Renesas e^2 studio

スマート・コンフィグレータ Application Examples: CMT, A/D, SCI, DMA, USB

要旨

スマート・コンフィグレータ（SC）は、コード生成機能と、ドライバ、ミドルウェア、端子の設定機能を持つ、GUI ベースのツールです。SC はルネサスの MCU ファミリそれぞれに適したコードを生成し、FIT モジュールによって生成されたコードをインポートする機能を持ちます。

このアプリケーションノートは、ユーザが e^2 studio 上の SC を使用して、ドライバを設定しコードを生成するためのガイドです。

対象デバイス/対応コンパイラ

サポートしているデバイス及びコンパイラは、以下の URL をご参照ください。

https://www.renesas.com/smart-configurator

ホスト PC の OS は以下をサポートします。

- Windows 7　32 ビット/64 ビット
- Windows 8.1　32 ビット/64 ビット
- Windows 10　32 ビット/64 ビット

ソフトウェアコンポーネント

スマート・コンフィグレータは、2 種類のソフトウェアコンポーネント（コード生成（CG）と、Firmware Integration Technology (FIT)）に対応します。それぞれのソフトウェアが対応するドライバとミドルウェアは、以下のとおりです。

- ベーシックドライバ：CG ドライバ（CMT、A/D コンバータ、SCI など）、
  FIT モジュール（CMT、DTC、DMAC、RSPI、SCIFA など）
- ミドルウェア：FIT モジュール（USB、Ethernet、フラッシュメモリ（内蔵フラッシュメモリ書き換え）など）

ベーシックドライバは、CMT や A/D コンバータ、SCI などマイコンの周辺機能の制御プログラムで、コード生成（CG）機能を使ったソフトウェアコンポーネント（CG ドライバ）の組み込みが便利です。また、USB や Ethernet、フラッシュメモリ（内蔵フラッシュメモリ書き換え）などのミドルウェアを含んだ FIT モジュールをソフトウェアコンポーネントとして組み込むことが可能です。

略称一覧：

CMT: Compare Match Timer

DTC: Data Transfer Controller

DMAC: Direct Memory Access Controller

RSPI: Serial Peripheral Interface

SCIFA: FIFO Embedded Serial Communications Interface

目次

1. 概要 4
1.1 本ドキュメントの目的 4
1.2 動作環境 4
1.3 スマート・コンフィグレータプロジェクトの基本操作フロー 5

2. Application Example 1 （スマート・コンフィグレータを使用した基本的なプログラムの作成） 6
2.1 ワークスペースの作成 8
2.2 プロジェクトの作成 8
2.3 ペリフェラルドライバの追加 12
2.4 コード生成 18
2.5 “r_cg_userdefine.h”ファイルに定義コードを追加 19
2.6 コンペアマッチタイマドライバにアプリケーションコードを追加 19
2.7 ビルドとハードウェアボードでの実行 21
2.8 LED2 の点滅インターバルの変更 25
2.9 ボードでの動作確認 26

3. Application Example 2（スマート・コンフィグレータでの機能追加） 27
3.1 ペリフェラルドライバの追加 29
3.2 コード生成 31
3.3 シングルスキャンモード S12AD ドライバにアプリケーションコードを追加 32
3.4 *main()*にアプリケーションコードを追加 33
3.5 ビルドとハードウェアボードでの実行 35
3.6 ボードでの動作確認 35

4. Application Example 3（スマート・コンフィグレータでの機能変更） 36
4.1 S12AD ドライバと関連するアプリケーションコードの削除 39
4.2 ペリフェラルドライバの追加 40
4.3 コード生成 43
4.4 SCI/SCIF 調歩同期式モードドライバにアプリケーションコードを追加 44
4.5 *main()*にアプリケーションコードを追加 47
4.6 ビルドとハードウェアボードでの実行 51
4.7 ボードでの動作確認 53

5. Application Example 4（DMA を使用したデータ転送） 54
5.1 周辺ドライバの追加と修正 58
5.1.1 DMAC0 ドライバの追加 58
5.1.2 DMAC1 ドライバの追加 60
5.1.3 SCI8 ドライバの修正 62
5.2 コード生成 62

5.3 DMAC0 ドライバにアプリケーションコードを追加 63
5.4 DMAC1 ドライバにアプリケーションコードを追加 64
5.5 SCI8 ドライバのアプリケーションコードを修正 67
5.6 *main()*にアプリケーションコードを追加 68
5.7 ビルドとハードウェアボードでの実行 71
5.8 ボードでの動作確認 71

6. Application Example 5（スマート・コンフィグレータを使った USB Function を含む FIT モジュールの組み込み） 72
6.1 SCI/SCIF 調歩同期式モードドライバと関連するアプリケーションコードの削除 75
6.2 ペリフェラルドライバの追加 77
6.3 コード生成 81
6.4 USB PCDC ドライバのヘッダファイルの追加 82
6.5 *main()*にアプリケーションコードを追加 90
6.6 ビルドとハードウェアボードでの実行 93
6.7 ボードでの動作確認 95
6.8 開発支援ツール（QE） 95

ホームページとサポート窓口 96

1. 概要

1.1 本ドキュメントの目的

Application Examples におけるスマート・コンフィグレータの使用方法をガイドします。

1.2 動作環境

対象デバイス	RX64M グループ RX65N, RX651 グループ RX130 グループ
デバッガ	E1 /E2 Lite
IDE	e2 studio V7.0.0 以降
ツールチェーン	RX ファミリ用ルネサス C/C++ コンパイラパッケージ

1.3　スマート・コンフィグレータプロジェクトの基本操作フロー

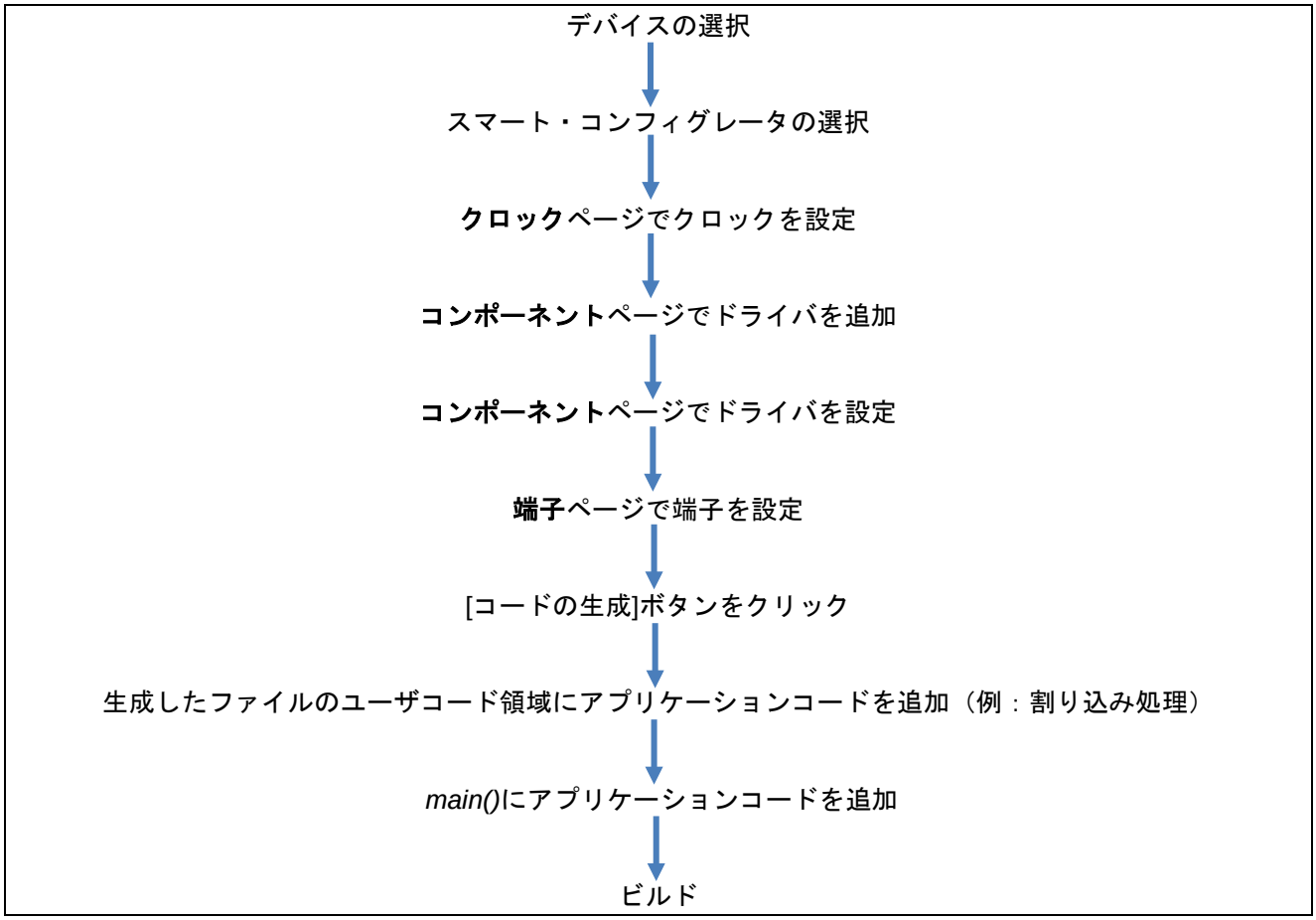


図 1-1-1　基本操作

スマート・コンフィグレータの詳細な操作については、“スマート・コンフィグレータ　ユーザガイド”を参照してください。

2. Application Example 1 （スマート・コンフィグレータを使用した基本的なプログラムの作成）

本章では、スマート・コンフィグレータ上でコード生成（CG）ドライバを使用して、LED 点滅プロジェクトを作成する方法について説明します。この例では、以下のリソースが必要です。

- IDE：e² studio V7.0 以降
- ツールチェーン：Renesas RXC Toolchain
- ボード：RX65N-2MB 用の Renesas Starter Kit+
- MCU：RX65N　R5F565NEDxFC
- デバッガ：E1 または E2 Lite

RX65N-2MB 用の Renesas Starter Kit+の回路図は、以下の通りです。この例では、5000 回カウントするたびに LED2 が点滅するように設定されます。



図 2-1　RX65N-2MB 用の Renesas Starter Kit+の回路図

下記の表に設定する CG ドライバを示します。

ドライバ	リソース	機能
ポート	PORTG	LED2 (PG6) に接続します。 LED2 を点灯させるために、ローレベル“0”を出力します。
コンペアマッチタイマ	CMT0	5000 回ごとに、出力コンペア割り込みハンドラを使用して LED を切り替えます。

下記の回路図では、RX65N メインクロックは 24MHz の水晶振動子に接続されています。このクロックは、2.7 章のデバッグ構成で、メインクロックとして設定します。

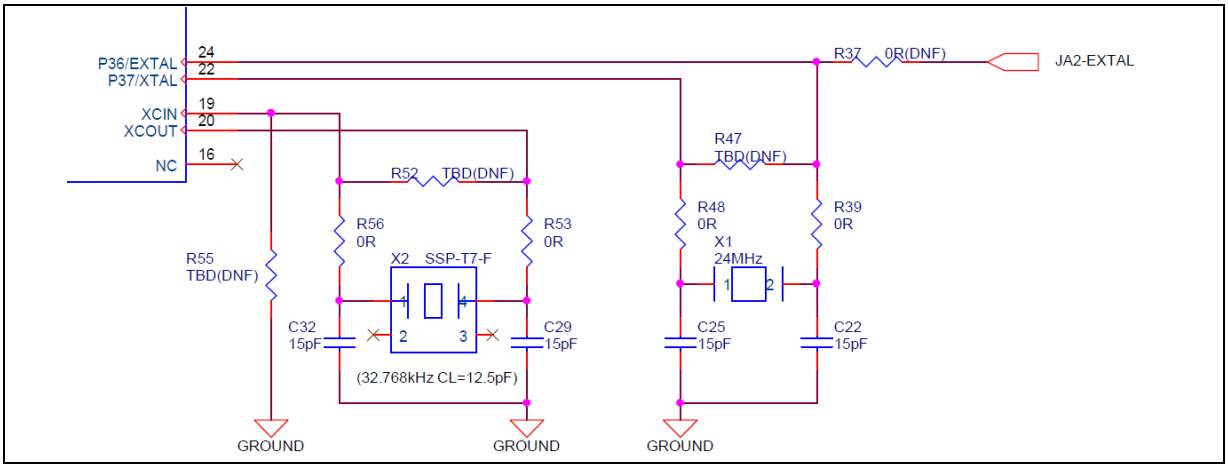


図 2-2　RX65N-2MB 用の Renesas Starter Kit+の回路図

プログラムフローチャートは、以下の通りです。

a)　main 関数：

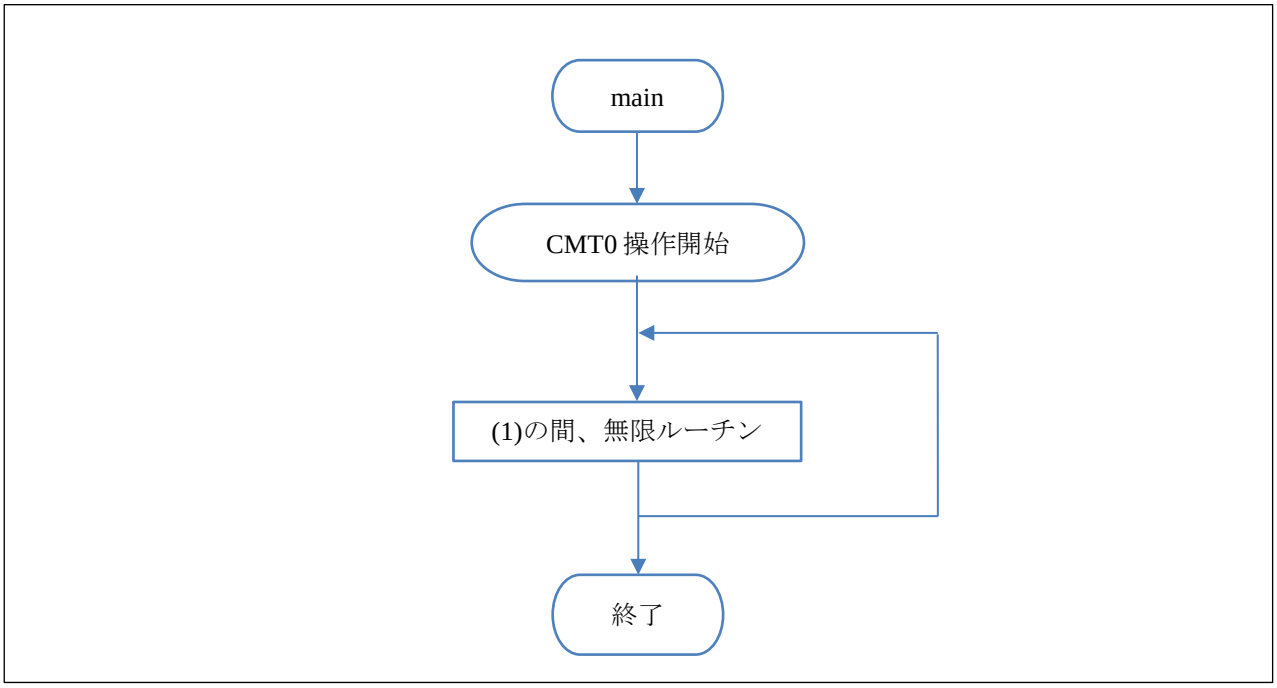


図 2-3　main フローチャート

b)　CMT0 割り込み関数：

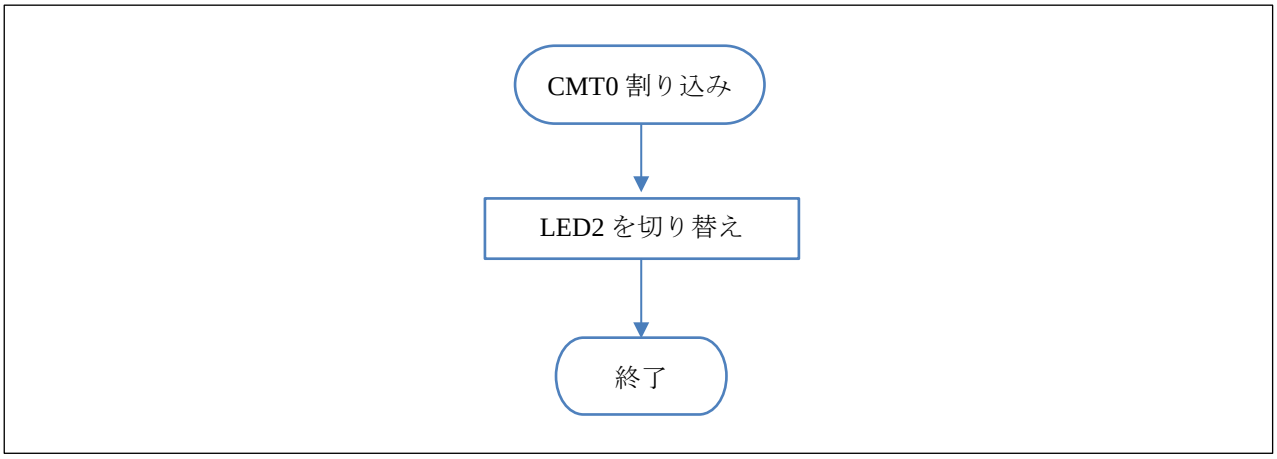


図 2-4　CMT0 割り込みフローチャート

2.1　ワークスペースの作成

1) Windows®スタートメニューから、e²studio を起動します。デフォルトのワークスペースフォルダを使用し、[OK]をクリックします。

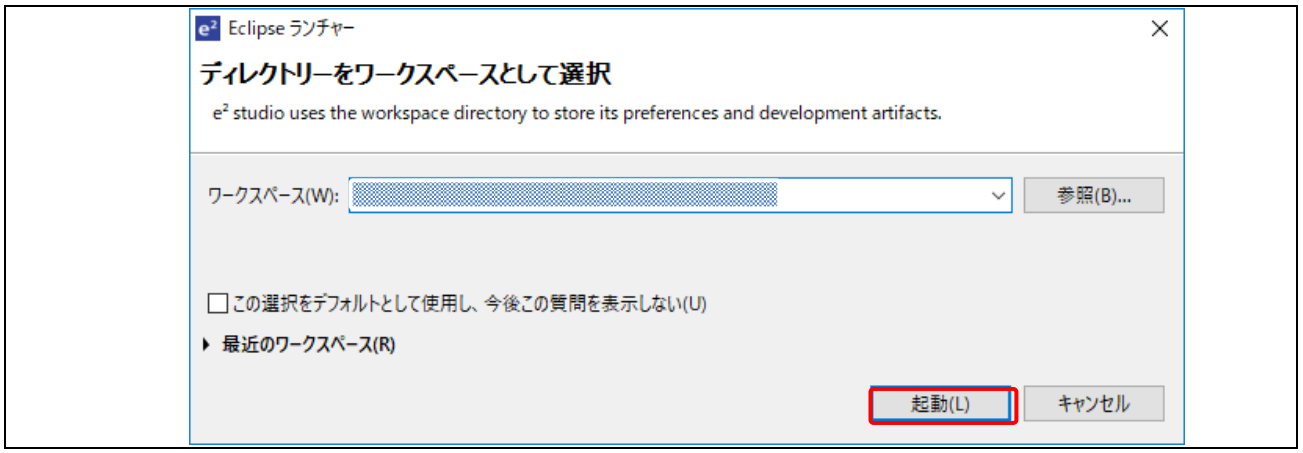


図 2-5　ワークスペースランチャー

2.2　プロジェクトの作成

1) e²studio で、新しい C プロジェクトを作成します。

[ファイル] → [新規] → [C/C++Project]を開き、新しいプロジェクトを作成します。



図 2-6　ファイルメニューからプロジェクトを作成

2)　[Renesas RX] → [Renesas CC-RX C/C++Executable Project]を選択し、[次へ]をクリックします。

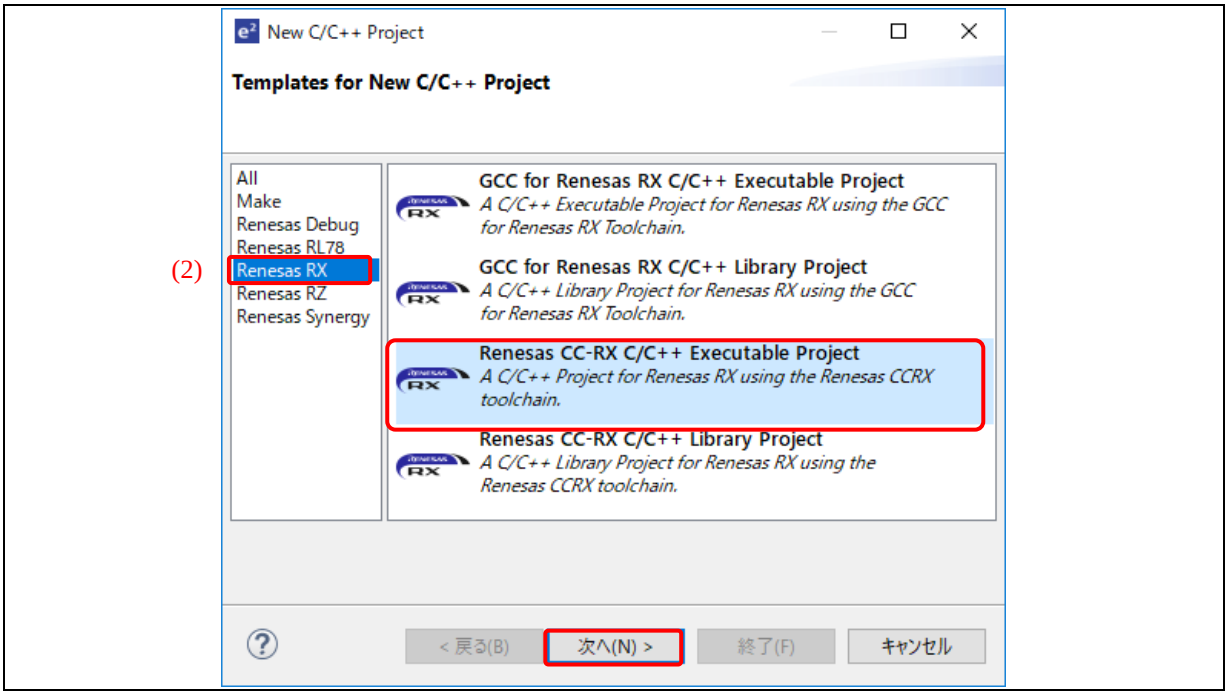


図 2-7　ファイルメニューからプロジェクトを作成

3)　プロジェクト名を入力し(ここでは“Smart_Configurator_Example”とします)、[次へ]に進みます。

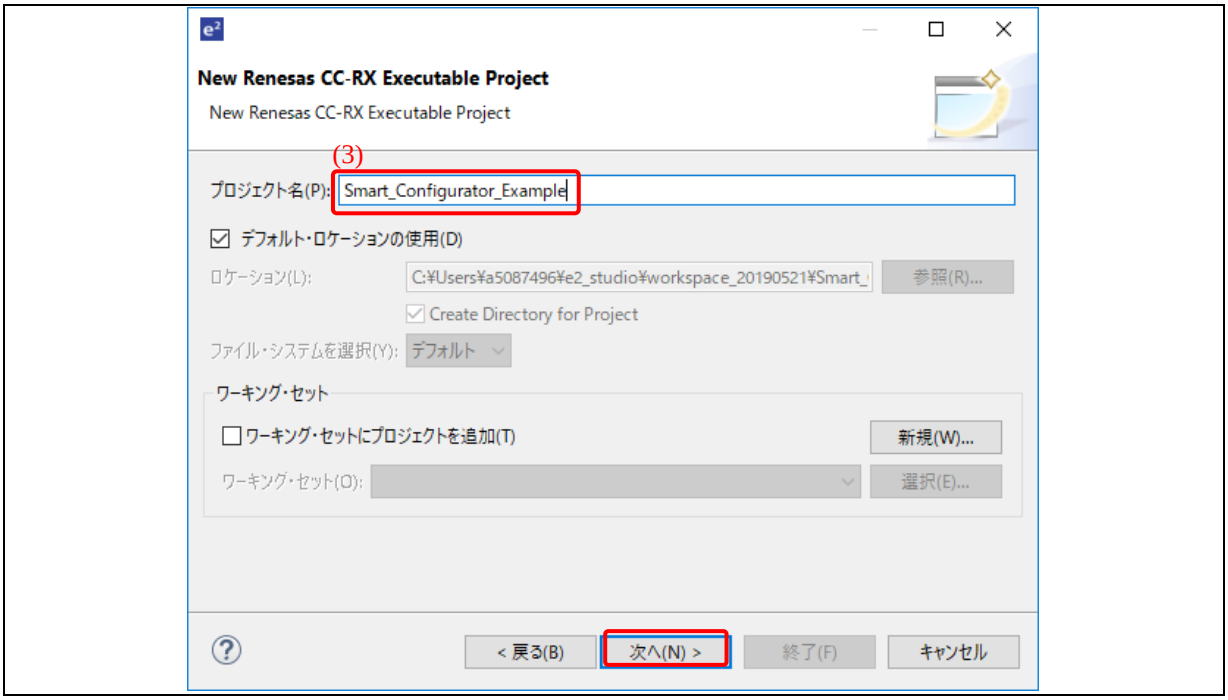


図 2-8　ファイルメニューからプロジェクトを作成

4) 言語は“C”を選択します。
5) ツールチェーンは“Renesas CCRX”を選択します。
6) ツールチェーン・バージョンを選択します。（例：v3.01.00）
7) ターゲット・デバイスを選択します：“RX600 > RX65N > RX65N - 176pin > R5F565NEDxFC”
8) [Hardware Debug 構成を生成]にチェックを入れ、エミュレータを選択します。（例：E2 Lite(RX)）
9) [次へ]をクリックします。

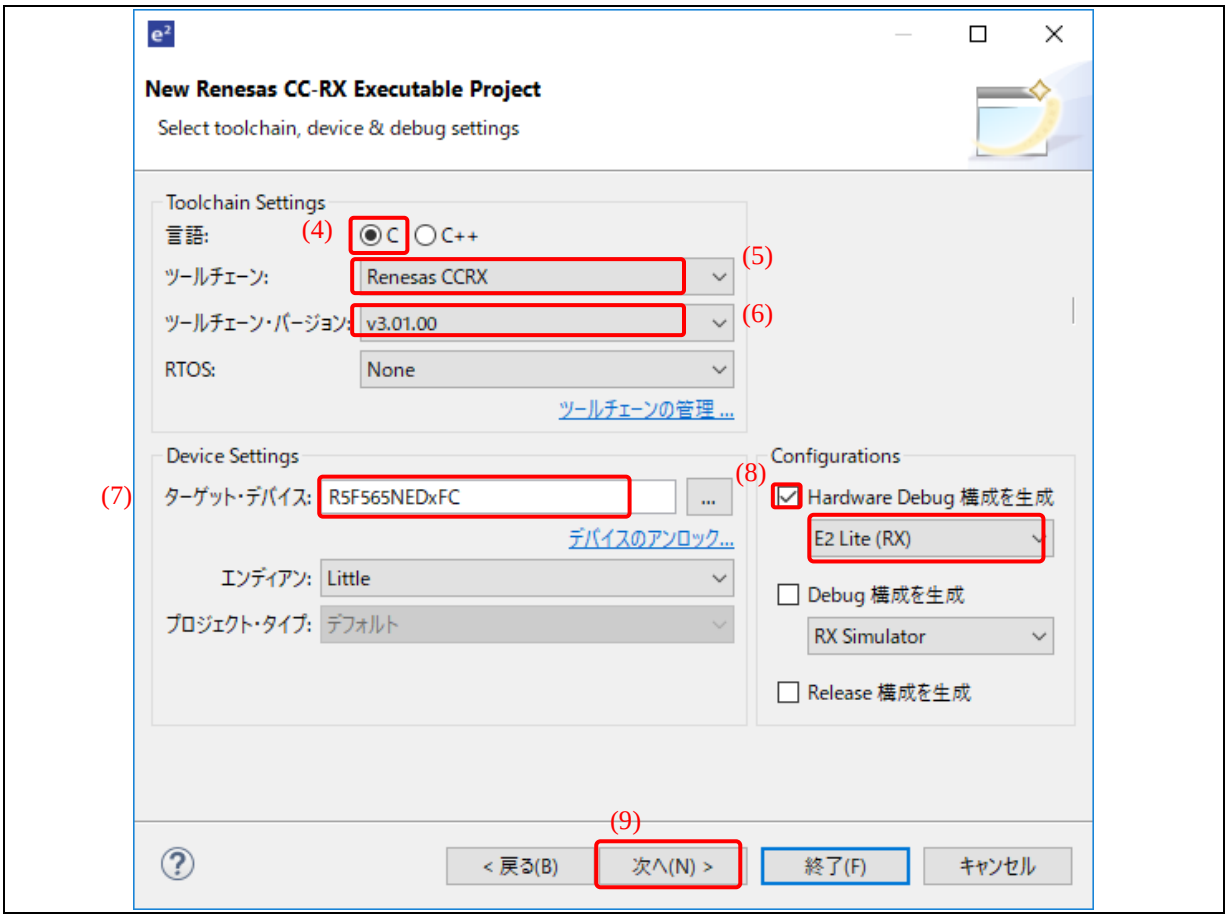


図 2-9　C プロジェクト：E2 Lite エミュレータを使用する場合の設定例

10) “コーディング・アシストツールの選択”ダイアログボックスで、“スマート・コンフィグレータを使用する”にチェックを入れます。

11) [終了]をクリックします。

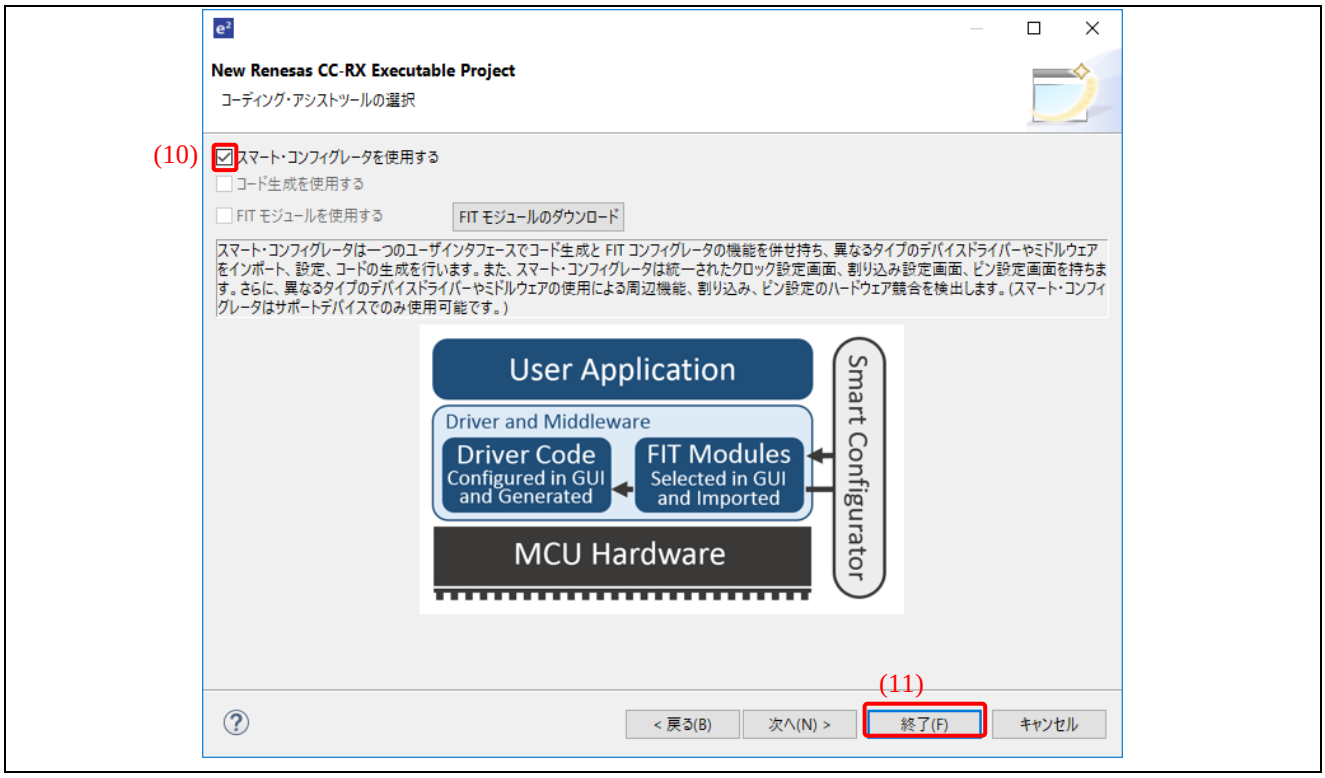


図 2-10　コーディング・アシストツールの選択

2.3　ペリフェラルドライバの追加

以下に示す手順で、スマート・コンフィグレータ・パースペクティブを開始します。

1)　Smart_Configurator_Example.scfg 画面で、[コンポーネント]タブをクリックします。

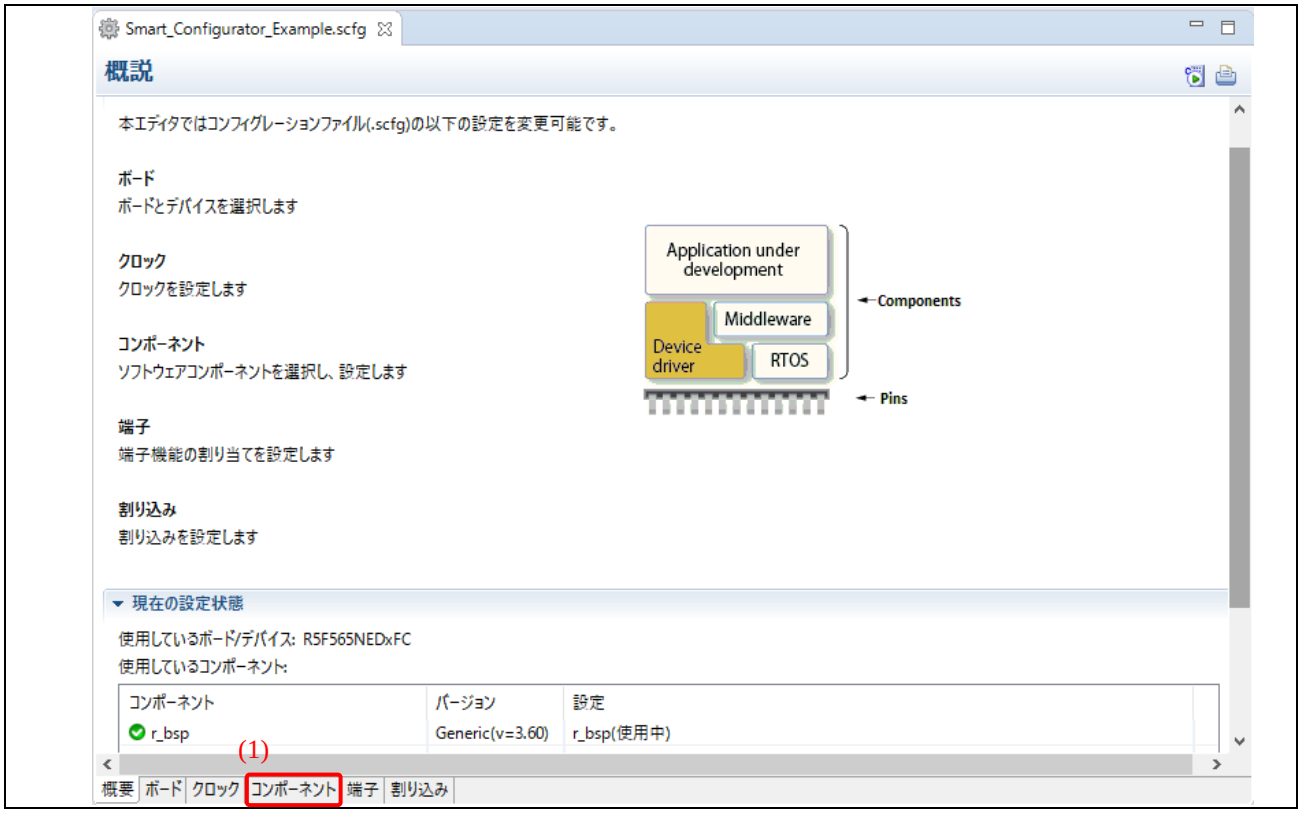


図 2-11　スマート・コンフィグレータ・パースペクティブ

2)　をクリックし、新しいコンポーネントを追加します。

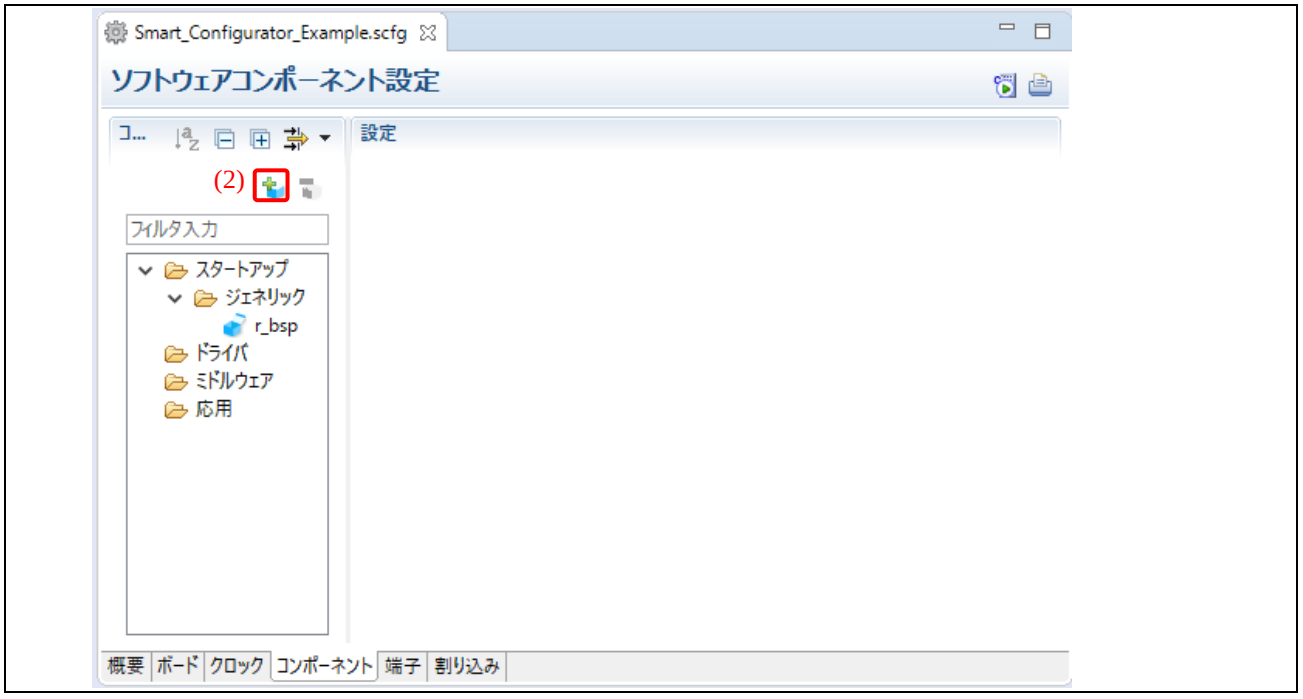


図 2-12　スマート・コンフィグレータでのソフトウェアコンポーネント設定

3) “コンペアマッチタイマ”ドライバをプロジェクトに追加します。

a. “機能”オプションから“タイマ (Drivers)”を選択します。
b. コンポーネントリストから “コンペアマッチタイマ”を選択します。
c. [次へ]をクリックし、続けます。

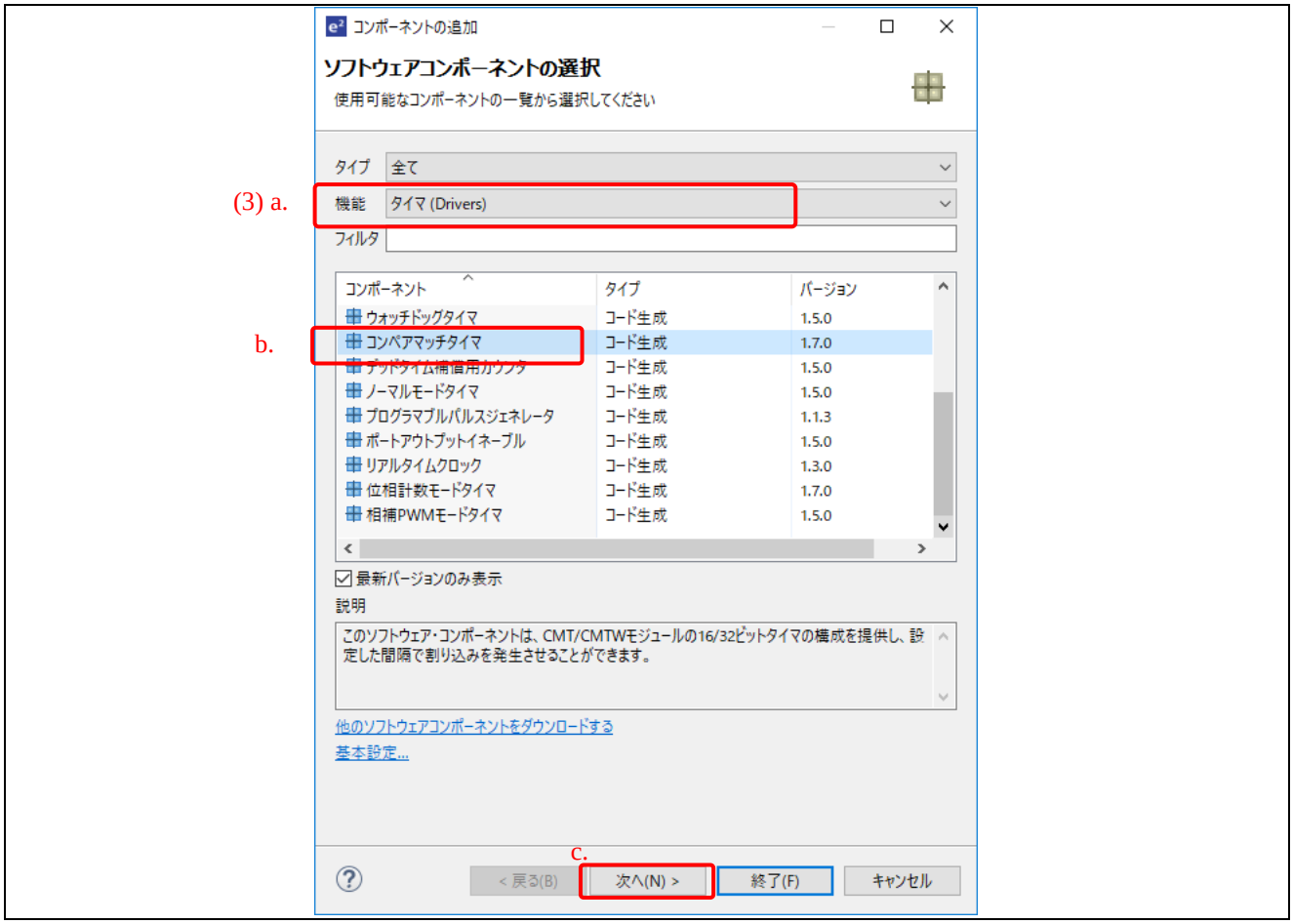


図 2-13　ソフトウェアコンポーネントの選択

d. リソースとして CMT0 を選択します。
e. デフォルトのコンフィグレーション名を、そのまま使用します。このコンフィグレーション名から、ソースファイルと API が生成されます。
f. [終了]をクリックします。

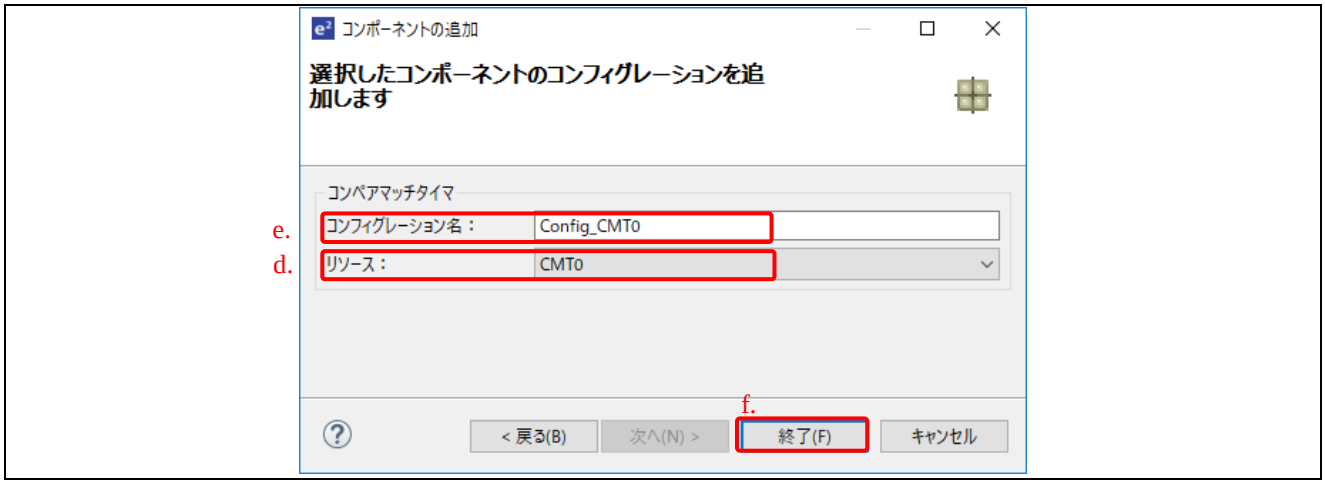


図 2-14　プロジェクトに新しいコンフィグレーションを追加

4) *Config_CMT0* の設定パネルで、

a. クロック設定の[PCLK/512]をクリックし、インターバル時間[5000 カウント]を設定します。

b. “コンペアマッチ割り込みを許可(CMI0)”にチェックが入っていることを確認します。

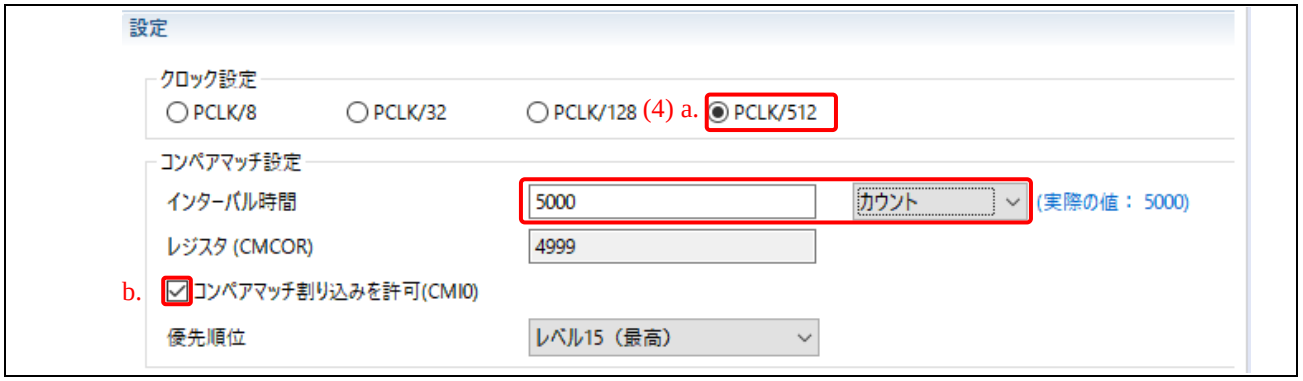


図 2-15　コンペアマッチタイマ設定

5) プロジェクトに“ポート”ドライバを追加します。

a. [コンポーネント]タブで、 をクリックし“ポート”を追加します。
b. “機能”オプションから“入出力ポート (Drivers)”を選択します。
c. “ポート”ドライバを選択します。
d. [次へ]をクリックし続けます。

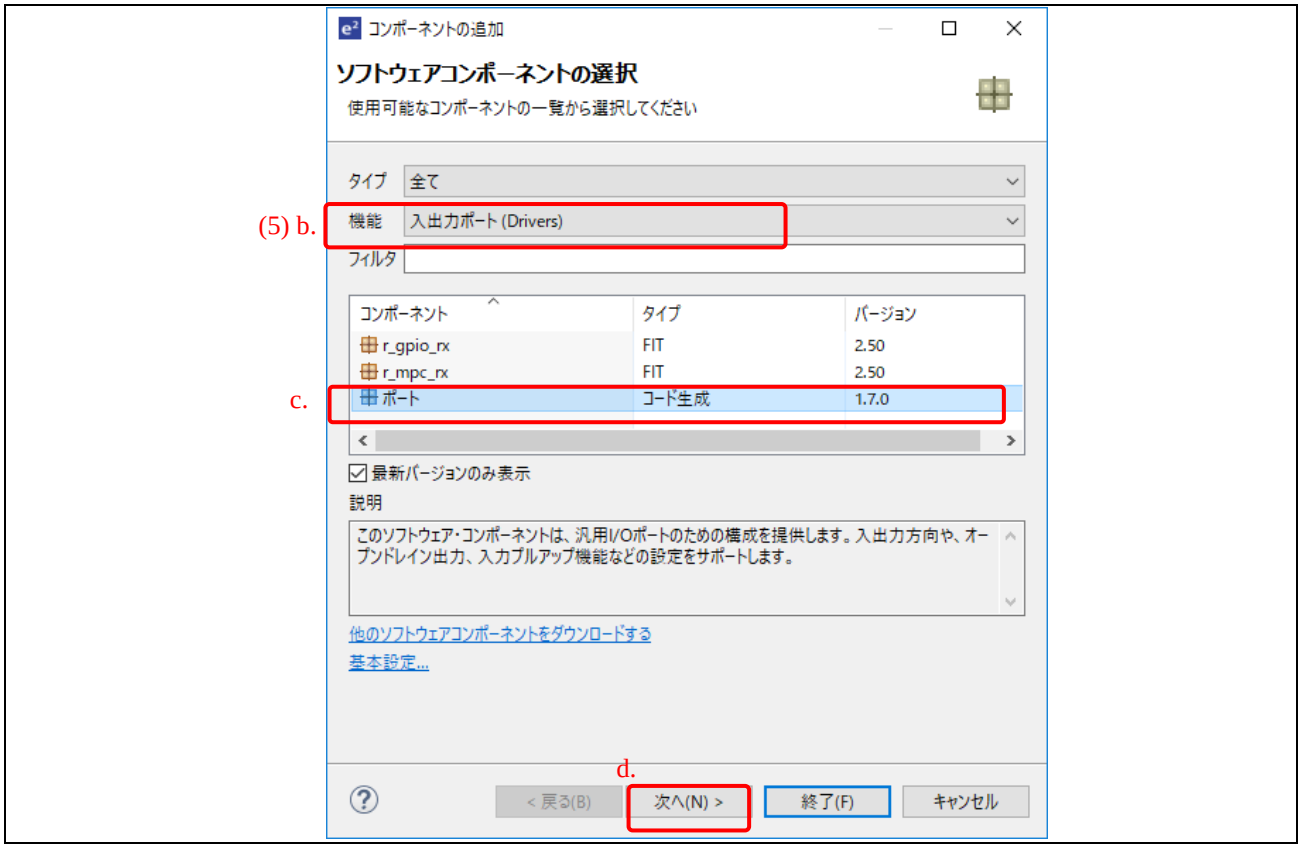


図 2-16　ソフトウェアコンポーネントの選択

e. デフォルトのコンフィグレーション名をそのまま使用し、[終了]をクリックします。

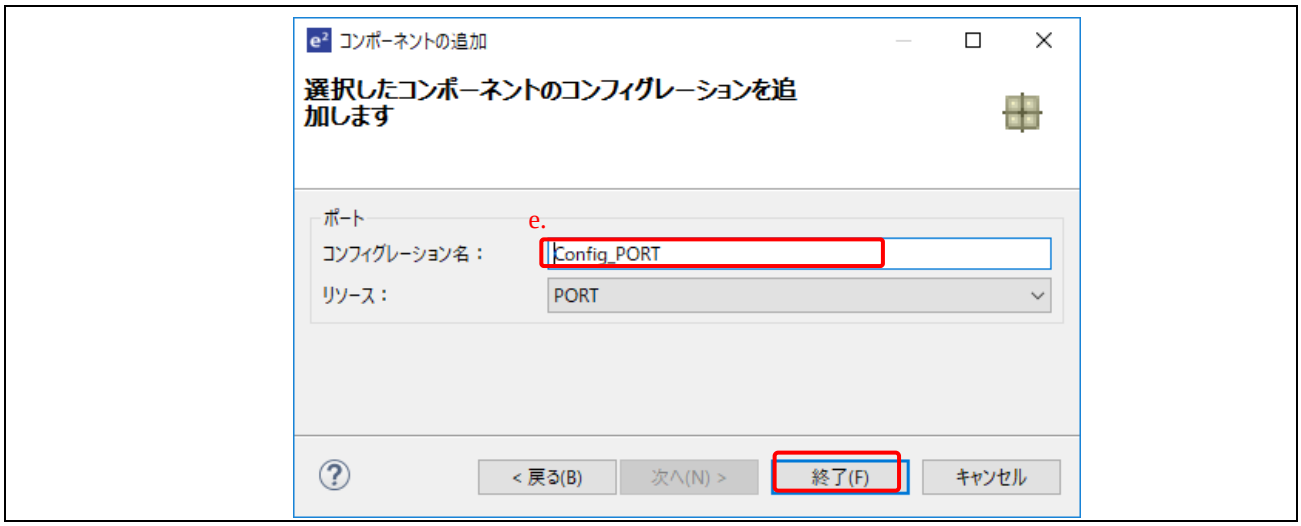


図 2-17　プロジェクトに新しいコンフィグレーションを追加

6) [ポート選択]タブで、PORTG のチェックボックスにチェックします。

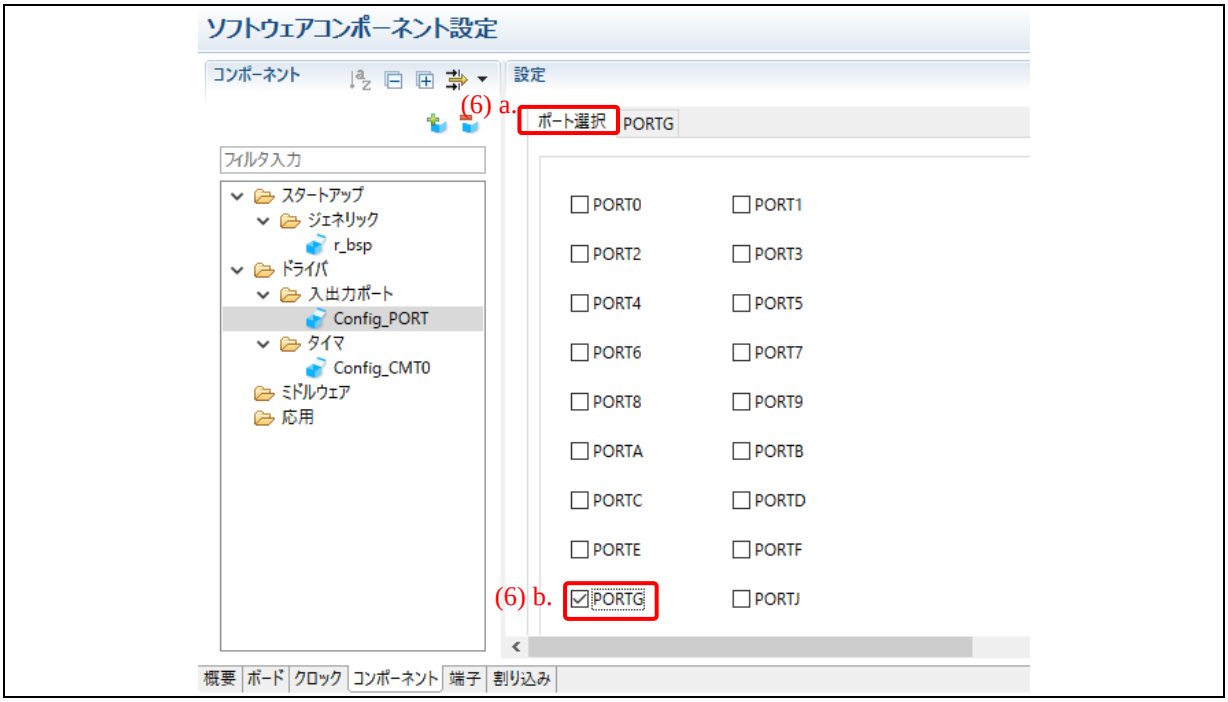


図 2-18　ポート選択タブ

7) [PORTG]タブで、出力端子として PG6 を設定します。

a. PG6 のセクションで、[出力]を選択します。

b. [1 を出力]にチェックを入れ、最初は LED2 がオフの状態になるように設定します。

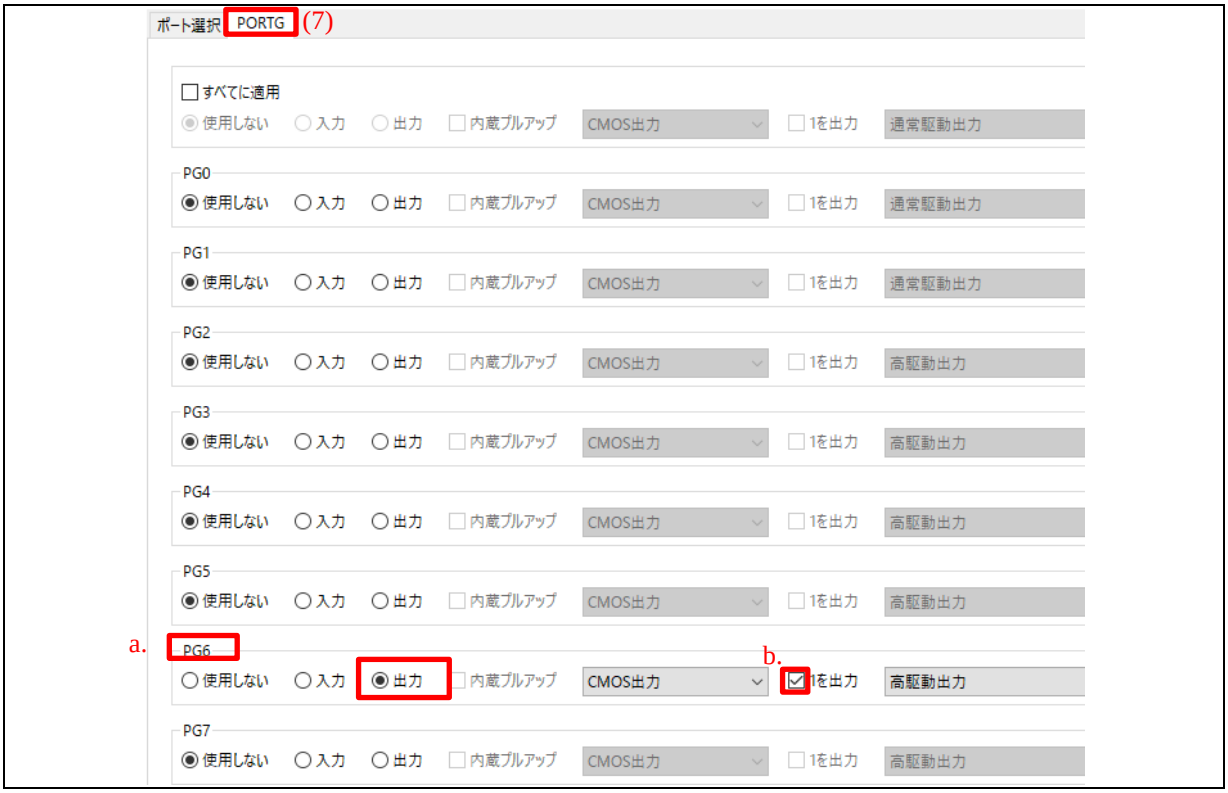


図 2-19　PORTG タブ

8) [端子]タブで  ボタンをクリックし、ツリーをソフトウェアコンポーネントビューに切り替えます。

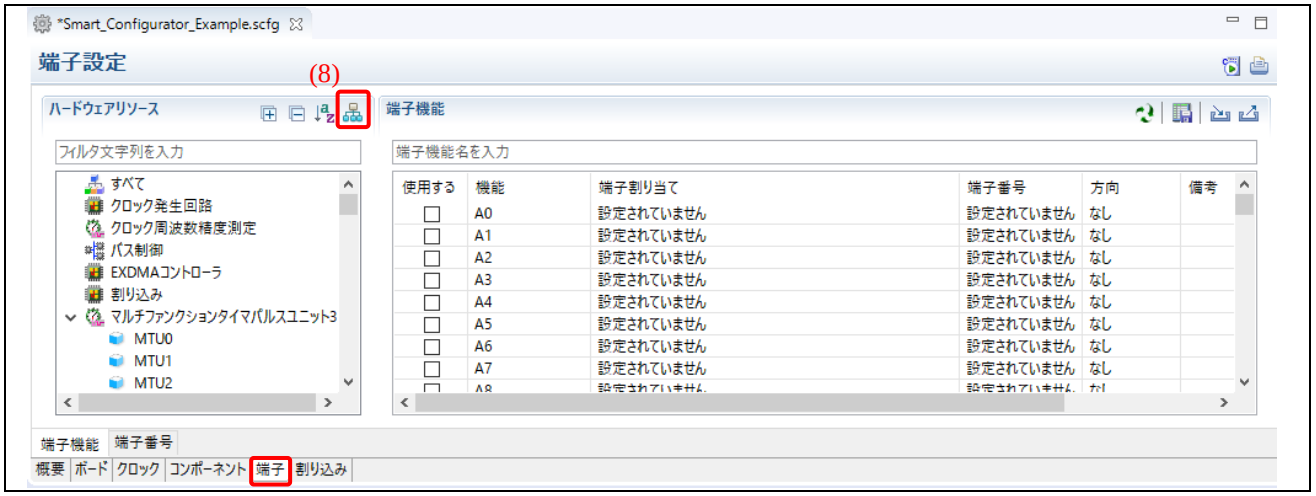


図 2-20　端子タブ

9) ツリーにある *Config_PORT* をクリックし、端子設定を表示します。
10) 端子機能フィルタに、“PG”と入力します。
11) PG6 端子が設定されていることを確認します。

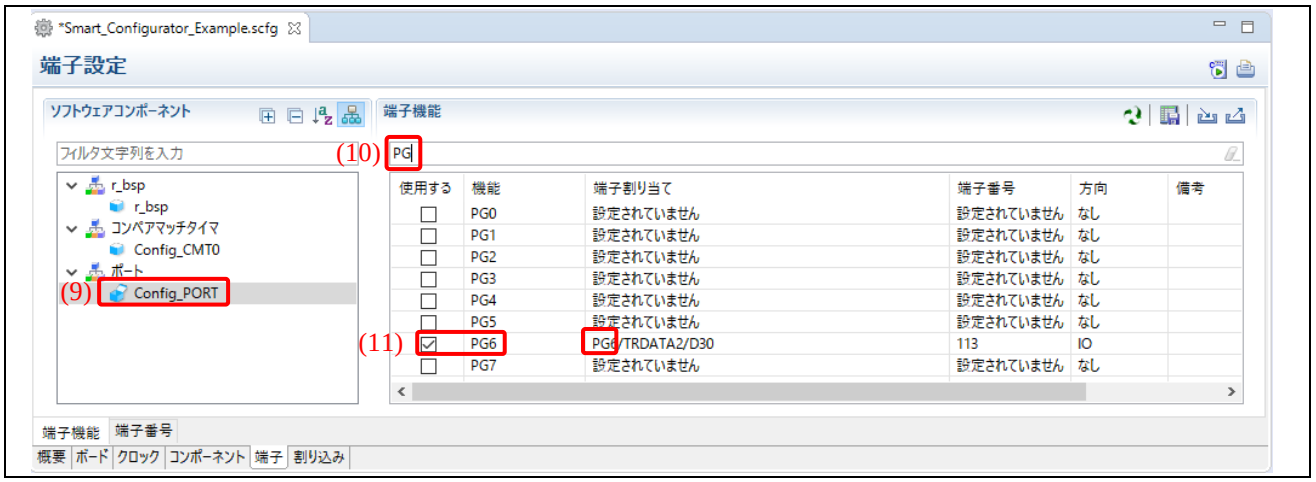


図 2-21　*Config_PORT* の端子設定

2.4　コード生成

1)　をクリックし、コードを生成します。

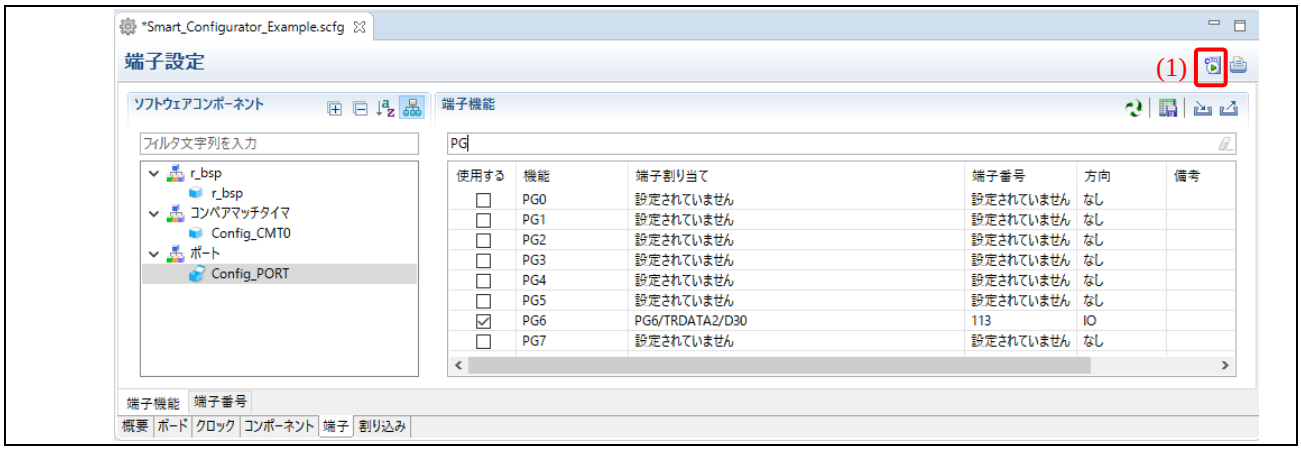


図 2-22　コード生成

2)　“コード生成の終了”というメッセージがコンソールに表示されます。

3)　プロジェクトの\src\smc_gen フォルダにファイルが生成されます。

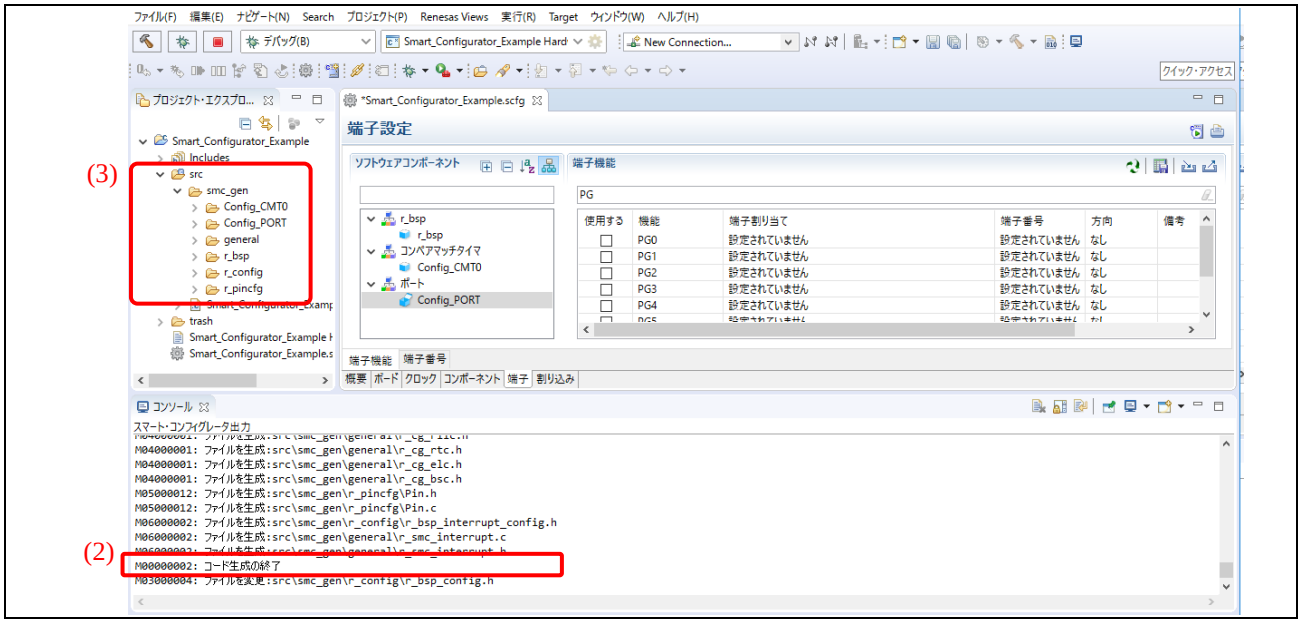


図 2-23　正常に終了したコード生成

2.5　“r_cg_userdefine.h”ファイルに定義コードを追加

1) プロジェクトツリーで、\src\smc_gen\general フォルダの“r_cg_userdefine.h”ファイルをダブルクリックして開きます。

a. 以下の LED2 ポート設定をコメントの始まりと終わりの間の部分に追加します。

```
/* LED ポート設定 */
#define LED2            (PORTG.PODR.BIT.B6)
```

ソースファイルは以下のようになります。

```
/***********************************************************************************
Macro definitions (Register bit)
***********************************************************************************/
/* Start user code for register. Do not edit comment generated here */

/* End user code. Do not edit comment generated here */

/***********************************************************************************
Macro definitions
***********************************************************************************/
/* Start user code for macro define. Do not edit comment generated here */
/* LED port settings */
#define LED2 (PORTG.PODR.BIT.B6)
/* End user code. Do not edit comment generated here */

/***********************************************************************************
Typedef definitions
***********************************************************************************/
/* Start user code for type define. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/***********************************************************************************
Global functions
***********************************************************************************/
/* Start user code for function. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#endif
```

図 2-24　r_cg_userdefine.h

2.6　コンペアマッチタイマドライバにアプリケーションコードを追加

1) プロジェクトの\src\smc_gen\Config_CMT0 フォルダにある “Config_CMT0_user.c”を開きます。

a. *r_Config_CMT0_cmi0_interrupt (void)* 関数で LED2 を切り替えます。

```
/* Toggle LED2 whenever timer is up */
LED2 ^= 1U;
```

ソースファイルは以下のようになります。

```
/***********************************************************************************************
* Function Name: r_Config_CMT0_cmi0_interrupt
* Description  : This function is CMI0 interrupt service routine
* Arguments    : None
* Return Value : None
***********************************************************************************************/
#if FAST_INTERRUPT_VECTOR == VECT_CMT0_CMI0
#pragma interrupt r_Config_CMT0_cmi0_interrupt(vect=VECT(CMT0,CMI0),fint)
#else
#pragma interrupt r_Config_CMT0_cmi0_interrupt(vect=VECT(CMT0,CMI0))
#endif
static void r_Config_CMT0_cmi0_interrupt(void)
{
    /* Start user code for r_Config_CMT0_cmi0_interrupt. Do not edit comment generated here */
    /* Toggle LED2 whenever timer is up */
    LED2 ^= 1U;
    /* End user code. Do not edit comment generated here */
}
```

図 2-25　Config_CMT0_user.c

2)　CMT0 のカウントを開始します。

a.　\src フォルダにある“Smart_Configurator_Example.c”を開き、*main()* 関数に以下のコードを追加します。

```
/* Start CMT0 counter operation */
R_Config_CMT0_Start();

while(1U)
{
        nop();
}
```

ソースファイルは以下のようになります。

```
*  FILE        : Smart_Configurator_Example.c
#include "r_smc_entry.h"

void main(void);

void main(void)
{
    /* Start CMT0 counter operation */
    R_Config_CMT0_Start();

    while(1U)
    {
        nop();
    }

}
```

図 2-26　main ファイルで CMT0 のカウントを開始

2.7　ビルドとハードウェアボードでの実行

1)　[プロジェクト]から[プロジェクトのビルド]を選択し、プロジェクトをビルドします。

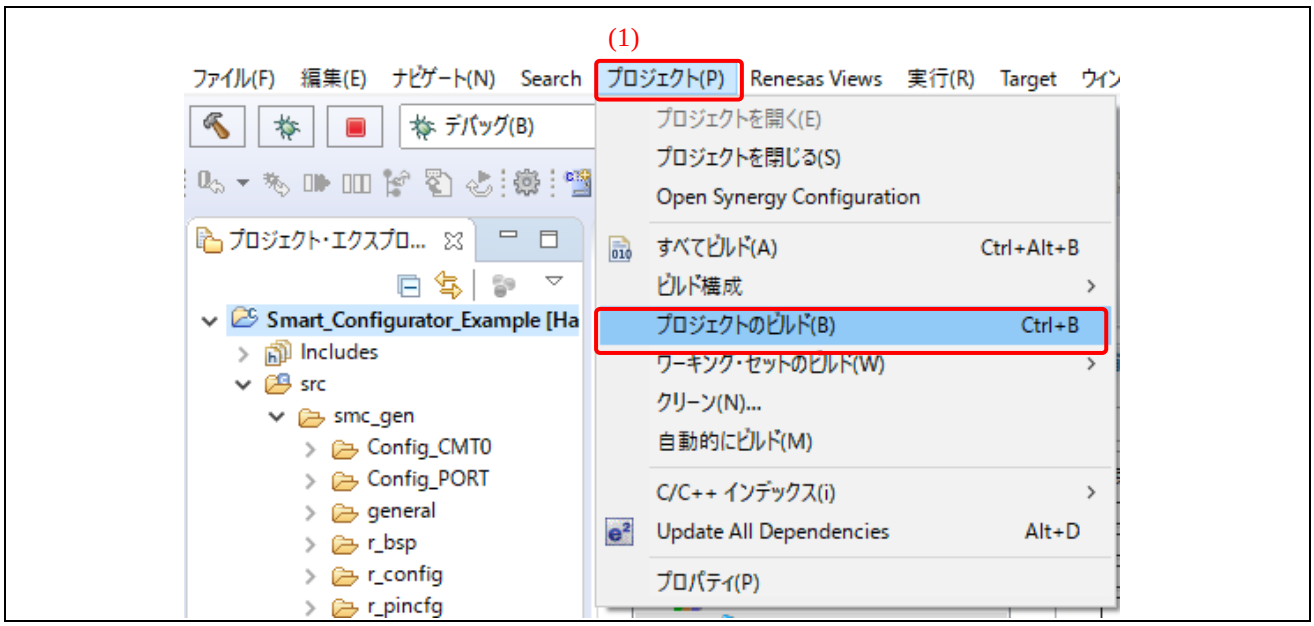


図 2-27　プロジェクトのビルド

2)　コードをデバッグします。

a.　RSK+RX65N 2MB ボードを E1/E2 Lite エミュレータに接続し、E1/E2 Lite エミュレータを PC に接続します。

b.　[実行]から[デバッグの構成...]またはアイコンの横にある下向きの矢印から[デバッグの構成...]を選択し、“デバッグ構成”ウインドウを開きます。

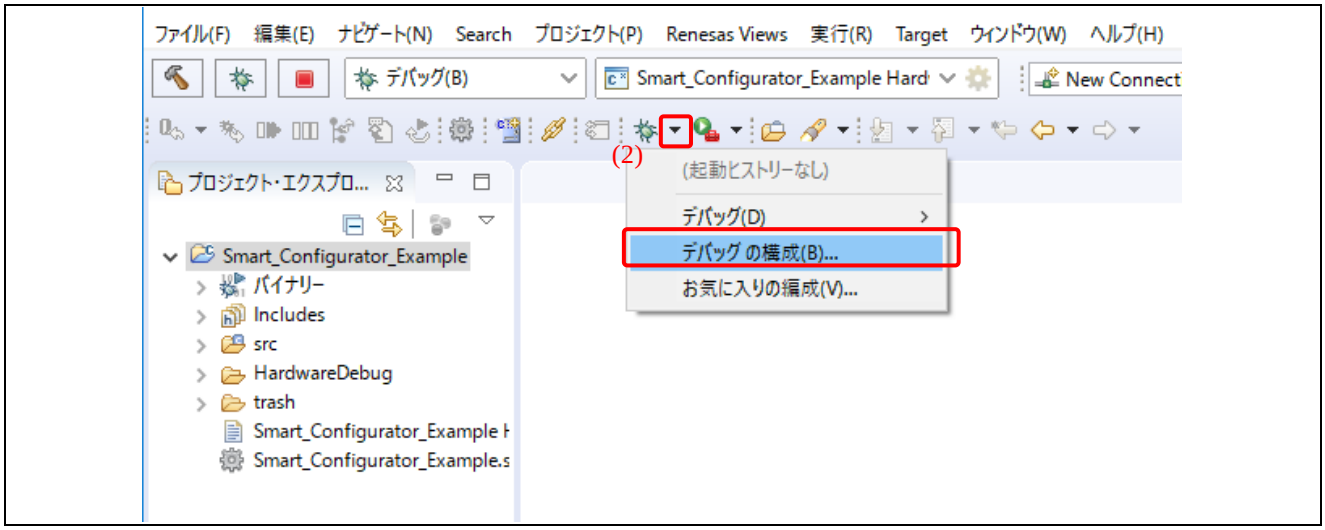


図 2-28　デバッグ構成ウインドウを開く

3) [Renesas GDB Hardware Debugging]を拡張し、[Smart_Configurator_Example_HardwareDebug]をクリックします。[デバッガ]タブ上で[Connection Settings]タブをクリックし、以下のように設定してください。

a. Debug hardware：E1 (RX)または E2 Lite (RX)を選択

b. Target Device： R5F565NEDxFC

c. メイン・クロック・ソース：EXTAL

d. EXTAL 周波数[MHz]：24MHz

e. 内部フラッシュメモリ書き換え時にクロックソースの変更を許可する：はい

f. エミュレータから電源を供給する(MAX 200mA)：はい

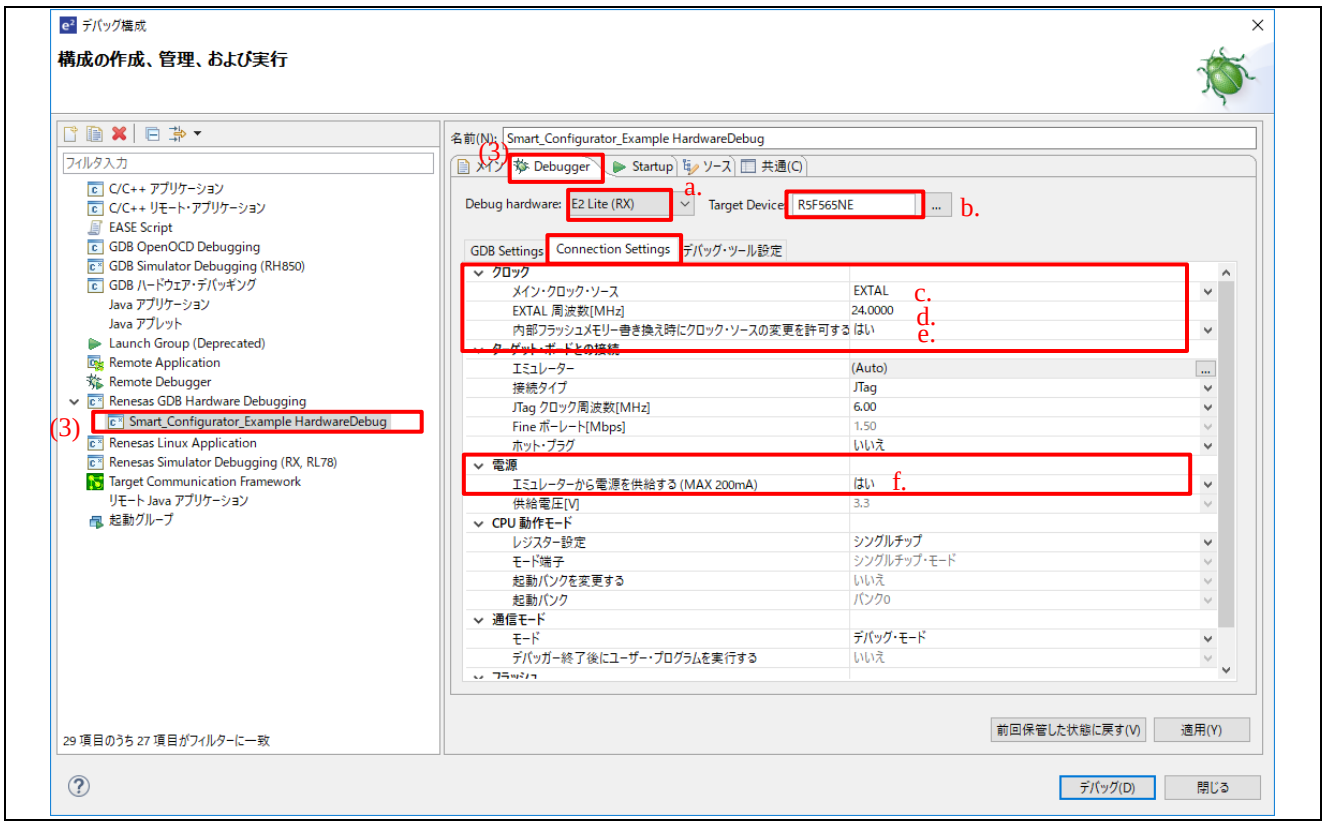


図 2-29　E2 (Lite) エミュレータ接続時のデバッグコンフィグレーション例

4) “Startup”タブで、“ブレークポイント設定先： main”のチェックを外します。

5) [デバッグ]をクリックし、デバッグを開始します。

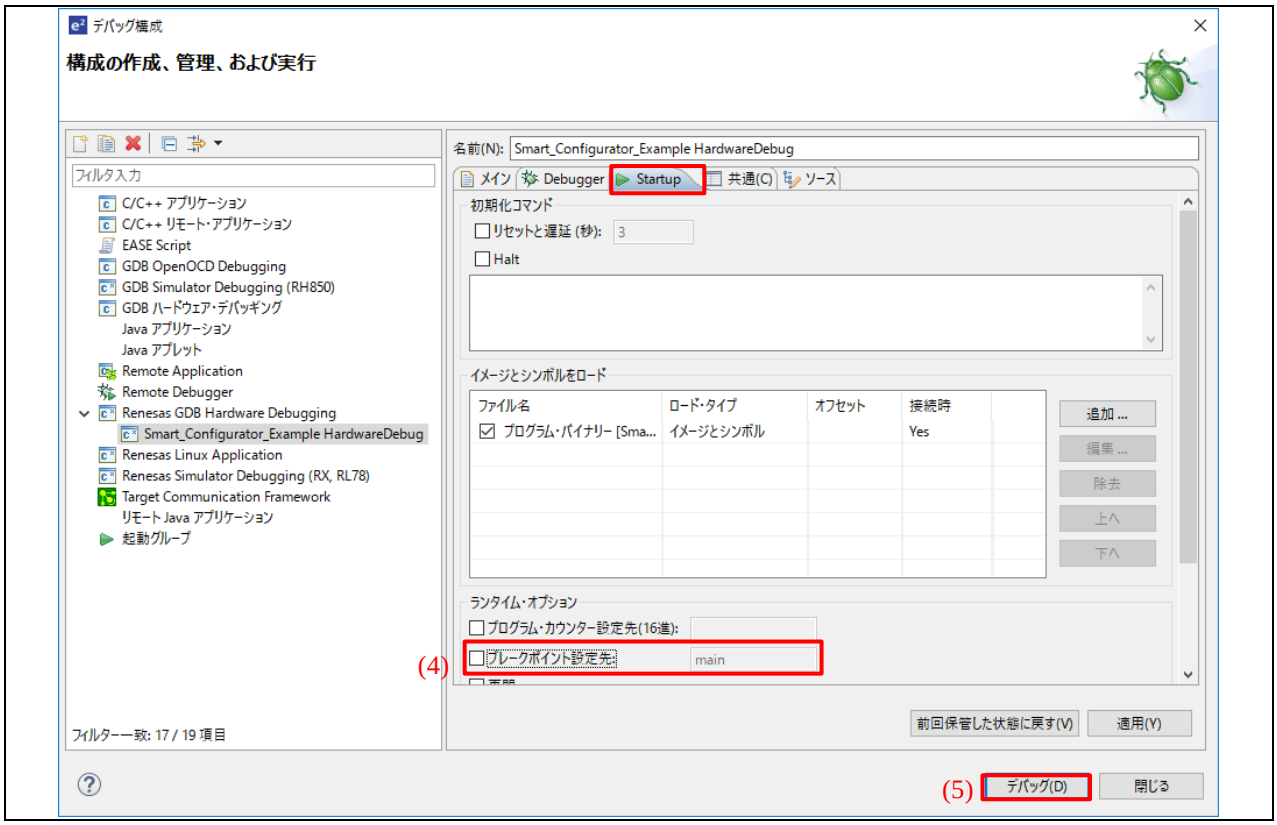


図 2-30　main のブレークポイント解除

6) ‘パースペクティブ切り替えの確認’ダイアログが表示されたら、[はい]をクリックし続けます。

7) をクリックし、プログラムを実行します。

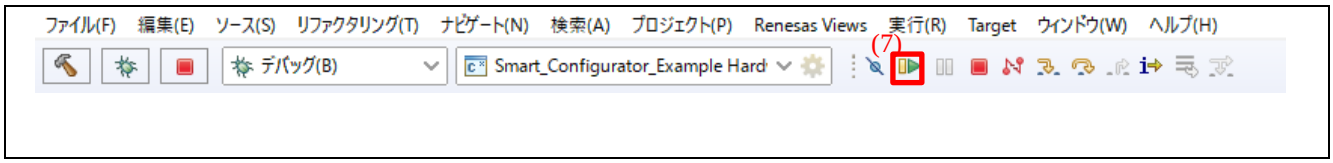


図 2-31　実行開始

8) LED2 が点滅していることを確認します。LED2 の位置については、2.9 章“ボードでの動作確認”を参照してください。

9)　中断ボタン をクリックし、実行を一時停止します。

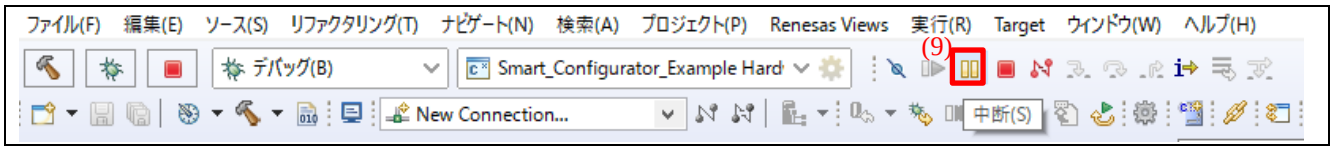


図 2-32　実行の中断

10) 実行を停止するには、切断ボタン をクリックし、デバッグセッションを終了します。

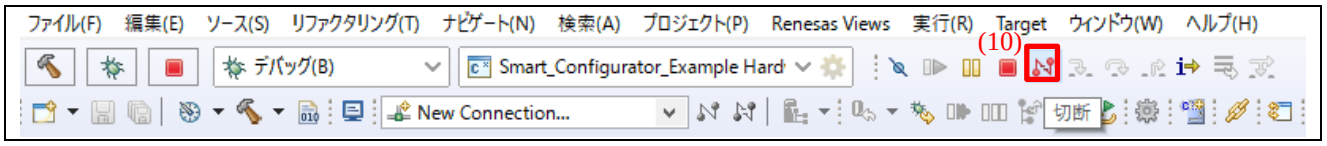


図 2-33　デバッグの停止

2.8　LED2 の点滅インターバルの変更

1) ツールバーの右手側にあるをクリックし、スマート・コンフィグレータ・パースペクティブに切り替えます。



図 2-34　スマート・コンフィグレータ・パースペクティブの変更

2) Smart_Configurator_Example.scfg 画面で、[コンポーネント]タブをクリックし、ツリーから *Config_CMT0* を選択します。

3) インターバル時間を[50000 カウント]に変え、LED2 の点滅スピードを下げます。

4) ツールバーのをクリックし変更を保存します。その後、をクリックしコードを再生成します。

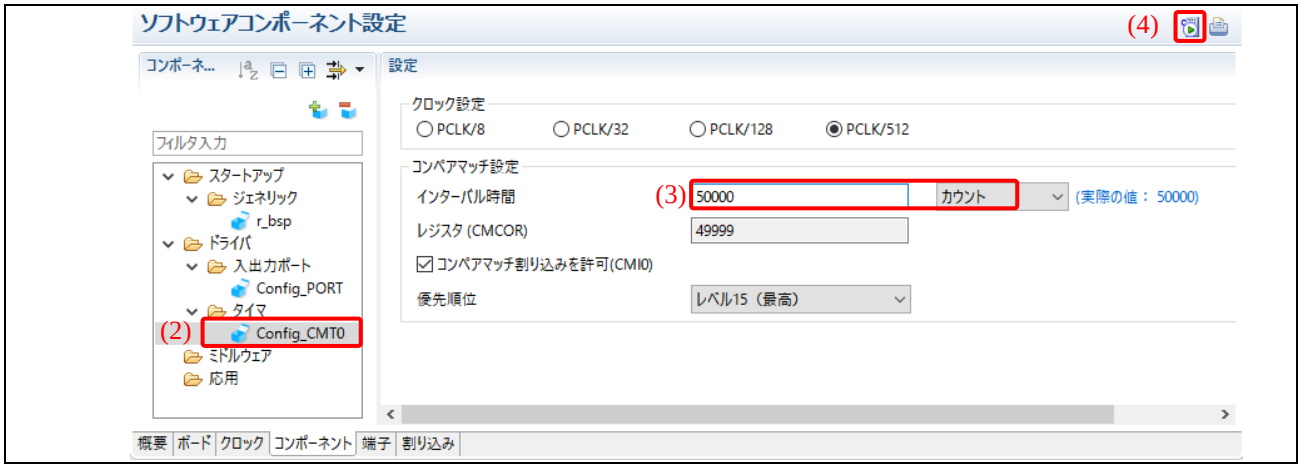


図 2-35　コンペアマッチタイマ設定

5) [プロジェクト]から[プロジェクトのビルド]をクリックし、プロジェクトをビルドします。

6) [はい]をクリックし、更新されたプロジェクトをリロードします。

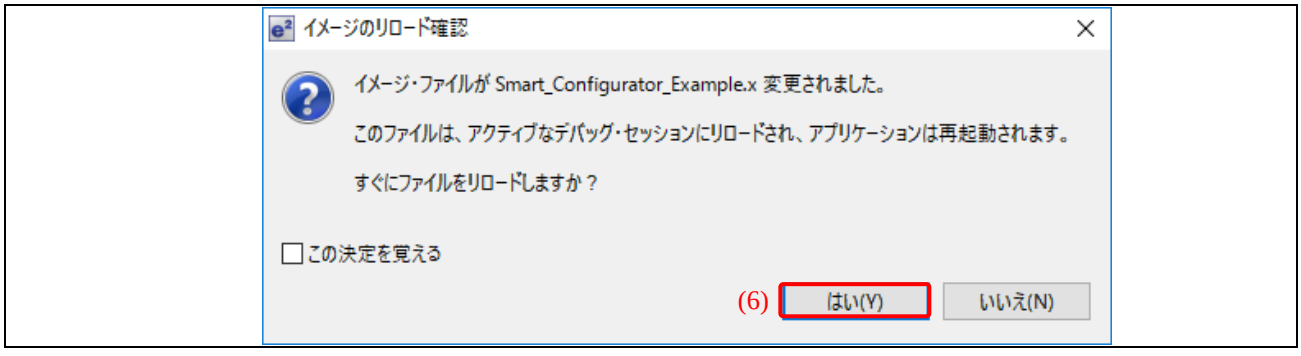


図 2-36　イメージのリロード確認

7) ‘パースペクティブ切り替えの確認’ダイアログが表示されたら、[はい]をクリックして続けます。

8)　アイコンをクリックして、実行を開始します。

LED2 は低速で点滅します。LED2 の位置については、2.9 章“ボードでの動作確認”を参照してください。

2.9　ボードでの動作確認

RSK+ RX65N 2MB ボードで、LED2 が点滅します。

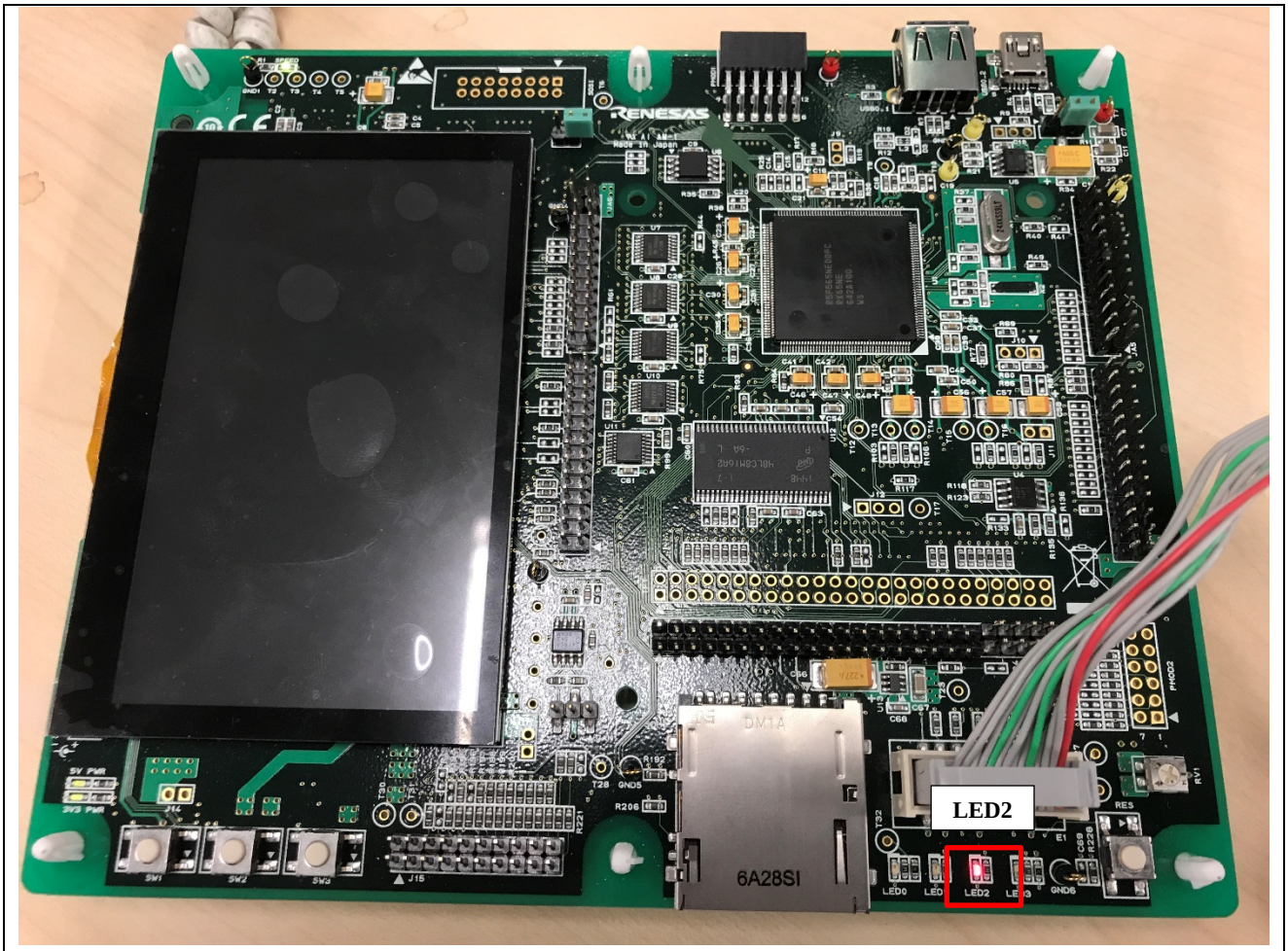


図 2-37　RSK+RX65N 2MB ボード上の LED

3. Application Example 2（スマート・コンフィグレータでの機能追加）

本章では、LED2 の点滅周期を調整するポテンショメータの使用方法を説明します。

下記の表は、ここで設定する CG ドライバを示します。

ドライバ	リソース	機能
シングルスキャンモード S12AD	S12AD	LED 点滅周期を調整するためのポテンショメータ値を読み込みます。

S12AD アプリケーションコードは、以下のようにプログラムに追加されます。

a) main 関数：

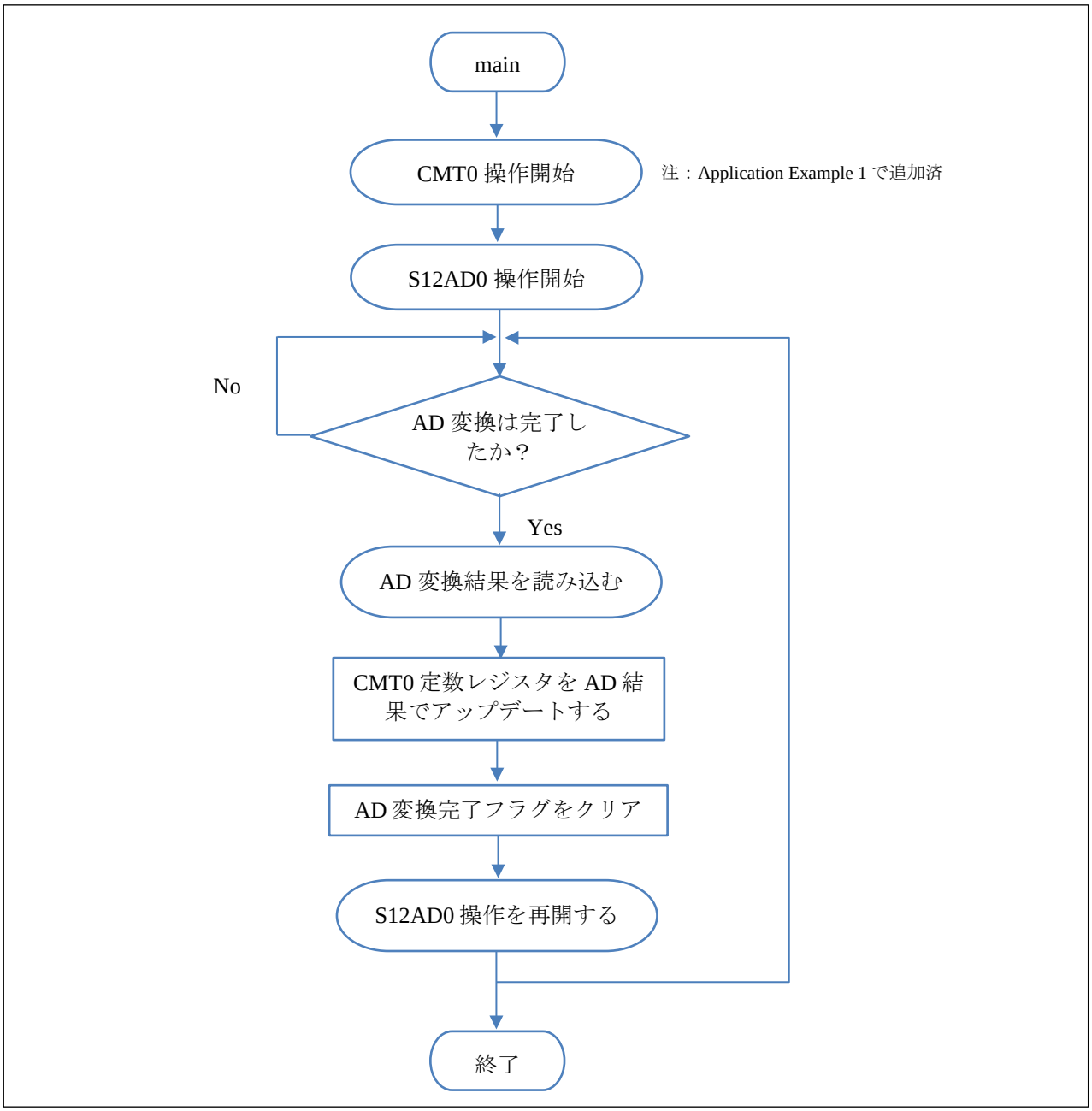


図 3-1　main フローチャート

b)　S12AD0 ユーザ関数：

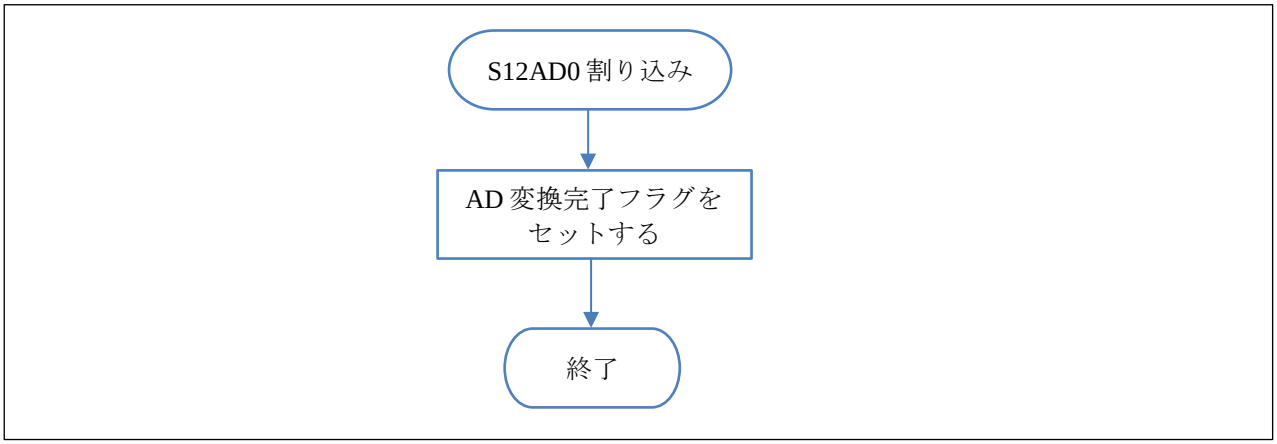


図 3-2　S12AD0 割り込みフローチャート

3.1　ペリフェラルドライバの追加

1) Smart_Configurator_Example.scfg 画面で、[コンポーネント]タブをクリックします。
2) をクリックして、新しいコンポーネントを追加します。

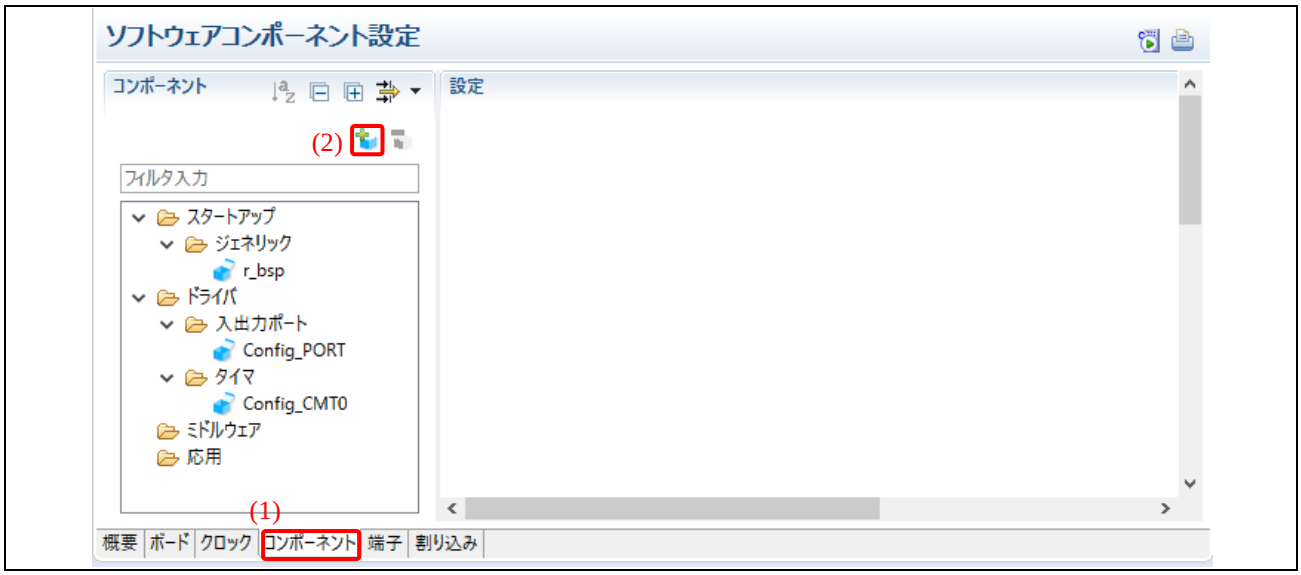


図 3-3　スマート・コンフィグレータでのソフトウェアコンポーネント設定

3) プロジェクトへ “シングルスキャンモード S12AD” ドライバを追加します。
   a. “機能” オプションから、“A/D コンバータ (Drivers)” を選択します。
   b. コンポーネントリストを表示させ、“シングルスキャンモード S12AD” を選択します。
   c. [次へ]をクリックして続けます。

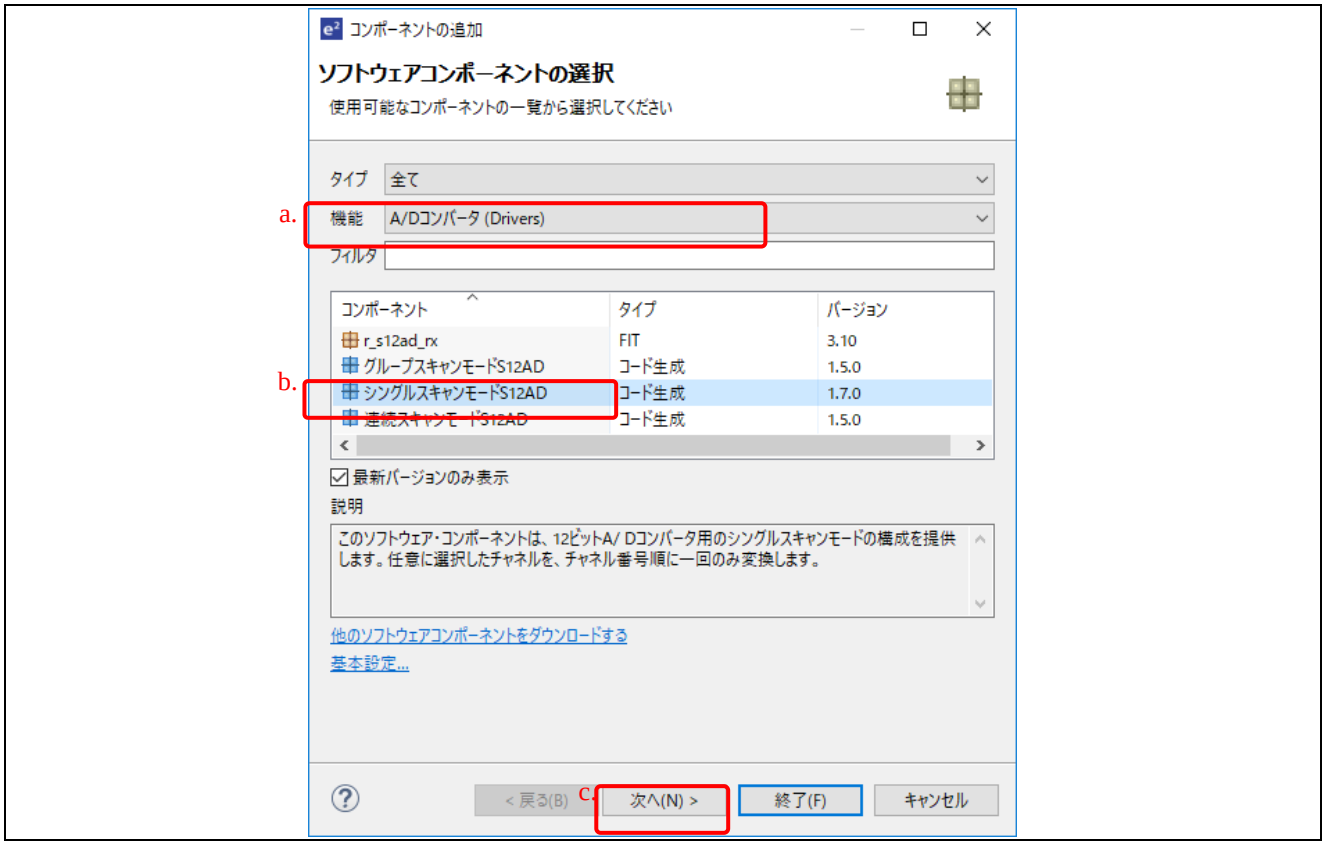


図 3-4　ソフトウェアコンポーネントの選択

d. リソースとして S12AD0 を選択します。
e. デフォルトのコンフィグレーション名を、そのまま使用します。このコンフィグレーション名から、ソースファイルと API が生成されます。
f. [終了]をクリックします。

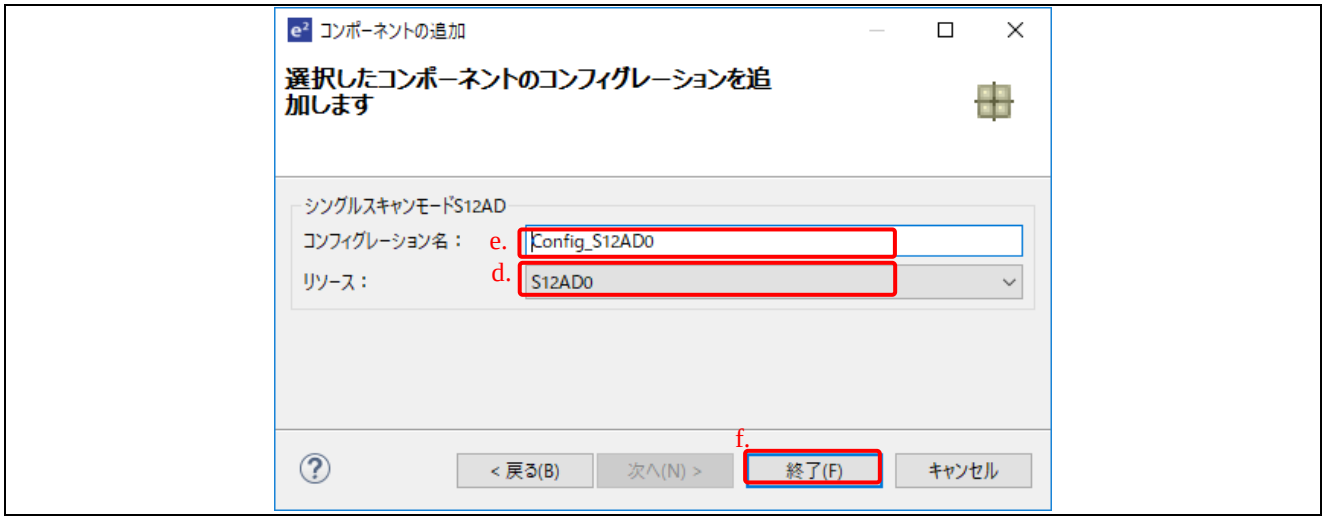


図 3-5　プロジェクトに新しいコンフィグレーションを追加

4) 設定画面のアナログ入力チャネル設定で[AN000]を選択します。

5) 開始トリガソースで[ソフトウェアトリガ]が選択され、[AD 変換終了割り込みを許可(S12ADI)]のチェックボックスにチェックが入っていることを確認します。

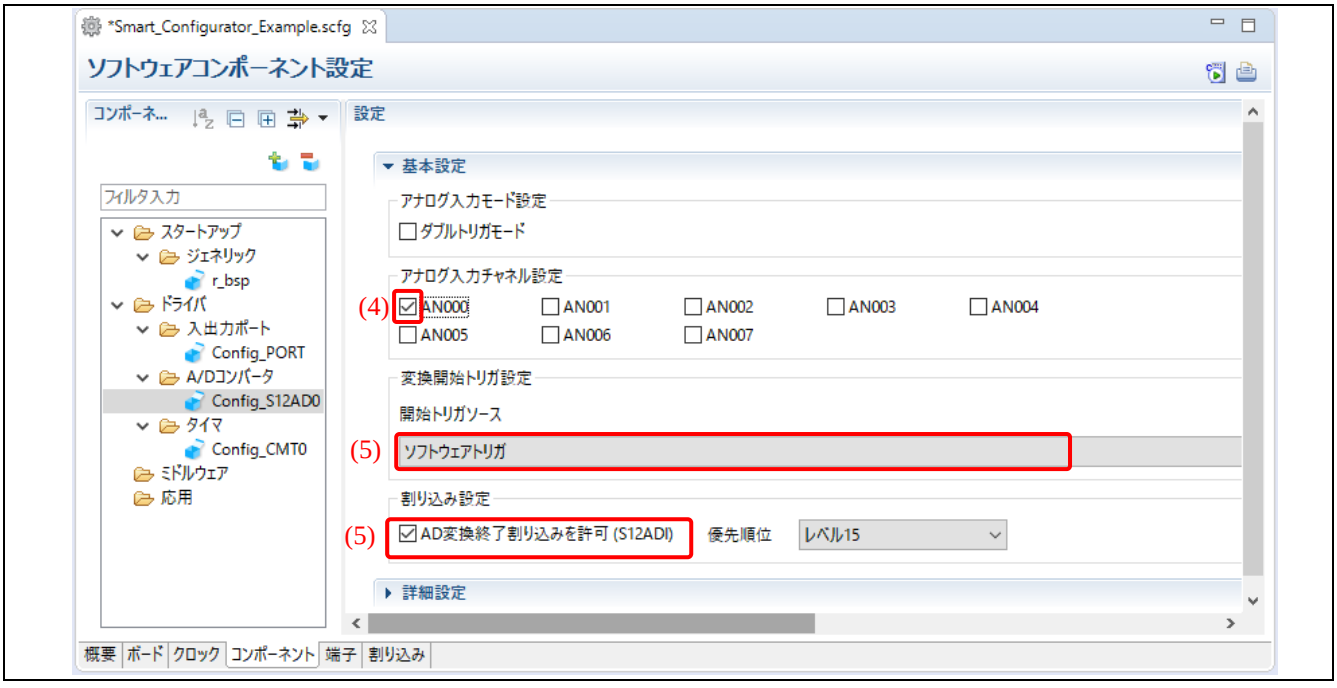


図 3-6　シングルスキャンモード S12AD 設定

6) [端子]タブで、ツリーから Config_S12AD0 を選択します。
7) 端子機能フィルタに何か入力されている場合は削除します。
8) AN000 端子が P40 に割り当てられ、アナログパワー端子（AVCC0、AVSS0、VREFH0、VREFL0）が使用可能になっていることを確認します。

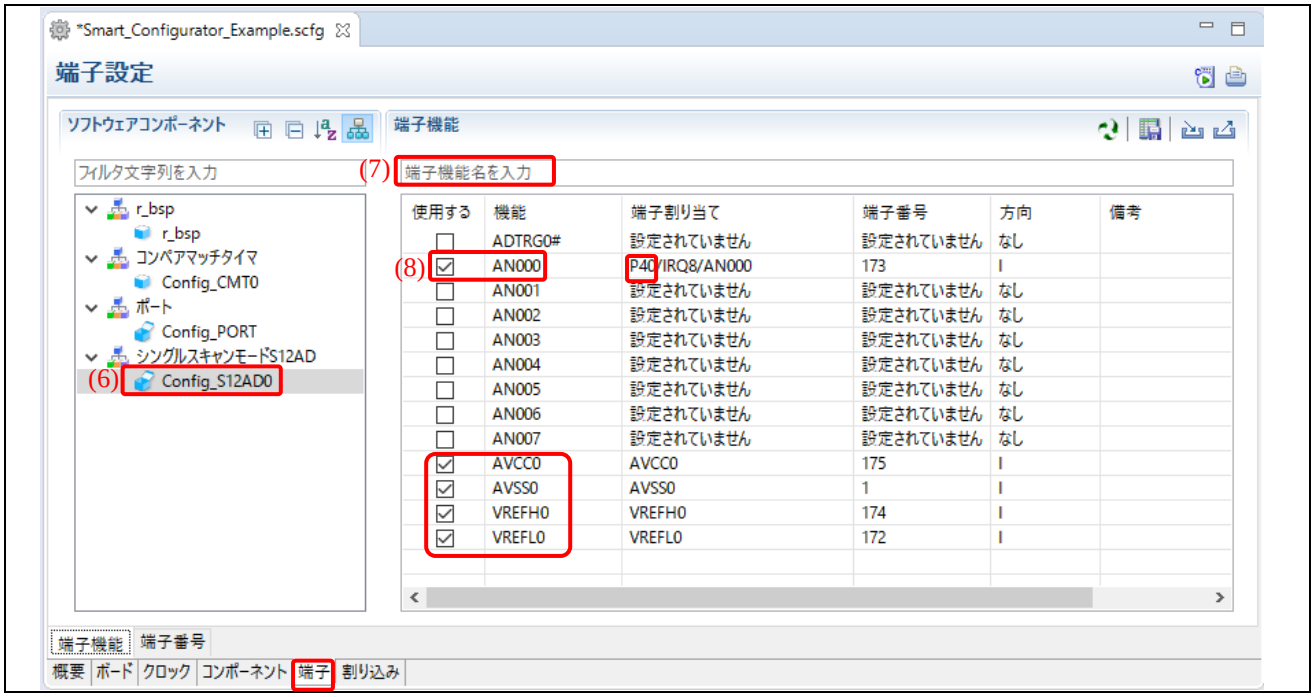


図 3-7　*Config_S12AD0* の端子設定

3.2　コード生成

1)  をクリックし、コードを生成します。

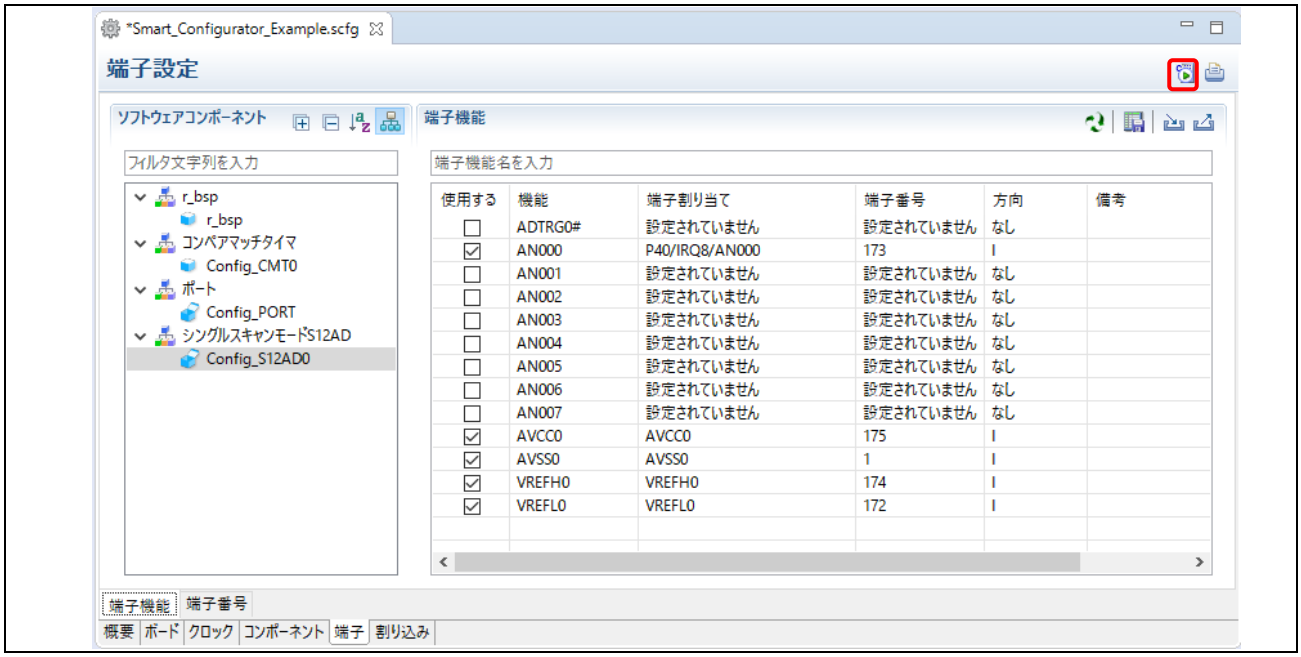


図 3-8　コード生成

3.3　シングルスキャンモード S12AD ドライバにアプリケーションコードを追加

1) プロジェクトの\src\smc_gen Config_S12AD0 フォルダにある“Config_S12AD0_user.c”を開きます。

a. ファイルの先頭近くにあるユーザコードエリアに、グローバルフラグと変数の宣言を追加します。

```
/* Global flag to indicate A/D conversion operation is completed */
uint8_t g_adc_flag;
```

b. *r_Config_S12AD0_interrupt (void)* 関数で AD 変換完了フラグをセットします。

```
/* Set A/D conversion completion flag */
g_adc_flag = 1U;
```

ソースファイルは以下のようになります。

```
* File Name    : Config_S12AD0_user.c

Pragma directive
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

Includes
#include "r_cg_macrodriver.h"
#include "r_cg_userdefine.h"
#include "Config_S12AD0.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

Global variables and functions
/* Start user code for global. Do not edit comment generated here */
/* Global flag to indicate A/D conversion operation is completed */
uint8_t g_adc_flag;
/* End user code. Do not edit comment generated here */

* Function Name: R_Config_S12AD0_Create_UserInit

void R_Config_S12AD0_Create_UserInit(void)
{
    /* Start user code for user init. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
}

* Function Name: r_Config_S12AD0_interrupt

#if FAST_INTERRUPT_VECTOR == VECT_PERIB_INTB186
#pragma interrupt r_Config_S12AD0_interrupt(vect=VECT(PERIB,INTB186),fint)
#else
#pragma interrupt r_Config_S12AD0_interrupt(vect=VECT(PERIB,INTB186))
#endif
static void r_Config_S12AD0_interrupt(void)
{
    /* Start user code for r_Config_S12AD0_interrupt. Do not edit comment generated here */
    /* Set A/D conversion completion flag */
    g_adc_flag = 1U;
    /* End user code. Do not edit comment generated here */
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

図 3-8　Config_S12AD0_user.c

3.4 *main()*にアプリケーションコードを追加

1) CMT0 と S12AD0 の操作を開始します。

a. プロジェクトツリーで、\src フォルダにある“Smart_Configurator_Example.c”を開きます。

b. ファイルの先頭近くにあるユーザコードエリアで、**#include** "r_smc_entry.h"の後に、グローバル変数の宣言を追加します。

```
/* Global variable for changing CMT0 interval */
uint16_t interval_level;

/* Global variable for storing the A/D conversion result */
uint16_t g_adc_result;

/* Global flag to indicate A/D conversion operation is completed */
extern uint8_t g_adc_flag;
```

c. *main()*関数の *while* ループの前に以下のコードを追加し、AD 変換を開始します。

```
/* Start performing A/D conversion */
R_Config_S12AD0_Start();
```

d. AD 変換が完了したら、AD 変換結果を読み込み、CMCOR 定数レジスタを更新します。

while ループの中の *nop()*命令を、以下のコードに置き換えます。

```
/* A/D conversion is completed */
if (g_adc_flag == 1U)
{
        /* Read the A/D conversion result and store in variable */
        R_Config_S12AD0_Get_ValueResult(ADCHANNEL0, &g_adc_result);

        /* Get new blink interval level from A/D conversion result */
        interval_level = g_adc_result / 500;

        /* Limit minimum level to 1 */
        if (interval_level < 1)
        {
              interval_level = 1;
        }
        else
        {
              /* Do nothing*/
        }

        /* Change blinking rate */
        CMT0.CMCOR = (uint16_t)(5000 * interval_level);

        /* Reset A/D conversion flag */
        g_adc_flag = 0U;

        /* Restart A/D conversion operation */
        R_Config_S12AD0_Start();
}
else
{
        /* Do nothing*/
}
```

ソースファイルは以下のようになります。

```
#include "r_smc_entry.h"

/* Global variable for changing CMT0 interval */
uint16_t interval_level;

/* Global variable for storing the A/D conversion result */
uint16_t g_adc_result;

/* Global flag to indicate A/D conversion operation is completed */
extern uint8_t g_adc_flag;

void main(void);


void main(void)
{
    /* Start CMT0 counter operation */
    R_Config_CMT0_Start();

    /* Start performing A/D conversion */
    R_Config_S12AD0_Start();

    while(1U)
    {
        /* A/D conversion is completed */
        if (g_adc_flag == 1U)
        {
            /* Read the A/D conversion result and store in variable */
            R_Config_S12AD0_Get_ValueResult(ADCHANNEL0, &g_adc_result);

            /* Get new blink interval level from A/D conversion result */
            interval_level = g_adc_result / 500;

            /* Limit minimum level to 1 */
            if (interval_level < 1)
            {
                interval_level = 1;
            }
            else
            {
                /* Do nothing*/
            }

            /* Change blinking rate */
            CMT0.CMCOR = (uint16_t)(5000 * interval_level);

            /* Reset A/D conversion flag */
            g_adc_flag = 0U;

            /* Restart A/D conversion operation */
            R_Config_S12AD0_Start();
        }
        else
        {
            /* Do nothing*/
        }
    }
}
```

図 3-9　*main* コードでの CMT0 と S12AD0 の動作開始

3.5　ビルドとハードウェアボードでの実行

このプロジェクトのビルド、デバッグについては、2.7 章を参照してください。

3.6　ボードでの動作確認

RSK+RX65N 2MB ボードでは、ポテンショメータを回転させると LED2 点滅周期が変わります。

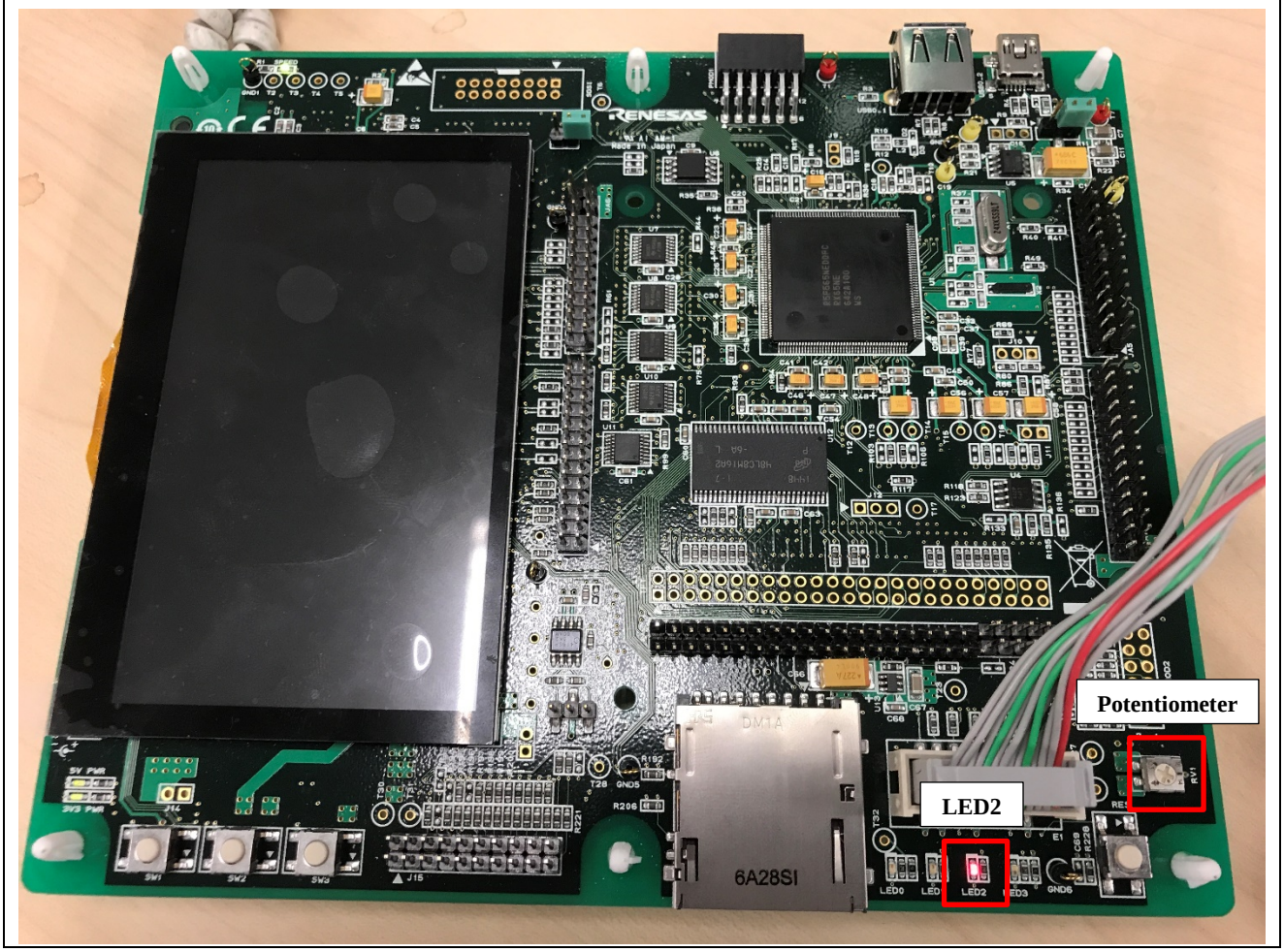


図 3-10　RSK+RX65N 2MB ボード上の LED とポテンショメータ

4. Application Example 3（スマート・コンフィグレータでの機能変更）

この前の章では、LED2 点滅周期を制御するためにポテンショメータを使用しました。

本章では、PC のターミナルプログラムを使い、シリアル通信で LED2 点滅周期を制御する方法を説明します。ここでは、RX65N のシリアル通信インタフェースモジュール(SCI8)は、RSK+ボード上の USB⇔シリアルコンバータ（RL78/G1C に実装されている）を介して PC に接続されています。

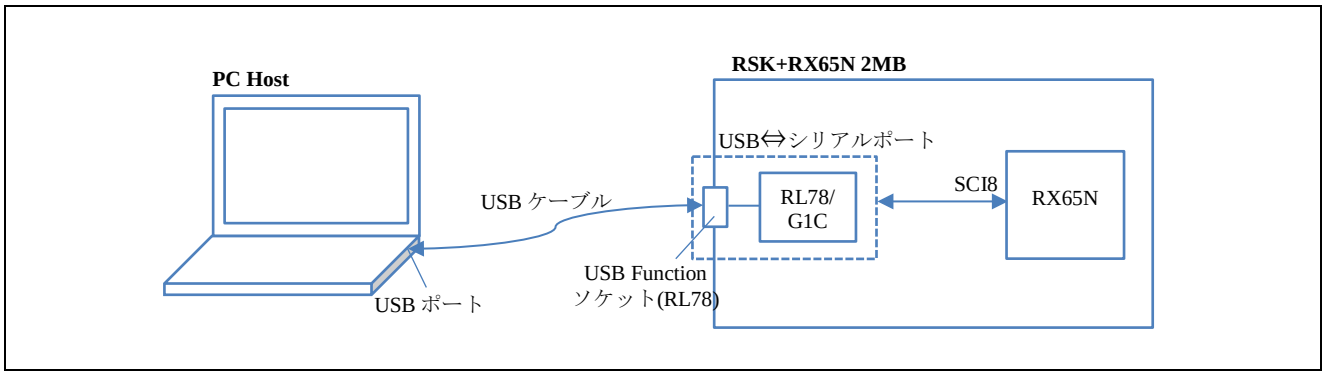


図 4-1　シリアル通信を経由したターミナルプログラムによる LED 点滅周期制御

RX65N 2MB 用 Renesas Starter Kit+の回路図を以下に示します。ここでは、SCI8 が RL78/G1C と通信するよう設定します。

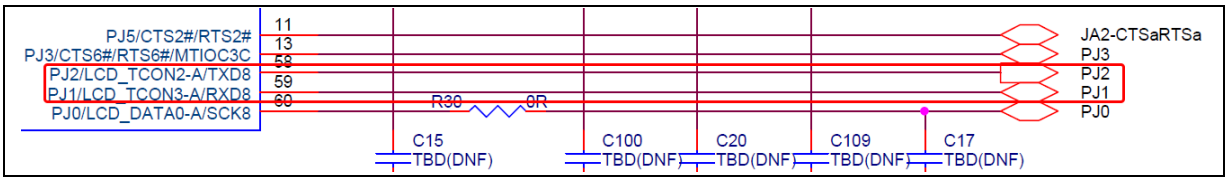


図 4-2　RX65N 2MB 用 Renesas Starter Kit+の回路図

ドライバの選択の変更を以下の表に示します。

	ドライバ	リソース	機能
1. ドライバの削除	シングルスキャンモード S12AD	S12AD	LED 点滅周期を制御するためのポテンショメータ値を読み込みます。
2. 新ドライバの追加	SCI/SCIF 調歩同期式モード	SCI8	オンボードの USB シリアルコンバータ（RL78/G1C）を経由した、RX65N と PC ターミナル間の通信。 PC ターミナルから受信したデータは、LED 点滅周期の制御に使用されます。

SCI/SCIF 調歩同期式モードアプリケーションコードは、以下のようなプログラムを追加します。

a)　main 関数：

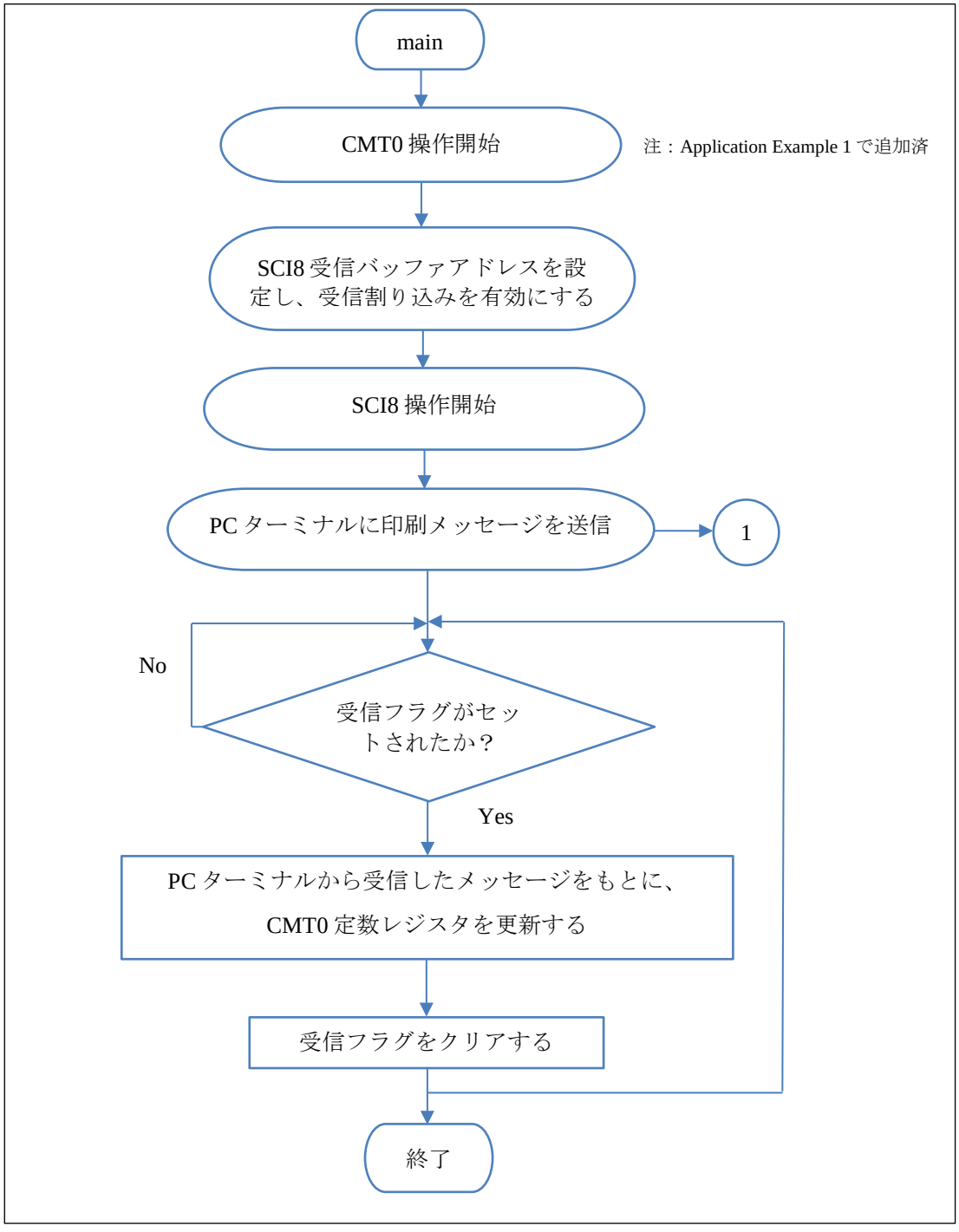


図 4-3　main フローチャート

b)　SCI8 ユーザ関数：

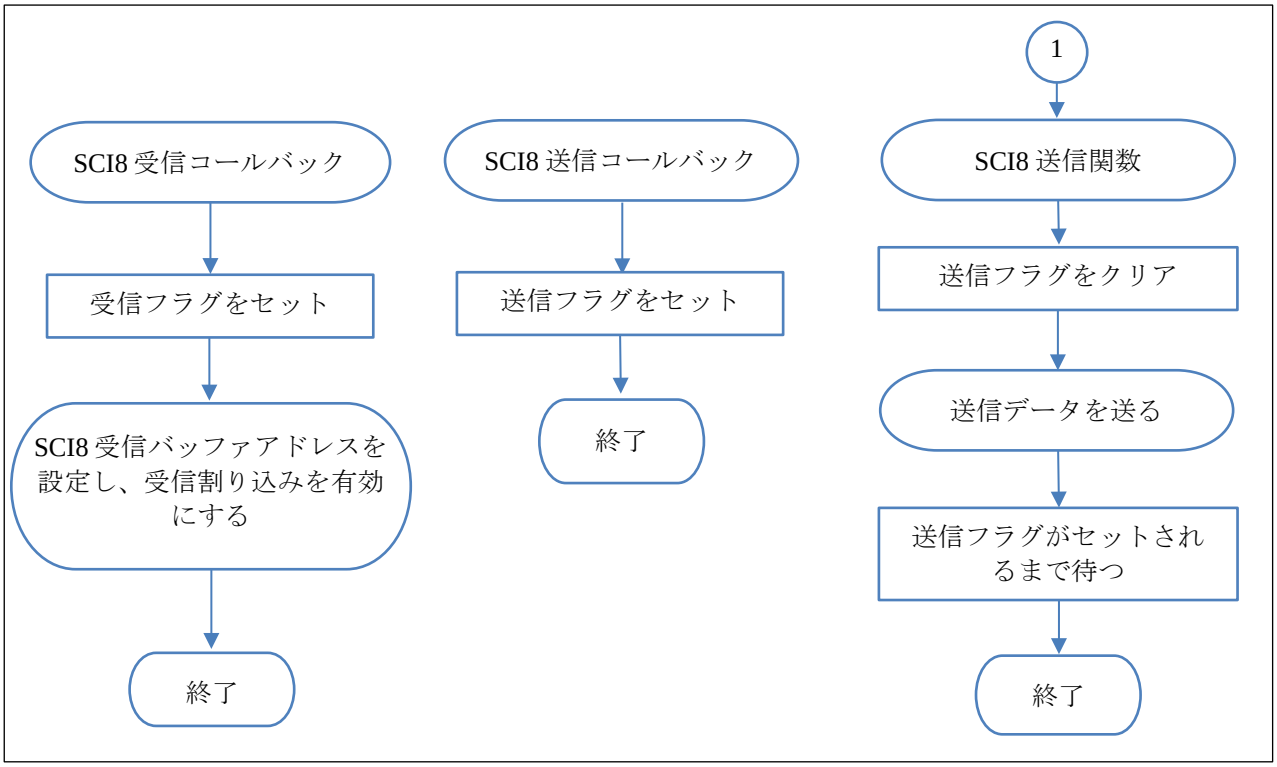


図 4-4　SCI ユーザ関数フローチャート

4.1　S12AD ドライバと関連するアプリケーションコードの削除

1) Smart_Configurator_Example.scfg 画面で、[コンポーネント]タブをクリックします。
2) コンポーネントツリーで、*Config_S12AD0* コンフィグレーションをクリックして選択します。
3)  をクリックして、このコンポーネントを削除します。



図 4-5　スマート・コンフィグレータでのソフトウェアコンポーネント設定

4)  をクリックし、コードを生成します。
コード生成操作時に、ソースフォルダ *Config_S12AD0* はプロジェクト\trash フォルダに移動します。
5) プロジェクトツリーで、\src フォルダにある“Smart_Configurator_Example.c”を開きます。
6) S12AD0 に関連するアプリケーションコードを削除します。
   a. グローバル変数の宣言を削除します。

```
/* Global variable for changing CMT0 interval */
uint16_t interval_level;

/* Global variable for storing the A/D conversion result */
uint16_t g_adc_result;

/* Global flag to indicate A/D conversion operation is completed */
extern uint8_t g_adc_flag;
```

   b. *R_Config_CMT0_Start* 関数の後のすべてのコードを削除します。

S12AD0 に関連するコードを削除したあと、main のファイルは以下のようになります。

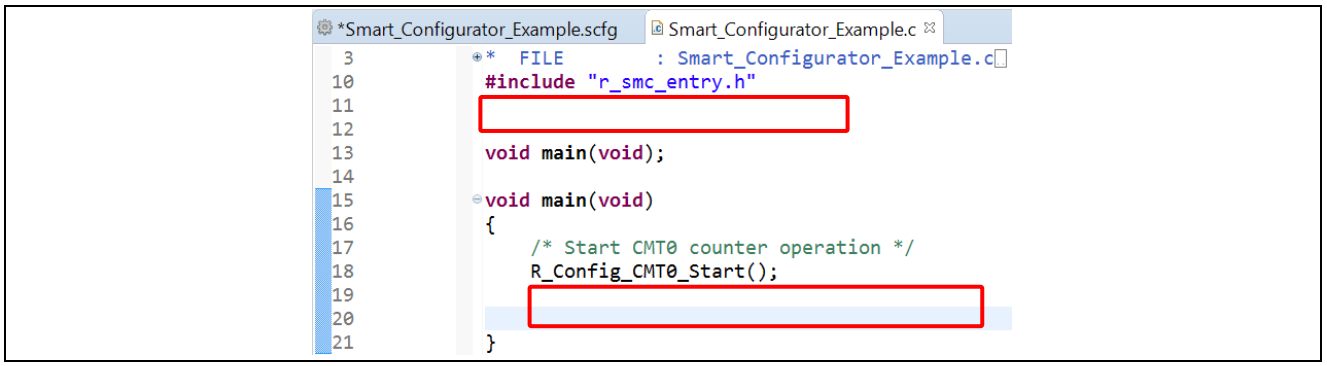


図 4-6　S12AD0 に関連するコードを削除した後の main ファイル

4.2　ペリフェラルドライバの追加

1) Smart_Configurator_Example.scfg 画面で、[コンポーネント]タブをクリックします。
2) をクリックし、新しいコンポーネントを追加します。

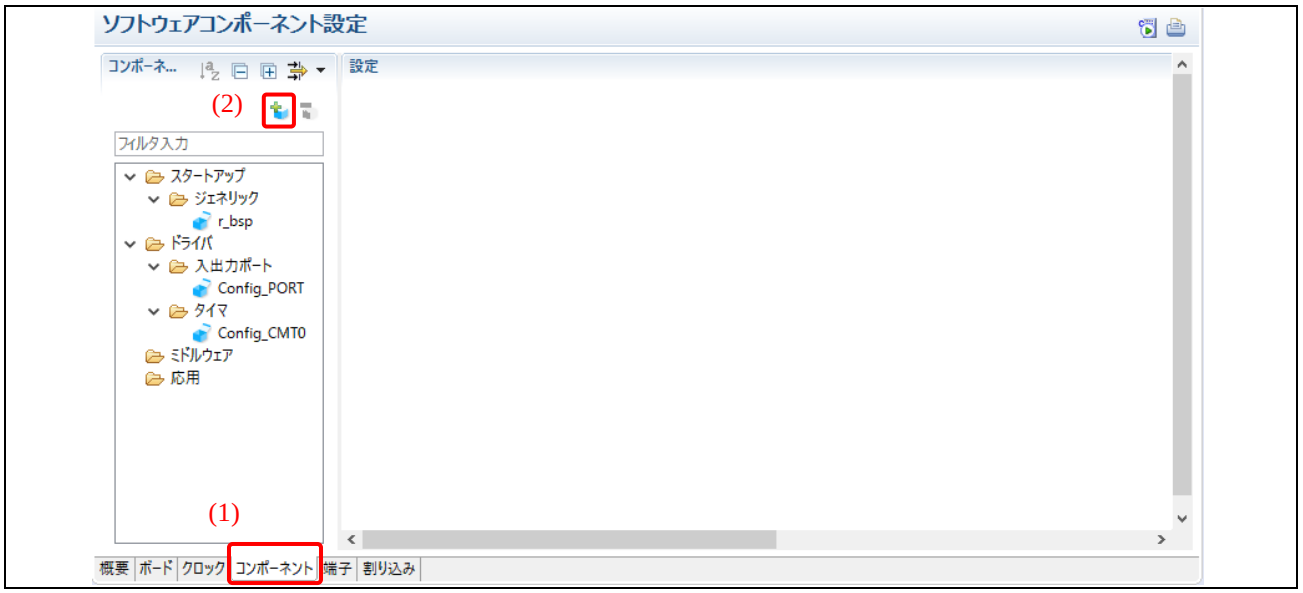


図 4-7　スマート・コンフィグレータでのソフトウェアコンポーネント設定

3) プロジェクトに“シリアル通信インタフェース”ドライバを追加します。
   a. “機能”オプションから“通信 (Drivers)”を選択します。
   b. コンポーネントリストを開き、“SCI/SCIF 調歩同期式モード”ドライバを選択します。
   c. [次へ]をクリックして続けます。

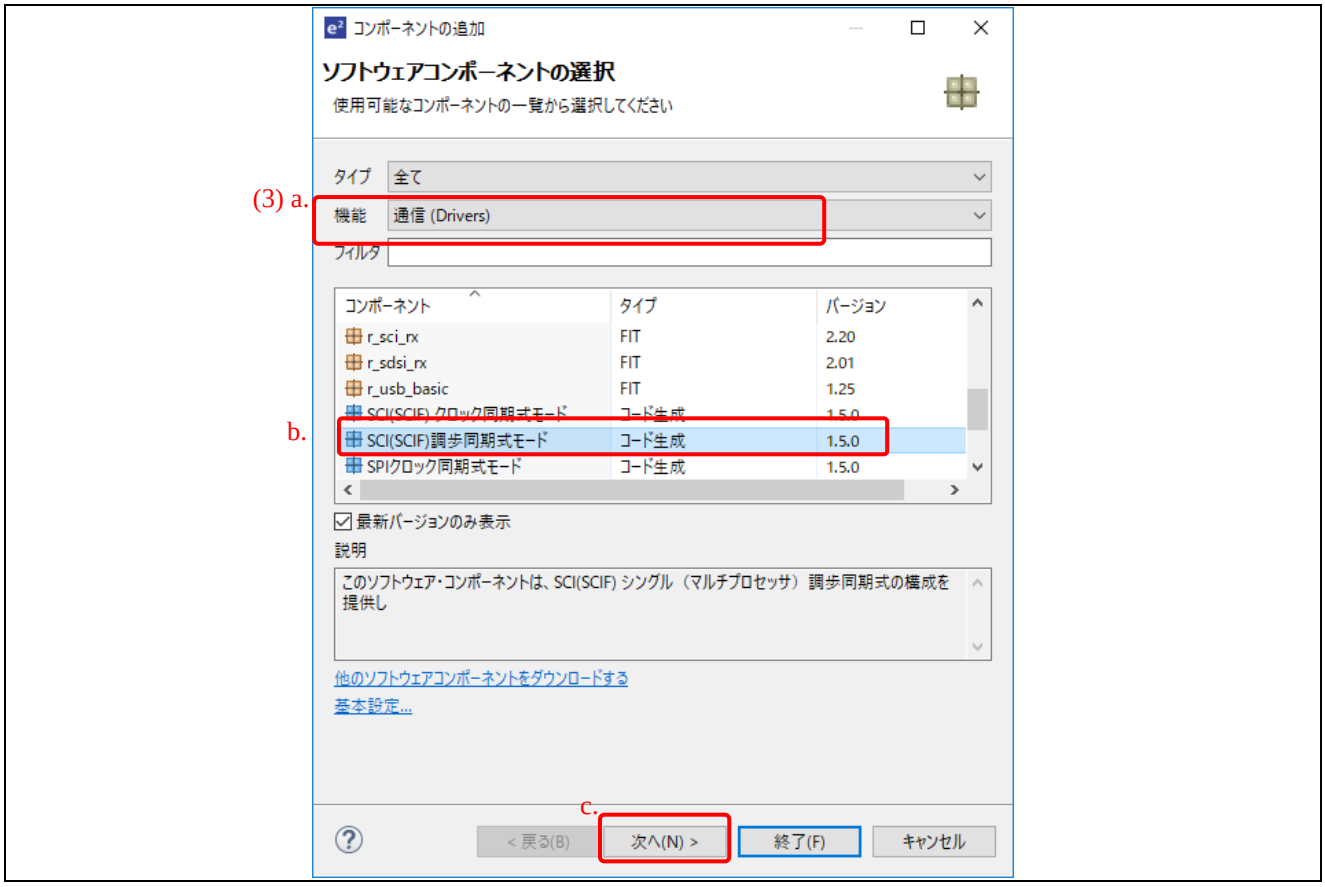


図 4-8　ソフトウェアコンポーネントの選択

d. リソースとして SCI8 を選択します。
e. 作業モードは、“送信/受信”を選択します。
f. デフォルトのコンフィグレーション名を、そのまま使用します。このコンフィグレーション名から、ソースファイルと API が生成されます。
g. [終了]をクリックします。

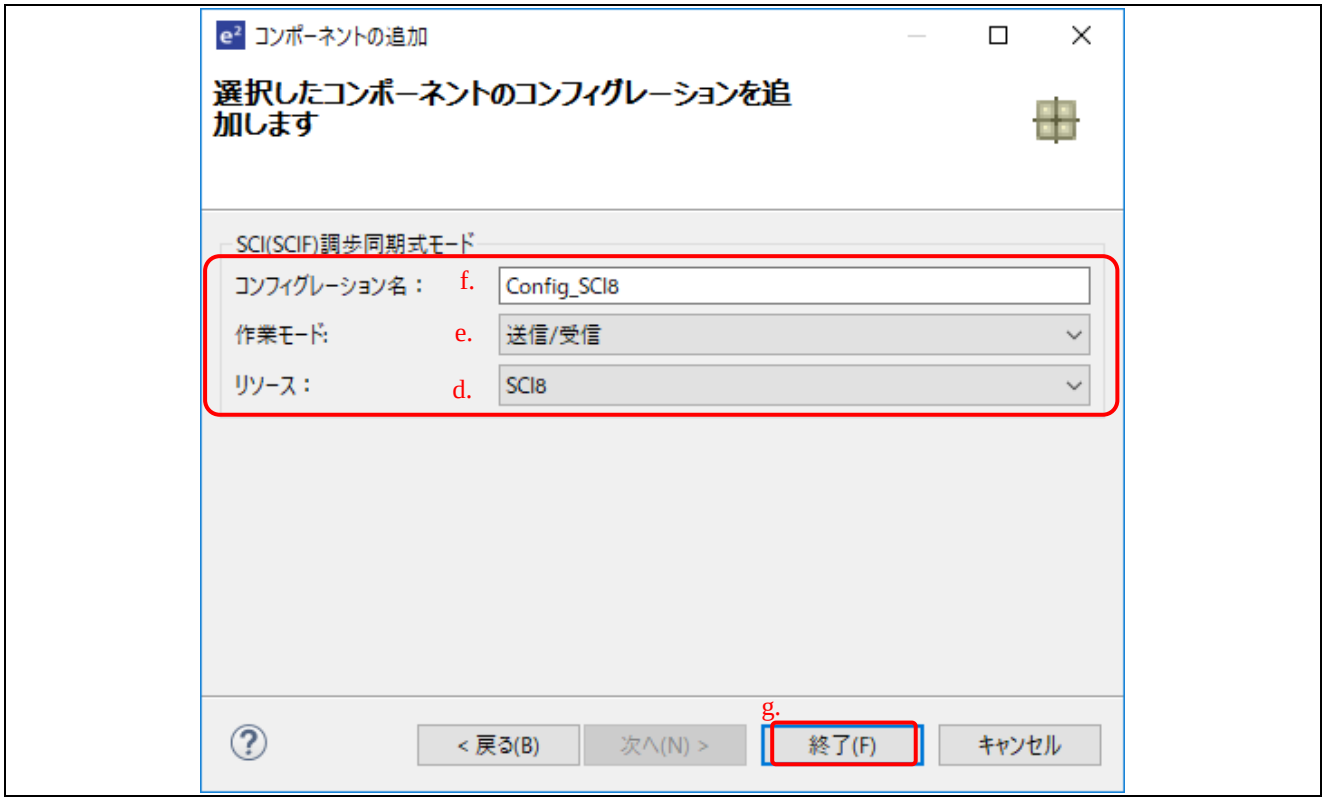


図 4-9　プロジェクトに新しいコンフィグレーションを追加

4) 設定画面では、以下が設定されていることを確認します。
- スタートビット検出設定：RXD8 端子の立下りエッジ
- データ・ビット長設定：8 ビット
- パリティ設定：禁止
- ストップビット設定：1 ビット
- ビットレート：9600 bps

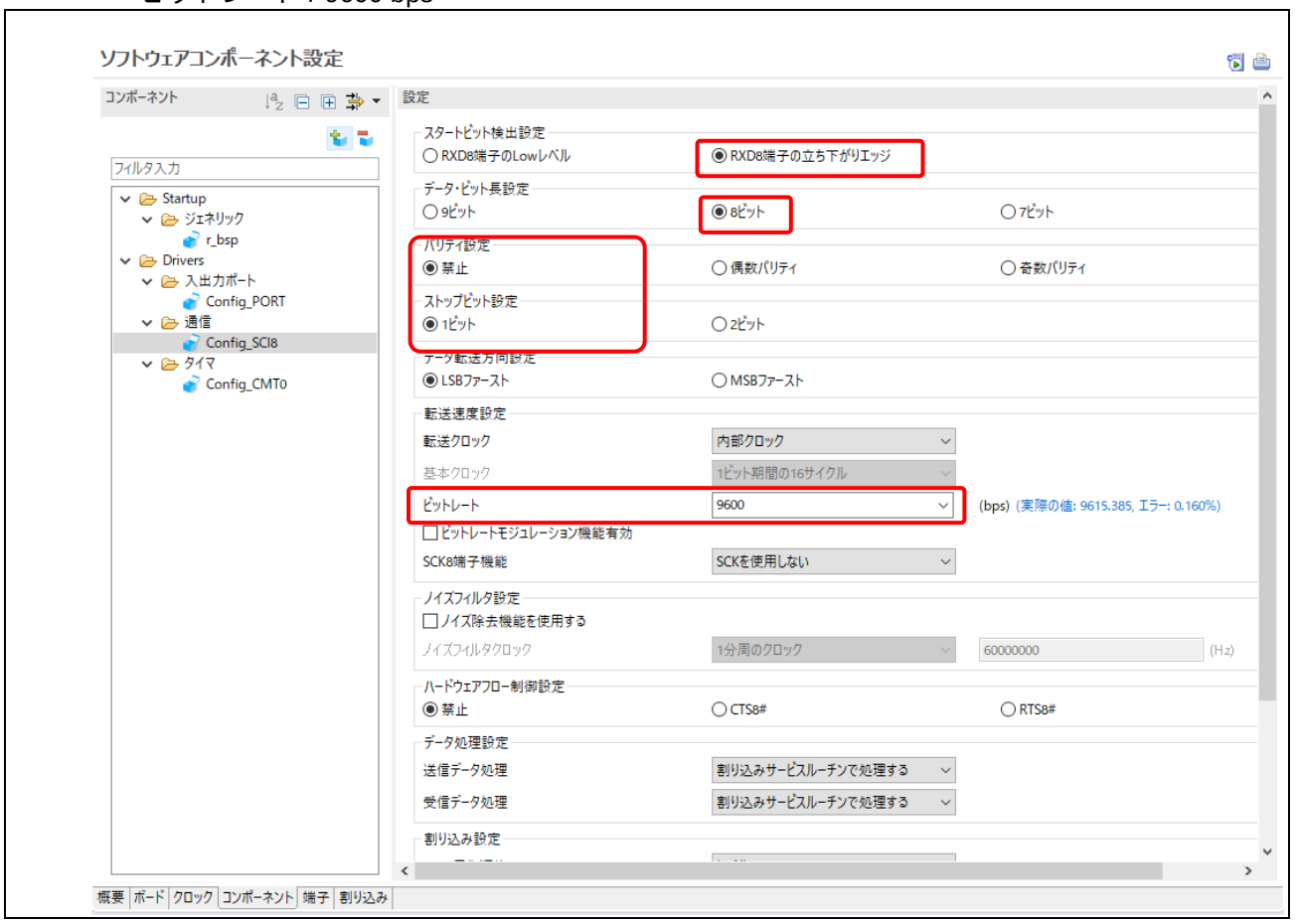


図 4-10　SCI/SCIF 調歩同期式モードドライバ設定

5) [端子]タブで  をクリックし、ツリーをソフトウェアコンポーネントビューに切り替えます。



図 4-11　端子タブ

6) ツリーで Config_SCI8 をクリックし、端子設定を表示します。
7) PJ1 に RXD8、PJ2 に TXD8 が割り当てられていることを確認します。

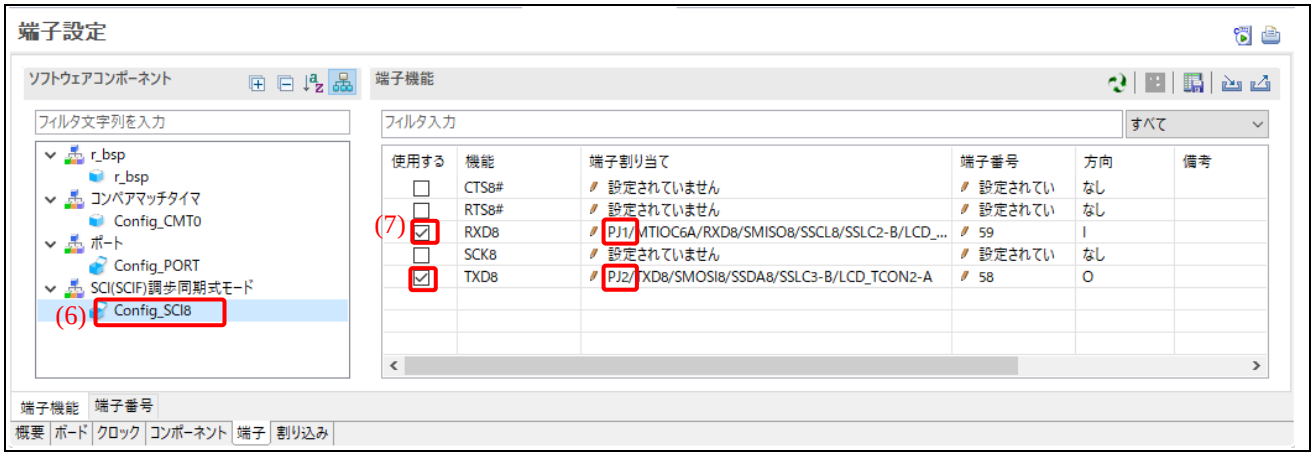


図 4-12　*Config_SCI8* の端子設定

4.3　コード生成

1) [ソフトウェアコンポーネント設定]タブで  をクリックし、コードを生成します。

4.4 SCI/SCIF 調歩同期式モードドライバにアプリケーションコードを追加

1) プロジェクトツリーで、\src\smc_gen\Config_SCI8 フォルダにある“Config_SCI8.h”を開きます。ファイルの最後にあるユーザコードエリアに、以下の宣言を追加します。

```
/* Sends SCI8 data and waits for transmit end flag */
MD_STATUS R_SCI8_AsyncTransmit(uint8_t * const tx_buf, const uint16_t
tx_num);
```

2) プロジェクトツリーで、\src\smc_gen\Config_SCI8 フォルダにある“Config_SCI8_user.c”を開きます。

a. ファイルの先頭近くにあるユーザコードエリアに、以下の宣言を追加します。

```
/* Global variable used to receive a character from PC terminal */
volatile uint8_t g_rx_char;

/* Flag used to detect whether data is received */
volatile uint8_t g_rx_flag;

/* Flag used to detect completion of transmission */
static volatile uint8_t SCI8_txdone;
```

b. *r_Config_SCI8_callback_transmitend (void)* 関数で、送信完了フラグをセットします。

```
/* Set transmit completion flag */
SCI8_txdone = 1U;
```

c. *r_Config_SCI8_callback_receiveend (void)* 関数で受信完了フラグをセットし、受信を再開します。

```
/* Set receive completion flag */
g_rx_flag = 1U;

/* Set SCI8 receive buffer address and restart reception */
R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);
```

d. ファイルの最後にあるユーザコードエリアに、SCI8 データを送信し送信完了フラグを待つ関数を追加します。

```
/***********************************************************************
*********
* Function Name: R_SCI8_AsyncTransmit
* Description  : This function sends SCI8 data and waits for the
transmit end flag
* Arguments    : tx_buf -
*                    transfer buffer pointer
*                tx_num -
*                    buffer size
* Return Value : status -
*                    MD_OK or MD_ARGERROR
***********************************************************************
********/
MD_STATUS R_SCI8_AsyncTransmit (uint8_t * const tx_buf, const uint16_t
tx_num)
{
    MD_STATUS status = MD_OK;

    /* Clear transmit completion flag before transmission */
    SCI8_txdone = 0U;

      /* Set SCI8 transmit buffer address and start transmission */
    status = R_Config_SCI8_Serial_Send(tx_buf, tx_num);

    /* Wait for transmit end flag */
    while (0U == SCI8_txdone)
    {
        /* Wait */
    }
    return (status);
}
/***********************************************************************
*********
* End of function R_SCI8_AsyncTransmit
      ***********************************************************************
**************/
```

ソースファイルは以下のようになります。

```
* File Name    : Config_SCI8_user.c

Pragma directive
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

Includes
#include "r_cg_macrodriver.h"
#include "Config_SCI8.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"

Global variables and functions
extern volatile uint8_t * gp_sci8_tx_address;                /* SCI8 transmit buffer address */
extern volatile uint16_t  g_sci8_tx_count;                   /* SCI8 transmit data number */
extern volatile uint8_t * gp_sci8_rx_address;                /* SCI8 receive buffer address */
extern volatile uint16_t  g_sci8_rx_count;                   /* SCI8 receive data number */
extern volatile uint16_t  g_sci8_rx_length;                  /* SCI8 receive data length */
/* Start user code for global. Do not edit comment generated here */
/* Global variable used to receive a character from PC terminal */
volatile uint8_t g_rx_char;

/* Flag used to detect whether data is received */
volatile uint8_t g_rx_flag;

/* Flag used to detect completion of transmission */
static volatile uint8_t SCI8_txdone;
/* End user code. Do not edit comment generated here */

[redacted]


/***********************************************************************************************************
* Function Name: r_Config_SCI8_callback_transmitend
* Description  : This function is a callback function when SCI8 finishes transmission
* Arguments    : None
* Return Value : None
***********************************************************************************************************/

static void r_Config_SCI8_callback_transmitend(void)
{
    /* Start user code for r_Config_SCI8_callback_transmitend. Do not edit comment generated here */
    /* Set transmit completion flag */
    SCI8_txdone = 1U;
    /* End user code. Do not edit comment generated here */
}

/***********************************************************************************************************
* Function Name: r_Config_SCI8_callback_receiveend
* Description  : This function is a callback function when SCI8 finishes reception
* Arguments    : None
* Return Value : None
***********************************************************************************************************/

static void r_Config_SCI8_callback_receiveend(void)
{
    /* Start user code for r_Config_SCI8_callback_receiveend. Do not edit comment generated here */
    /* Set receive completion flag */
    g_rx_flag = 1U;

    /* Set SCI8 receive buffer address and restart reception */
    R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);

    /* End user code. Do not edit comment generated here */
}

[redacted]
```

```
/* Start user code for adding. Do not edit comment generated here */
/***********************************************************************
* Function Name: R_SCI8_AsyncTransmit
* Description  : This function sends SCI8 data and waits for the transmit end flag
* Arguments    : tx_buf -
*                    transfer buffer pointer
*                tx_num -
*                    buffer size
* Return Value : status -
*                    MD_OK or MD_ARGERROR
***********************************************************************/
MD_STATUS R_SCI8_AsyncTransmit (uint8_t * const tx_buf, const uint16_t tx_num)
{
    MD_STATUS status = MD_OK;

    /* Clear transmit completion flag before transmission */
    SCI8_txdone = 0U;

    /* Set SCI8 transmit buffer address and start transmission */
    status = R_Config_SCI8_Serial_Send(tx_buf, tx_num);

    /* Wait for transmit end flag */
    while (0U == SCI8_txdone)
    {
        /* Wait */
    }
    return (status);
}
/***********************************************************************
* End of function R_SCI8_AsyncTransmit
***********************************************************************/

/* End user code. Do not edit comment generated here */
```

図 4-13　Config_SCI8_user.c

4.5　*main()*にアプリケーションコードを追加

1)　Smart_Configurator_Example.c ファイルの先頭近くに、ヘッダファイルと宣言を追加します。

```
#include <stdio.h>
#include <string.h>

/* Global variable for changing CMT0 interval */
volatile uint16_t interval_level = 5;

/* String used to print message at PC terminal */
static char print_str[80];

/* Flag used to detect whether data is received from PC terminal */
extern volatile uint8_t g_rx_flag;

/* Global variable used for storing data received from PC terminal */
extern volatile uint8_t g_rx_char;
```

2) 以下のコードを *main()*関数の *R_Config_CMT0_Start()*の後に追加し、ターミナルプログラムとの通信を開始します。

```
/* Set SCI8 receive buffer address and enable receive interrupt */
R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);

/* Enable SCI8 operation */
R_Config_SCI8_Start();

/* Print instruction onto PC terminal */
sprintf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not case sensitive)\r\n");
R_SCI8_AsyncTransmit((uint8_t *)print_str, (uint16_t)strlen(print_str));
sprintf(print_str, "\r\n a ----> Slower\n");
R_SCI8_AsyncTransmit((uint8_t *)print_str, (uint16_t)strlen(print_str));
sprintf(print_str, "\r\n b ----> Faster\n");
R_SCI8_AsyncTransmit((uint8_t *)print_str, (uint16_t)strlen(print_str));

while (1U)
{
        /* Get new blink interval level from PC terminal */
        if (g_rx_flag == 1U)
        {
                /* Instruction to slow down blinking rate is received */
                if (('A' == g_rx_char) || ('a' == g_rx_char))
                {
                        R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);

                        /* Notify the character received and inform next action */
                        sprintf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink at
                slower rate.\r\n");
                        R_SCI8_AsyncTransmit((uint8_t *)print_str, (uint16_t)strlen(print_str));

                        /* Get new blink interval level. Maximum level is 9. */
                        if (interval_level < 10)
                        {
                                interval_level++;
                        }
                        else
                        {
                                sprintf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to
                        blink faster. \r\n");
                                R_SCI8_AsyncTransmit((uint8_t *)print_str,
                        (uint16_t)strlen(print_str));
                        }
                }
                /* Instruction to increase blinking rate is received */
                else if (('B' == g_rx_char) || ('b' == g_rx_char))
                {
                        R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);

                        /* Notify the character received and inform next action */
                        sprintf(print_str, "\r\n Character 'b' or 'B' is received. LED2 will blink at
                faster rate.\r\n");
                        R_SCI8_AsyncTransmit((uint8_t *)print_str, (uint16_t)strlen(print_str));

                        /* Get new blink interval level. Minimum level is 1 */
                        if (interval_level > 1)
                        {
                                interval_level--;
                        }
                        else
                        {
                                sprintf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to
                        blink slower. \r\n");
                                R_SCI8_AsyncTransmit((uint8_t *)print_str,
                        (uint16_t)strlen(print_str));
                        }
                }
                else
                {
                        /* Do nothing */
                }

                /* Change blinking rate */
                CMT0.CMCOR = (uint16_t)(5000 * interval_level);

                /* Reset SCI8 reception flag*/
                g_rx_flag = 0U;
        }
        else
        {
        /* Do nothing */
        }
}
```

ソースファイルは以下のようになります。

```
 *  FILE        : Smart_Configurator_Example.c
#include "r_smc_entry.h"
#include <stdio.h>
#include <string.h>

/* Global variable for changing CMT0 interval */
volatile uint16_t interval_level = 5;

/* String used to print message at PC terminal */
static char print_str[80];

/* Flag used to detect whether data is received from PC terminal */
extern volatile uint8_t g_rx_flag;

/* Global variable used for storing data received from PC terminal */
extern volatile uint8_t g_rx_char;
```

```
void main(void);

void main(void)
{
    /* Start CMT0 counter operation */
    R_Config_CMT0_Start();

    /* Set SCI8 receive buffer address and enable receive interrupt */
    R_Config_SCI8_Serial_Receive((uint8_t *) &g_rx_char, 1);

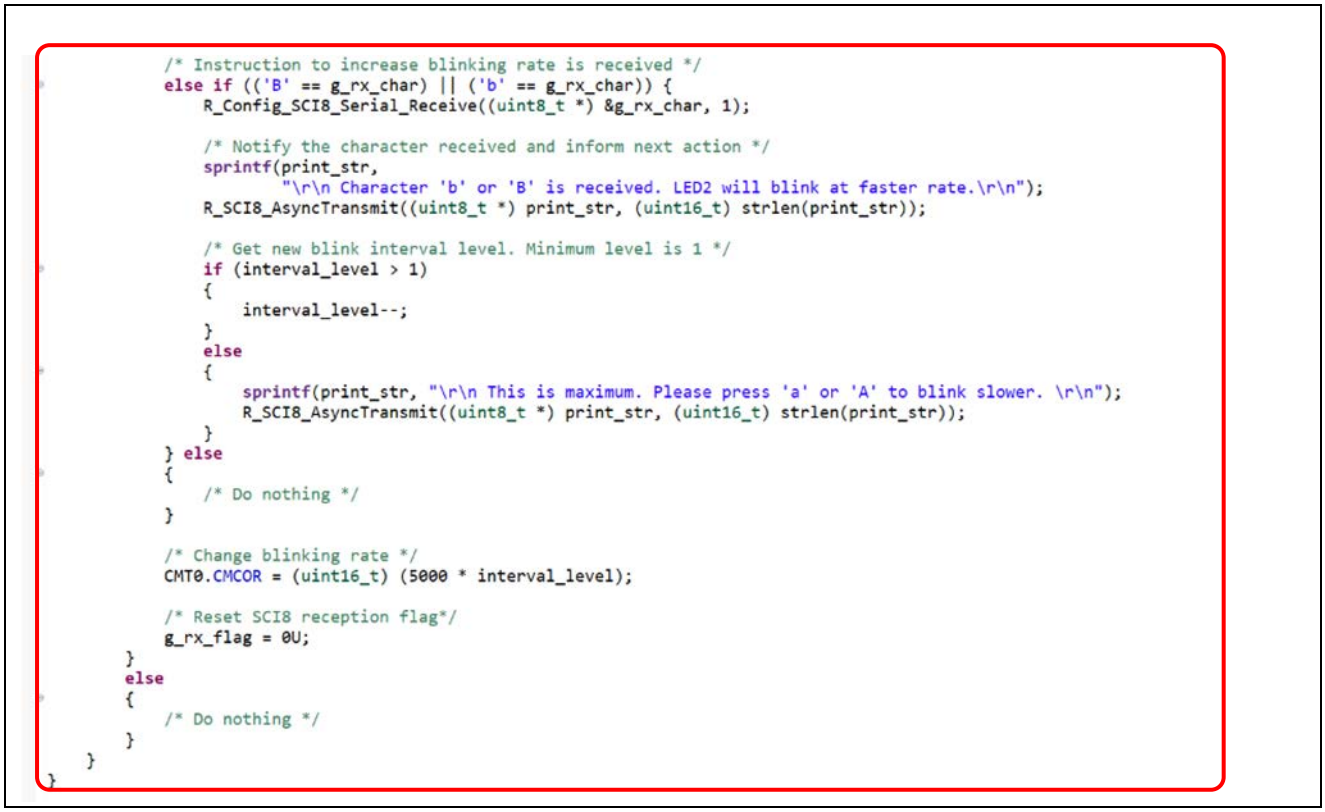
    /* Enable SCI8 operation */
    R_Config_SCI8_Start();

    /* Print instruction onto PC terminal */
    sprintf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not case sensitive)\r\n");
    R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
    sprintf(print_str, "\r\n a ----> Slower\n");
    R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
    sprintf(print_str, "\r\n b ----> Faster\n");
    R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

    while (1U)
    {
        /* Get new blink interval level from PC terminal */
        if (g_rx_flag == 1U)
        {
            /* Instruction to slow down blinking rate is received */
            if (('A' == g_rx_char) || ('a' == g_rx_char))
            {
                R_Config_SCI8_Serial_Receive((uint8_t *) &g_rx_char, 1);

                /* Notify the character received and inform next action */
                sprintf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink at slower rate.\r\n");
                R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

                /* Get new blink interval level. Maximum level is 9. */
                if (interval_level < 10)
                {
                    interval_level++;
                }
                else
                {
                    sprintf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to blink faster. \r\n");
                    R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
                }
            }
```



```
            /* Instruction to increase blinking rate is received */
            else if (('B' == g_rx_char) || ('b' == g_rx_char)) {
                R_Config_SCI8_Serial_Receive((uint8_t *) &g_rx_char, 1);

                /* Notify the character received and inform next action */
                sprintf(print_str,
                        "\r\n Character 'b' or 'B' is received. LED2 will blink at faster rate.\r\n");
                R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

                /* Get new blink interval level. Minimum level is 1 */
                if (interval_level > 1)
                {
                    interval_level--;
                }
                else
                {
                    sprintf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to blink slower. \r\n");
                    R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
                }
            } else
            {
                /* Do nothing */
            }

            /* Change blinking rate */
            CMT0.CMCOR = (uint16_t) (5000 * interval_level);

            /* Reset SCI8 reception flag*/
            g_rx_flag = 0U;
        }
        else
        {
            /* Do nothing */
        }
    }
}
```

図 4-14　*main* 関数内の調歩同期式アプリケーションコード

4.6　ビルドとハードウェアボードでの実行

プロジェクトをビルドした後、プロジェクトをデバッグする前に以下の手順を実行します。

ステップ 1.　PC の COM ポートと、RX65N 2MB 用 Renesas Starter Kit+上の USB シリアルコネクタ (G1CUSB0)を、USB ケーブル (ミニ B) を使用して接続します。（シリアルポートの位置は、4.7 章を参照してください。）

ステップ 2.　PC でターミナルプログラム(例　Tera Term)を開き、以下の手順を実行します。

- RSK USB Serial Port を選択します。

  注：ポート名は PC のドライバによって変わります。



図 4-15　USB シリアルポート

- [設定]メニューから [シリアルポート…]を選択します。

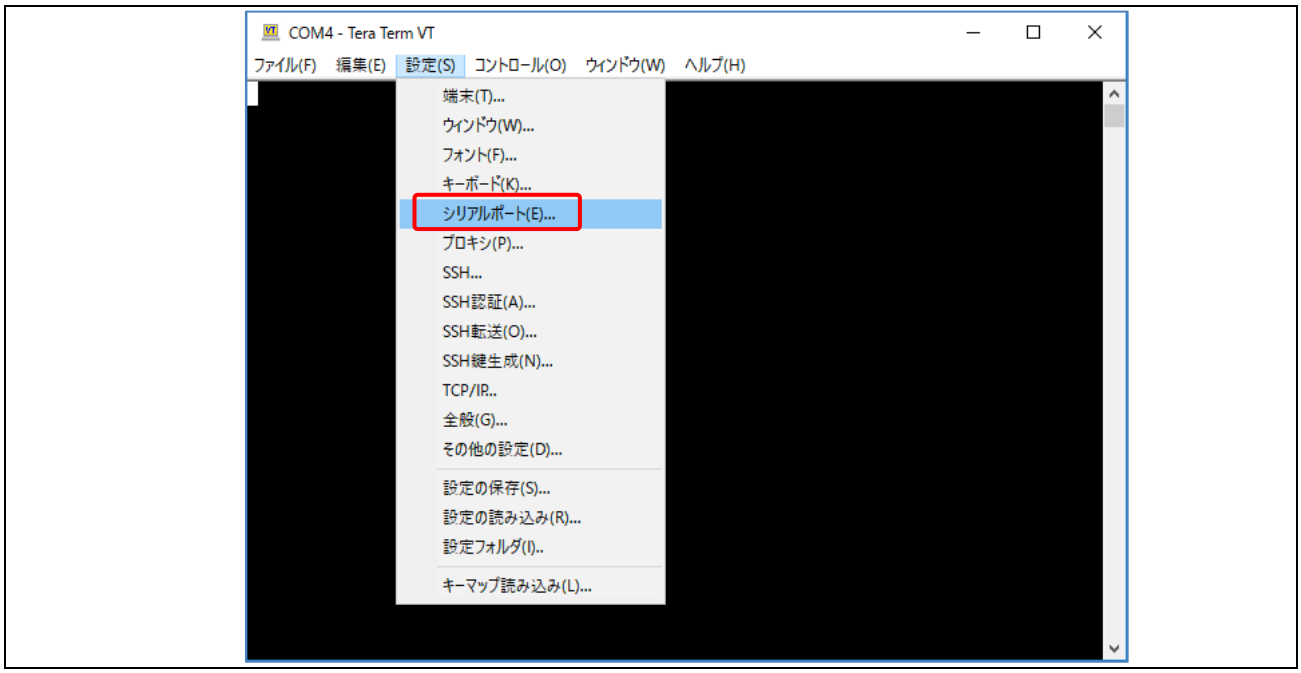


図 4-16　PC のターミナルウィンドウ

- [シリアルポート設定]ウインドウで、SCI/SCIF 調歩同期式モードドライバの設定に合うように設定し、[OK]をクリックします。
  - データ・ビット長設定：8 ビット
  - パリティ設定：禁止
  - ストップビット設定：1 ビット
  - ビットレート：9600 bps

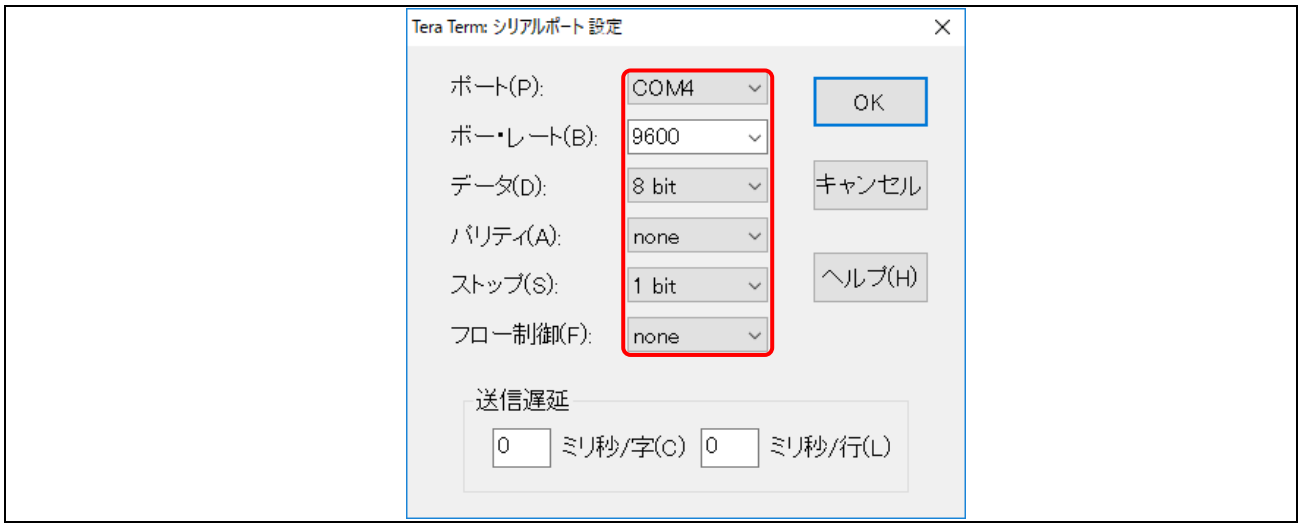


図 4-17　シリアルポート設定

ステップ 3.　e² studio で、デバッグアイコン をクリックします。

ステップ 4.　デバッグ・パースペクティブで再開ボタン をクリックし、プロジェクトを実行します。

PC ターミナルウィンドウに以下のメッセージが表示されます。メッセージに従い、LED2 点滅速度を上げるか、または下げてください。

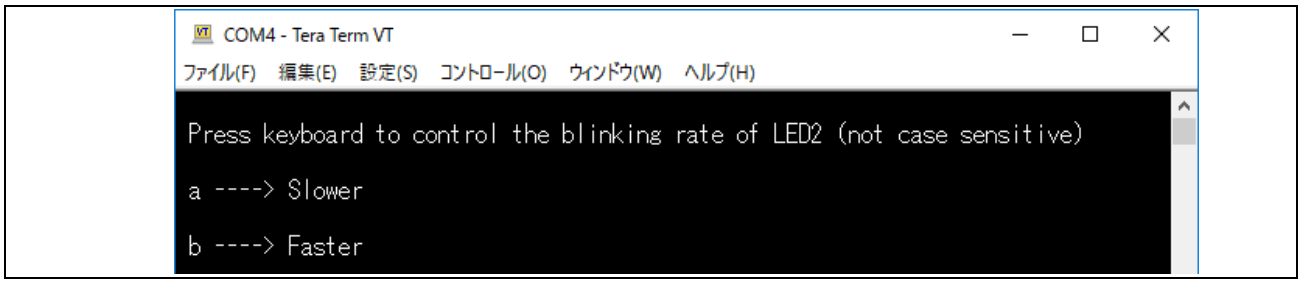


図 4-18　PC のターミナルウィンドウ

これらのメッセージは、RSK ボードが PC からの入力を受信した場合に表示されます。

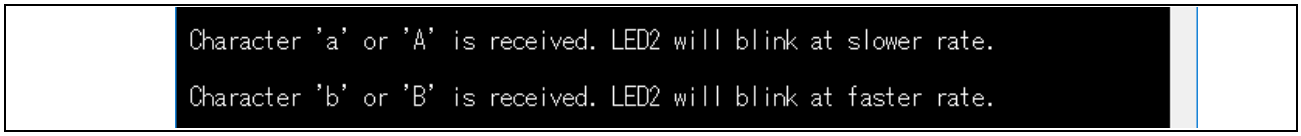


図 4-19　PC からの入力を受信した場合の RSK ボードからの返答

4.7　ボードでの動作確認

PC ターミナル上のキーが'a'または'b'のとき、RX65N 2MB ボード用 Renesas Starter Kit+上の LED2 の点滅周期が変わります。

- a：LED2 の点滅は遅くなります
- b：LED2 の点滅は早くなります

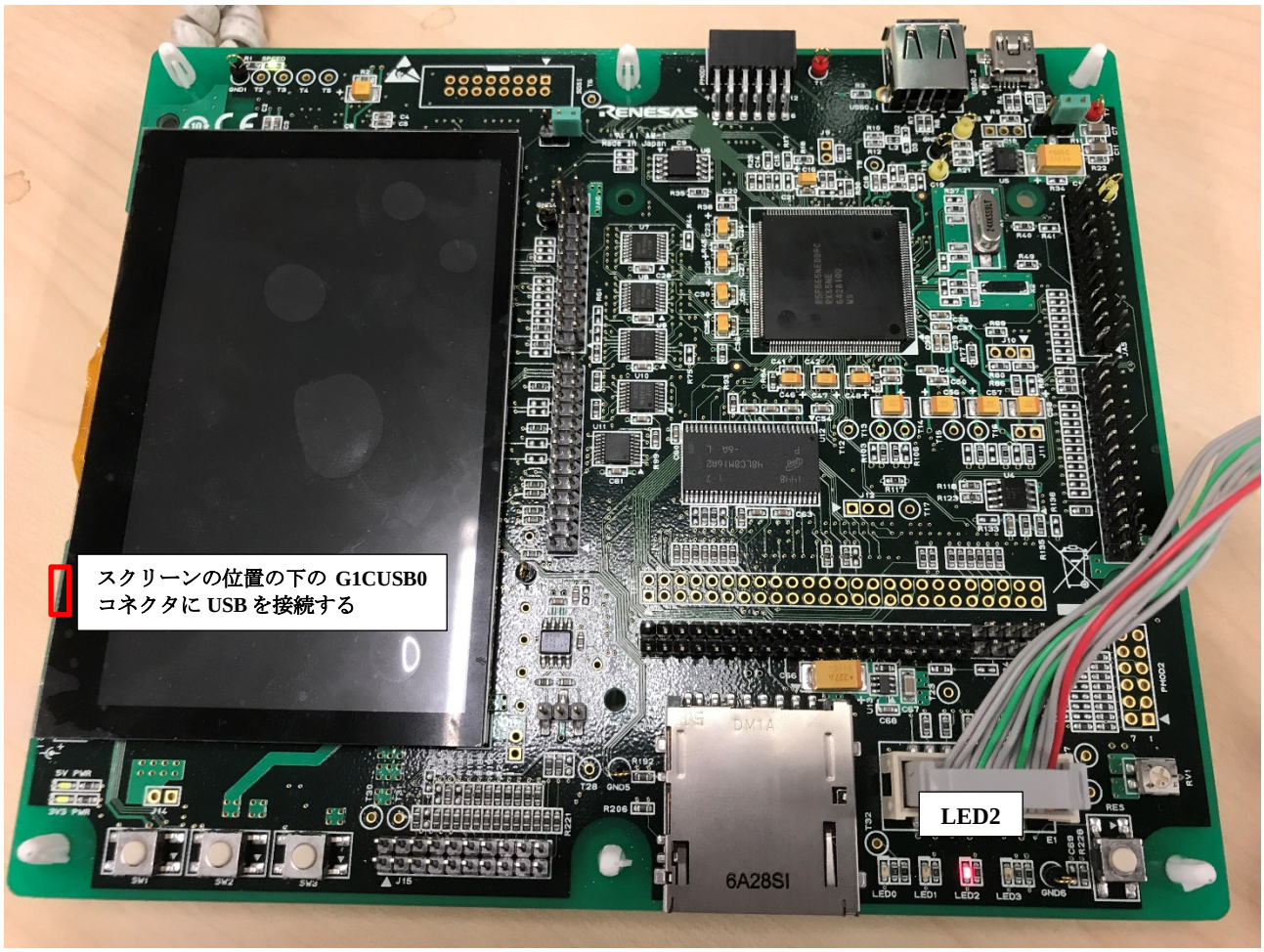


図 4-20　RX65N 2MB ボード上の LED2 とシリアルポート

5. Application Example 4（DMA を使用したデータ転送）

この前の章では、SCI8 と内蔵 RAM の間のデータ転送に、割り込みサービスルーチンを使用しました。データ転送は、CPU によって行われます。

本章では、DMA コントローラを使用したデータ転送について説明します。DMA コントローラを使用することで、メモリ操作の速度が向上し、CPU はデータ転送から解放されます。

注：　ダイレクト・メモリ・アクセス (DMA)は、入出力(I/O)デバイスがメインメモリとデータを直接送受信できるようにする方法です。CPU を介さないため、メモリ操作の速度を上げることができます。

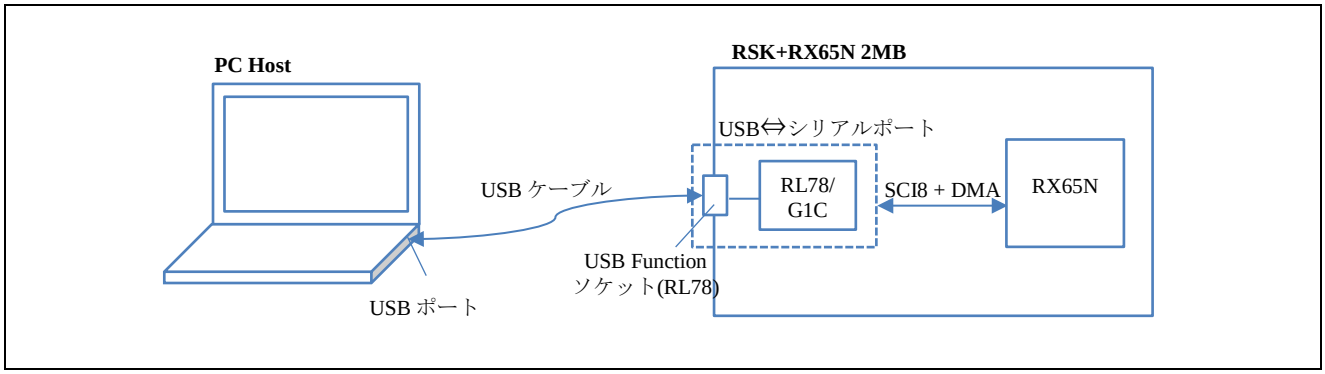


図 5-1　DMAC によるデータ処理

ドライバの選択の変更を以下の表に示します。

	ドライバ	リソース	機能
1.新ドライバの追加	DMA コントローラ	DMAC0	SCI8 から RAM へのデータ転送（受信）
		DMAC1	RAM から SCI8 へのデータ転送（送信）
2.ドライバの修正	SCI/SCIF 調歩同期式モード	SCI8	LED 点滅周期を制御するための、PC ターミナルとターゲットボード間の通信

SCI-DMA アプリケーションコードは、以下のようなプログラムを追加します。

a) main 関数：

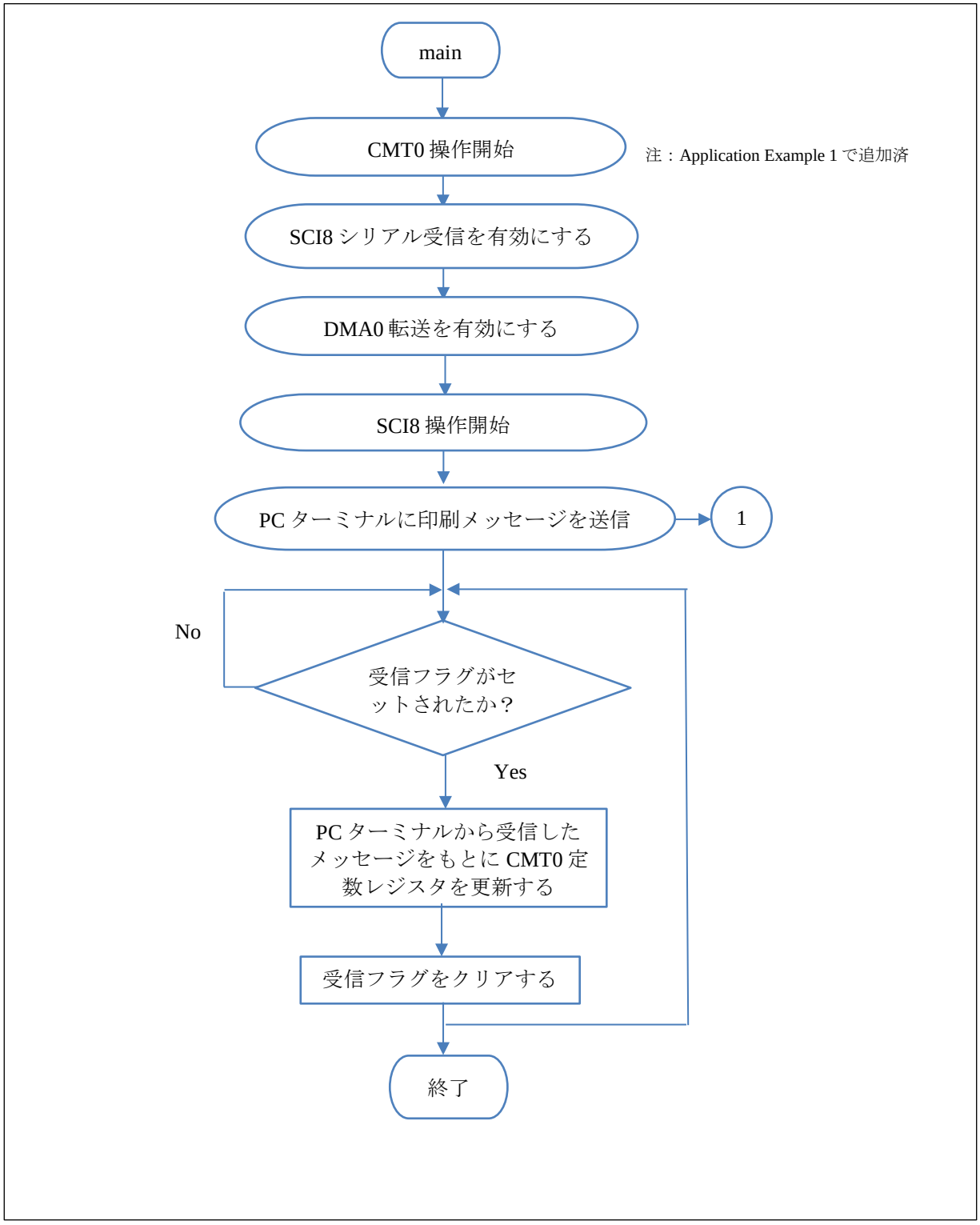


図 5-2　main フローチャート

b)　SCI8 ユーザ関数：

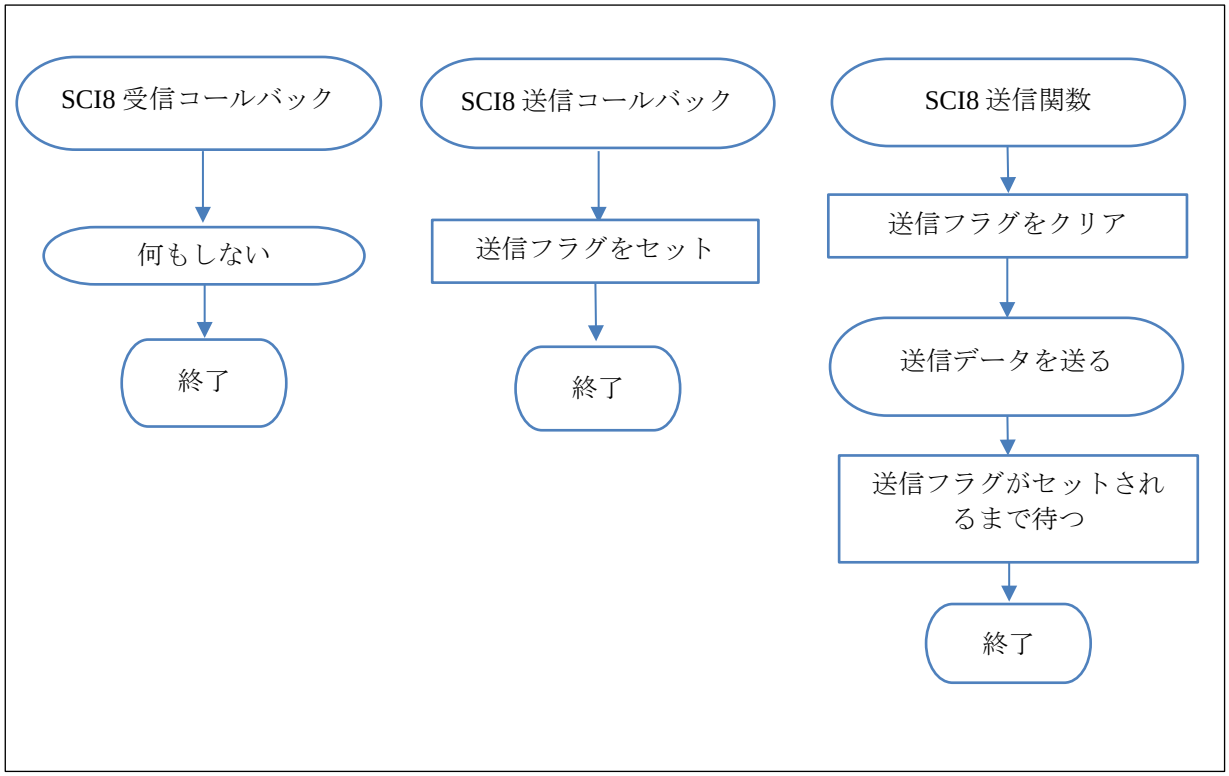


図 5-3　SCI ユーザ関数フローチャート

c)　DMAC0 ユーザ関数：

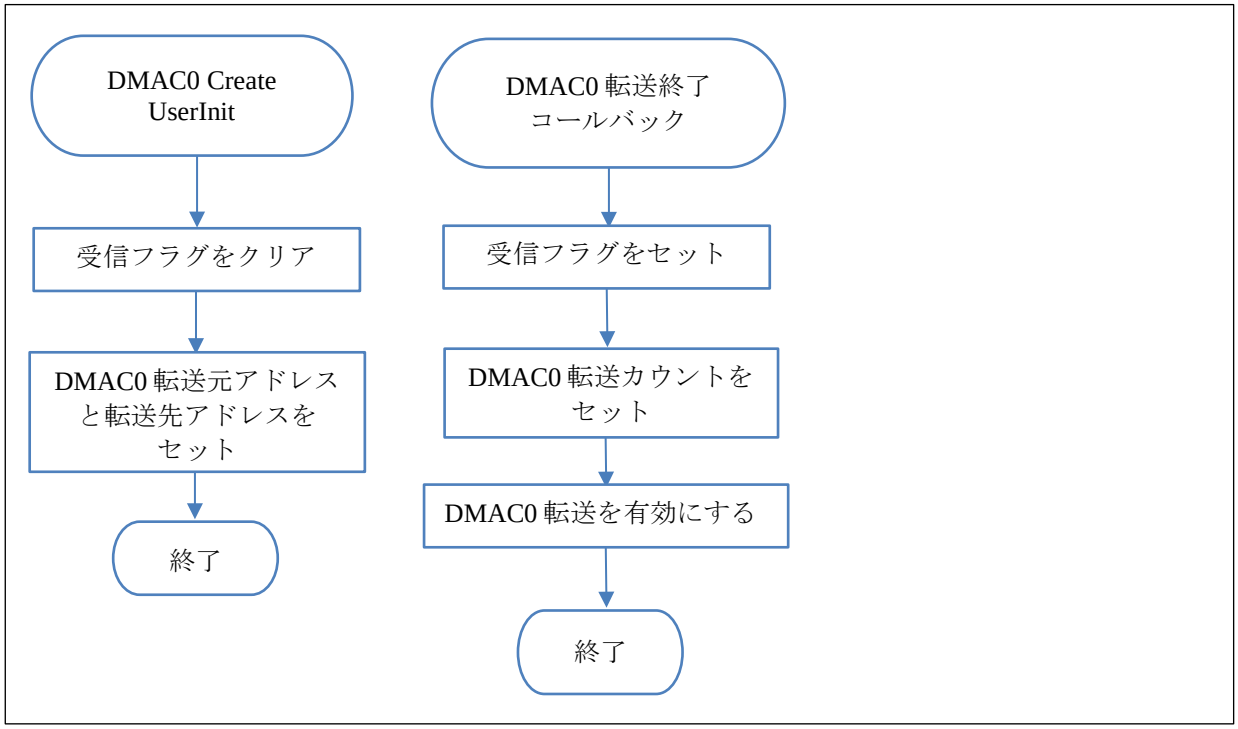


図 5-4　DMAC0 ユーザ関数フローチャート

d) DMAC1 ユーザ関数：

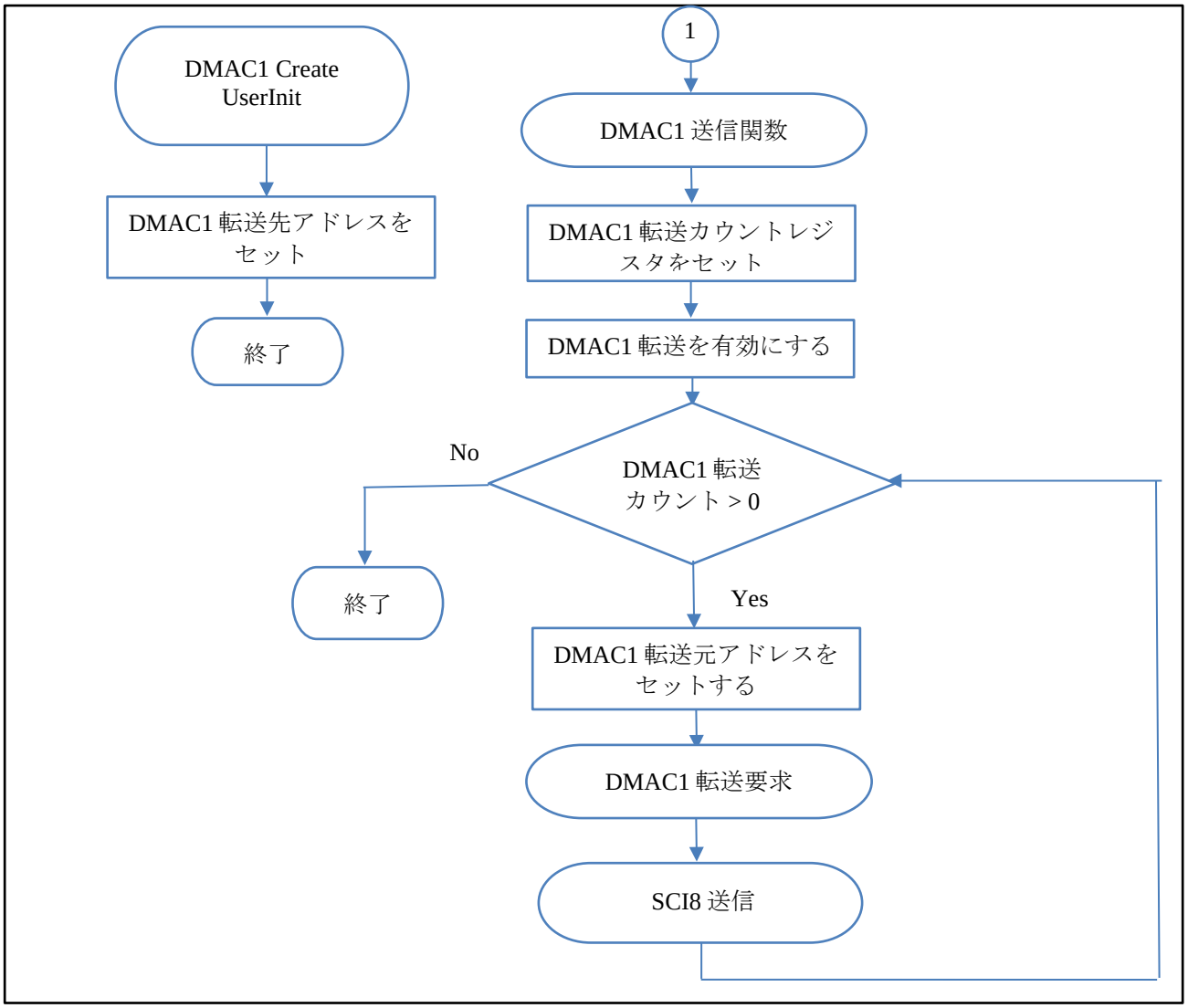


図 5-5　DMAC1 ユーザ関数フローチャート

5.1　周辺ドライバの追加と修正

5.1.1　DMAC0 ドライバの追加

1) Smart_Configurator_Example.scfg 画面で、[コンポーネント]タブをクリックします。
2) をクリックして、新しいコンポーネントを追加します。

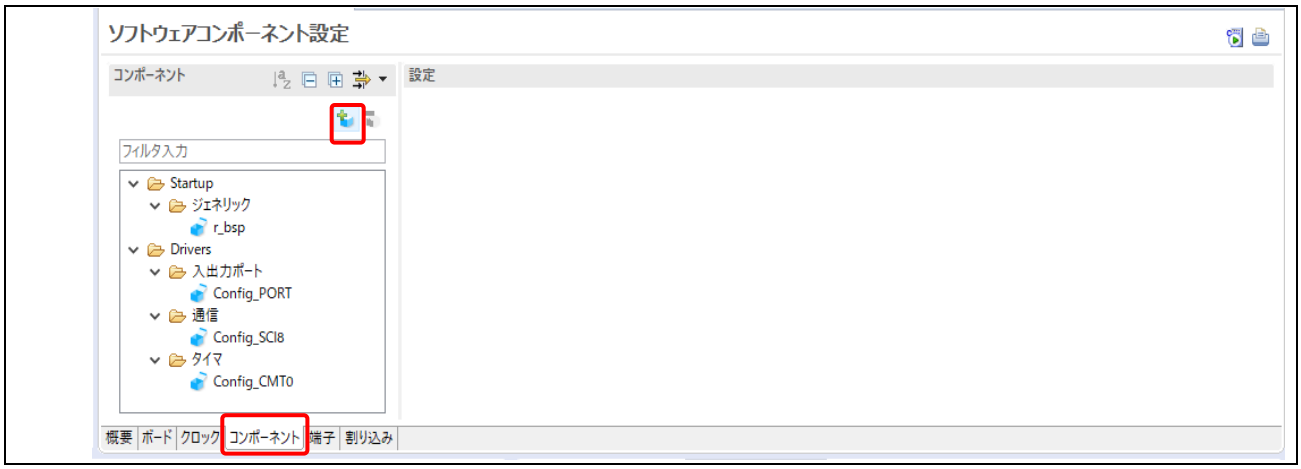


図 5-6　スマート・コンフィグレータでのソフトウェアコンポーネント設定

3) “DMA コントローラ”ドライバをプロジェクトに追加します。
   - “フィルタ”ボックスに“DMA”と入力します。
   - コンポーネントリストを開き、“DMA コントローラ”ドライバを選択します。
   - [次へ]をクリックします。

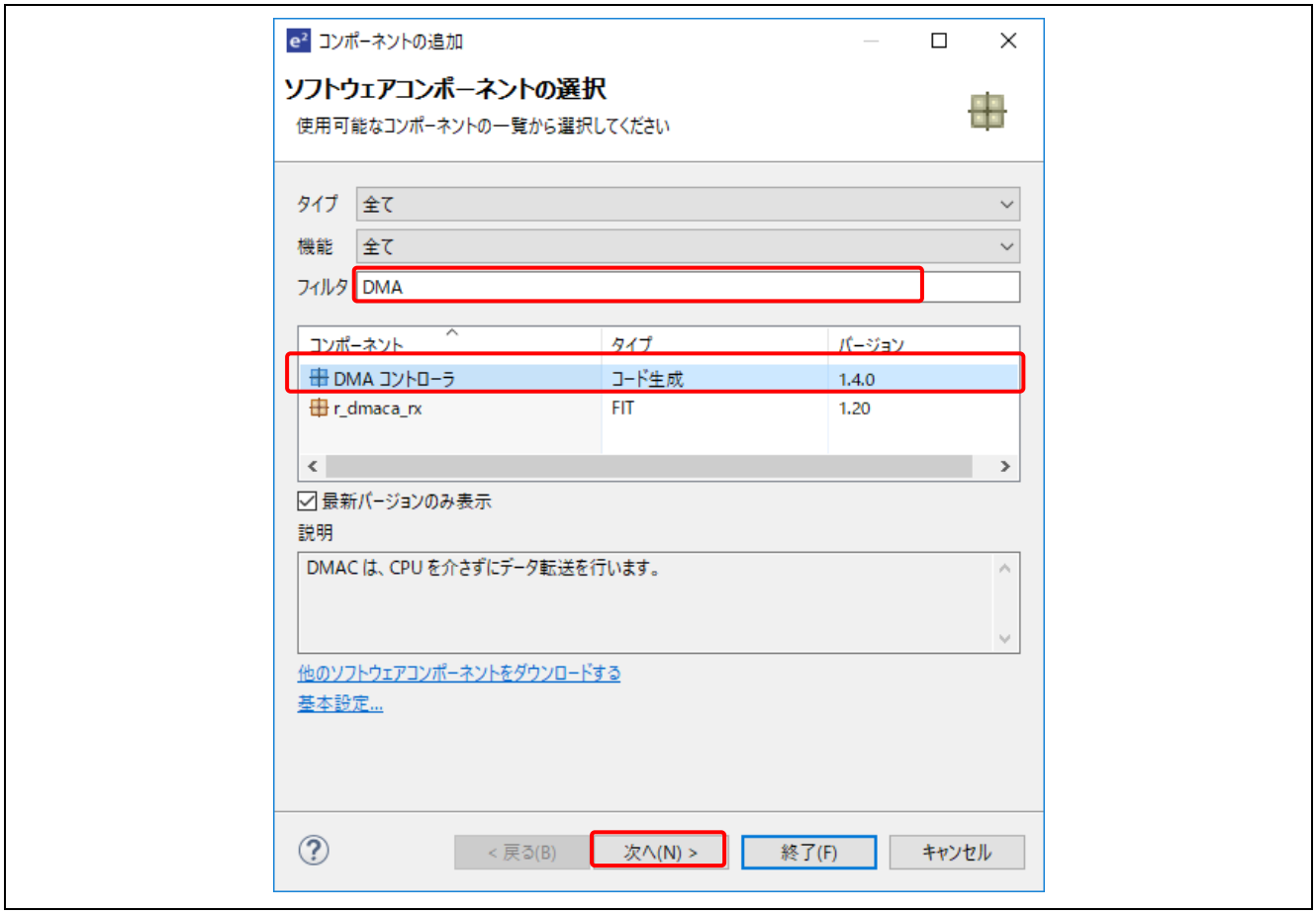


図 5-7　ソフトウェアコンポーネントの選択

4) [コンポーネントの追加]ダイアログで、以下を行います。
   - DMAC0 をリソースとして選択します。
   - デフォルトのコンフィグレーション名のままにします。このコンフィグレーション名を元に、ソースファイルと API が作成されます。
   - [終了]をクリックします。

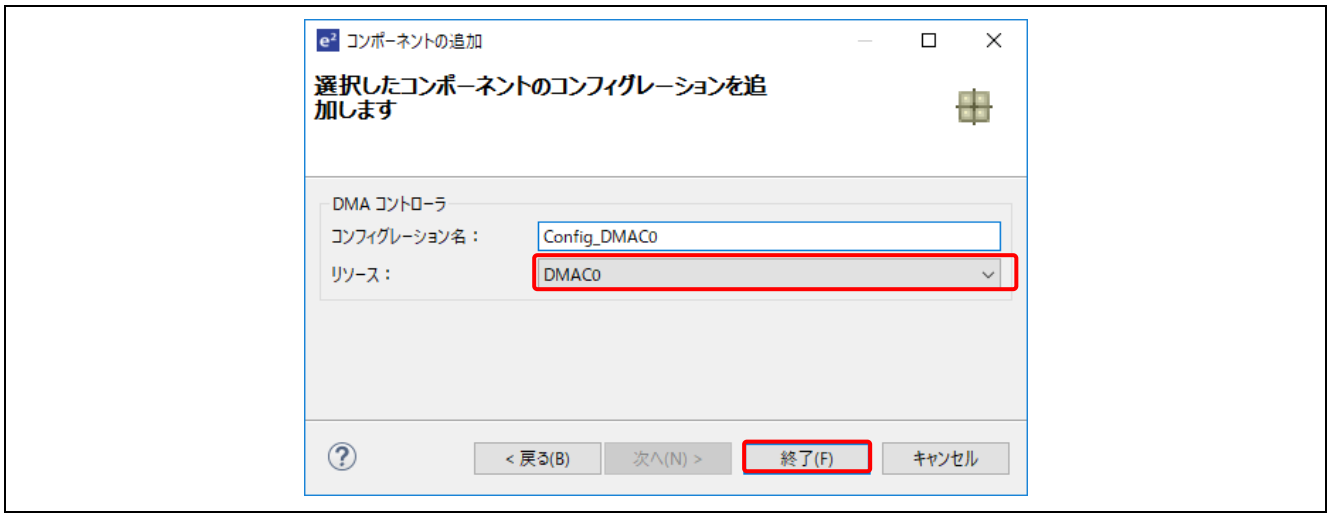


図 5-8　プロジェクトに新しいコンフィグレーションを追加

5) 設定画面で、下記のように設定します。
   - 起動要因：SCI8 (RXI8)
   - 転送終了割り込みを許可：選択
   - 他のデフォルト設定はそのままにします。

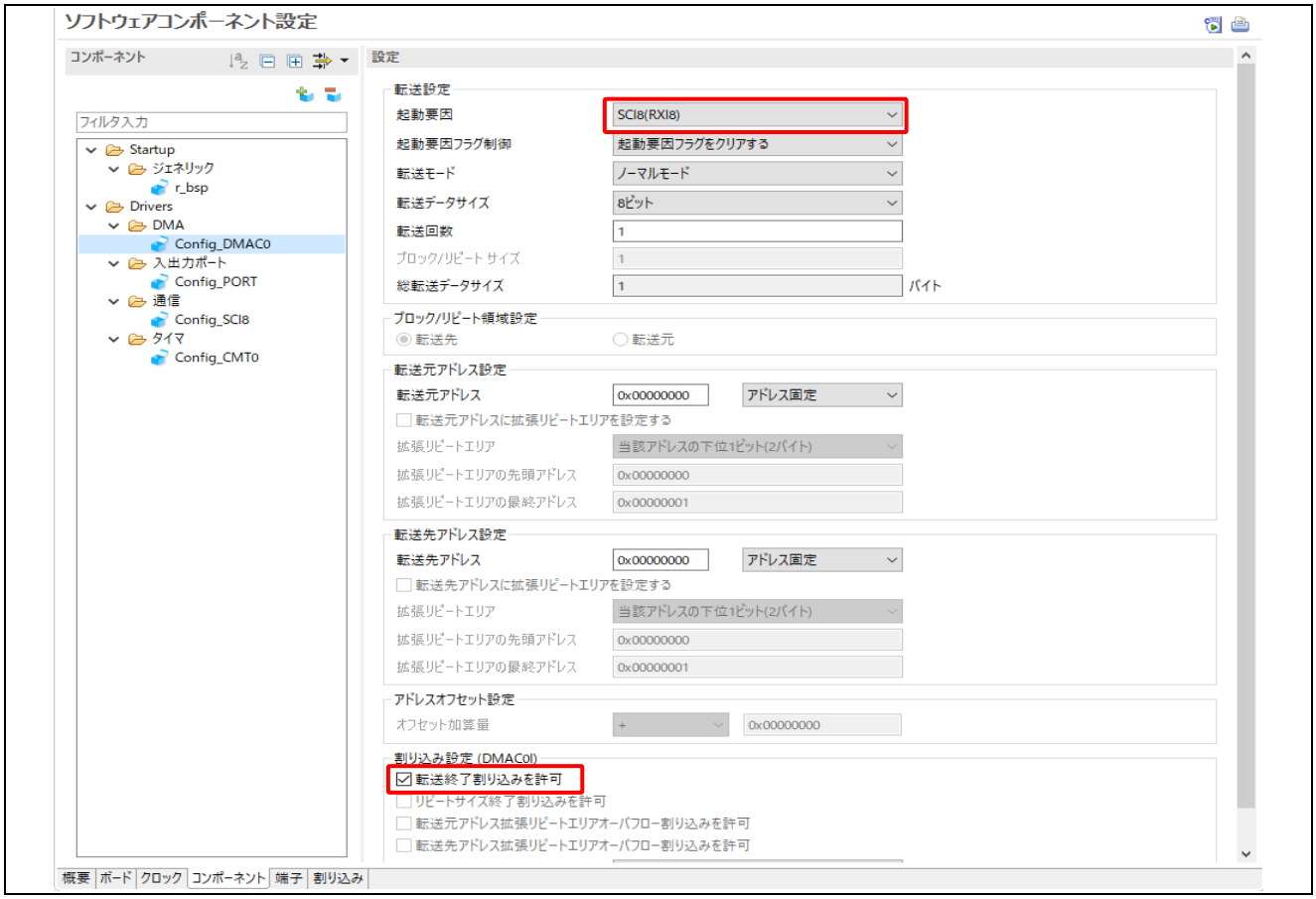


図 5-9　DMAC0 ドライバ設定

5.1.2　DMAC1 ドライバの追加

1) Smart_Configurator_Example.scfg 画面で、[コンポーネント]タブをクリックします。
2) をクリックし、新しいコンポーネントを追加します。

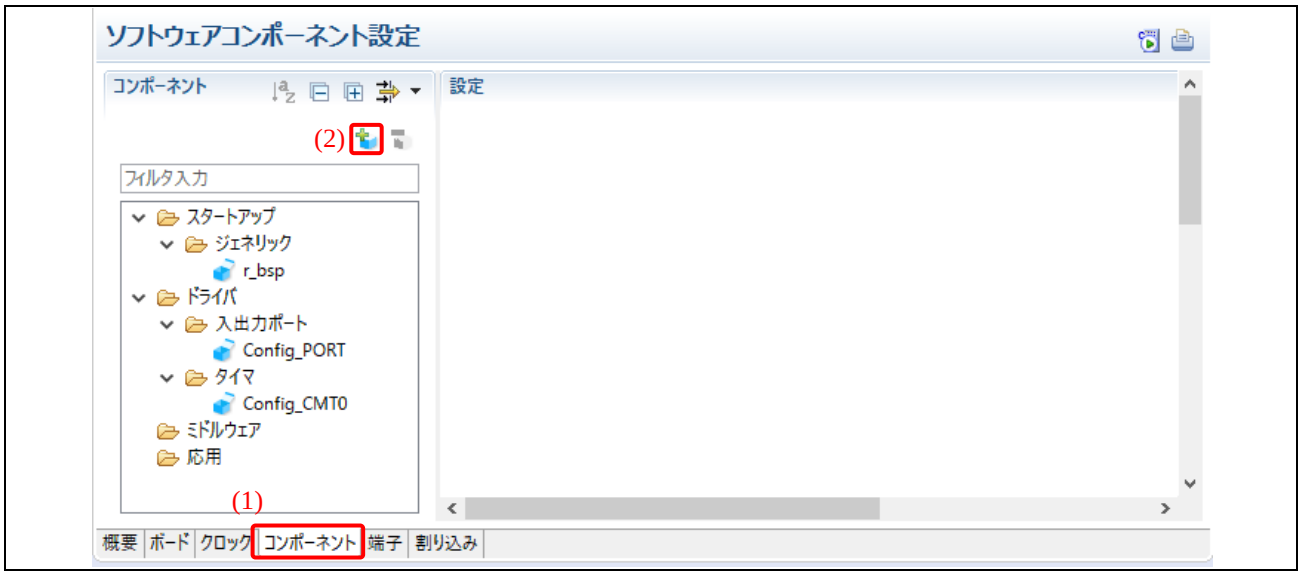


図 5-10　スマート・コンフィグレータでのソフトウェアコンポーネント設定

3) “DMA コントローラ”ドライバをプロジェクトに追加します。
   - “フィルタ”ボックスに“DMA”と入力します。
   - コンポーネントリストを開き、“DMA コントローラ”ドライバを選択します。
   - [次へ]をクリックします。

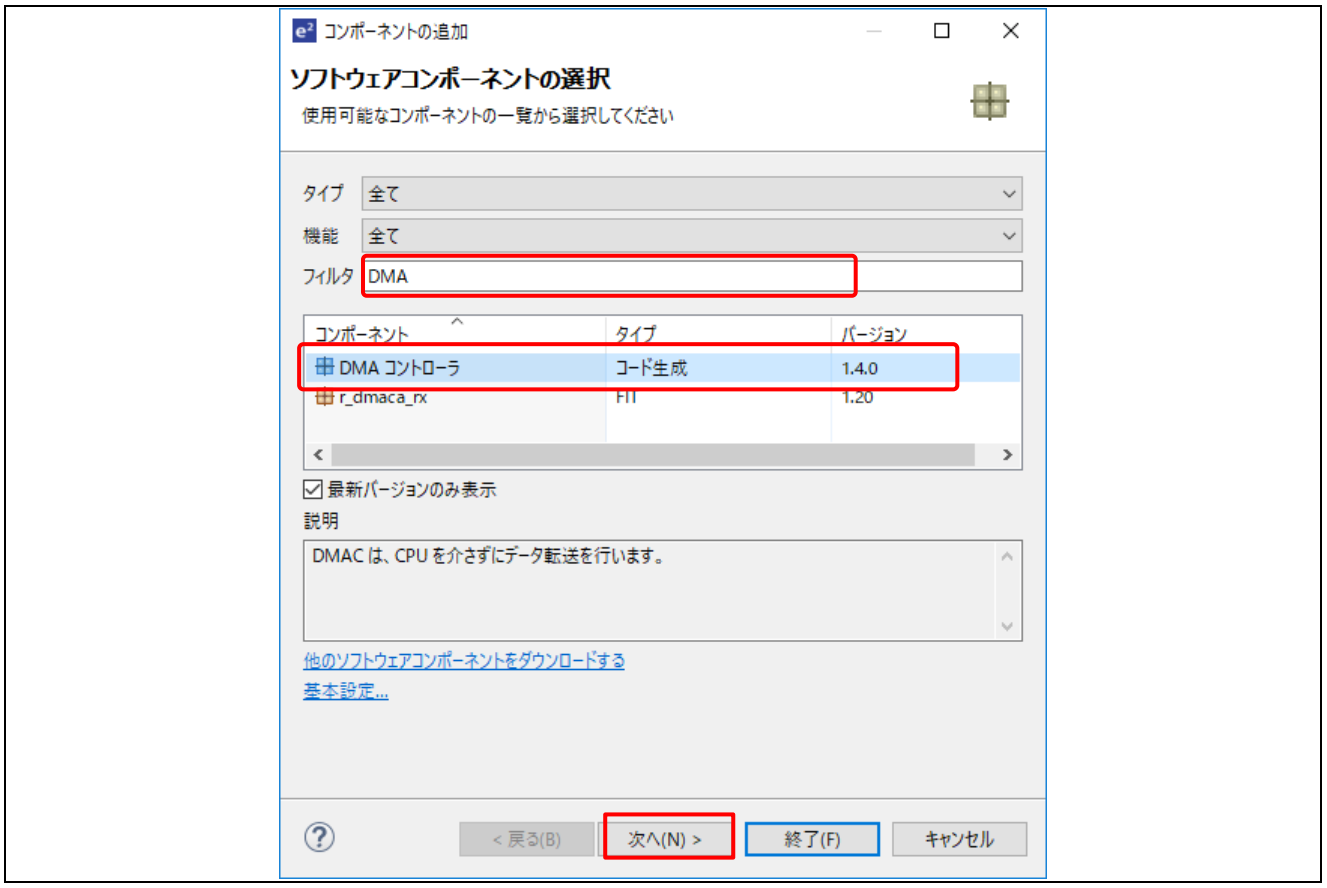


図 5-11　ソフトウェアコンポーネントの選択

4) [コンポーネントの追加]ダイアログで、以下を行います。

- DMAC1 をリソースとして選択します。
- デフォルトのコンフィグレーション名のままにします。このコンフィグレーション名を元に、ソースファイルと API が作成されます。
- [終了]をクリックします。

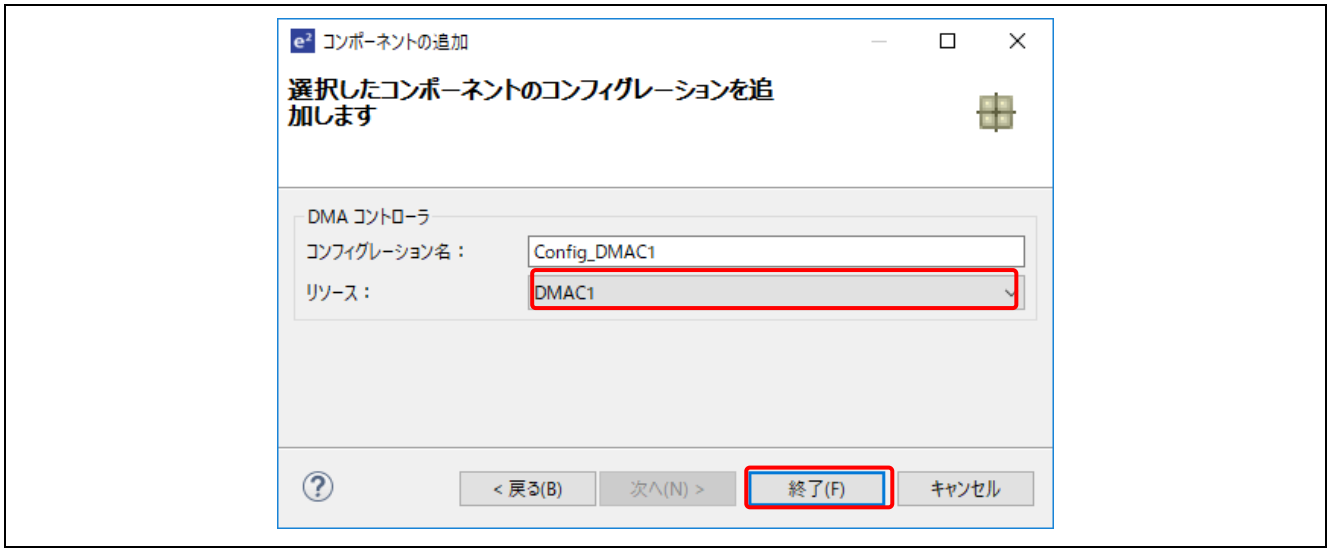


図 5-12　プロジェクトに新しいコンフィグレーションを追加する

5) 設定画面で、下記のように設定します。

- 起動要因：ソフトウェア起動
- 他のデフォルト設定はそのままにします。

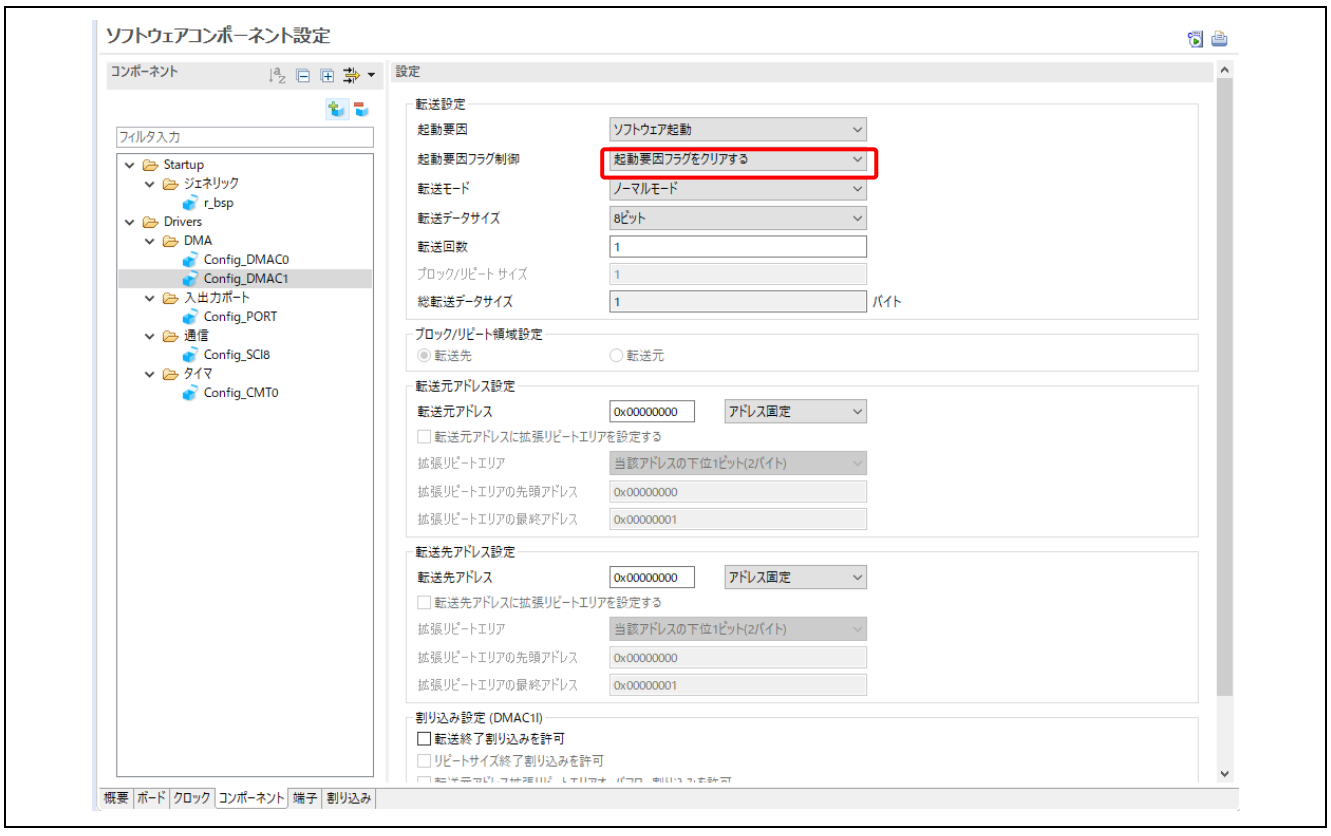


図 5-13　DMAC1 ドライバ設定

5.1.3　SCI8 ドライバの修正

1)　Smart_Configurator_Example 画面で、[コンポーネント]タブをクリックします。

2)　[コンポーネント]パネルで Config_SCI8 を選択します。

3)　設定画面で、以下のように設定します。

- 送信データ処理：DMAC で処理する
- 受信データ処理：DMAC で処理する
- 他の設定はそのままにします。

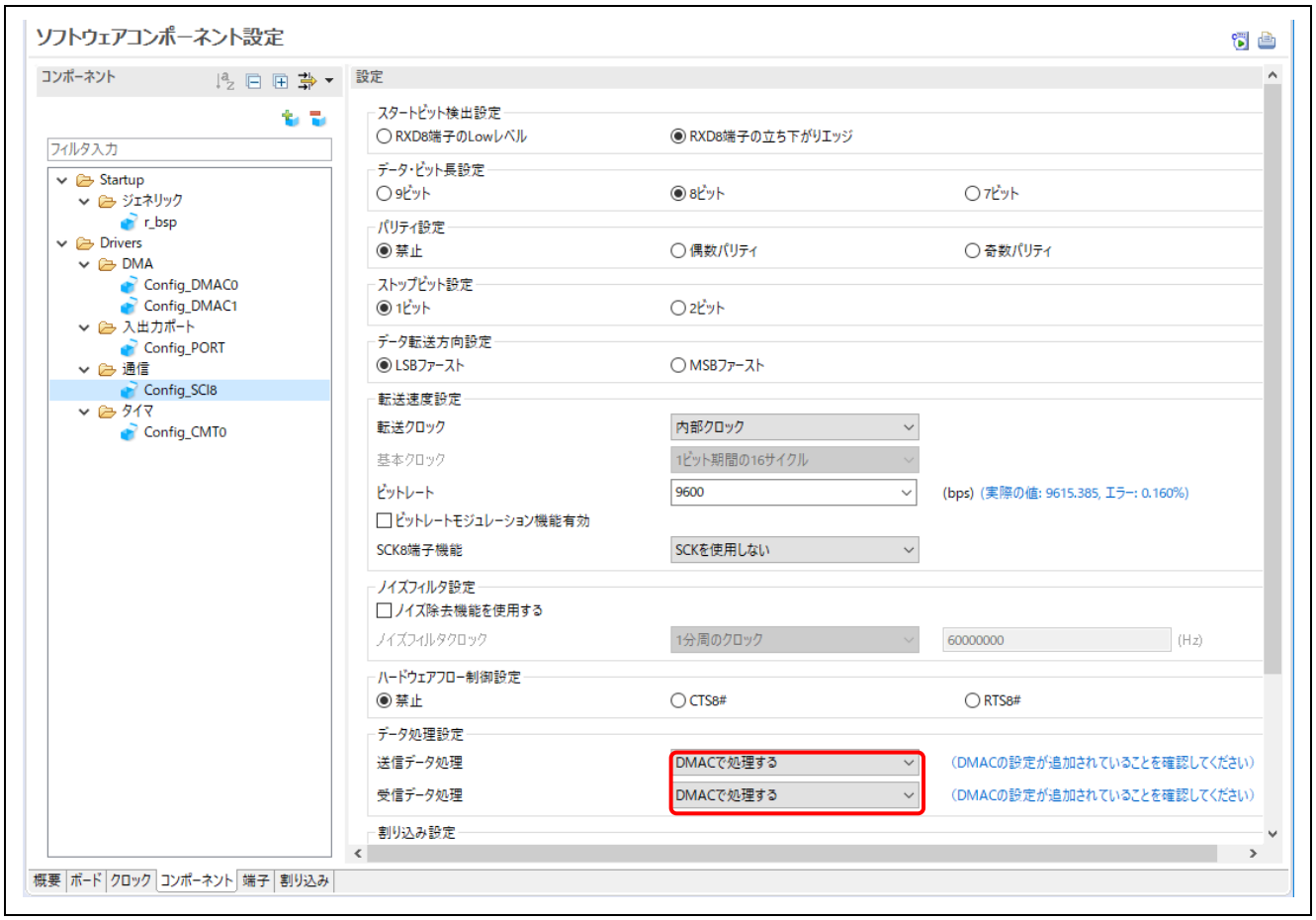


図 5-14　SCI8 ドライバ設定

5.2　コード生成

[ソフトウェアコンポーネント設定]タブで  をクリックし、コードを生成します。

5.3　DMAC0 ドライバにアプリケーションコードを追加

プロジェクト・エクスプローラー・ツリーで、\src\smc_gen\Config_DMAC0 フォルダにある”Config_DMAC0_user.c”を開きます。

a.　ファイルの先頭近くにあるユーザコードエリアに、以下の宣言を追加します。

```
/* Global variable used to receive a character from SCI8 */
volatile uint8_t g_rx_char_dmac;

/* Flag used to detect whether data is received */
volatile uint8_t g_rx_flag_dmac;
```

b.　受信完了フラグと、R_Config_DMAC0_Create_UserInit(void)関数内にある DMAC0 転送元アドレスと DMAC0 転送先アドレスの初期値を設定するコードを追加します。

```
/* Clear receive flag */
g_rx_flag_dmac = 0U;

/* Set DMAC0 source address */
DMAC0.DMSAR = (void *)&SCI8.RDR;

/* Set DMAC0 destination address */
DMAC0.DMDAR = (void *)&g_rx_char_dmac;
```

c.　r_dmac0_callback_transfer_end(void)関数で、以下を行います。

i.　受信完了フラグをセットします。

ii.　転送カウントをセットし、DMAC0 転送を有効にし、受信を再開します。

```
/* Set receive completion flag */
g_rx_flag_dmac = 1U;

/* Set DMAC0 transfer count */
DMAC0.DMCRA = _00000001_DMAC0_DMCRA_COUNT;

/* Enable DMAC0 transfer */
R_Config_DMAC0_Start();
```

ソースファイルは以下のようになります。

```
2          * DISCLAIMER
19
21         * File Name     : Config_DMAC0_user.c
27
29         Pragma directive
31         /* Start user code for pragma. Do not edit comment generated here */
32          /* End user code. Do not edit comment generated here */
33
35         Includes
37          #include "r_cg_macrodriver.h"
38          #include "Config_DMAC0.h"
39         /* Start user code for include. Do not edit comment generated here */
40          /* End user code. Do not edit comment generated here */
41          #include "r_cg_userdefine.h"
42
44         Global variables and functions
46         /* Start user code for global. Do not edit comment generated here */
47          /* Global variable used to receive a character from SCI8 */
48         volatile uint8_t g_rx_char_dmac;
49
50          /* Flag used to detect whether data is received */
51         volatile uint8_t g_rx_flag_dmac;
52          /* End user code. Do not edit comment generated here */
53
55         * Function Name: R_Config_DMAC0_Create_UserInit
60
61         void R_Config_DMAC0_Create_UserInit(void)
62          {
63              /* Start user code for user init. Do not edit comment generated here */
64              /* Clear receive flag */
65              g_rx_flag_dmac = 0U;
66
67              /* Set DMAC0 source address */
68              DMAC0.DMSAR = (void *) &SCI8.RDR;
69
70              /* Set DMAC0 destination address */
71              DMAC0.DMDAR = (void *) &g_rx_char_dmac;
72              /* End user code. Do not edit comment generated here */
73          }
74
97         * Function Name: r_dmac0_callback_transfer_end
102
103        static void r_dmac0_callback_transfer_end(void)
104         {
105             /* Start user code for r_dmac0_callback_transfer_end. Do not edit comment generated here */
106             /* Set receive completion flag */
107             g_rx_flag_dmac = 1U;
108
109             /* Set DMAC0 transfer count */
110             DMAC0.DMCRA = _00000001_DMAC0_DMCRA_COUNT;
111
112             /* Enable DMAC0 transfer */
113             R_Config_DMAC0_Start();
114             /* End user code. Do not edit comment generated here */
115         }
116
117        /* Start user code for adding. Do not edit comment generated here */
118         /* End user code. Do not edit comment generated here */
```

図 5-15　Config_DMAC0_user.c

5.4　DMAC1 ドライバにアプリケーションコードを追加

プロジェクト・エクスプローラー・ツリーで、\src\smc_gen\Config_DMAC1 フォルダにある”Config_DMAC1_user.c”を開きます。

a.　ファイルの先頭近くにあるユーザコードエリアに、ヘッダファイルを追加します。

```
#include "Config_SCI8.h"
```

b.　R_Config_DMAC1_Create_UserInit(void)関数内にある、DMAC1 転送先アドレスの初期値を設定するコードを追加します。

```
/* Set DMAC1 destination address */
DMAC1.DMDAR = (void *)&SCI8.TDR;
```

c. データを SCI8 へ転送したあと SCI8 関数を呼び出して PC にデータを送信する関数を、ファイルの最後にあるユーザコードエリアに追加します。

```
/***********************************************************************
*********************************************
* Function Name: R_DMAC1_AsyncTransmit
* Description  : This function transfer data to SCI8 and call SCI8
function to send data to PC.
* Arguments    : tx_buf -
*                    transfer buffer pointer
*                tx_num -
*                    buffer size
* Return Value : status -
*                    MD_OK or MD_ARGERROR
***********************************************************************
*********************************************/
MD_STATUS R_DMAC1_AsyncTransmit(uint8_t * const tx_buf, const uint16_t
tx_num)
{
    MD_STATUS status = MD_OK;

    uint8_t * source_address = tx_buf;

      // Set DMAC1 transfer count
      DMAC1.DMCRA = tx_num;

      // Enables DMA transfer
      R_Config_DMAC1_Start();

      while(DMAC1.DMCRA > 0)
      {
            // Set DMAC1 source address
            DMAC1.DMSAR = (void *) source_address;
            source_address++;

            // DMA transfer request
            R_Config_DMAC1_Set_SoftwareTrigger();

            // Sends SCI8 data
            R_SCI8_AsyncTransmit(NULL, 1);
      }

      return (status);
}
/***********************************************************************
*********
* End of function R_DMAC1_AsyncTransmit
      ***********************************************************************
**************/
```

ソースファイルは、以下のようになります。

Config_DMAC1_user.c

```
 2  * DISCLAIMER
19
21  * File Name     : Config_DMAC1_user.c
27
29  Pragma directive
31  /* Start user code for pragma. Do not edit comment generated here */
32  /* End user code. Do not edit comment generated here */
33
35  Includes
37  #include "r_cg_macrodriver.h"
38  #include "Config_DMAC1.h"
39  /* Start user code for include. Do not edit comment generated here */
40  #include "Config_SCI8.h"
41  /* End user code. Do not edit comment generated here */
42  #include "r_cg_userdefine.h"
43
45  Global variables and functions
47  /* Start user code for global. Do not edit comment generated here */
48  /* End user code. Do not edit comment generated here */
49
51  * Function Name: R_Config_DMAC1_Create_UserInit
56
57  void R_Config_DMAC1_Create_UserInit(void)
58  {
59      /* Start user code for user init. Do not edit comment generated here */
60      /* Set DMAC1 destination address */
61      DMAC1.DMDAR = (void *)&SCI8.TDR;
62      /* End user code. Do not edit comment generated here */
63  }
64
65  /* Start user code for adding. Do not edit comment generated here */
66  /***********************************************************************************************************************
67  * Function Name: R_DMAC1_AsyncTransmit
68  * Description  : This function transfer data to SCI8 and call SCI8 function to send data to PC.
69  * Arguments    : tx_buf -
70  *                    transfer buffer pointer
71  *                tx_num -
72  *                    buffer size
73  * Return Value : status -
74  *                    MD_OK or MD_ARGERROR
75  ***********************************************************************************************************************/
76  MD_STATUS R_DMAC1_AsyncTransmit(uint8_t * const tx_buf, const uint16_t tx_num)
77  {
78      MD_STATUS status = MD_OK;
79
80      uint8_t * source_address = tx_buf;
81
82      // Set DMAC1 transfer count
83      DMAC1.DMCRA = tx_num;
84
85      // Enables DMA transfer
86      R_Config_DMAC1_Start();
87
88      while(DMAC1.DMCRA > 0)
89      {
90          // Set DMAC1 source address
91          DMAC1.DMSAR = (void *) source_address;
92          source_address++;
93
94          // DMA transfer request
95          R_Config_DMAC1_Set_SoftwareTrigger();
96
97          // Sends SCI8 data
98          R_SCI8_AsyncTransmit(NULL, 1);
99      }
100
101     return (status);
102 }
103 /***********************************************************************************************************************
104 * End of function R_DMAC1_AsyncTransmit
105 ***********************************************************************************************************************/
106 /* End user code. Do not edit comment generated here */
107
```

図 5-16　Config_DMAC1_user.c

5.5　SCI8 ドライバのアプリケーションコードを修正

1) プロジェクト・エクスプローラー・ツリーで、\src\smc_gen\Config_SCI8 フォルダにある”Config_SCI8_user.c”を開きます。

a. ファイルの先頭近くのユーザコードエリアにある宣言”g_rx_char”と”g_rx_flag”を削除します。

```
/* Global variable used to receive a character from PC terminal */
volatile uint8_t g_rx_char;

/* Flag used to detect whether data is received */
volatile uint8_t g_rx_flag;

/* Flag used to detect completion of transmission */
static volatile uint8_t SCI8_txdone;
```

b. r_Config_SCI8_callback_receiveend(void)関数のコードを削除します。

```
/* Set receive completion flag */
g_rx_flag = 1U;

/* Set SCI8 receive buffer address and restart reception */
R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);
```

ソースファイルは、以下のようになります。

```
Config_SCI8_user.c
 2  * DISCLAIMER
19
21  * File Name    : Config_SCI8_user.c
27
29  Pragma directive
31  /* Start user code for pragma. Do not edit comment generated here */
32  /* End user code. Do not edit comment generated here */
33
35  Includes
37  #include "r_cg_macrodriver.h"
38  #include "Config_SCI8.h"
39  /* Start user code for include. Do not edit comment generated here */
40  /* End user code. Do not edit comment generated here */
41  #include "r_cg_userdefine.h"
42
44  Global variables and functions
46  /* Start user code for global. Do not edit comment generated here */
47  /* Flag used to detect completion of transmission */
48  static volatile uint8_t SCI8_txdone;
49  /* End user code. Do not edit comment generated here */
146 * Function Name: r_Config_SCI8_callback_receiveend
151
152 static void r_Config_SCI8_callback_receiveend(void)
153 {
154     /* Start user code for r_Config_SCI8_callback_receiveend. Do not edit comment generated h
155     /* End user code. Do not edit comment generated here */
156 }
157
```

図 5-17　Config_SCI8_user.c

5.6 *main()*にアプリケーションコードを追加

1) Smart_Configurator_Example.c ファイルの先頭近くに、SCI8 に使用される２つの外部宣言（g_rx_char と g_rx_flag）があり、Config_SCI8_user.c ファイルで定義されています。DMAC を SCI8 の代わりに使用するので、これらの宣言は削除してください。

```
/* Flag used to detect whether data is received from PC terminal */
extern volatile uint8_t g_rx_flag;

/* Global variable used for storing data received from PC terminal */
extern volatile uint8_t g_rx_char;
```

新しい外部宣言を、DMAC0 変数用に追加します。（5.3 章の DMAC Config_DMAC0_user.c に定義されています。）

```
/* Flag used to detect whether data is received from PC terminal */
extern volatile uint8_t g_rx_flag_dmac;

/* Global variable used for storing data received from PC terminal */
extern volatile uint8_t g_rx_char_dmac;
```

2) *main()* 関数内のすべてのコードを、以下のコードに置き換えてください。

```
            /* Start CMT0 counter operation */
            R_Config_CMT0_Start();

            /* Set SCI8 receive buffer address and enable receive interrupt */
            R_Config_SCI8_Serial_Receive(NULL, 0);

            /* Enable DMAC0 operation */
            R_Config_DMAC0_Start();

            /* Enable SCI8 operation */
            R_Config_SCI8_Start();

            /* Print instruction onto PC terminal */
            sprintf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not case sensitive)\r\n");
            R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
            sprintf(print_str, "\r\n a ----> Slower\n");
            R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
            sprintf(print_str, "\r\n b ----> Faster\n");
            R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

            while (1U)
            {
                        /* Get new blink interval level from PC terminal */
                        if (g_rx_flag_dmac == 1U)
                        {
                                    /* Instruction to slow down blinking rate is received */
                                    if (('A' == g_rx_char_dmac) || ('a' == g_rx_char_dmac))
                                    {
                                                R_Config_SCI8_Serial_Receive(NULL, 0);

                                                /* Notify the character received and inform next action */
                                                sprintf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink at slower
rate.\r\n");
                                                R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

                                                /* Get new blink interval level. Maximum level is 9. */
                                                if (interval_level < 10)
                                                {
                                                            interval_level++;
                                                }
                                                else
                                                {
                                                            sprintf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to blink faster.
\r\n");
                                                            R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
                                                }
                                    }
                                    /* Instruction to increase blinking rate is received */
                                    else if (('B' == g_rx_char_dmac) || ('b' == g_rx_char_dmac))
                                    {
                                                R_Config_SCI8_Serial_Receive(NULL, 0);

                                                /* Notify the character received and inform next action */
                                                sprintf(print_str, "\r\n Character 'b' or 'B' is received. LED2 will blink at faster
rate.\r\n");
                                                R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

                                                /* Get new blink interval level. Minimum level is 1 */
                                                if (interval_level > 1)
                                                {
                                                            interval_level--;
                                                }
                                                else
                                                {
                                                            sprintf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to blink slower.
\r\n");
                                                            R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
                                                }
                                    } else
                                    {
                                                /* Do nothing */
                                    }

                                    /* Change blinking rate */
                                    CMT0.CMCOR = (uint16_t) (5000 * interval_level);

                                    /* Reset SCI8 reception flag*/
                                    g_rx_flag_dmac = 0U;
                        }
                        else
                        {
                                    /* Do nothing */
                        }

            }
```

ソースファイルは以下のようになります。

```
 3          *  FILE        : Smart_Configurator_Example.c
10          #include "r_smc_entry.h"
11          #include <stdio.h>
12          #include <string.h>
13
14          /* Global variable for changing CMT0 interval */
15          volatile uint16_t interval_level = 5;
16
17          /* String used to print message at PC terminal */
18          static char print_str[80];
19
20          /* Flag used to detect whether data is received from PC terminal */
21          extern volatile uint8_t g_rx_flag_dmac;
22
23          /* Global variable used for storing data received from PC terminal */
24          extern volatile uint8_t g_rx_char_dmac;
25
26          void main(void);
27
28          void main(void)
29          {
30              /* Start CMT0 counter operation */
31              R_Config_CMT0_Start();
32
33              /* Set SCI8 receive buffer address and enable receive interrupt */
34              R_Config_SCI8_Serial_Receive(NULL, 0);
35
36              /* Enable DMAC0 operation */
37              R_Config_DMAC0_Start();
38
39              /* Enable SCI8 operation */
40              R_Config_SCI8_Start();
41
42              /* Print instruction onto PC terminal */
43              sprintf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not case sensitive)\r\n");
44              R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
45              sprintf(print_str, "\r\n a ----> Slower\n");
46              R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
47              sprintf(print_str, "\r\n b ----> Faster\n");
48              R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
49
50              while (1U)
51              {
52                  /* Get new blink interval level from PC terminal */
53                  if (g_rx_flag_dmac == 1U)
54                  {
55                      /* Instruction to slow down blinking rate is received */
56                      if (('A' == g_rx_char_dmac) || ('a' == g_rx_char_dmac))
57                      {
58                          R_Config_SCI8_Serial_Receive(NULL, 0);
59
60                          /* Notify the character received and inform next action */
61                          sprintf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink at slower rate.\r\n");
62                          R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
63
64                          /* Get new blink interval level. Maximum level is 9. */
65                          if (interval_level < 10)
66                          {
67                              interval_level++;
68                          }
69                          else
70                          {
71                              sprintf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to blink faster. \r\n");
72                              R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
73                          }
74                      }
75                      /* Instruction to increase blinking rate is received */
76                      else if (('B' == g_rx_char_dmac) || ('b' == g_rx_char_dmac))
77                      {
78                          R_Config_SCI8_Serial_Receive(NULL, 0);
79
80                          /* Notify the character received and inform next action */
81                          sprintf(print_str, "\r\n Character 'b' or 'B' is received. LED2 will blink at faster rate.\r\n");
82                          R_DMAC1_AsyncTransmit((uint8_t *) print_str,    (uint16_t) strlen(print_str));
83
84                          /* Get new blink interval level. Minimum level is 1 */
85                          if (interval_level > 1)
86                          {
87                              interval_level--;
88                          }
89                          else
90                          {
91                              sprintf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to blink slower. \r\n");
92                              R_DMAC1_AsyncTransmit((uint8_t *) print_str,    (uint16_t) strlen(print_str));
93                          }
94                      } else
95                      {
96                          /* Do nothing */
97                      }
98
99                      /* Change blinking rate */
00                      CMT0.CMCOR = (uint16_t) (5000 * interval_level);
01
02                      /* Reset SCI8 reception flag*/
03                      g_rx_flag_dmac = 0U;
04                  }
05                  else
06                  {
07                      /* Do nothing */
08                  }
09              }
10          }
11
```

5.7　ビルドとハードウェアボードでの実行

4.6 章を参照してください。

5.8　ボードでの動作確認

4.7 章を参照してください。

6. Application Example 5（スマート・コンフィグレータを使った USB Function を含む FIT モジュールの組み込み）

前の章では、シリアル通信（調歩同期式モード）を使用して、LED2 点滅周期を制御しました。

本章では、USB シリアル通信による PC でのターミナルプログラムを使い、LED2 点滅周期を制御する方法を説明します。ここでは、RX65N 上のシリアル COM ポートをエミュレートして PC と通信するため、2 つの FIT モジュールをプロジェクトに組み込みます。

R01AN2025：USB Basic Host and Peripheral Driver
R01AN2030：USB Peripheral Communications Device Class Driver (PCDC)

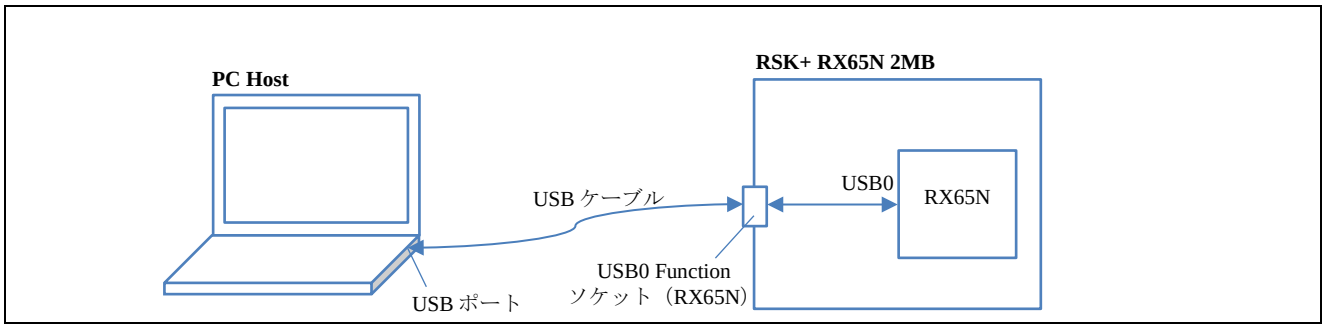


図 6-1　USB シリアル通信を経由したターミナルプログラムによる LED 点滅周期制御

RX65N 2MB 用 Renesas Starter Kit+の回路図を以下に示します。ここでは、USB0 function で使用するので USB0_VBUS (P16)を使用します。J7 の端子 2 と端子 3 がショートしていることを確認してください。（J7 の位置については、6.8 章を参照してください）

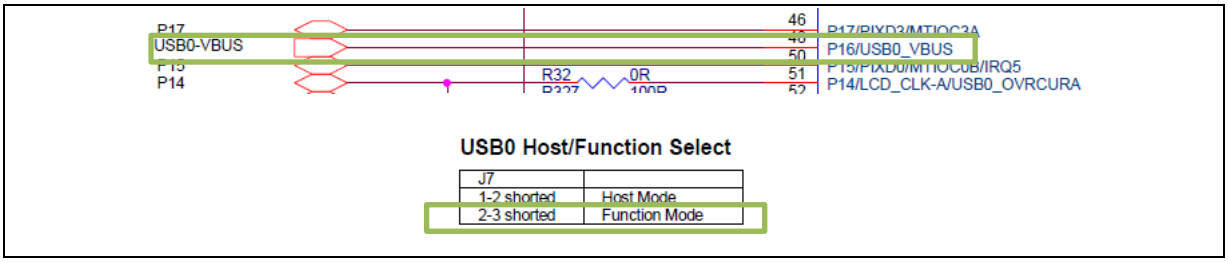


図 6-2　RX65N 2MB 用 Renesas Starter Kit+の回路図

ドライバの選択の変更を以下の表に示します。

	ドライバ	リソース	機能
1.ドライバの削除	SCI/SCIF 調歩同期式モード	SCI8	LED 点滅周期を制御するための、PC ターミナルとターゲットボード間の通信
	DMA コントローラ	DMAC0	SCI8 から RAM へのデータ転送（受信）
		DMAC1	RAM から SCI8 へのデータ転送（送信）
2.新ドライバの追加	r_usb_basic	USB0	USB Basic Host and Peripheral Firmware USB ハードウェア制御を行います。
	r_usb_pcdc	USB0	USB Peripheral Communications Device Class Driver (PCDC)として動作する USB host との通信を有効にする

USB PCDC モードアプリケーションコードは、以下のようにプログラムに追加されます。

a)　main 関数：

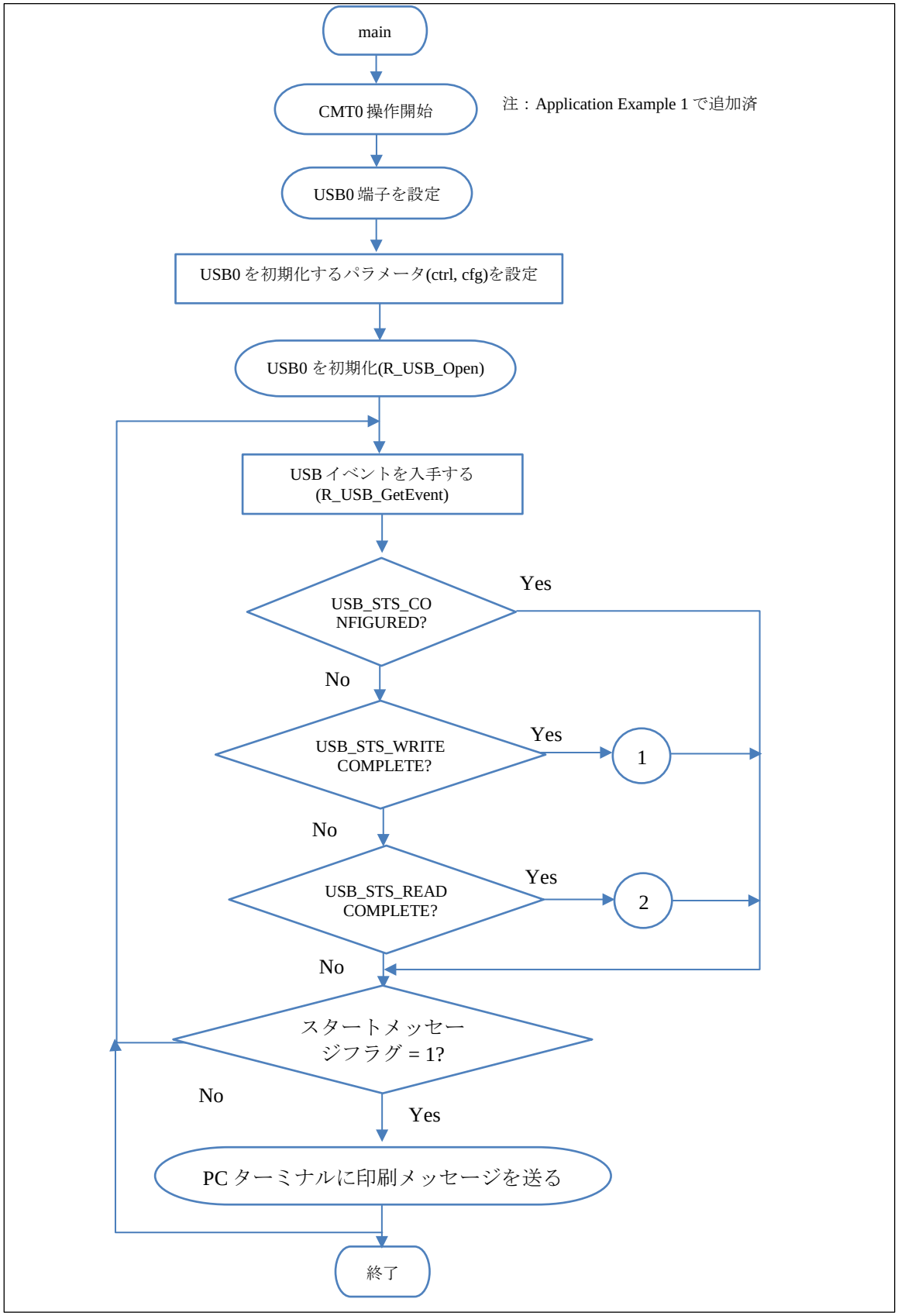


図 6-3　main フローチャート

b)　USB_STS_WRITE_COMPLETE が検出された場合、以下の処理を行います。



図 6-4　USB_STS_WRITE_COMPLETE が検出されたときの処理

c)　USB_STS_READ_COMPLETE が検出されたときの処理：

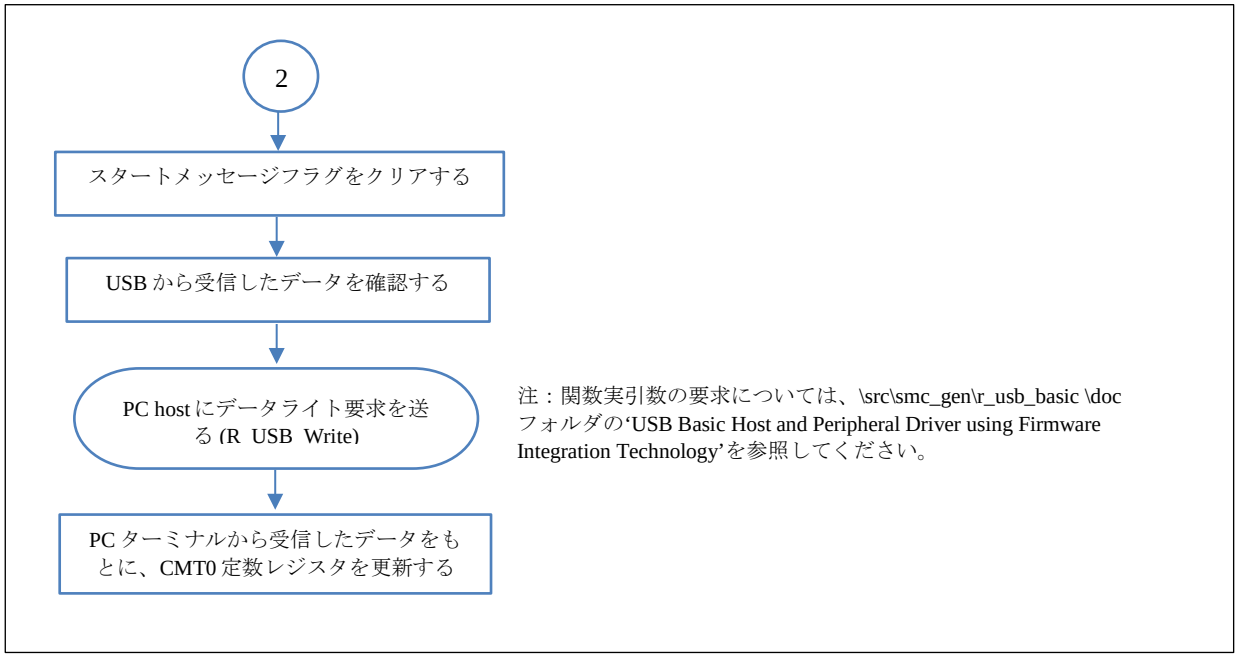


図 6-5　USB_STS_READ_COMPLETE が検出されたときの処理

6.1　SCI/SCIF 調歩同期式モードドライバと関連するアプリケーションコードの削除

1) Smart_Configurator_Example.scfg 画面で、[コンポーネント]タブをクリックします。
2) コンポーネントツリーで、“Config_SCI8”コンフィグレーションをクリックして選択します。


をクリックして、このコンフィグレーションを削除します。同様に、“Config_DMAC0”と“Config_DMAC1”も削除します。



図 6-6　スマート・コンフィグレータでのソフトウェアコンポーネント設定

3) 
をクリックして、コードを生成します。
コード生成操作時に、ソースフォルダ “Config_SCI8”、“Config_DMAC0”、“Config_DMAC1”はプロジェクト\trash フォルダに移動します。
4) プロジェクトツリーで、\src フォルダにある“Smart_Configurator_Example.c”を開きます。

5) SCI8、DMAC0、DMAC1 に関連するアプリケーションコードを削除します。

a. 変数の宣言を削除します。

```
#include <stdio.h>
#include <string.h>

/* Global variable used for changing CMT0 interval */
volatile uint16_t interval_level = 5;

/* String used to print message at PC terminal */
static char print_str[70];

/* Flag used to detect whether data is received from PC terminal */
extern volatile uint8_t g_rx_flag_dmac;

/* Global variable used for storing data received from PC terminal */
extern volatile uint8_t g_rx_char_dmac;
```

b. *R_Config_CMT0_Start* 関数の後のすべてのコードを削除します。

SCI8 に関連するコードを削除した後の main のファイルは以下のようになります。

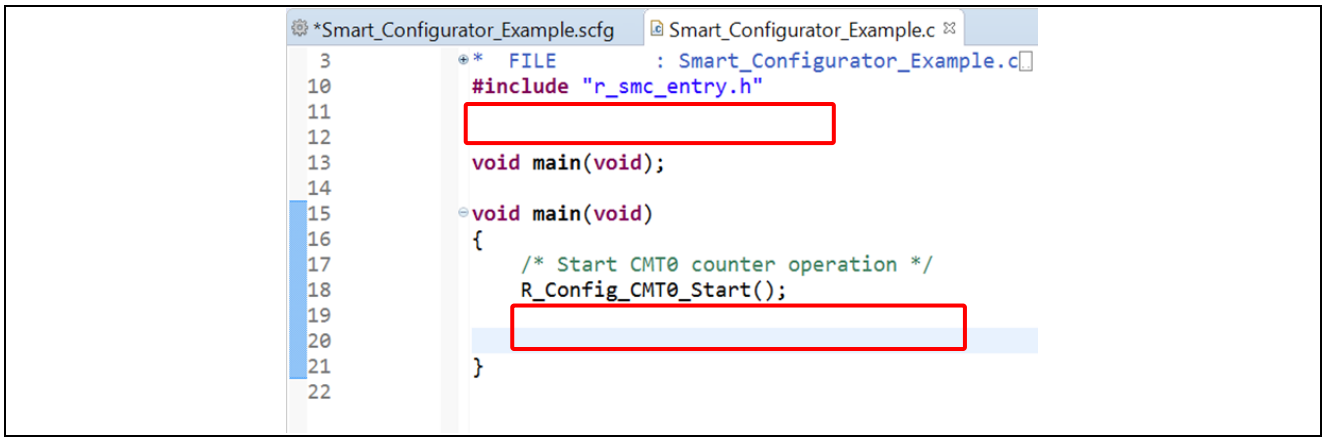


図 6-7　SCI8 に関連するコードを削除した後の main ファイル

6.2　ペリフェラルドライバの追加

1) Smart_Configurator_Example.scfg 画面で、[コンポーネント]タブをクリックします。
2) をクリックし、新しいコンポーネントを追加します。

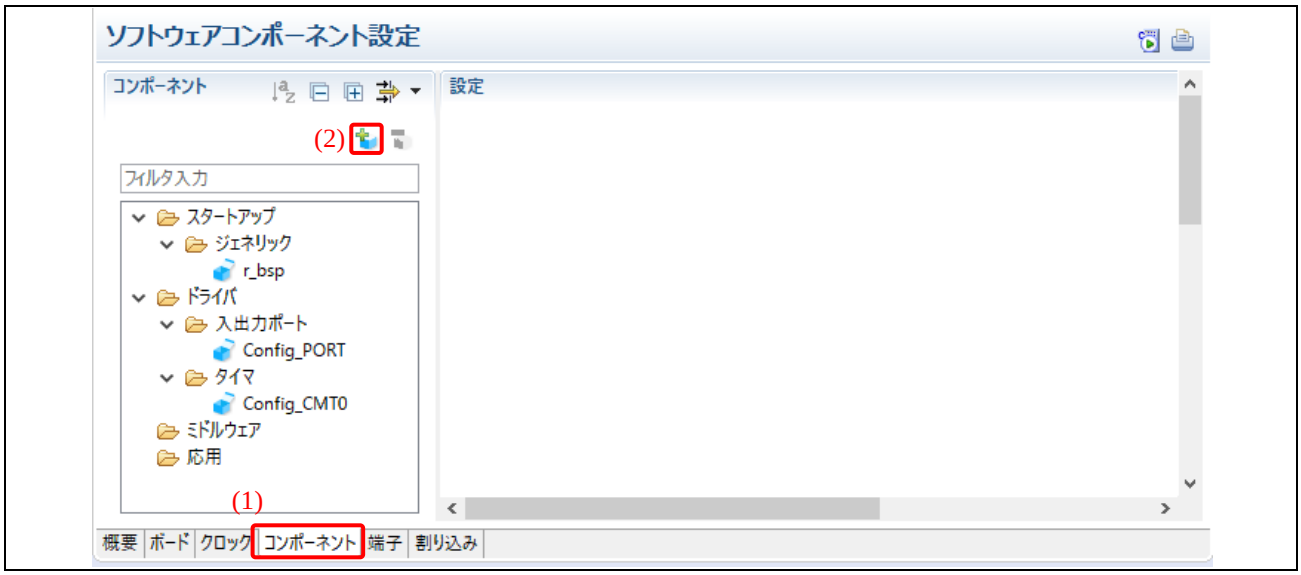


図 6-8　スマート・コンフィグレータでのソフトウェアコンポーネント設定

3) プロジェクトに“USB Basic”ドライバと“USB PCDC”ドライバを追加します。
   a. “フィルタ”欄に“usb”と入力します。
   b. コンポーネントリストを開き、“r_usb_basic”ドライバを選択します。
   c. Ctrl キーを押したまま、“r_usb_pcdc”ドライバをクリックします。
   注：上記のドライバがリストにない場合、[他のソフトウェアコンポーネントをダウンロードする]をクリックし、FIT モジュールをダウンロードします。
   d. [終了]をクリックします。

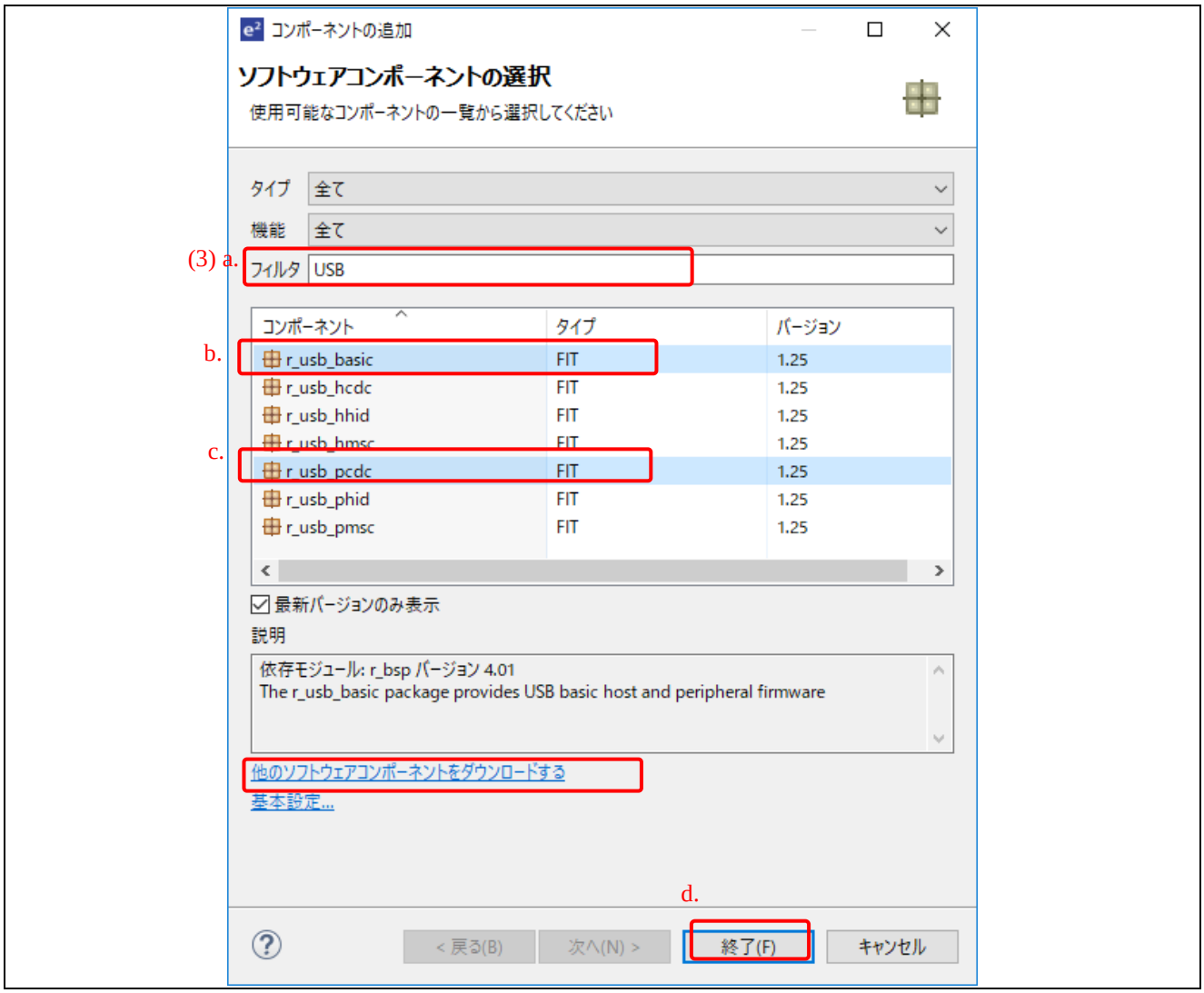


図 6-9　ソフトウェアコンポーネントの選択

4)　コンポーネントのページで、以下を行います。

a.　*r_usb_basic* をクリックし、設定タブを開きます。

b.　“Configurations”の下にある[Setting whether the received class request is supported.]を“Not supporting the class request”に設定します。

c.　[USB0_PERI]と[USB0_VBUS 端子]のチェックボックスにチェックを入れます。

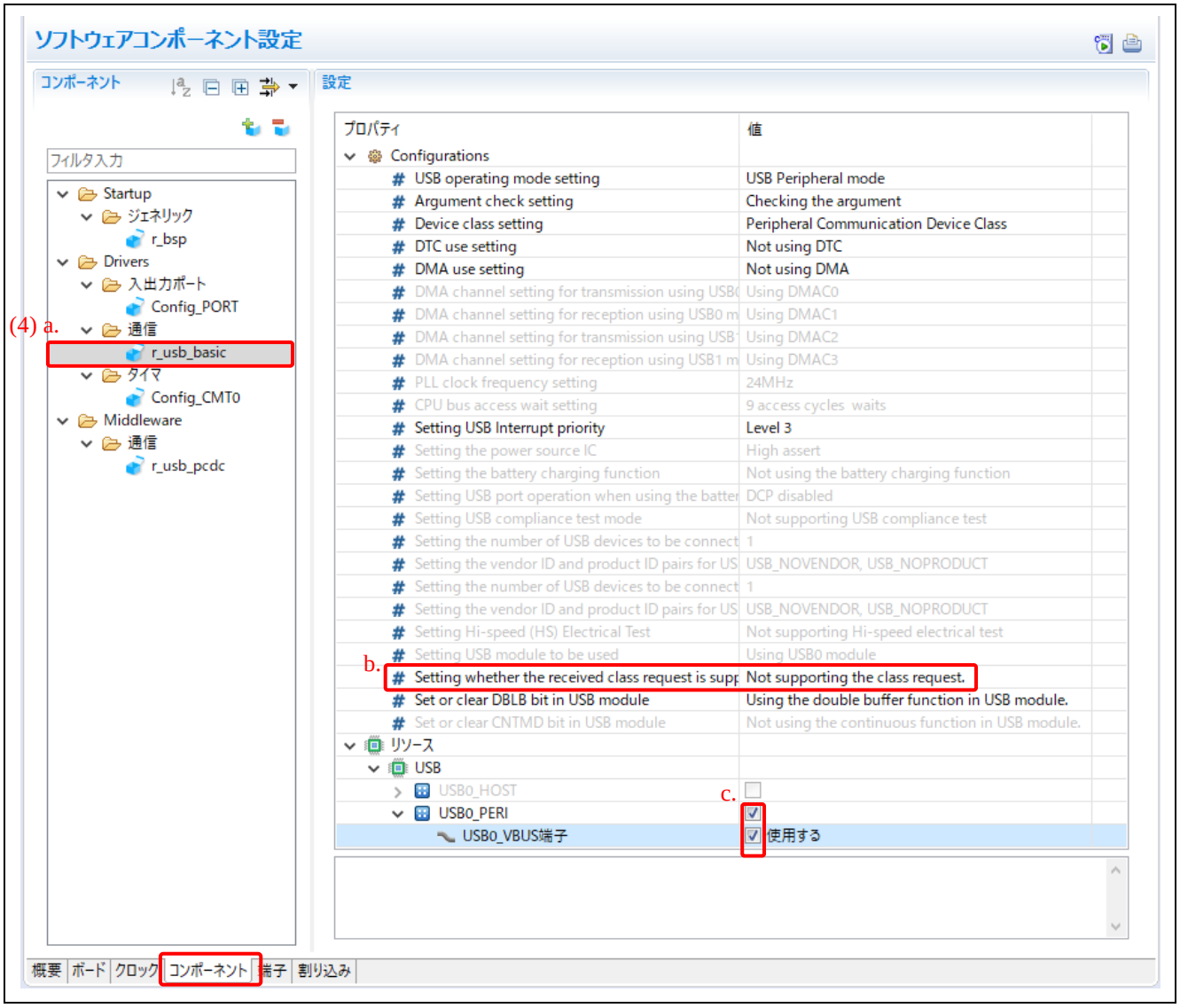


図 6-10　ソフトウェアコンポーネントの設定

5)　端子のページで、以下を行います。

a.　をクリックします。

b.　*r_usb_basic* をクリックし、対応する端子機能タブを開きます。

c.　[USB0_VBUS]のチェックボックスにチェックを入れます。

d.　その行の[端子割り当て]欄をクリックし、P16 を機能[USB0_VBUS]に割り当てます。

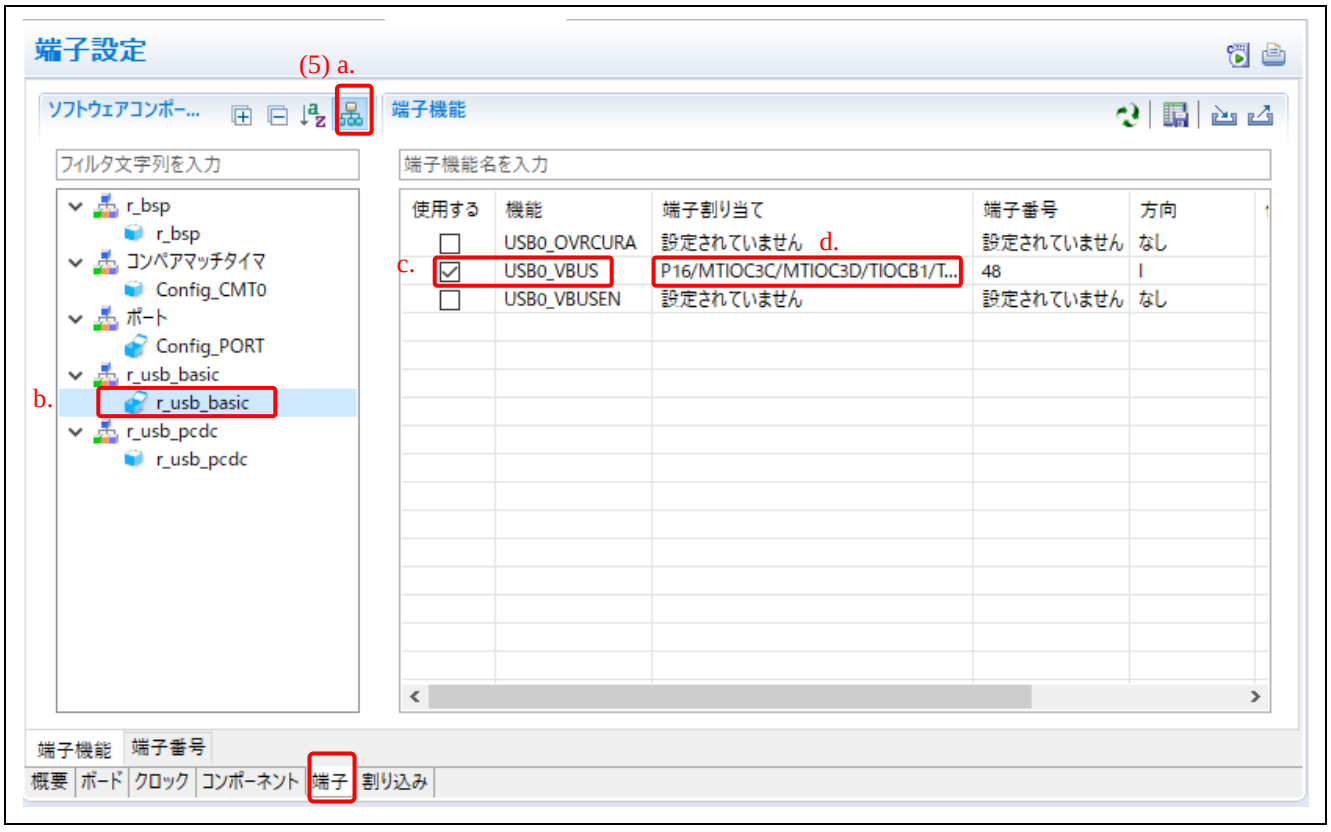


図 6-11　ソフトウェアコンポーネントの設定

6.3　コード生成

1) [ソフトウェアコンポーネント設定]タブで  をクリックし、コードを生成します。

2) *USB Basic Host*、*Peripheral Driver*、*USB PCDC Driver* コードは\src\smc_gen の下にある 3 つのフォルダに生成されます。

フォルダ	サブフォルダ/ ファイル	説明
r_usb_basic	**doc**	*USB Basic Host* と *Peripheral Driver* (*r_usb_basic*)のアプリケーションノート
	ref	コンフィグレーションファイルの参照
	src	ソースファイルとヘッダファイル
	r_usb_basic_if.h	すべての API 呼び出しと *r_usb_basic* インタフェース定義のリスト。 アプリケーションでの操作にこのファイルを参照します。
r_usb_pcdc	**doc**	*USB PCDC Driver (r_usb_pcdc)*のアプリケーションノート
	ref	コンフィグレーションファイルの参照
	src	ソースファイルとヘッダファイル
	utilities	システム定義ファイル(CDC_Demo_Win7.inf)を含みます。 Windows 7 PC host にこのドライバをインストールします。
	r_usb_pcdc_if.h	すべての API 呼び出しと *r_usb_pcdc* インタフェース定義のリスト。 アプリケーションでの操作にこのファイルを参照します。
r_config	*r_usb_basic_config.h*	*r_usb_basic* のコンフィグレーション アプリケーションの要求としてこのファイルを設定します。 このファイルを変更する前に、*r_usb_basic* のアプリケーションノートを参照してください。
	r_usb_pcdc_config.h	*r_usb_pcdc* のコンフィグレーション アプリケーションの要求としてこのファイルを設定します。 このファイルを変更する前に、*r_usb_basic* と *r_usb_pcdc* のアプリケーションノートを参照してください。

3) この Application Example では、

   a. USB ドライバは Peripheral モード（デバイス）に設定します。

   b. ヘッダファイル r_usb_pcdc_descriptor.h をディスクリプタの構成用に作成します。

   c. API 関数：*R_USB_Open, R_USB_Read, R_USB_Write* をアプリケーションコードで呼び出します。

   d. USB ドライバ状態：*USB_STS_WRITE_COMPLETE, USB_STS_READ_COMPLETE* をアプリケーションコードでチェックし操作します。

6.4　USB PCDC ドライバのヘッダファイルの追加

1) プロジェクト・エクスプローラー・ツリーで、[src]フォルダを右クリックし、[新規]→[ヘッダファイル]を選択します。

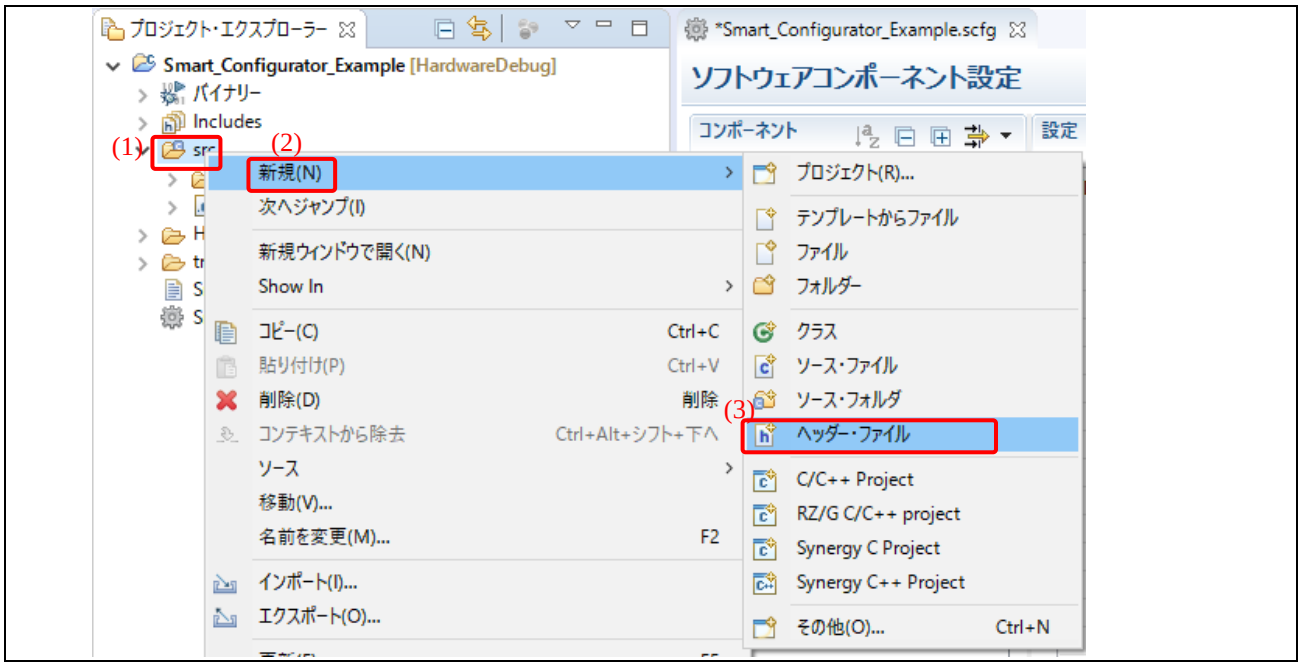


図 6-12　ヘッダファイルの追加

2) ヘッダファイル名（例 *r_usb_pcdc_descriptor.h*）を入力し、[終了]をクリックします。

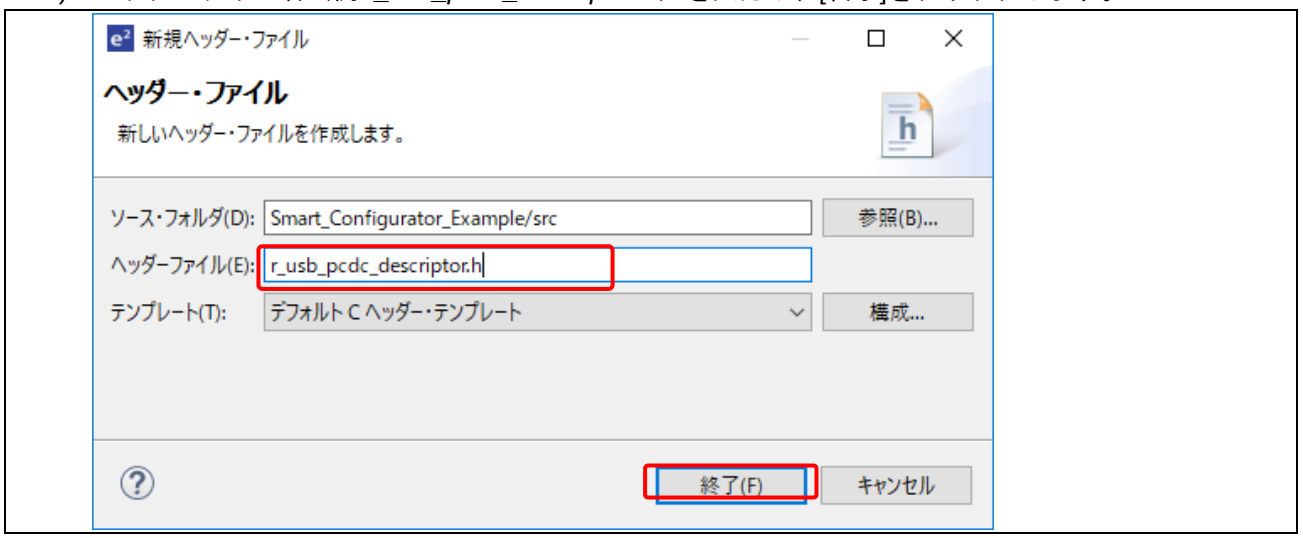


図 6-13　ヘッダファイルの追加

3) \src フォルダにある“r_usb_pcdc_descriptor.h”ファイルを開きます。

USB 2.0 仕様に関する詳細は、\src\smc_gen\r_usb_basic \doc フォルダの下にある‘USB Basic Host and Peripheral Driver using Firmware Integration Technology’のアプリケーションノートや、USB-IF ウェブサイト(http://www.usb.org)を参照してください。

以下のコードを“r_usb_pcdc_descriptor.h”に入力し、この Application Example 用のディスクリプタを構成してください。

注：デバイスディスクリプタと INF ファイルに、ベンダーID と製品 ID を必ず指定してください。

```
#include "r_usb_basic_if.h"
/******************************************************************************
 Macro definitions
 ******************************************************************************/
/* bcdUSB */
#define USB_BCDNUM       (0x0200u)       /* 0x0100 (USB1.0), 0x0110 (USB1.1), 0x0200 (USB2.0) */
/* Release Number */
#define USB_RELEASE      (0x0200u)       /* 0x0100 (USB1.0), 0x0110 (USB1.1), 0x0200 (USB2.0) */
/* DCP max packet size */
#define USB_DCPMAXP      (64u)           /* Max packet size for endpoint 0. 8/16/32/64 */
/* Configuration number */
#define USB_CONFIGNUM    (1u)            /* Total number of possible configurations for the device */
/* Vendor ID */
#define USB_VENDORID     (0x0000u)       /* Must be obtained from USB-IF */
/* Product ID*/
#define USB_PRODUCTID    (0x0002u)       /* Assigned by the manufacturer */

/* Class-Specific Configuration Descriptors */
#define     USB_PCDC_CS_INTERFACE        (0x24u) /* Assigned by USB-IF*/

/* bDescriptor SubType in Communications Class Functional Descriptors */
/* Header Functional Descriptor */
#define     USB_PCDC_DT_SUBTYPE_HEADER_FUNC                     (0x00u) /* Assigned by USB-IF */
/* Call Management Functional Descriptor. */
#define     USB_PCDC_DT_SUBTYPE_CALL_MANAGE_FUNC                (0x01u) /* Assigned by USB-IF */
/* Abstract Control Management Functional Descriptor. */
#define     USB_PCDC_DT_SUBTYPE_ABSTRACT_CTR_MANAGE_FUNC        (0x02u) /* Assigned by USB-IF */
/* Union Functional Descriptor */
#define     USB_PCDC_DT_SUBTYPE_UNION_FUNC                      (0x06u) /* Assigned by USB-IF */

/* Communications Class Subclass Codes */
#define     USB_PCDC_CLASS_SUBCLASS_CODE_ABS_CTR_MDL     (0x02u) /*SubClass Code assigned by USB-IF */

/* USB Class Definitions for Communications Devices Specification
 release number in binary-coded decimal. */
#define     USB_PCDC_BCD_CDC             (0x0110u) /* Assigned by USB-IF */

/* Descriptor length */
#define USB_PCDC_DD_LEN                  (18u)
#define USB_PCDC_QD_LEN                  (10u)
#define USB_PCDC_CD1_LEN                 (67u)
#define STRING_DESCRIPTOR0_LEN           (4u)
#define STRING_DESCRIPTOR1_LEN           (16u)
#define STRING_DESCRIPTOR2_LEN           (8u)
#define STRING_DESCRIPTOR3_LEN           (22u)
#define STRING_DESCRIPTOR4_LEN           (22u)
#define STRING_DESCRIPTOR5_LEN           (18u)
#define STRING_DESCRIPTOR6_LEN           (28u)

/* Standard Device Descriptor */
uint8_t g_apl_device[USB_PCDC_DD_LEN + ( USB_PCDC_DD_LEN % 2)] =
{
    USB_PCDC_DD_LEN,                                    /* 0:bLength */
    USB_DT_DEVICE,                                      /* 1:bDescriptorType */
    (USB_BCDNUM & (uint8_t) 0xffu),                     /* 2:bcdUSB_lo */
    ((uint8_t) (USB_BCDNUM >> 8) & (uint8_t) 0xffu),    /* 3:bcdUSB_hi */
    USB_IFCLS_CDCC,                                     /* 4:bDeviceClass */
    0,                                                  /* 5:bDeviceSubClass */
    0,                                                  /* 6:bDeviceProtocol */
    (uint8_t) USB_DCPMAXP,                              /* 7:bMAXPacketSize(for DCP) */
    (USB_VENDORID & (uint8_t) 0xffu),                   /* 8:idVendor_lo */
    ((uint8_t) (USB_VENDORID >> 8) & (uint8_t) 0xffu),  /* 9:idVendor_hi */
    ((uint16_t) USB_PRODUCTID & (uint8_t) 0xffu),       /* 10:idProduct_lo */
    ((uint8_t) (USB_PRODUCTID >> 8) & (uint8_t) 0xffu), /* 11:idProduct_hi */
    (USB_RELEASE & (uint8_t) 0xffu),                    /* 12:bcdDevice_lo */
     ((uint8_t) (USB_RELEASE >> 8) & (uint8_t) 0xffu),  /* 13:bcdDevice_hi */
    1,                                                  /* 14:iManufacturer */
    2,                                                  /* 15:iProduct */
    6,                                                  /* 16:iSerialNumber */
    USB_CONFIGNUM                                       /* 17:bNumConfigurations */
};
```

```
/************************************************************
 *  Configuration or other Speed Configuration Descriptor   *
 ************************************************************/
/* For Full-Speed */
uint8_t g_apl_configuration[USB_PCDC_CD1_LEN + ( USB_PCDC_CD1_LEN % 2)] =
{
        9,                                              /* 0:bLength */
        USB_SOFT_CHANGE,                                /* 1:bDescriptorType */
        USB_PCDC_CD1_LEN % 256,                         /* 2:wTotalLength(L) */
        USB_PCDC_CD1_LEN / 256,                         /* 3:wTotalLength(H) */
        2,                                              /* 4:bNumInterfaces */
        1,                                              /* 5:bConfigurationValue */
        0,                                              /* 6:iConfiguration */
        USB_CF_RESERVED | USB_CF_SELFP,                 /* 7:bmAttributes */
        (10 / 2),                                       /* 8:MAXPower (2mA unit) */

        /* Interface Descriptor */
        9,                                              /* 0:bLength */
        USB_DT_INTERFACE,                               /* 1:bDescriptor */
        0,                                              /* 2:bInterfaceNumber */
        0,                                              /* 3:bAlternateSetting */
        1,                                              /* 4:bNumEndpoints */
        USB_IFCLS_CDCC,                                 /* 5:bInterfaceClass */
        USB_PCDC_CLASS_SUBCLASS_CODE_ABS_CTR_MDL,       /* 6:bInterfaceSubClass */
        1,                                              /* 7:bInterfaceProtocol */
        0,                                              /* 8:iInterface */

        /* Communications Class Functional Descriptors */
        5,                                              /* 0:bLength */
        USB_PCDC_CS_INTERFACE,                          /* 1:bDescriptorType */
        USB_PCDC_DT_SUBTYPE_HEADER_FUNC,                /* 2:bDescriptorSubtype */
        USB_PCDC_BCD_CDC % 256,                         /* 3:bcdCDC_lo */
        USB_PCDC_BCD_CDC / 256,                         /* 4:bcdCDC_hi */

        /* Communications Class Functional Descriptors */
        4,                                              /* 0:bLength */
        USB_PCDC_CS_INTERFACE,                          /* 1:bDescriptorType */
        USB_PCDC_DT_SUBTYPE_ABSTRACT_CTR_MANAGE_FUNC,   /* 2:bDescriptorSubtype */
        2,                                              /* 3:bmCapabilities */

        /* Communications Class Functional Descriptors */
        5,                                              /* 0:bLength */
        USB_PCDC_CS_INTERFACE,                          /* 1:bDescriptorType */
        USB_PCDC_DT_SUBTYPE_UNION_FUNC,                 /* 2:bDescriptorSubtype */
        0,                                              /* 3:bMasterInterface */
        1,                                              /* 4:bSlaveInterface0 */

        /* Communications Class Functional Descriptors */
        5,                                              /* 0:bLength */
        USB_PCDC_CS_INTERFACE,                          /* 1:bDescriptorType */
        USB_PCDC_DT_SUBTYPE_CALL_MANAGE_FUNC,           /* 2:bDescriptorSubtype */
        /* D1:1-Device can send/receive call management
         information over a Data Class interface. */
        /* D0:1-Device handles call management itself. */
        3,                                              /* 3:bmCapabilities */
        1,                                              /* 4:bDataInterface */

        /* Endpoint Descriptor 0 */
        7,                                              /* 0:bLength */
        USB_DT_ENDPOINT,                                /* 1:bDescriptorType */
        USB_EP_IN | USB_EP3,                            /* 2:bEndpointAddress */
        USB_EP_INT,                                     /* 3:bmAttribute */
        16,                                             /* 4:wMAXPacketSize_lo */
        0,                                              /* 5:wMAXPacketSize_hi */
        0x10,                                           /* 6:bInterval */
```

```
        /* Interface Descriptor */
        9,                                              /* 0:bLength */
        USB_DT_INTERFACE,                               /* 1:bDescriptor */
        1,                                              /* 2:bInterfaceNumber */
        0,                                              /* 3:bAlternateSetting */
        2,                                              /* 4:bNumEndpoints */
        USB_IFCLS_CDCD,                                 /* 5:bInterfaceClass */
        0,                                              /* 6:bInterfaceSubClass */
        0,                                              /* 7:bInterfaceProtocol */
        0,                                              /* 8:iInterface */

        /* Endpoint Descriptor 0 */
        7,                                              /* 0:bLength */
        USB_DT_ENDPOINT,                                /* 1:bDescriptorType */
        USB_EP_IN | USB_EP1,                            /* 2:bEndpointAddress */
        USB_EP_BULK,                                    /* 3:bmAttribute */
        64,                                             /* 4:wMAXPacketSize_lo */
        0,                                              /* 5:wMAXPacketSize_hi */
        0,                                              /* 6:bInterval */

        /* Endpoint Descriptor 1 */
        7,                                              /* 0:bLength */
        USB_DT_ENDPOINT,                                /* 1:bDescriptorType */
        USB_EP_OUT | USB_EP2,                           /* 2:bEndpointAddress */
        USB_EP_BULK,                                    /* 3:bmAttribute */
        64,                                             /* 4:wMAXPacketSize_lo */
        0,                                              /* 5:wMAXPacketSize_hi */
        0,                                              /* 6:bInterval */
};

/**********************************
 *    String Descriptor             *
 **********************************/
/* UNICODE 0x0409 English (United States) */
uint8_t g_cdc_string_descriptor0[STRING_DESCRIPTOR0_LEN + (STRING_DESCRIPTOR0_LEN % 2)] =
{
        STRING_DESCRIPTOR0_LEN,                         /* 0:bLength */
        USB_DT_STRING,                                  /* 1:bDescriptorType */
        0x09, 0x04                                      /* 2:wLANGID[0] */
};
/* iManufacturer */
uint8_t g_cdc_string_descriptor1[STRING_DESCRIPTOR1_LEN + (STRING_DESCRIPTOR1_LEN % 2)] =
{
        STRING_DESCRIPTOR1_LEN,                         /* 0:bLength */
        USB_DT_STRING,                                  /* 1:bDescriptorType */
        'R', 0x00,                                      /* 2:wLANGID[0] */
        'E', 0x00,
        'N', 0x00,
        'E', 0x00,
        'S', 0x00,
        'A', 0x00,
        'S', 0x00,
};

/* iProduct */
uint8_t g_cdc_string_descriptor2[STRING_DESCRIPTOR2_LEN + (STRING_DESCRIPTOR2_LEN % 2)] =
{
        STRING_DESCRIPTOR2_LEN,                         /* 0:bLength */
        USB_DT_STRING,                                  /* 1:bDescriptorType */
        'U', 0x00,
        'S', 0x00,
        'B', 0x00,
};
```

```
/* iInterface */
uint8_t g_cdc_string_descriptor3[STRING_DESCRIPTOR3_LEN + (STRING_DESCRIPTOR3_LEN % 2)] =
{
        STRING_DESCRIPTOR3_LEN,                         /* 0:bLength */
        USB_DT_STRING,                                  /* 1:bDescriptorType */
        'C', 0x00,
        'o', 0x00,
        'm', 0x00,
        ' ', 0x00,
        'D', 0x00,
        'e', 0x00,
        'v', 0x00,
        'i', 0x00,
        'c', 0x00,
        'e', 0x00,
};

/* iConfiguration */
uint8_t g_cdc_string_descriptor4[STRING_DESCRIPTOR4_LEN + (STRING_DESCRIPTOR4_LEN % 2)] =
{
        STRING_DESCRIPTOR4_LEN,                         /* 0:bLength */
        USB_DT_STRING,                                  /* 1:bDescriptorType */
        'F', 0x00,                                      /* 2:wLANGID[0] */
        'u', 0x00,
        'l', 0x00,
        'l', 0x00,
        '-', 0x00,
        'S', 0x00,
        'p', 0x00,
        'e', 0x00,
        'e', 0x00,
        'd', 0x00
};


uint8_t *g_apl_string_table[] =
{
        g_cdc_string_descriptor0,
        g_cdc_string_descriptor1,
        g_cdc_string_descriptor2,
        g_cdc_string_descriptor3,
        g_cdc_string_descriptor4,
};

/***********************************************************************************
 Renesas Abstracted Peripheral Communications Devices Class Driver API functions
 ***********************************************************************************/
```

ソースファイルは以下のようになります。

```
 * r_usb_pcdc_descriptor.h

#ifndef R_USB_PCDC_DESCRIPTOR_H_
#define R_USB_PCDC_DESCRIPTOR_H_
#include "r_usb_basic_if.h"
/******************************************************************************
 Macro definitions
 ******************************************************************************/
/* bcdUSB */
#define USB_BCDNUM                      (0x0200u)/* 0x0100 (USB1.0), 0x0110 (USB1.1), 0x0200 (USB2.0)*/
/* Release Number */
#define USB_RELEASE                     (0x0200u)/* 0x0100 (USB1.0), 0x0110 (USB1.1), 0x0200 (USB2.0)*/
/* DCP max packet size */
#define USB_DCPMAXP                     (64u)  /*Max packet size for endpoint 0.Must be 8, 16, 32 or 64*/
/* Configuration number */
#define USB_CONFIGNUM                   (1u)/*Specifies the total number of possible configurations for the device.*/
/* Vendor ID */
#define USB_VENDORID                    (0x0000u)/*must be obtained from USB-IF*/
/* Product ID*/
#define USB_PRODUCTID                   (0x0002u)/*assigned by the manufacturer */

/* Class-Specific Configuration Descriptors */
#define     USB_PCDC_CS_INTERFACE                               (0x24u) /*assigned from USB-IF*/

/* bDescriptor SubType in Communications Class Functional Descriptors */
/* Header Functional Descriptor */
#define     USB_PCDC_DT_SUBTYPE_HEADER_FUNC                     (0x00u)/*assigned from USB-IF*/
/* Call Management Functional Descriptor. */
#define     USB_PCDC_DT_SUBTYPE_CALL_MANAGE_FUNC                (0x01u)/*assigned from USB-IF*/
/* Abstract Control Management Functional Descriptor. */
#define     USB_PCDC_DT_SUBTYPE_ABSTRACT_CTR_MANAGE_FUNC        (0x02u)/*assigned from USB-IF*/
/* Union Functional Descriptor */
#define     USB_PCDC_DT_SUBTYPE_UNION_FUNC                      (0x06u)/*assigned from USB-IF*/

/* Communications Class Subclass Codes */
#define     USB_PCDC_CLASS_SUBCLASS_CODE_ABS_CTR_MDL            (0x02u) /*SubClass Code assigned by USB-IF*/

/* USB Class Definitions for Communications Devices Specification
 release number in binary-coded decimal. */
#define     USB_PCDC_BCD_CDC          (0x0110u) /*assigned from USB-IF*/

/* Descriptor length */
#define USB_PCDC_DD_LEN               (18u)
#define USB_PCDC_QD_LEN               (10u)
#define USB_PCDC_CD1_LEN              (67u)
#define STRING_DESCRIPTOR0_LEN        (4u)
#define STRING_DESCRIPTOR1_LEN        (16u)
#define STRING_DESCRIPTOR2_LEN        (8u)
#define STRING_DESCRIPTOR3_LEN        (22u)
#define STRING_DESCRIPTOR4_LEN        (22u)
#define STRING_DESCRIPTOR5_LEN        (18u)
#define STRING_DESCRIPTOR6_LEN        (28u)

/* Standard Device Descriptor */
uint8_t g_apl_device[USB_PCDC_DD_LEN + ( USB_PCDC_DD_LEN % 2)] =
{
    USB_PCDC_DD_LEN,                                    /*  0:bLength */
    USB_DT_DEVICE,                                      /*  1:bDescriptorType */
    (USB_BCDNUM & (uint8_t) 0xffu),                     /*  2:bcdUSB_lo */
    ((uint8_t) (USB_BCDNUM >> 8) & (uint8_t) 0xffu),    /*  3:bcdUSB_hi */
    USB_IFCLS_CDCC,                                     /*  4:bDeviceClass */
    0,                                                  /*  5:bDeviceSubClass */
    0,                                                  /*  6:bDeviceProtocol */
    (uint8_t) USB_DCPMAXP,                              /*  7:bMAXPacketSize(for DCP) */
    (USB_VENDORID & (uint8_t) 0xffu),                   /*  8:idVendor_lo */
    ((uint8_t) (USB_VENDORID >> 8) & (uint8_t) 0xffu),  /*  9:idVendor_hi */
    ((uint16_t) USB_PRODUCTID & (uint8_t) 0xffu),       /* 10:idProduct_lo */
    ((uint8_t) (USB_PRODUCTID >> 8) & (uint8_t) 0xffu), /* 11:idProduct_hi */
    (USB_RELEASE & (uint8_t) 0xffu),                    /* 12:bcdDevice_lo */
    ((uint8_t) (USB_RELEASE >> 8) & (uint8_t) 0xffu),   /* 13:bcdDevice_hi */
    1,                                                  /* 14:iManufacturer */
    2,                                                  /* 15:iProduct */
    6,                                                  /* 16:iSerialNumber */
    USB_CONFIGNUM                                       /* 17:bNumConfigurations */
};
```

```
 *  Configuration Or Other_Speed_Configuration Descriptor   *
/* For Full-Speed */
uint8_t g_apl_configuration[USB_PCDC_CD1_LEN + ( USB_PCDC_CD1_LEN % 2)] =
{
    9,                              /*  0:bLength */
    USB_SOFT_CHANGE,                /*  1:bDescriptorType */
    USB_PCDC_CD1_LEN % 256,         /*  2:wTotalLength(L) */
    USB_PCDC_CD1_LEN / 256,         /*  3:wTotalLength(H) */
    2,                              /*  4:bNumInterfaces */
    1,                              /*  5:bConfigurationValue */
    0,                              /*  6:iConfiguration */
    USB_CF_RESERVED | USB_CF_SELFP, /*  7:bmAttributes */
    (10 / 2),                       /*  8:MAXPower (2mA unit) */

    /* Interface Descriptor */
    9,                              /*  0:bLength */
    USB_DT_INTERFACE,               /*  1:bDescriptor */
    0,                              /*  2:bInterfaceNumber */
    0,                              /*  3:bAlternateSetting */
    1,                              /*  4:bNumEndpoints */
    USB_IFCLS_CDCC,                 /*  5:bInterfaceClass */
    USB_PCDC_CLASS_SUBCLASS_CODE_ABS_CTR_MDL,   /*  6:bInterfaceSubClass */
    1,                              /*  7:bInterfaceProtocol */
    0,                              /*  8:iInterface */

    /* Communications Class Functional Descriptors */
    5,                          /*  0:bLength */
    USB_PCDC_CS_INTERFACE,      /*  1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_HEADER_FUNC,    /*  2:bDescriptorSubtype */
    USB_PCDC_BCD_CDC % 256,     /*  3:bcdCDC_lo */
    USB_PCDC_BCD_CDC / 256,     /*  4:bcdCDC_hi */

    /* Communications Class Functional Descriptors */
    4,                          /*  0:bLength */
    USB_PCDC_CS_INTERFACE,      /*  1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_ABSTRACT_CTR_MANAGE_FUNC,   /*  2:bDescriptorSubtype */
    2,                          /*  3:bmCapabilities */

    /* Communications Class Functional Descriptors */
    5,                          /*  0:bLength */
    USB_PCDC_CS_INTERFACE,      /*  1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_UNION_FUNC, /*  2:bDescriptorSubtype */
    0,                          /*  3:bMasterInterface */
    1,                          /*  4:bSlaveInterface0 */

    /* Communications Class Functional Descriptors */
    5,                          /*  0:bLength */
    USB_PCDC_CS_INTERFACE,      /*  1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_CALL_MANAGE_FUNC,   /*  2:bDescriptorSubtype */
    /* D1:1-Device can send/receive call management
     information over a Data Class interface. */
    /* D0:1-Device handles call management itself. */
    3,                          /*  3:bmCapabilities */
    1,                          /*  4:bDataInterface */

    /* Endpoint Descriptor 0 */
    7,                          /*  0:bLength */
    USB_DT_ENDPOINT,            /*  1:bDescriptorType */
    USB_EP_IN | USB_EP3,        /*  2:bEndpointAddress */
    USB_EP_INT,                 /*  3:bmAttribute */
    16,                         /*  4:wMAXPacketSize_lo */
    0,                          /*  5:wMAXPacketSize_hi */
    0x10,                       /*  6:bInterval */

    /* Interface Descriptor */
    9,                              /*  0:bLength */
    USB_DT_INTERFACE,               /*  1:bDescriptor */
    1,                              /*  2:bInterfaceNumber */
    0,                              /*  3:bAlternateSetting */
    2,                              /*  4:bNumEndpoints */
    USB_IFCLS_CDCD,                 /*  5:bInterfaceClass */
    0,                              /*  6:bInterfaceSubClass */
    0,                              /*  7:bInterfaceProtocol */
    0,                              /*  8:iInterface */

    /* Endpoint Descriptor 0 */
    7,                          /*  0:bLength */
    USB_DT_ENDPOINT,            /*  1:bDescriptorType */
    USB_EP_IN | USB_EP1,        /*  2:bEndpointAddress */
    USB_EP_BULK,                /*  3:bmAttribute */
    64,                         /*  4:wMAXPacketSize_lo */
    0,                          /*  5:wMAXPacketSize_hi */
    0,                          /*  6:bInterval */
```

```
        /* Endpoint Descriptor 1 */
        7,                              /*  0:bLength */
        USB_DT_ENDPOINT,                /*  1:bDescriptorType */
        USB_EP_OUT | USB_EP2,           /*  2:bEndpointAddress */
        USB_EP_BULK,                    /*  3:bmAttribute */
        64,                             /*  4:wMAXPacketSize_lo */
        0,                              /*  5:wMAXPacketSize_hi */
        0,                              /*  6:bInterval */
};
/**********************************
 *    String Descriptor             *
 **********************************/
/* UNICODE 0x0409 English (United States) */
uint8_t g_cdc_string_descriptor0[STRING_DESCRIPTOR0_LEN + ( STRING_DESCRIPTOR0_LEN % 2)] =
{
    STRING_DESCRIPTOR0_LEN,     /*  0:bLength */
    USB_DT_STRING,              /*  1:bDescriptorType */
    0x09, 0x04                  /*  2:wLANGID[0] */
};
/* iManufacturer */
uint8_t g_cdc_string_descriptor1[STRING_DESCRIPTOR1_LEN + ( STRING_DESCRIPTOR1_LEN % 2)] =
{
    STRING_DESCRIPTOR1_LEN,     /*  0:bLength */
    USB_DT_STRING,              /*  1:bDescriptorType */
    'R', 0x00,                  /*  2:wLANGID[0] */
    'E', 0x00,
    'N', 0x00,
    'E', 0x00,
    'S', 0x00,
    'A', 0x00,
    'S', 0x00,
};
/* iProduct */
uint8_t g_cdc_string_descriptor2[STRING_DESCRIPTOR2_LEN + ( STRING_DESCRIPTOR2_LEN % 2)] =
{
    STRING_DESCRIPTOR2_LEN, /*  0:bLength */
    USB_DT_STRING,          /*  1:bDescriptorType */
    'U', 0x00,
    'S', 0x00,
    'B', 0x00,
};
/* iInterface */
uint8_t g_cdc_string_descriptor3[STRING_DESCRIPTOR3_LEN + ( STRING_DESCRIPTOR3_LEN % 2)] =
{
    STRING_DESCRIPTOR3_LEN, /*  0:bLength */
    USB_DT_STRING,          /*  1:bDescriptorType */
    'C', 0x00,
    'o', 0x00,
    'm', 0x00,
    ' ', 0x00,
    'D', 0x00,
    'e', 0x00,
    'v', 0x00,
    'i', 0x00,
    'c', 0x00,
    'e', 0x00,
};
/* iConfiguration */
uint8_t g_cdc_string_descriptor4[STRING_DESCRIPTOR4_LEN + ( STRING_DESCRIPTOR4_LEN % 2)] =
{
    STRING_DESCRIPTOR4_LEN, /*  0:bLength */
    USB_DT_STRING,          /*  1:bDescriptorType */
    'F', 0x00,              /*  2:wLANGID[0] */
    'u', 0x00,
    'l', 0x00,
    'l', 0x00,
    '-', 0x00,
    'S', 0x00,
    'p', 0x00,
    'e', 0x00,
    'e', 0x00,
    'd', 0x00
};
uint8_t *g_apl_string_table[] =
{
    g_cdc_string_descriptor0,
    g_cdc_string_descriptor1,
    g_cdc_string_descriptor2,
    g_cdc_string_descriptor3,
    g_cdc_string_descriptor4,
};
/***********************************************************************
 Renesas Abstracted Peripheral Communications Devices Class Driver API functions
 ***********************************************************************/
#endif /* R_USB_PCDC_DESCRIPTOR_H_ */
```

図 6-14　r_usb_pcdc_descriptor.h

6.5 *main()*にアプリケーションコードを追加

1) Smart_Configurator_Example.c ファイルの先頭近くに、ヘッダファイルのインクルードと宣言を追加します。

```
#include "r_usb_basic_if.h"
#include "r_usb_pcdc_if.h"
#include "r_usb_pcdc_descriptor.h"
#include <stdio.h>
#include <string.h>

void R_USB_PinSet_USB0_PERI();  /* Initialize USB0_VBUS pin */
static uint8_t  g_buf[1];               /* Variable to store input
character from PC terminal */
volatile unsigned int flag_start = 1;    /* Flag to print start message
*/
volatile uint16_t interval_level = 1;    /* Variable to change CMT0
interval */
static char print_str[120];              /* String to print message at
PC terminal */
volatile uint32_t slength;               /* String length */

extern uint8_t g_apl_device[];
extern uint8_t g_apl_configuration[];
extern uint8_t *g_apl_string_table[];
const static usb_descriptor_t usb_descriptor =
{
    g_apl_device,                   /* Pointer to the device descriptor
*/
    g_apl_configuration,            /* Pointer to the configuration
descriptor for Full-speed */
    USB_NULL,                       /* Pointer to the configuration
descriptor for Hi-speed */
    USB_NULL,                       /* Pointer to the qualifier
descriptor */
    g_apl_string_table              /* Pointer to the string descriptor
table */
};
```

2) 以下のコードを *main()* 関数の *R_Config_CMT0_Start()*の前に追加します。

```
    usb_ctrl_t  ctrl;
    usb_cfg_t   cfg;
```

3) 以下のコードを同じ *main()* 関数の *R_Config_CMT0_Start()*の後に追加します。これにより、USB0を初期化し、PC host 上のターミナルプログラムとの通信を開始します。

```
    R_USB_PinSet_USB0_PERI();            /* USB MCU pin setting */

    ctrl.module     = USB_IP0;    /* USB0 module */
    ctrl.type       = USB_PCDC;   /* Peripheral Communication Device Class*/
    cfg.usb_speed   = USB_FS;     /* USB_HS/USB_FS */
    cfg.usb_mode    = USB_PERI;
    cfg.p_usb_reg   = (usb_descriptor_t *)&usb_descriptor;
    R_USB_Open(&ctrl, &cfg);      /* Initializes the USB module */

        /* Loop back between PC Terminal and USB MCU */
        while (1)
        {
                switch (R_USB_GetEvent(&ctrl))
                {
                case USB_STS_CONFIGURED :
                        break;

                case USB_STS_WRITE_COMPLETE :
                        /* Read the input from PC terminal*/
                        R_USB_Read(&ctrl, g_buf, 1);
                        break;

                case USB_STS_READ_COMPLETE :
                        /* Clear start message flag*/
                        flag_start = 0;
                        /* Instruction to slow down blinking rate is received */
                        if (('a' == *g_buf)|| ('A' == *g_buf))
                        {
                                /* Get new blink interval level. Maximum level is 9. */
                                if (interval_level < 10)
                                {/* Notify the character received and inform next action */
                                sprintf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink
at slower rate.\r\n");
                                interval_level++;
                                }
                                else
                                {
                                sprintf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to blink
faster. \r\n");
                                }
                        }
                        else if (('b' == *g_buf)|| ('B' == *g_buf))
                        {
                                /* Get new blink interval level. Minimum level is 1 */
                                if (interval_level > 1)
                                {
                                        /* Instruction to increase blinking rate is received */
                                sprintf(print_str, "\r\n Character 'b' or 'B' is received. LED2 will blink
at faster rate.\r\n");
                                interval_level--;
                                }
                                else
                                {
                                sprintf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to blink
slower. \r\n");
                                }
                        }
                        else
                        {
                        sprintf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not
case sensitive)\n\r\n a ----> Slower\n\r\n b ----> Faster\n\r\n");
                        }
                        /* Print message at PC terminal */
                        slength = strlen(print_str);
                        R_USB_Write(&ctrl, (uint8_t *)print_str, slength);

                        /* Change blinking rate */
                        CMT0.CMCOR = (uint16_t)(5000 * interval_level);
                        break;

                default :
                        break;
                }

                if (flag_start == 1)
                {
                        /* Print start message until the 1st input from PC host is received */
                        sprintf(print_str, "Press space bar twice to start ...\r ");
                        slength = strlen(print_str);
                        R_USB_Write(&ctrl, (uint8_t *)print_str, slength);
        }
}
```

ソースファイルは以下のようになります。

```
#include "r_smc_entry.h"

#include "r_usb_basic_if.h"
#include "r_usb_pcdc_if.h"
#include "r_usb_pcdc_descriptor.h"
#include <stdio.h>
#include <string.h>

static void     usb_pin_setting (void);  /* Initialize USB0_VBUS pin */
static uint8_t  g_buf[1];                /* Variable to store input character from PC terminal */
volatile unsigned int flag_start = 1;    /* Flag to print start message */
volatile uint16_t interval_level = 1;    /* Variable to change CMT0 interval */
static char print_str[120];              /* String to print message at PC terminal */
volatile uint32_t slength;               /* String length */

extern uint8_t g_apl_device[];
extern uint8_t g_apl_configuration[];
extern uint8_t *g_apl_string_table[];
const static usb_descriptor_t usb_descriptor =
{
    g_apl_device,                   /* Pointer to the device descriptor */
    g_apl_configuration,            /* Pointer to the configuration descriptor for Full-speed */
    USB_NULL,                       /* Pointer to the configuration descriptor for Hi-speed */
    USB_NULL,                       /* Pointer to the qualifier descriptor */
    g_apl_string_table              /* Pointer to the string descriptor table */
};

[redacted]

void main(void)
{
    usb_ctrl_t  ctrl;
    usb_cfg_t   cfg;

    /* Start CMT0 counter operation */
    R_Config_CMT0_Start();

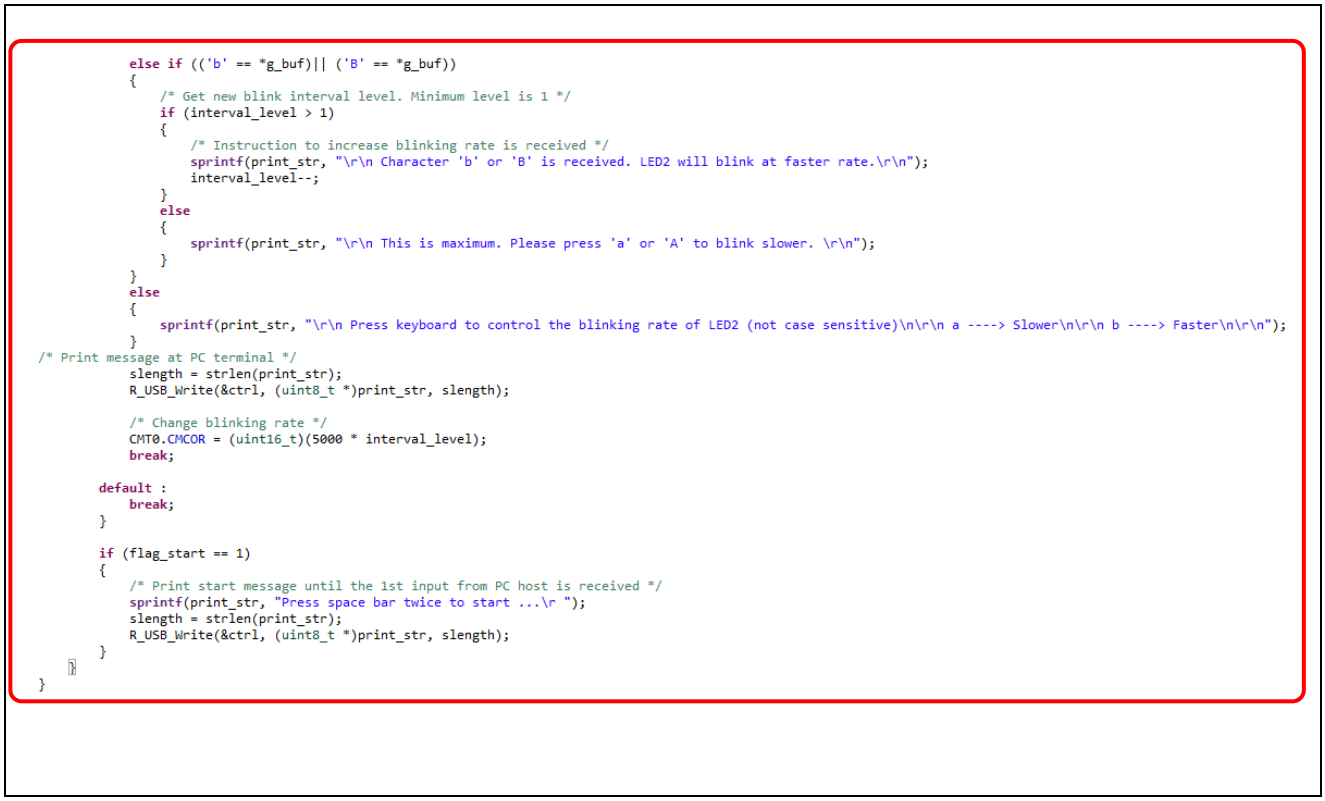
    usb_pin_setting();            /* USB MCU pin setting */

    ctrl.module     = USB_IP0;    /* USB0 module */
    ctrl.type       = USB_PCDC;   /* Peripheral Communication Device Class*/
    cfg.usb_speed   = USB_FS;     /* USB_HS/USB_FS */
    cfg.usb_mode    = USB_PERI;
    cfg.p_usb_reg   = (usb_descriptor_t *)&usb_descriptor;
    R_USB_Open(&ctrl, &cfg);      /* Initializes the USB module */

    /* Loop back between PC Terminal and USB MCU */
    while (1)
    {
        switch (R_USB_GetEvent(&ctrl))
        {
        case USB_STS_CONFIGURED :
            break;

        case USB_STS_WRITE_COMPLETE :
            /* Read the input from PC terminal*/
            R_USB_Read(&ctrl, g_buf, 1);
            break;

        case USB_STS_READ_COMPLETE :
            /* Clear start message flag*/
            flag_start = 0;
            /* Instruction to slow down blinking rate is received */
            if (('a' == *g_buf)|| ('A' == *g_buf))
            {
                /* Get new blink interval level. Maximum level is 9. */
                if (interval_level < 10)
                {
                    /* Notify the character received and inform next action */
                    sprintf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink at slower rate.\r\n");
                    interval_level++;
                }
                else
                {
                    sprintf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to blink faster. \r\n");
                }
            }
```



```
            else if (('b' == *g_buf)|| ('B' == *g_buf))
            {
                /* Get new blink interval level. Minimum level is 1 */
                if (interval_level > 1)
                {
                    /* Instruction to increase blinking rate is received */
                    sprintf(print_str, "\r\n Character 'b' or 'B' is received. LED2 will blink at faster rate.\r\n");
                    interval_level--;
                }
                else
                {
                    sprintf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to blink slower. \r\n");
                }
            }
            else
            {
                sprintf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not case sensitive)\n\r\n a ----> Slower\n\r\n b ----> Faster\n\r\n");
            }
/* Print message at PC terminal */
            slength = strlen(print_str);
            R_USB_Write(&ctrl, (uint8_t *)print_str, slength);

            /* Change blinking rate */
            CMT0.CMCOR = (uint16_t)(5000 * interval_level);
            break;

        default :
            break;
        }

        if (flag_start == 1)
        {
            /* Print start message until the 1st input from PC host is received */
            sprintf(print_str, "Press space bar twice to start ...\r ");
            slength = strlen(print_str);
            R_USB_Write(&ctrl, (uint8_t *)print_str, slength);
        }
    }
}
```

図 6-15　*main* 関数での USB PCDC アプリケーションコード

6.6　ビルドとハードウェアボードでの実行

プロジェクトをビルドした後、デバッグアイコンをクリックし、そのプロジェクトをデバッグします。

ステップ 1.　PC host と、RX65N 2MB 用 Renesas Starter Kit+上の USB function コネクタを、USB ケーブル (ミニ B)を使用して接続します。（USB function ポートの位置は、6.8 章を参照してください）

ステップ 2.　e² studio デバッグ・パースペクティブで、をクリックしてプロジェクトを実行します。プロジェクトの状態は、IDE の左下にある実行状態を示す

に表示されます。

ステップ 3.　PC でターミナルプログラム(例：Tera Term)を開き、以下の手順を実行します。

1) [シリアル]ラジオボタンを選択します。
2) [ポート]オプションで、USB シリアルデバイスに接続する COM を選択 します。
3) [OK]をクリックします。

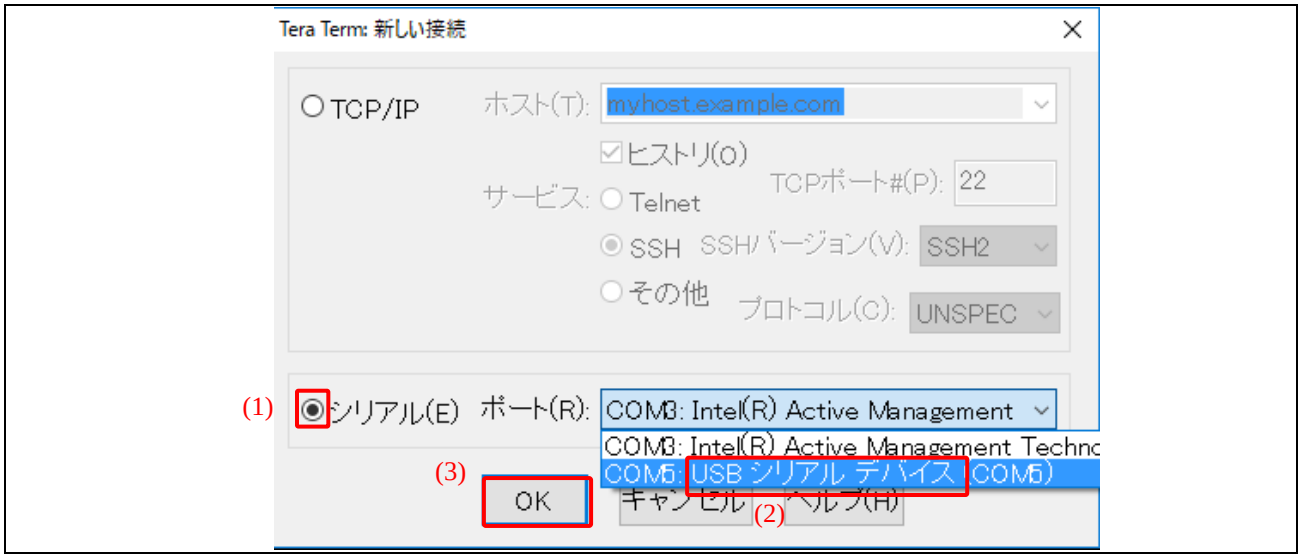


図 6-16　USB シリアルデバイス選択

注：Windows 7 PC では、デバイスドライバソフトウェアが正常にインストールされない場合があります。

\src\smc_gen\r_usb_pcdc\utilities\CDC_Demo_Win7.inf とは別に、システム定義ファイルをインストールしてください。

インストール後、ターミナルプログラムは COM を“CDC USB Demonstration”として表示します。

ステップ 4.　LED2 点滅周期を変更するには、ターミナルプログラムのメッセージに従ってください。

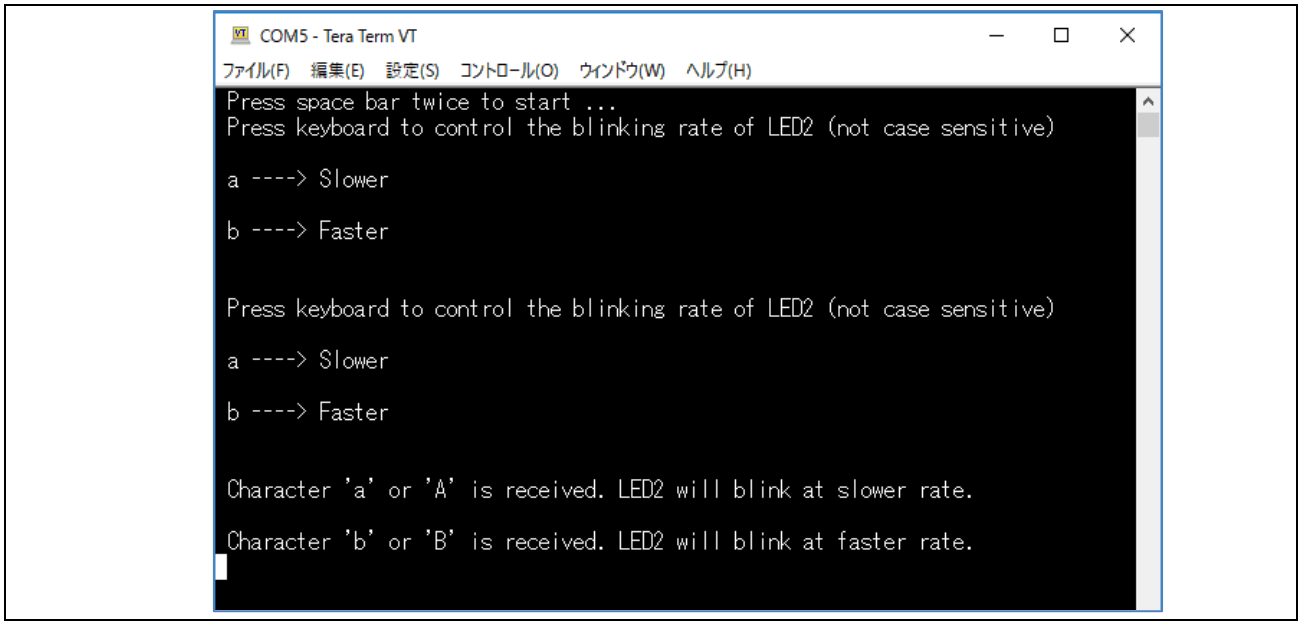


図 6-17　PC のターミナルウィンドウ

6.7　ボードでの動作確認

PC ターミナル上のキーが'a'または'b'のとき、RX65N 2MB ボード用 Renesas Starter Kit+上の LED2 の点滅周期が変わります。

- a：LED2 の点滅は遅くなります
- b：LED2 の点滅は早くなります

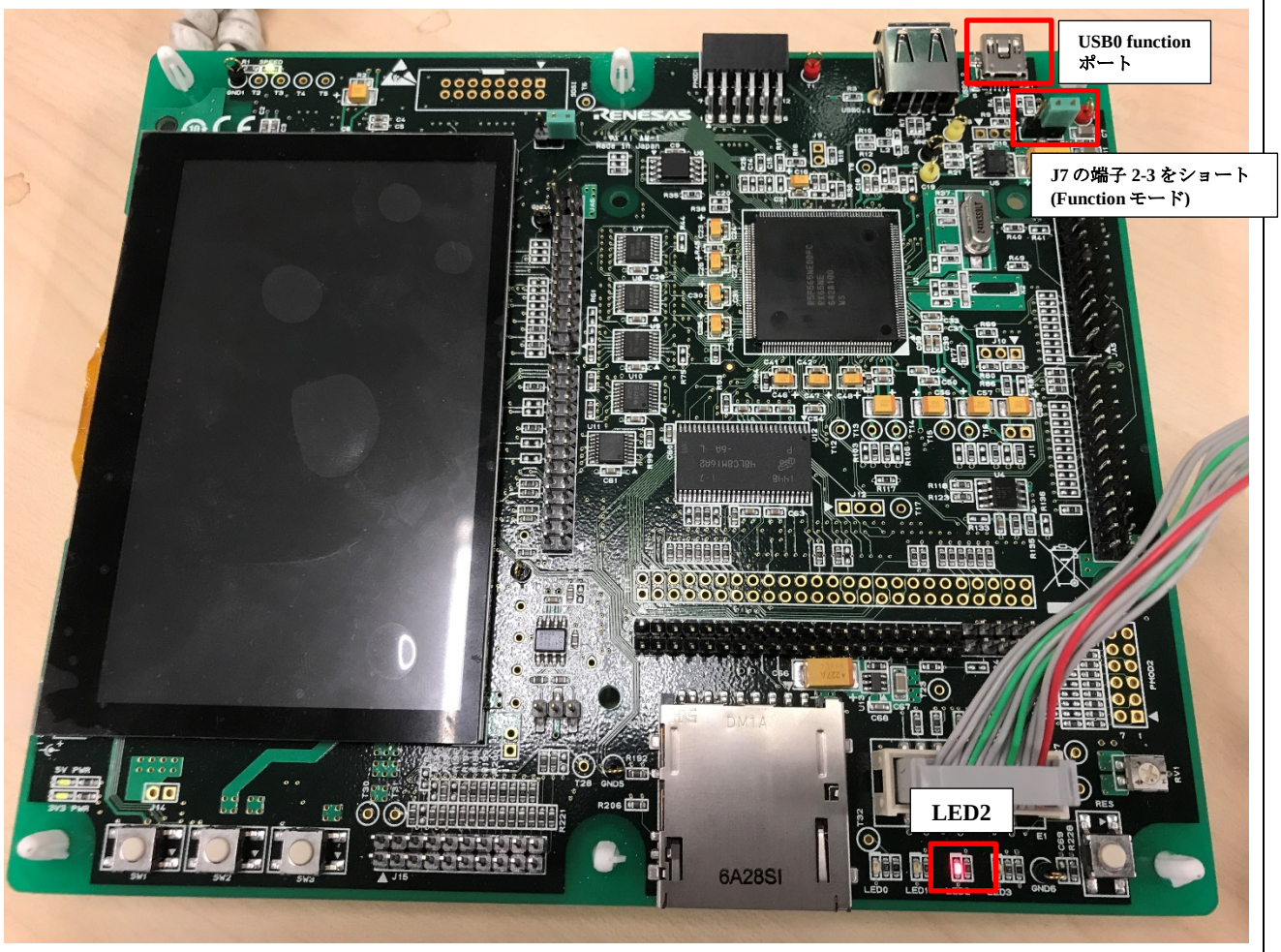


図 6-18　RX65N 2MB ボード上の LED2 と USB Function ポート

6.8　開発支援ツール（QE）

開発の効率向上を支援するため、各種アプリケーション用の開発ツールとして、ルネサスは豊富な Quick and Effective Tool Solution (QE)を提供しています。USB function を組み込む場合には、QE for USB の使用をお勧めします。QE for USB は、USB function アプリケーションのデバッグを支援する機能を持っています。

ルネサス IDE でサポートするアプリケーションや、QE については、以下のリンクを参照してください。

QE:https://www.renesas.com/qe

QE for USB:https://www.renesas.com/qe-usb

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://www.renesas.com/

お問合せ先
http://www.renesas.com/contact/

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	2017.11.10	-	初版発行
1.10	2018.03.30	1	ドキュメント名を変更
		10	e² studio V6.2 を元に更新 図 2-9 E2 Lite エミュレータを使用する場合の設定例を変更
		13	e² studio V6.2 を元に更新 図 2-13 ソフトウェアコンポーネントの選択を変更
		15	e² studio V6.2 を元に更新 図 2-16 ソフトウェアコンポーネントの選択を変更
		29	e² studio V6.2 を元に更新 図 3-4 ソフトウェアコンポーネントの選択を変更
		40	e² studio V6.2 を元に更新 図 4-8 ソフトウェアコンポーネントの選択を変更
		61	e² studio V6.2 を元に更新 図 6-9 ソフトウェアコンポーネントの選択を変更
		62	e² studio V6.2 を元に更新 図 6-10 ソフトウェアコンポーネントの設定を追加
		63	e² studio V6.2 を元に更新 図 6-11 ソフトウェアコンポーネントの設定を追加
		65	5.4 デフォルトコンフィグレーションの変更を削除
		74	e² studio V6.2 を元に更新 新しいスマート・コンフィグレータに合わせるため、コードを修正
1.20	2019.06.20	36	SCI8 を使用するため、Application Example 3 を更新
		54	Application Example 4 - DMA を使用したデータ転送を追加

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. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   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. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「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。
5. クロックについて
   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。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   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、将来の拡張機能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。
8. 製品間の相違について
   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、フラッシュメモリ、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、ノイズ幅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、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、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（宇宙機器と、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想定していません。たとえ、当社が想定してい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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、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。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. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
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. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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.4.0-1　2017.11)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。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。

© 2019 Renesas Electronics Corporation. All rights reserved.