします。

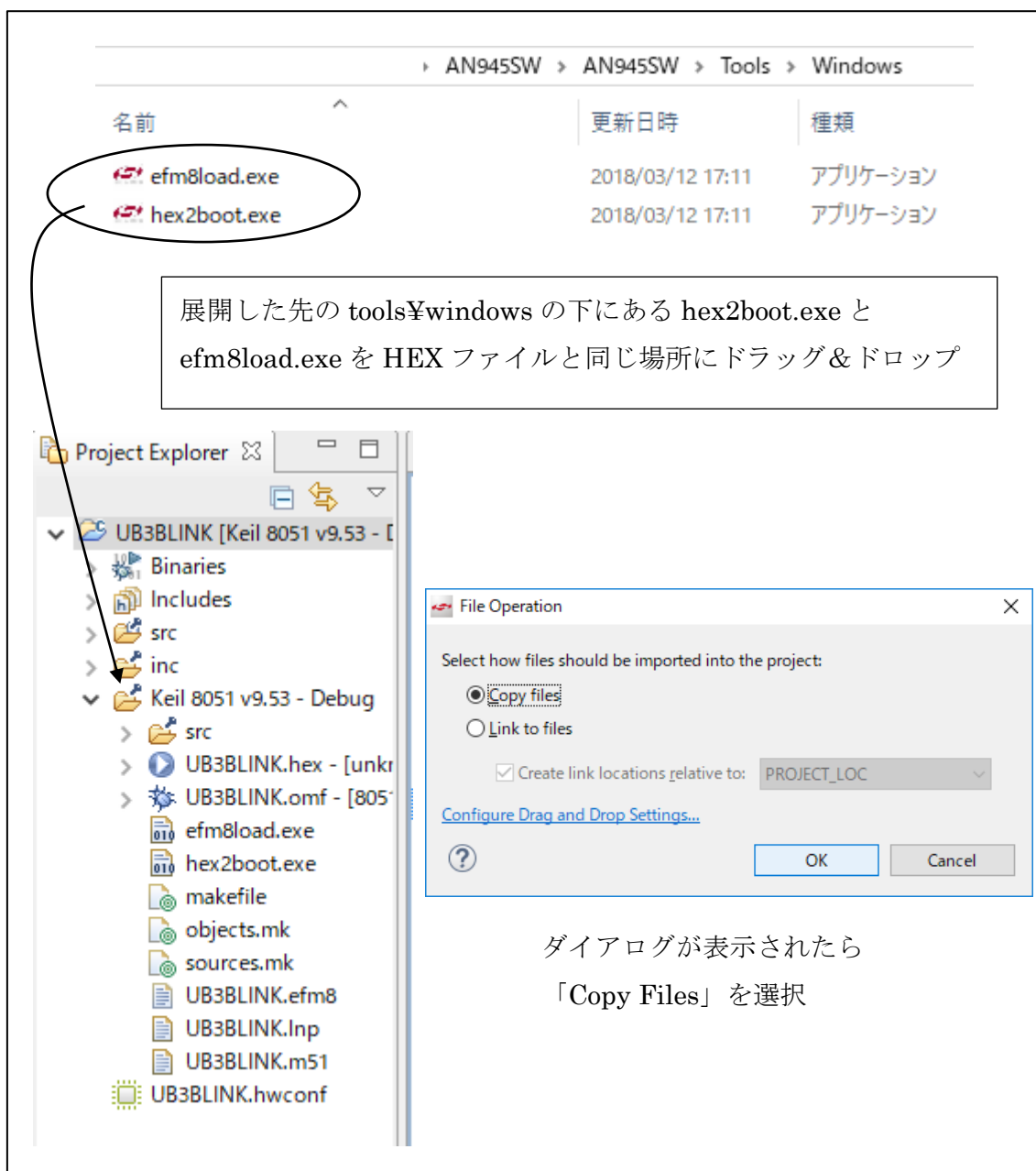


図 16 : ダウンロード用ツールのコピー

3.12 PM-EBU02 をブートローダモードで起動

3.13 ブートローダモードで起動

図 17 のように、P2 の 1-7 番ピンを短絡した状態で、USB ケーブルを PC に接続します。EFM8UB3 の USB ブートローダが起動し、PC に USB デバイスとして認識されます。

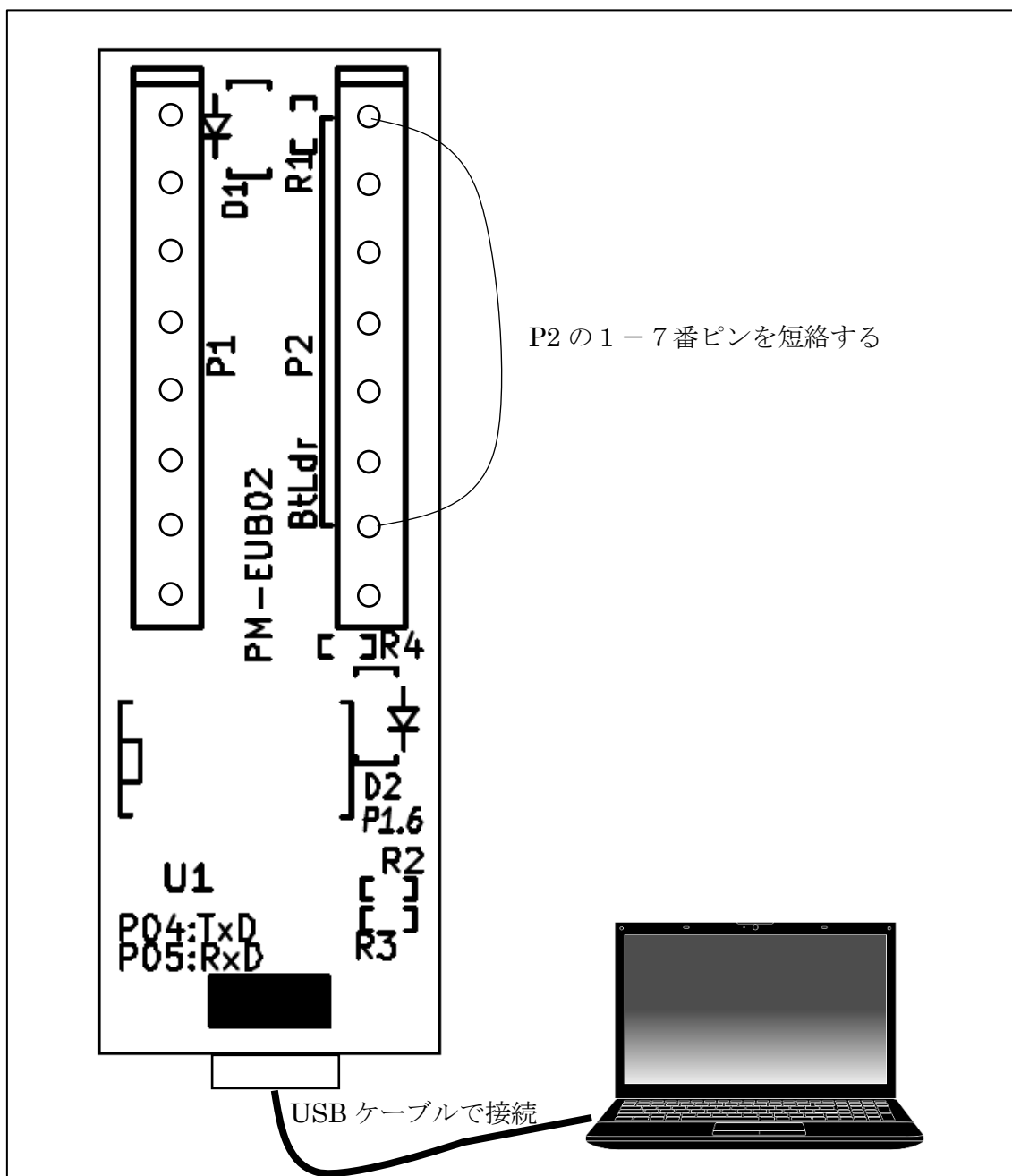


図 17: ブートローダモードで起動

3.14 コマンドラインを開く

次に HEX ファイルの変換とダウンロードを行います。この作業はコマンドライン（DOS 窓）を使います。図 18 のように、HEX ファイルを右クリックして、「Open Command Line Here」を選択します。

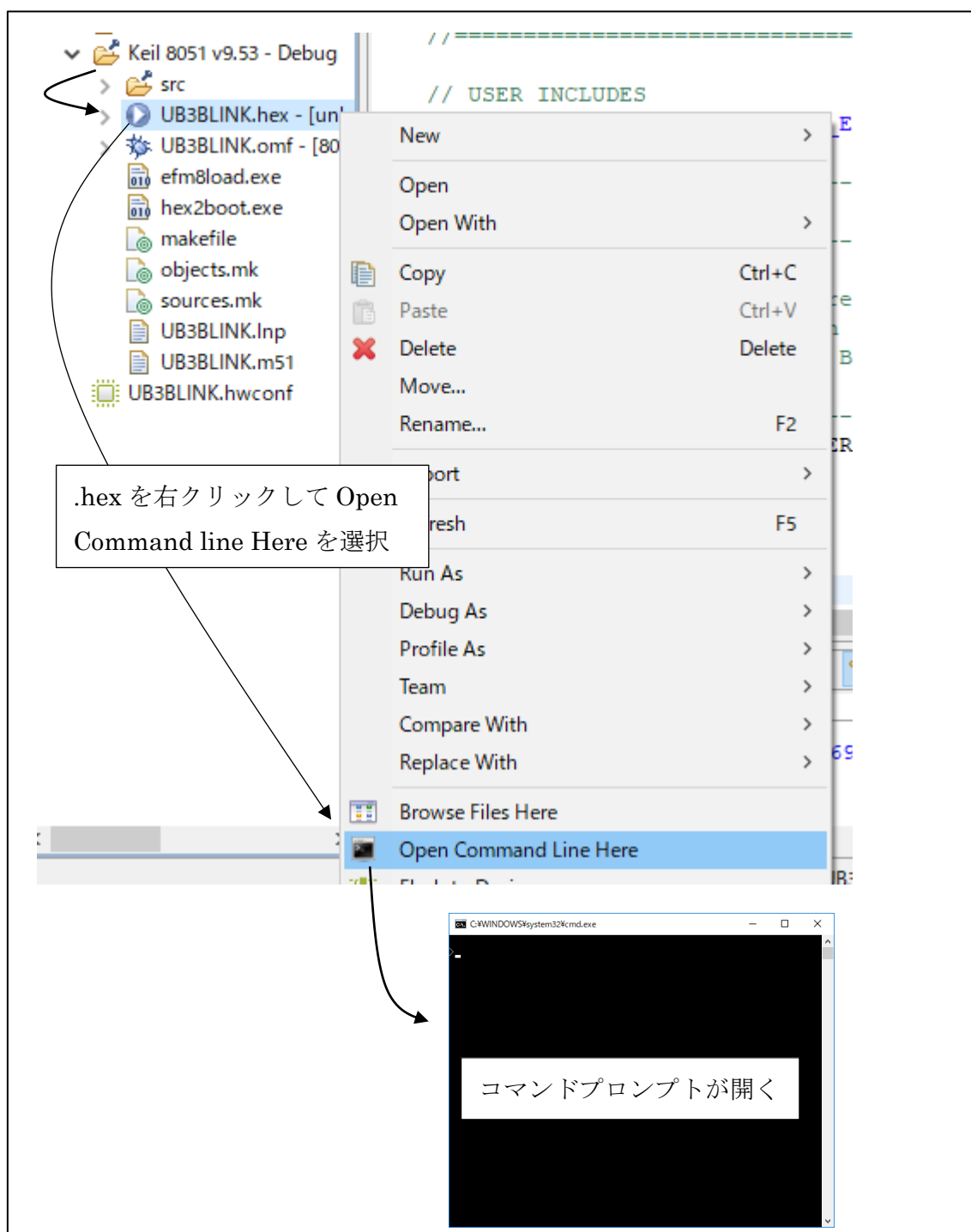


図 18 : コマンドプロンプトを開く

3.15 ファイル変換と書き込み

コマンドプロンプトが開いたら図 19 のように、

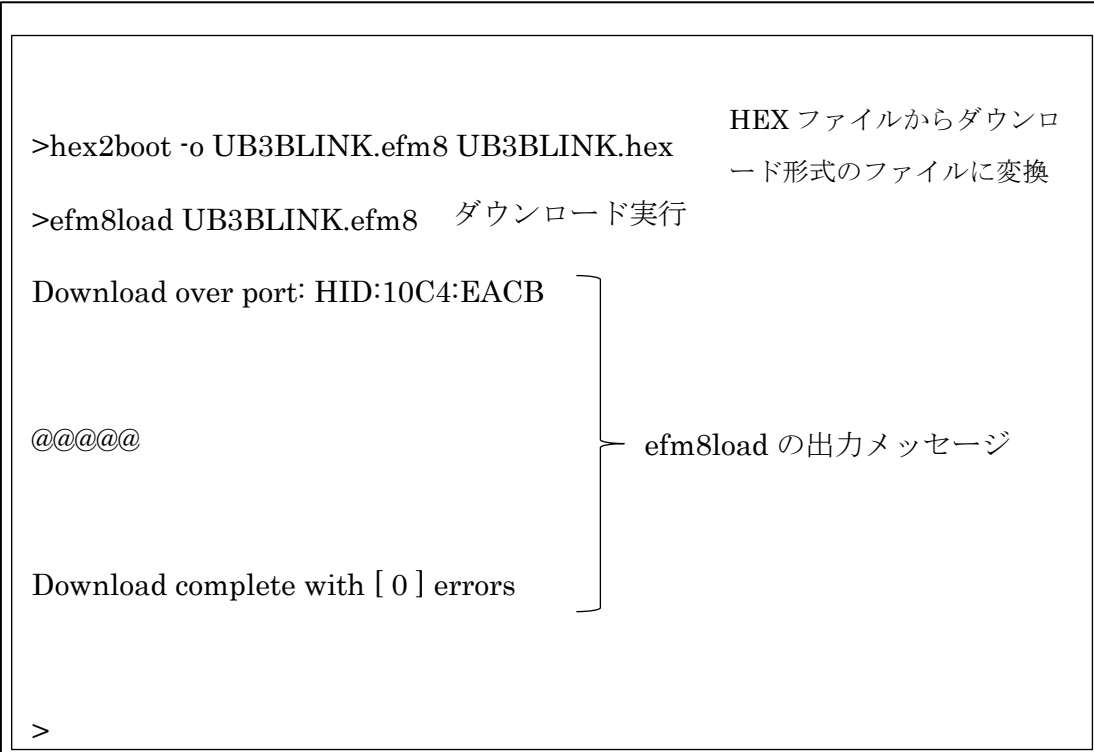
```
hex2boot -o UB3BLINK.efm8 UB3BLINK.hex
```

としてダウンロード形式のファイルに変換します。**-o** オプションの後に出力ファイル名を付けます。ファイル名は拡張子も含めて自由につけることができますが、わかりやすいように拡張子を**.efm8** としました。

変換できたファイルは **efm8load.exe** を使ってダウンロードします。

```
efm8load UB3BLINK.efm8
```

とすると、ブートローダモードになっている **EFM8** デバイスを自動的に見つけて、ダウンロード（書き込み）が行われます。ダウンロード後、基板上の **LED (D2)** が 1/10 ごとに点灯／消灯を繰り返えます。



```
>hex2boot -o UB3BLINK.efm8 UB3BLINK.hex
>efm8load UB3BLINK.efm8
Download over port: HID:10C4:EACB
@@@@@
Download complete with [ 0 ] errors
>
```

HEX ファイルからダウンロード形式のファイルに変換

ダウンロード実行

efm8load の出力メッセージ

図 19 : ファイル変換と書き込みの実行

4 付録

4.1 回路図

PM-EUB02 の回路図です。図中の部品型名・手配コードは参考です。

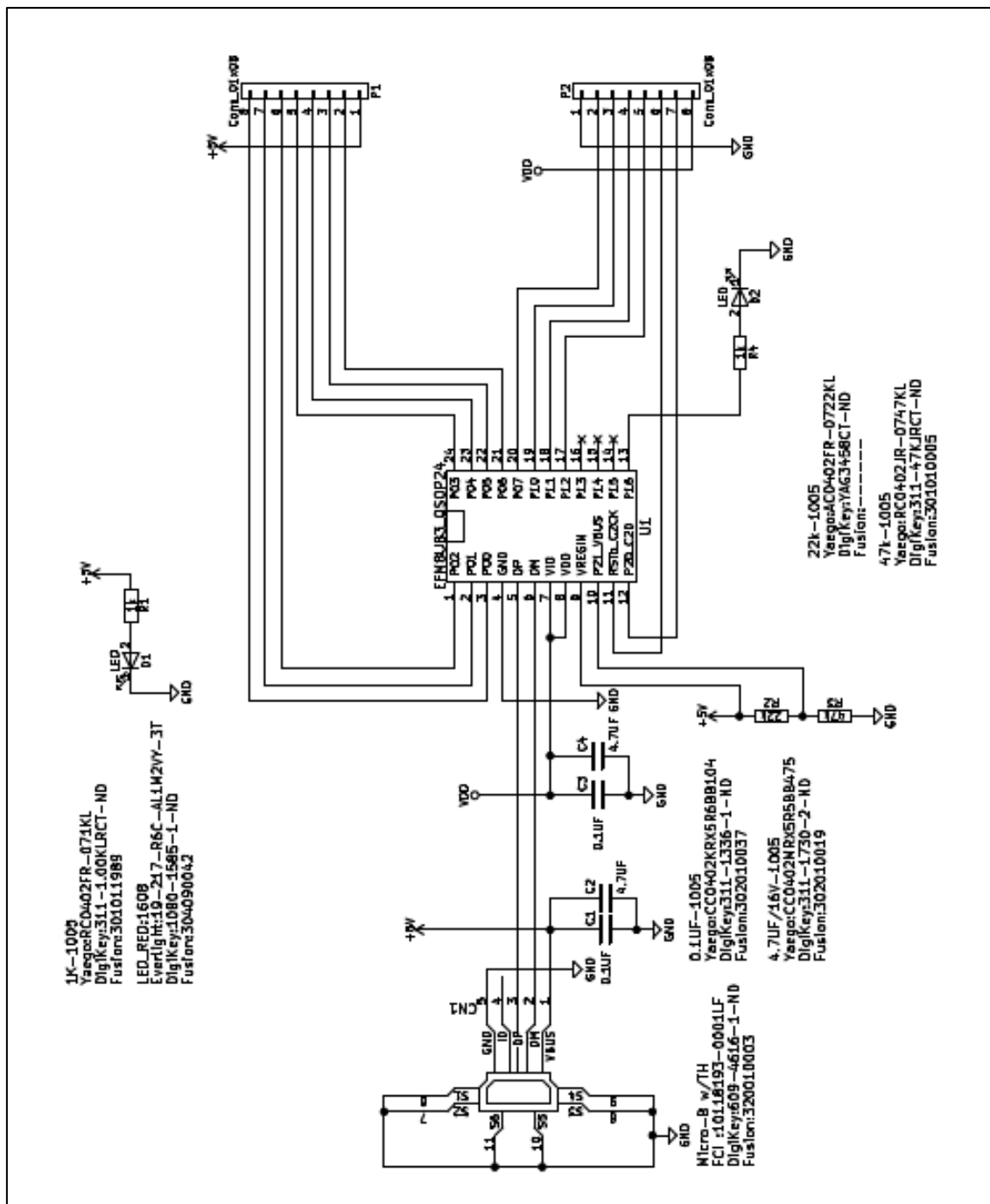


図 20 : PM-EUB02 回路図 (参考)

4.2 電源系統

PM-EUB02 の電源系統は図 21 のようになっています。PM-EUB への供給電源は +5V で、USB コネクタ (VBUS) または I/O コネクタの +5V 端子から供給できます。

この両者は直結されていますので、USB コネクタから電源供給したときには I/O コネクタの +5V 端子は電源出力になります。逆接防止などはありませんので、両方から +5V を供給しないように気をつけてください。

供給された 5V は CPU の内部レギュレータで 3.3V に落とされ、これが CPU の動作の電源や I/O 電圧として利用されます。+3.3V レギュレータの最大出力電流は 100mA (データシートによる) ですが、CPU 内部の消費電流は 10mA 程度ですので、3.3V を外部回路で利用することもできます。

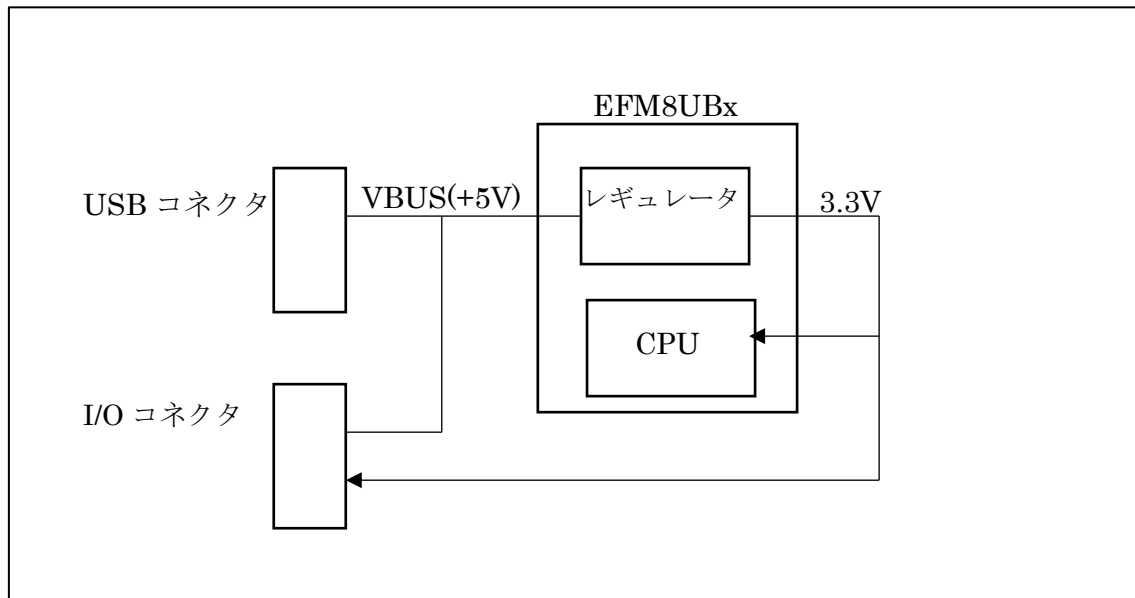


図 21 : 電源系統図

4.3 CLU について

EFM8UB31 の大きな特徴は、内部に CLU（Configurable Logic Unit：プログラマブルな論理回路ユニット）を持っていることです。CLU ブロックには図 23 のように 4 つの CLU ユニットがあります。入力にはタイマや UART、コンパレータ、I/O ピンなど接続されており、出力は I/O ピン（CLUxOUT）に出力したり、割り込みとして使うことができます。。

各 CLU の内部は図 24 のように、3 入力の LUT（セレクト）と 1 つの D/FF で構成されています。LUT の入力（FNSEL）は 8 ビットのレジスタになっていて、LUT の 3 つの入力信号によってこのうちひとつが選ばれて出力されます。

たとえば、LUT をエラー! 参照元が見つかりません。の左のように設定すると、右側の論理回路と同じ結果が得られます。このように、LUT によって最大 3 入力の論理回路が組めることになります。

詳細についてはデータシートをごらんください。

入力[2:0]	出力
000	0
001	0
010	0
011	1
100	0
101	1
110	0
111	1

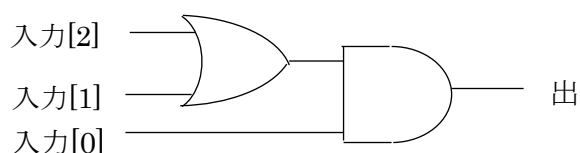


図 22 : LUT の例と、機能互換の論理

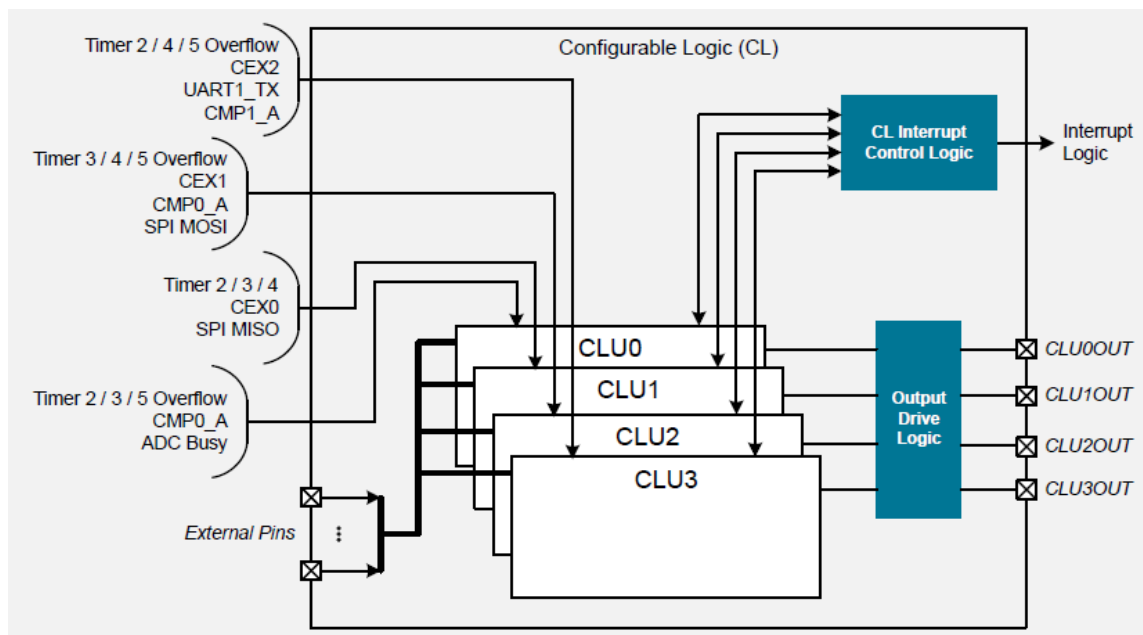


図 23 : CLU ブロック (efm8ub3 reference manual より抜粋)

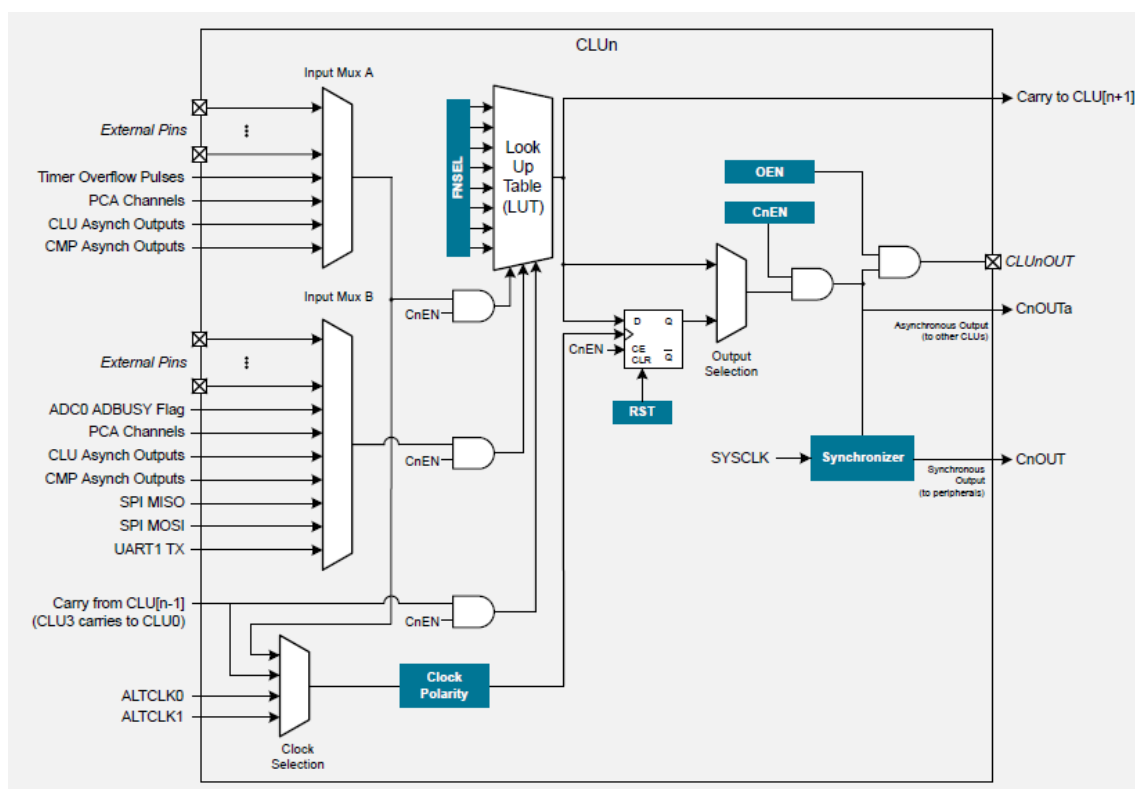


図 24 : CLU の内部構造 (efm8ub3 reference manual より抜粋)