コード生成プラグイン学習ガイド

ルネサス システムデザイン株式会社
ツールビジネス事業部 ツール技術部

2014/10/01　Rev. 1.00

R20UT3230JJ0100

はじめに

みなさんは、近所の方に「何のお仕事しているの？」と聞かれて、一言で答えられますか？
私は、無理です。マイコン、開発ツール、コード生成…説明しても理解してもらえるかわからない言葉だらけで、いつも適当に「はい、IT系です」と誤魔化しています。

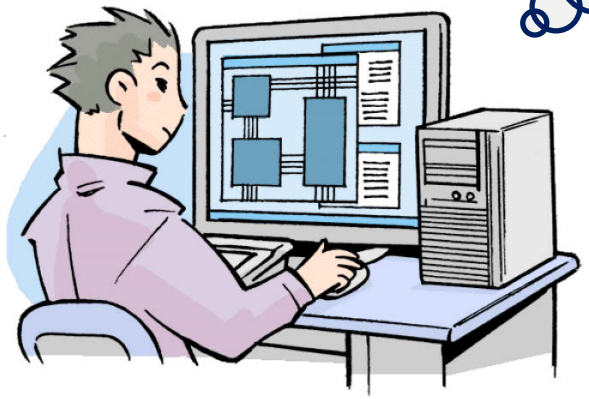
同じように、開発といっても、ある機能のほんの一部分を担当されている方もいると思います。
日々の仕事って何になるんだろうと考えることもありますよね？
先日、実家に帰省した際、93歳になる祖父に会いました。
ふと、祖父に「今まで生きてきた中で1番の変化（出来事）って何？」と聞いたところ、
「人生で一番の変化は、コンピュータができたことかなぁ」と答えました。
戦争ではなく、こんな山の中で農業をして暮らしているのに、
ITによる変化が1番だなんて！
驚くとともに、とてもうれしかったのを覚えています。
なぜって、その変化の中の、ほんの少しだろうけど、私の仕事も含まれていると思えたから。
開発をやり遂げた自己満足も良いですが、多くの人に変化を与えられるようなモノを目指したいと思っています。そんな気持ちを込めたガイドです。

コード生成って?

マニュアルが分厚くて、どこから読んでいいかわからないよ。

間違った使い方をして、壊してしまわないかな？

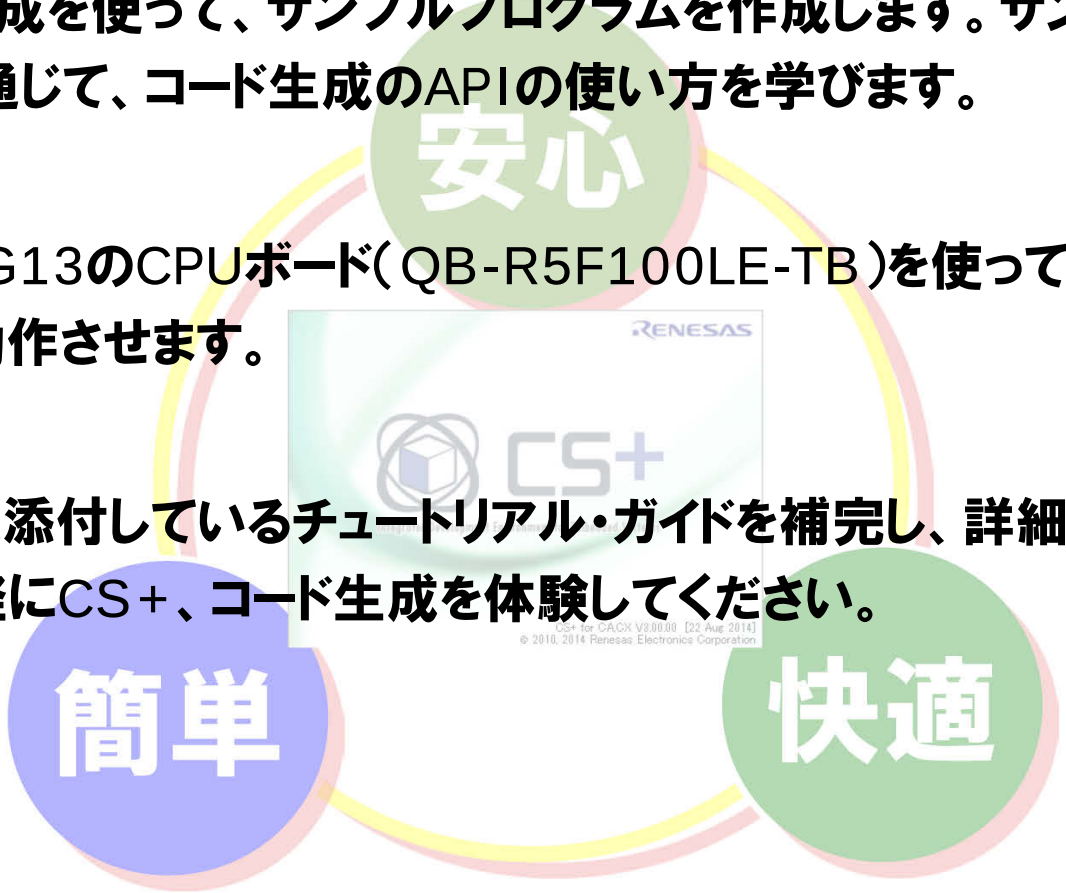
マイコンが高性能になるのは良いけれど、使うハードルが高くないかな？



- マイコンの使用したい機能を動作させるプログラムを、コード生成プラグインの簡単な操作で自動生成することができます。
- 誤った設定には、ワーニングが出るので安心です。
- 開発の最初の手掛かりに、マイコンの学習に役立ちます。

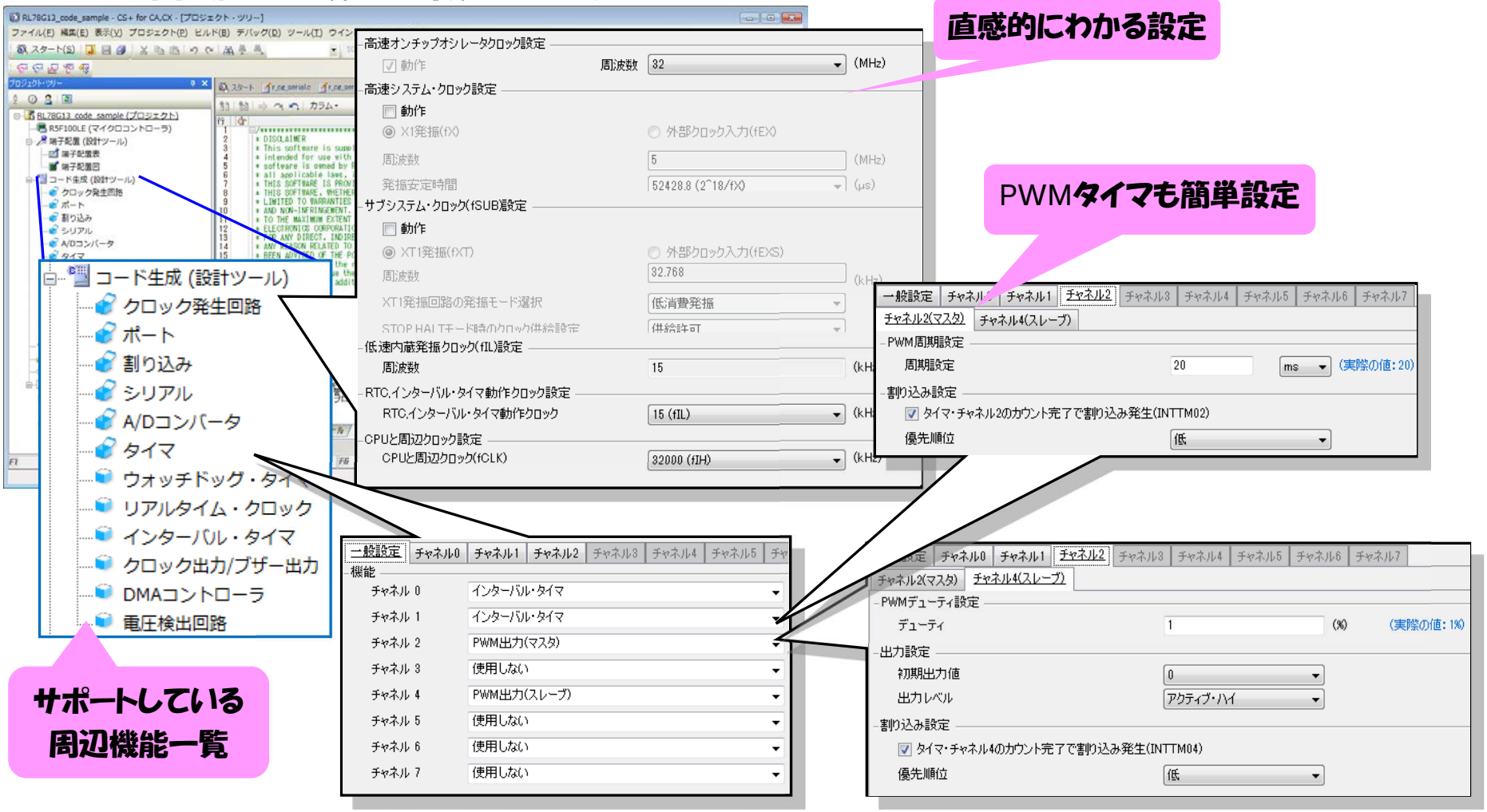
目的

- コード生成を使って、サンプルプログラムを作成します。サンプルプログラム作成を通じて、コード生成のAPIの使い方を学びます。
- RL78/G13のCPUボード（QB-R5F100LE-TB）を使って、作成したプログラムを動作させます。
- CS＋ に添付しているチュートリアル・ガイドを補完し、詳細な解説を加えました。気軽にCS＋、コード生成を体験してください。



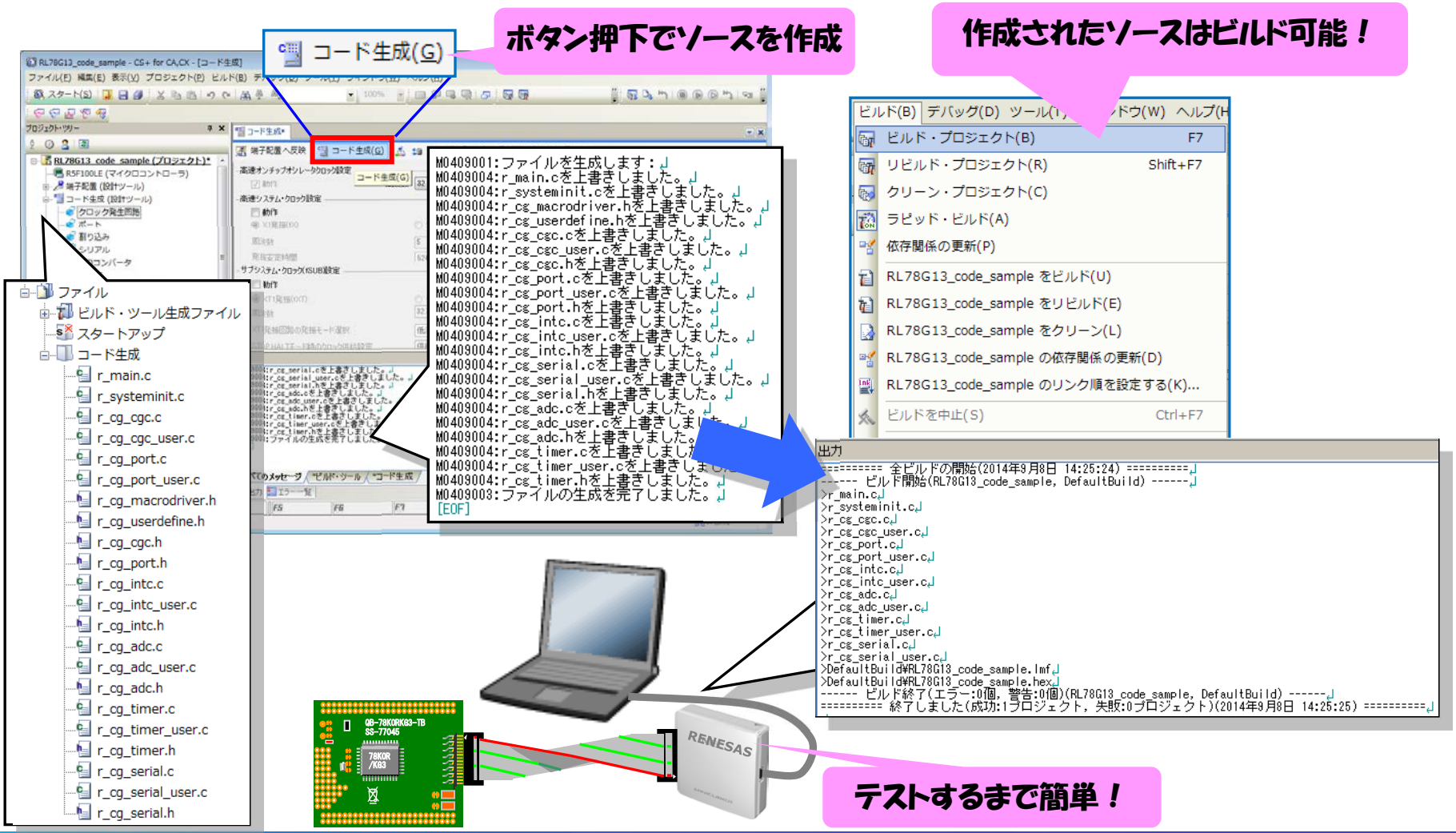
コード生成プラグイン紹介(1/3)

- GUIを使った簡単な操作でマイコンの内蔵周辺機能を動作させるプログラムを自動で生成する機能です。



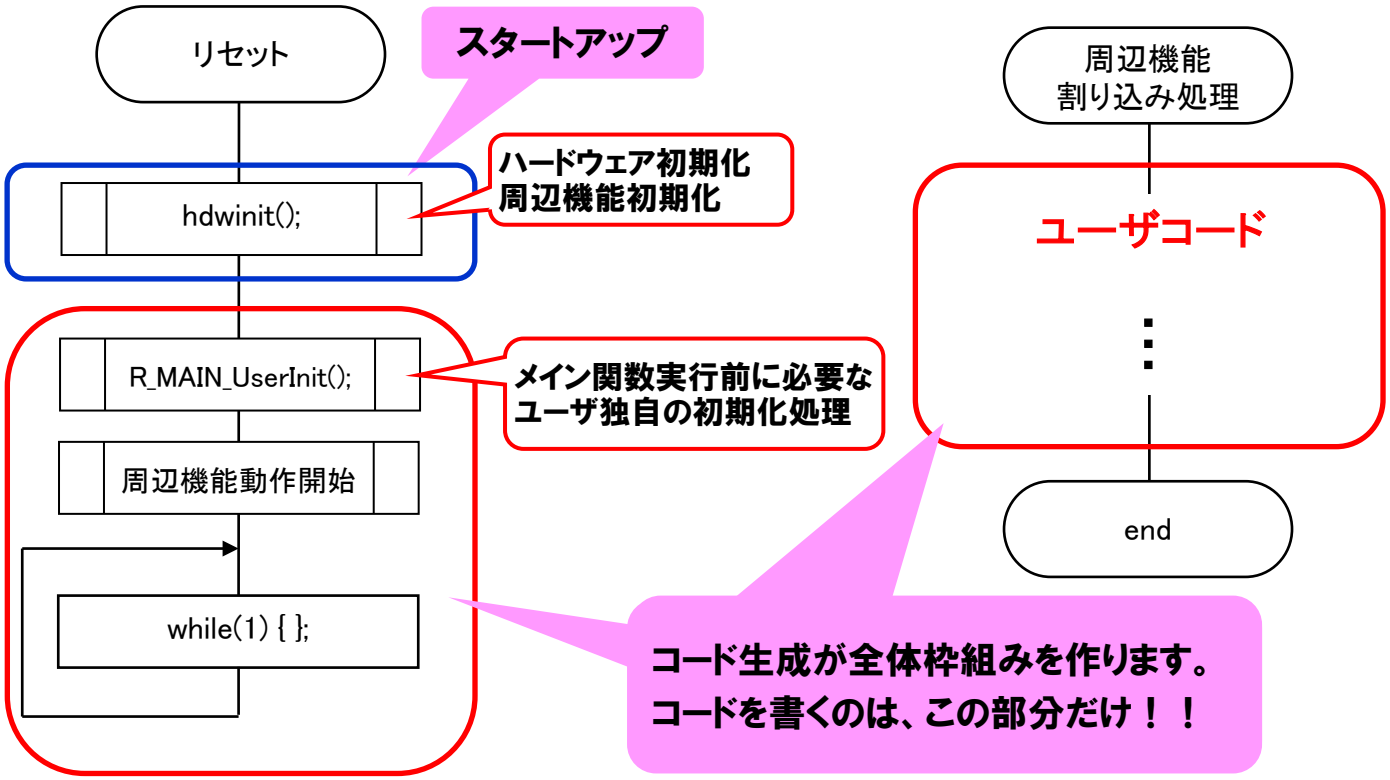
コード生成プラグイン紹介(2/3)

- **ボタン押下のみで、ビルド可能なソースを作成します。**



コード生成プラグイン紹介(3/3)

- main関数、使用する機能に応じたAPI(Application Program Interface)関数セットが生成されるので、これらを組み合わせるだけで様々なプログラムが作成可能です。
- 生成するプログラムはC言語ですので、処理が理解しやすくなっています。



プログラム概要(1/2)

- **ポート、タイマ機能を使用して、7セグLEDに数字を表示させましょう。**
  - ✓ **4桁とも同じ数字(“0000”、“1111”、… 、“9999”、“. . . . ”)を表示(ダイナミック点灯)させます。**

- **A/Dコンバータ、ポート、タイマ機能を使用して、ポテンションメータの値をA/D変換して7セグLEDに表示させましょう。**
  - ✓ **A/D変換値(10bit分解能(0～1023))を7セグLEDに表示(ダイナミック点灯)させます。**

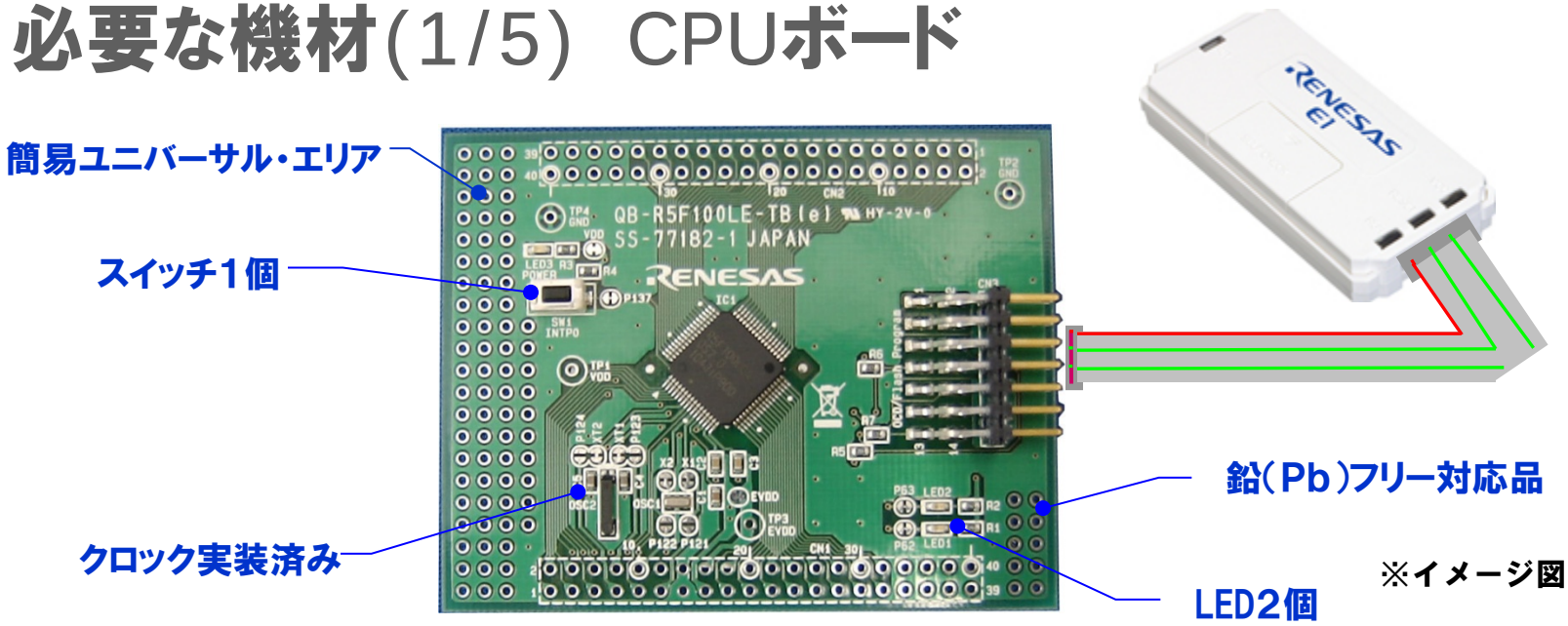
- **シリアル機能を使用して、UART通信させましょう。**
  - ✓ **ターミナルを使って、PCのキー入力を送受信(コールバック)させます。**
  - ✓ **スイッチボードで割り当てた機能をターミナルにメニュー表示させます。**

プログラム概要(2/2)

- **タイマ機能(PWM)を使用して、LEDの輝度を変化させましょう。**
  - ✓ CPUボードのLEDをダイナミック点灯させます。
  - ✓ 点灯期間と消灯期間を割合を制御しLEDの輝度を変化させます。

- **割り込み機能を使用して、スイッチボードで作成したプログラムを選択できるようにしましょう。**
  - ✓ SW1-4押下で、各機能を動作させます。
  - ✓ リセットSW(CPUボード上SW)押下で、再選択可能状態にします。

必要な機材(1/5)　CPUボード



RL78/G13（R5F100LEAFB）仕様

電源電圧		VDD = 1.6 - 5.5 V
動作周波数		1 - 32 MHz
ROM容量		64KBフラッシュ・メモリ、4 KBデータフラッシュ
RAM容量		4 KB
I/Oポート		16-120本(N-chオープン・ドレーン: 0-4本)
タイマ		
	16ビット・タイマ	8-16チャネル
	ウォッチドッグ・タイマ	1チャネル
	リアルタイム・クロック	1チャネル
	インターバル・タイマ	1チャネル
シリアル・インタフェース		
	CSI	2～8チャネル
	UART／UART(LIN-bus対応)	2～4チャネル
	I2C／簡易I2C	2～8チャネル
8/10ビット分解能A/Dコンバータ		6-26チャネル
オンチップ・デバッグ機能、電圧検出(LVD)回路、キー割り込み機能、クロック出力／ブザー出力制御回路内蔵		

必要な機材(2/5) その他機材

- ポテンションメータ(半固定抵抗)
- 7セグLED *1
- スイッチボード *1
- シリアル-USB変換ボード *2
- USBケーブル
- ブレッドボード
- TBアダプタ
- 配線材
- E1本体、ユーザシステムインタフェースケーブル、USBケーブル

- ターミナルソフト *3

*1:7セグLEDボード、スイッチボードは、workshop Nak様の製品です。 http://www.wsnak.com
*2:シリアル-USB変換ボードは、sparkfun様の製品です。https://www.sparkfun.com/products/9716
*3:ここでは、Teratermを使用しました。http://sourceforge.jp/projects/ttssh2/

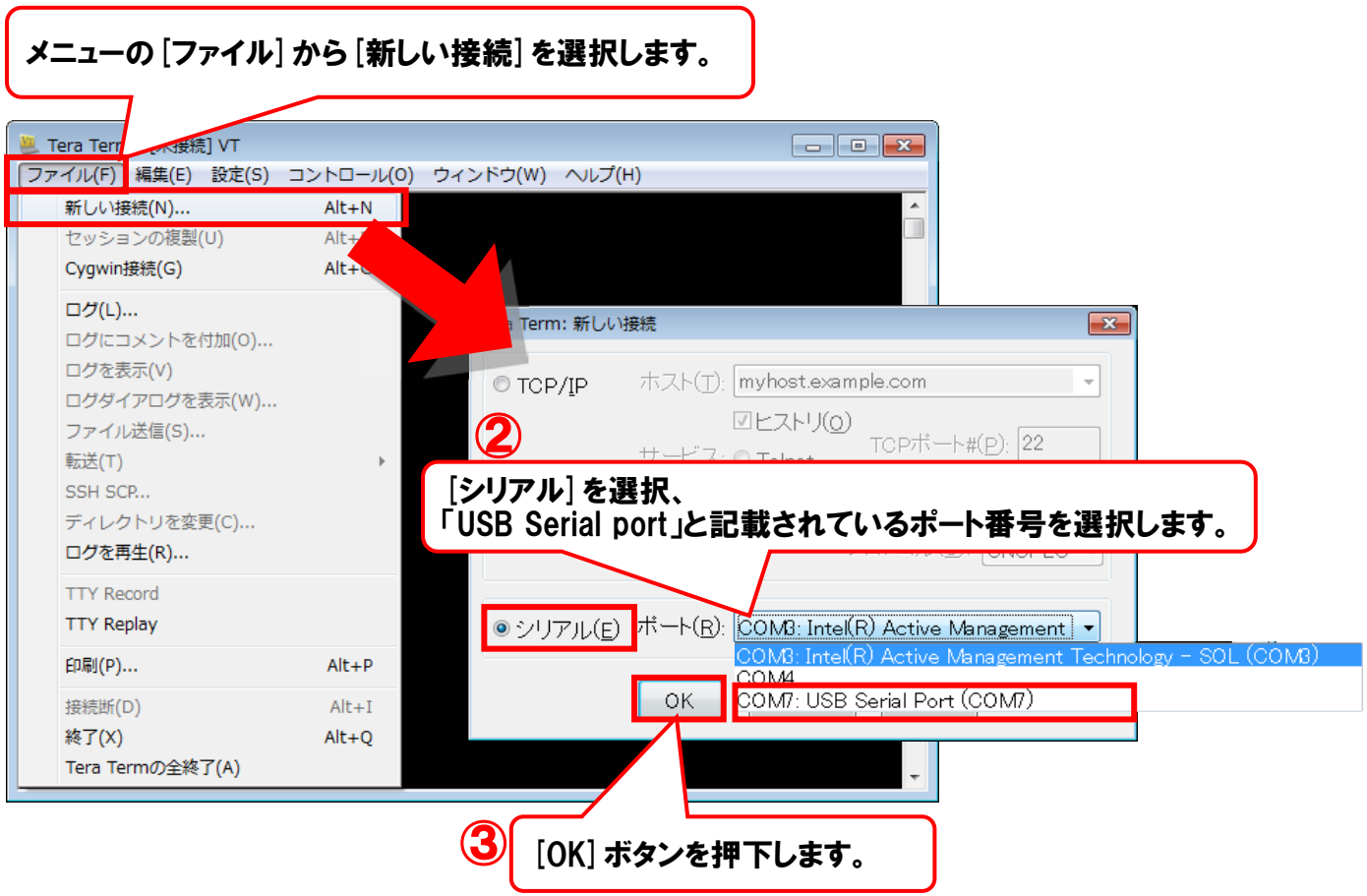
必要な機材(3/5)　ターミナルソフト

- シリアル通信のためのターミナルソフトとして、今回はTeraterm(フリーソフト)を使用しました。

  - インストーラを下記などのサイトよりダウンロードし、インストールしてください。
    http://sourceforge.jp/projects/ttssh2/
  - シリアル接続可能であれば、他のターミナルソフトを使用しても構いません。

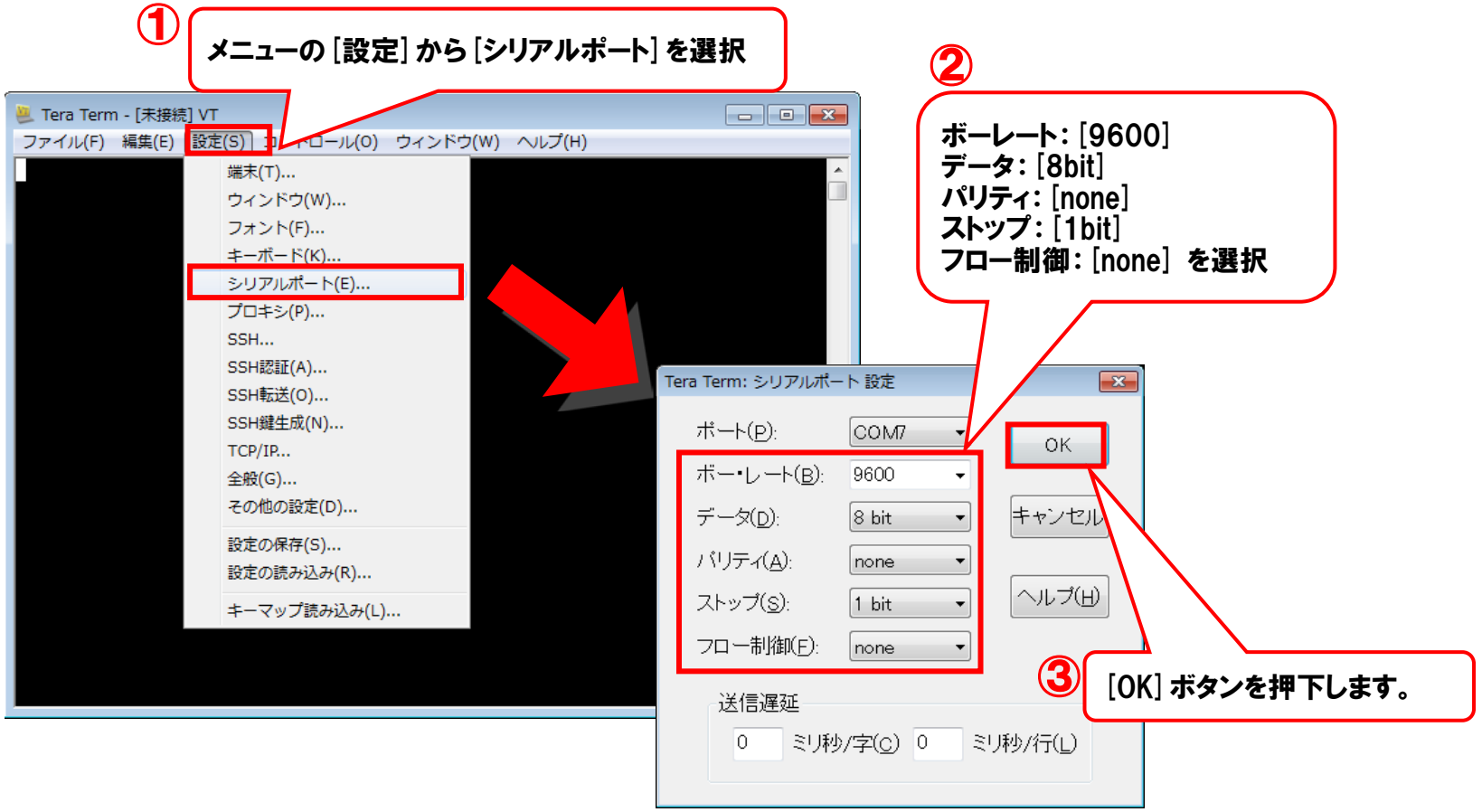
必要な機材(4/5)　ターミナルソフト

- Teratermは、以下のように設定し使用してください。
  - PC と シリアル-USB変換ボード を接続し、Teratermを起動します。

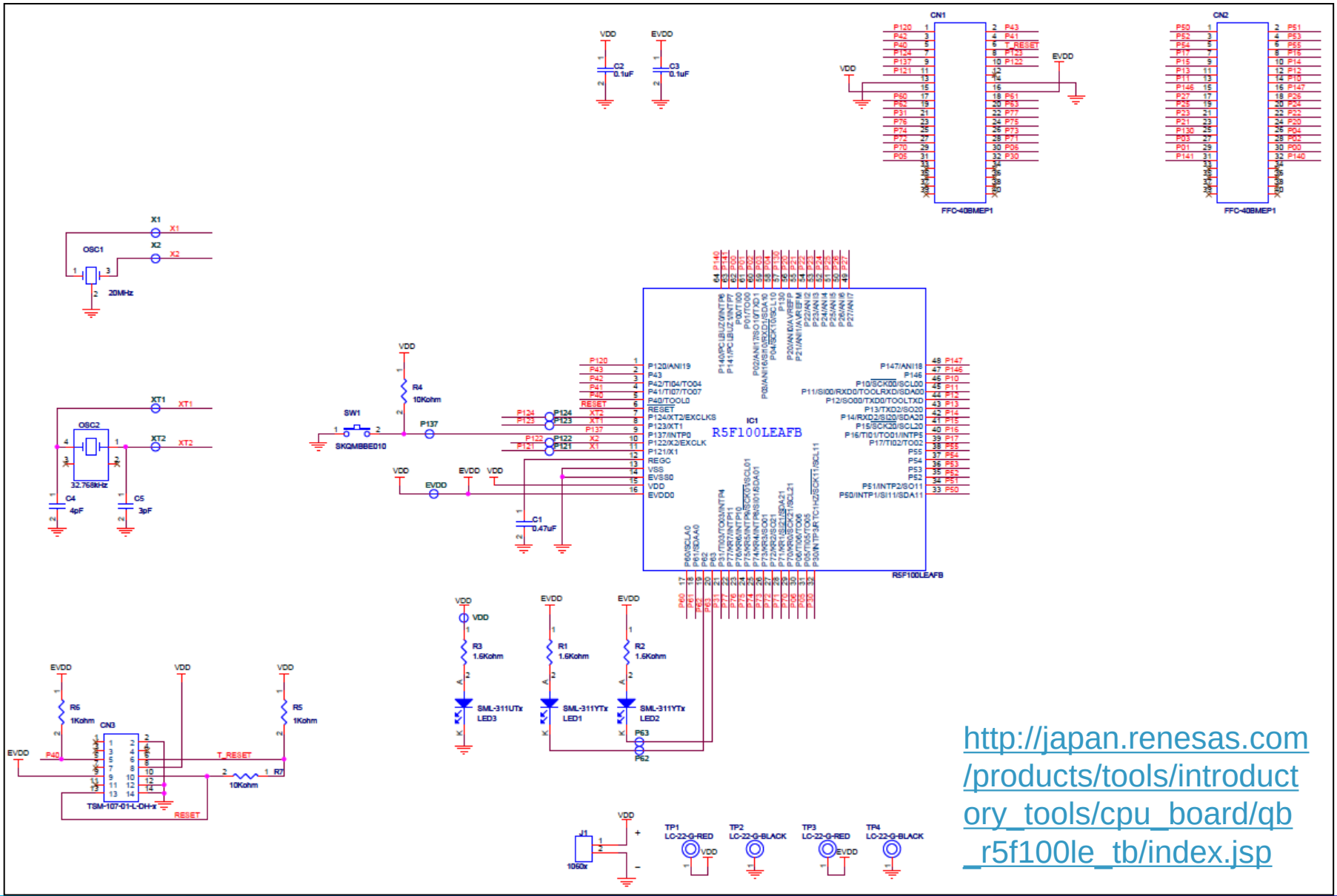


必要な機材(5/5)　ターミナルソフト

- 以下のように、シリアルポートの設定をします。



ハードウェア設計　回路図（CPUボード）



http://japan.renesas.com/products/tools/introductory_tools/cpu_board/qb_r5f100le_tb/index.jsp

ハードウェア設計　回路図（4桁7セグメントLED）

A～GのLEDを点灯させて
数字表示させる。

TITLE		DRAWING_No.
#233　4桁7セグメントLED BBアダプタ		
SHEET	DATE	DESIGN
1／1	09/02/07	T.Nakao

workshop Nak
www.wsnak.com

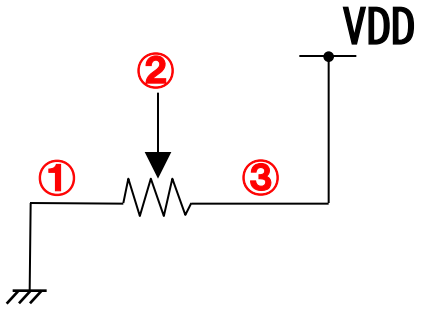
http://www.wsnak.com/kit/233/index.htm

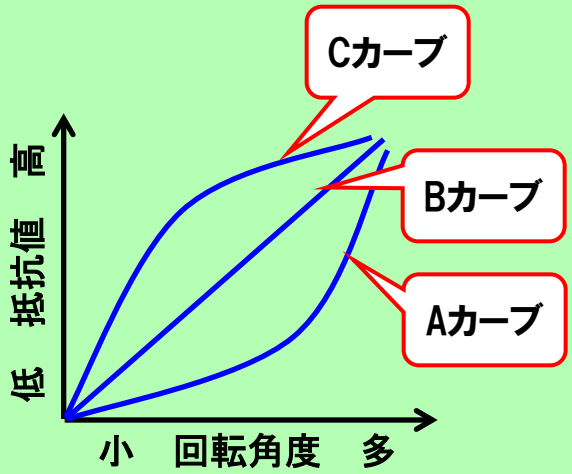
ハードウェア設計　回路図（ポテンションメータ）

GND(①)

マイコンのアナログ端子へ接続(②)

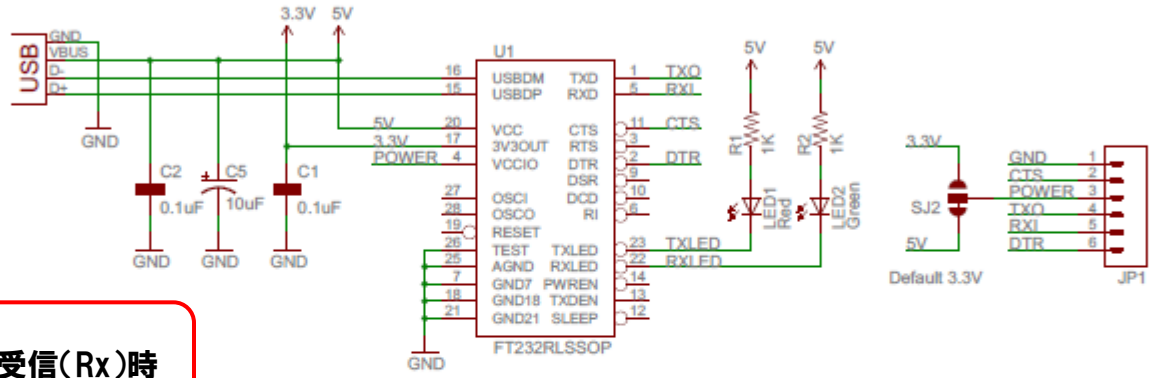
VDD(③)

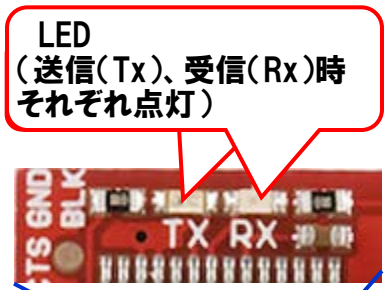


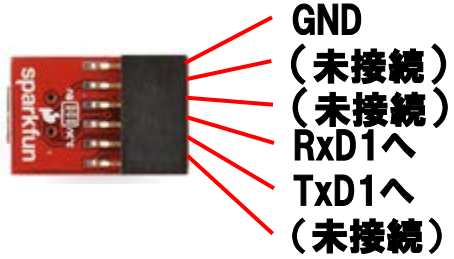


左図は、半固定抵抗器の特性を示したものです。
半固定抵抗器は、一般的にポテンションメータと呼ばれます。
今回扱う部品はBカーブのものを使います。一般的な半固定抵抗器（ボリューム）はAカーブやBカーブが多く、回転角度と抵抗値が一定のBカーブの部品が扱いやすいので、今回採用しました。

ハードウェア設計　回路図(シリアル-USB変換ボード)





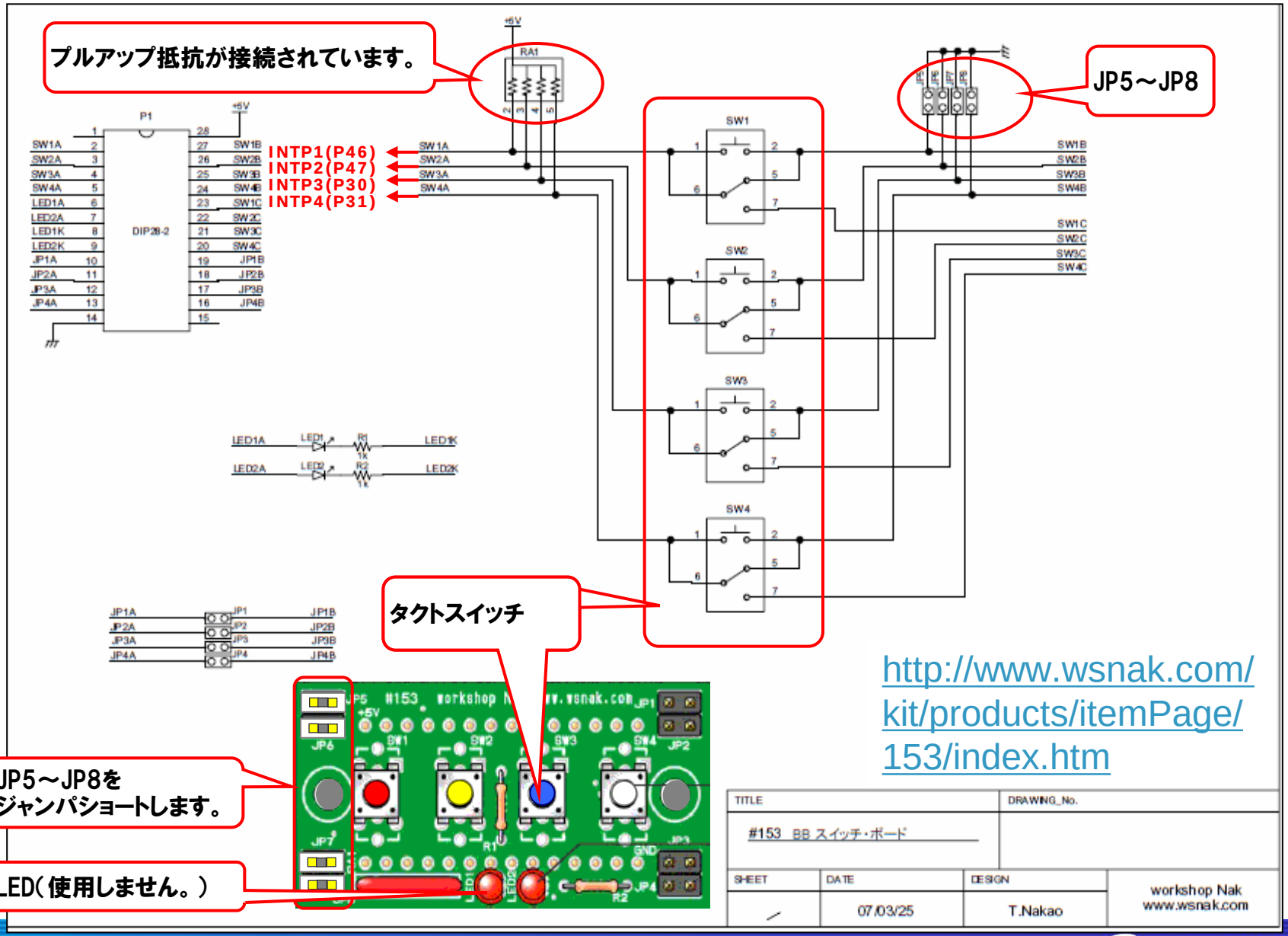


USBケーブル

https://www.sparkfun.com/products/9716

TITLE: FTDI Basic-v21-5V	SFE	SFE
Document Number:		REV:
Date: 12/2/2010 11:42:33 AM	Sheet: 1/1	

ハードウェア設計　回路図（SWボード）



ハードウェア設計

- CPUボード
  メイン・クロック動作周波数：20MHz （発振子搭載）
  評価用LED: 黄x2 (LED1 はP62,LED2 はP63 へ接続)
  評価用SW: SW1 (INTP0 へ接続)

- 7セグLED
  表示データを マイコンのP10-P17に接続
  出力セレクタ信号を、マイコンのP52-P55に接続

- ポテンションメータ
  可変端子を マイコンのANI0に接続

端子が競合しないように設計してください。
ここで決めた情報をコード生成に入力していきます。

ハードウェア設計

- **シリアル-USB変換ボード**
  マイコンのSAU0チャネル、チャネル2をUARTとして使用し、
  Tx1、Rx1、GNDを接続する。

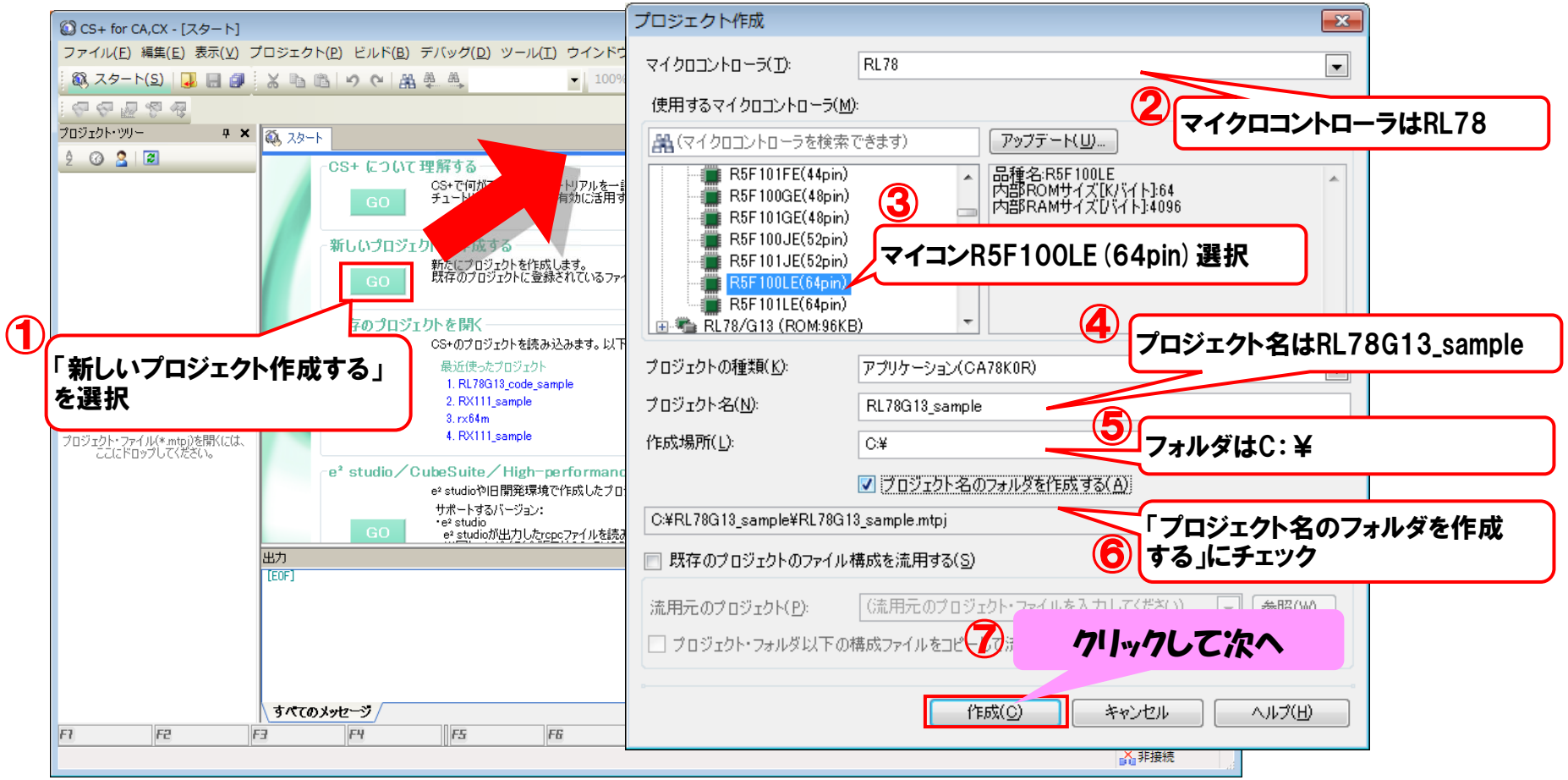
  <UART設定条件>
  データ・ビット長：8bit
  パリティ：無し
  ストップ・ビット長：1bit
  データ転送方向：LSBファースト
  転送レート：9600bps

- **スイッチボード**
  SW1-SW4を それぞれマイコンの INTP1-INTP4に接続する。

端子が競合しないように設計してください。
ここで決めた情報をコード生成に入力していきます。

CS+ プロジェクトの作成

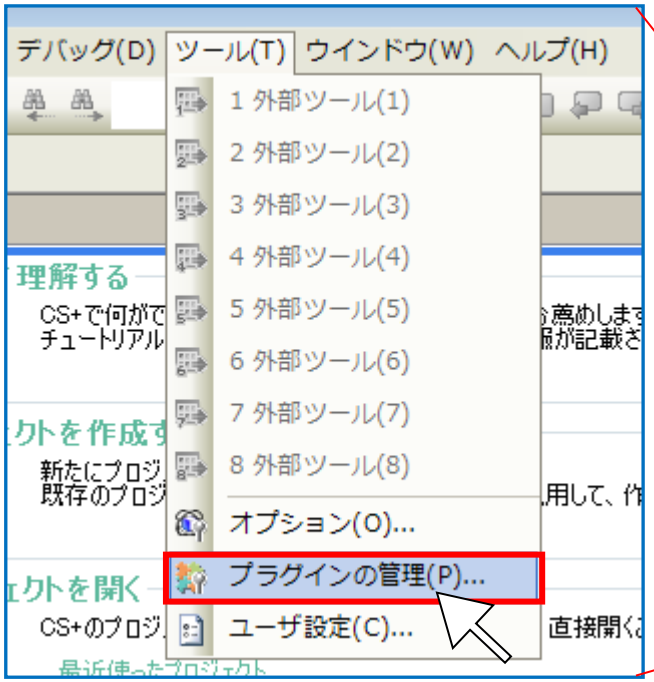
- CS+ for CA,CXを起動し、新しいプロジェクトを作成します。

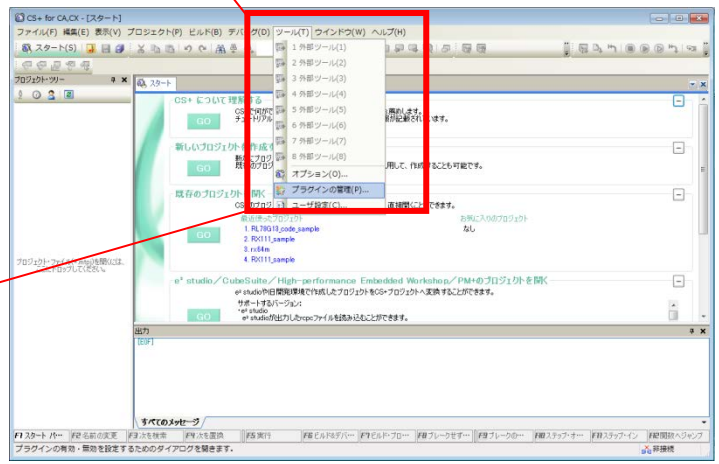


既存プロジェクトをコピーして新しいプロジェクトを作成することも可能です。
ただし、その場合、コード生成の設定は引き継がれません。

CS+ プラグインの管理 (1/2)

- メニューの [ツール] から [プラグイン管理] を選択してください。

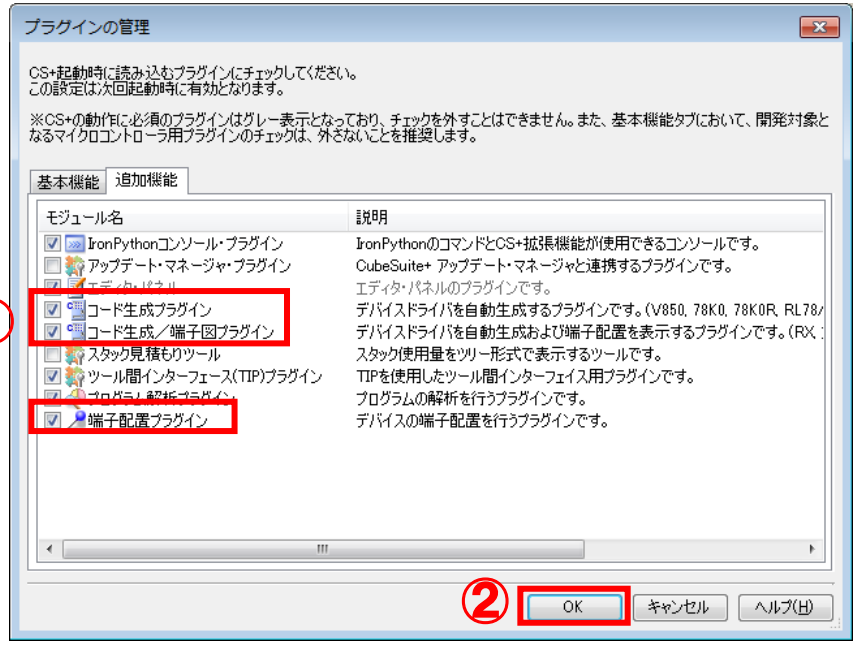




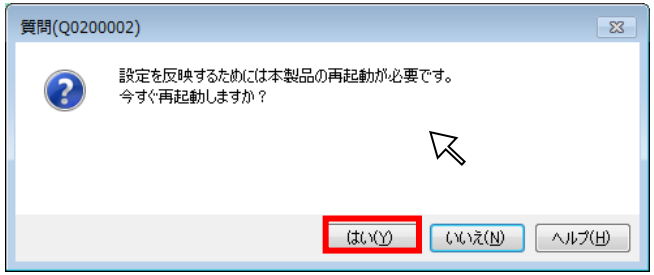
デフォルトの端子配置機能やコード生成機能はOFF状態になっていますので、プラグインをONにする必要があります。

CS+ プラグインの管理 (2/2)

- 下図のように情報を設定後、[OK] ボタンをクリックしてください。

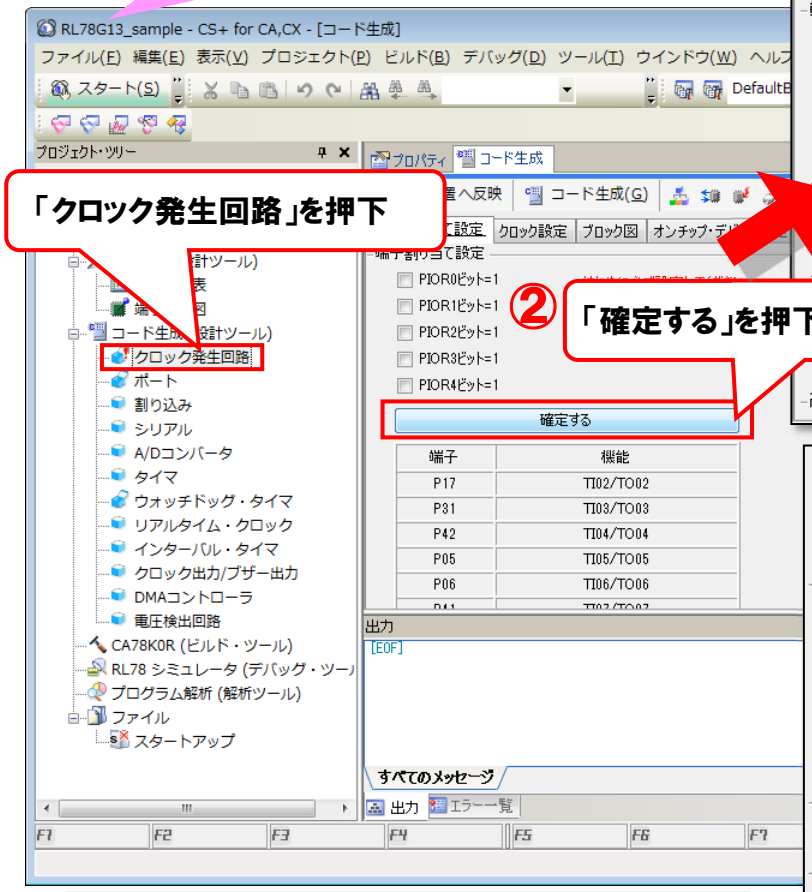


- [はい] ボタンを押下し、CS+を再起動してください。



CS+ コード生成設定(システム)

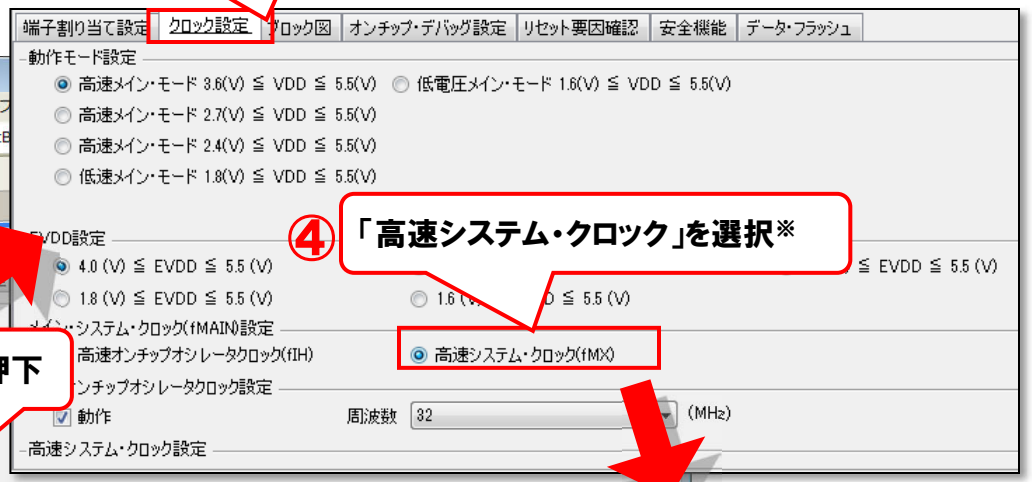
プロジェクトが作成される



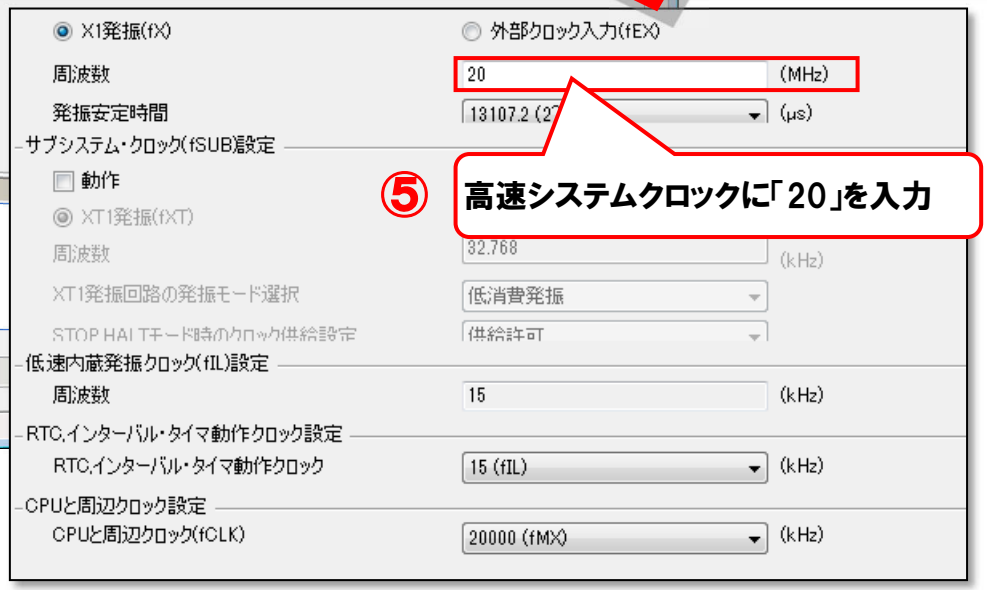
① 「クロック発生回路」を押下

② 「確定する」を押下

③ 「クロック設定」タブに切り替え



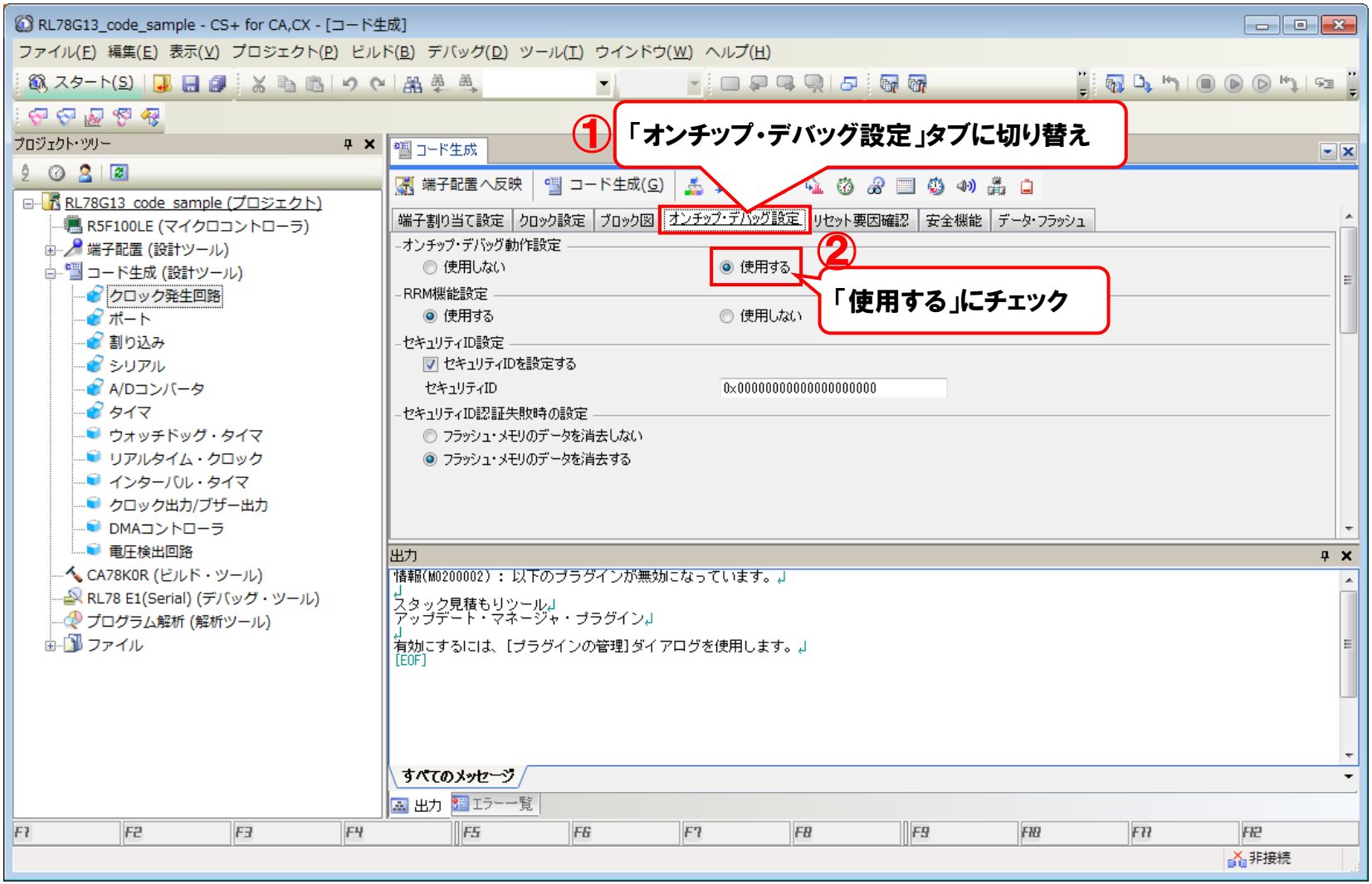
④ 「高速システム・クロック」を選択※



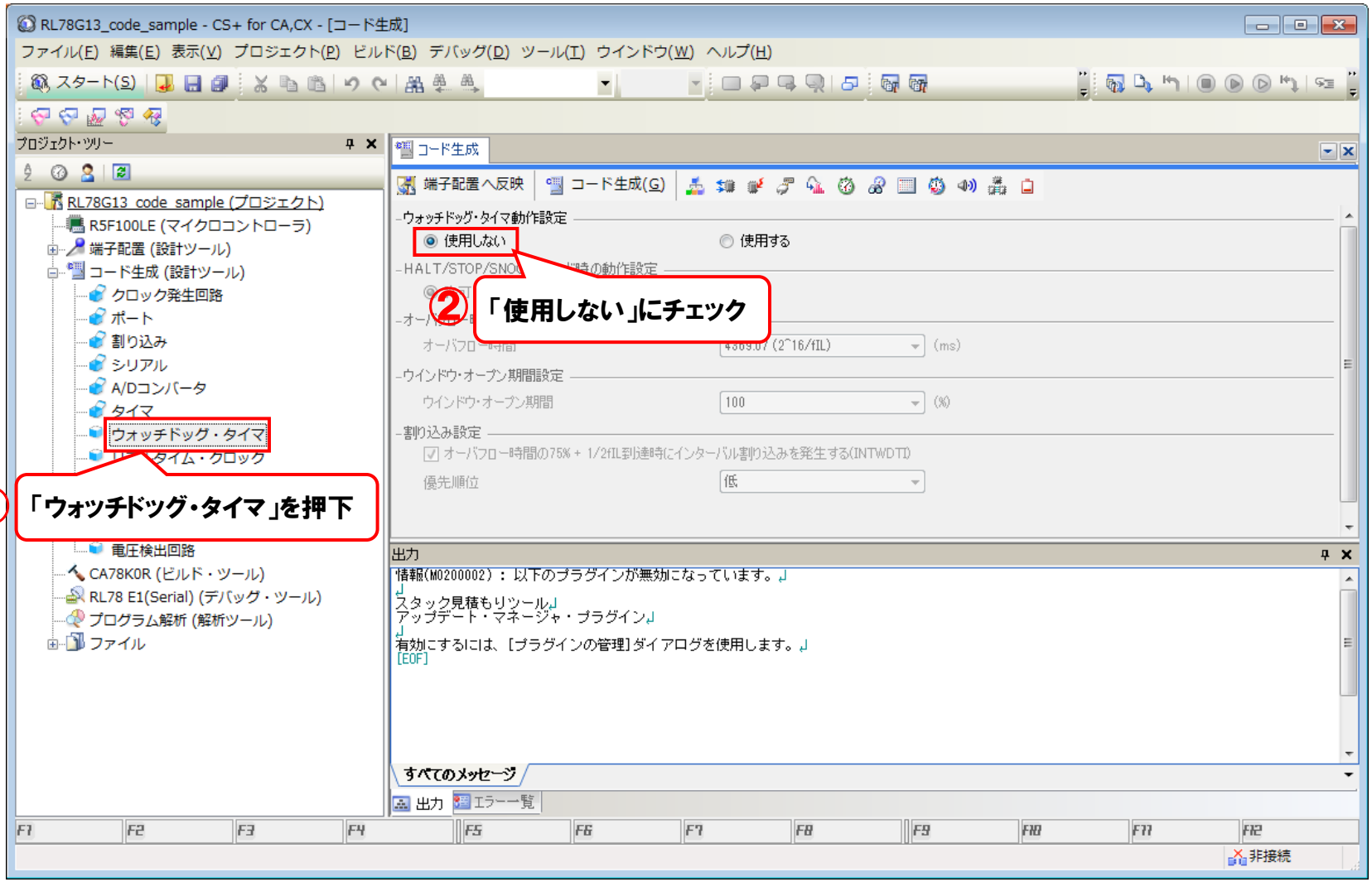
⑤ 高速システムクロックに「20」を入力

※ クロックはデフォルトのままでも動作します。
QB-R5F100LE-TBはX1,X2に20MHz発振子が搭載されているので、それを使う設定にします。

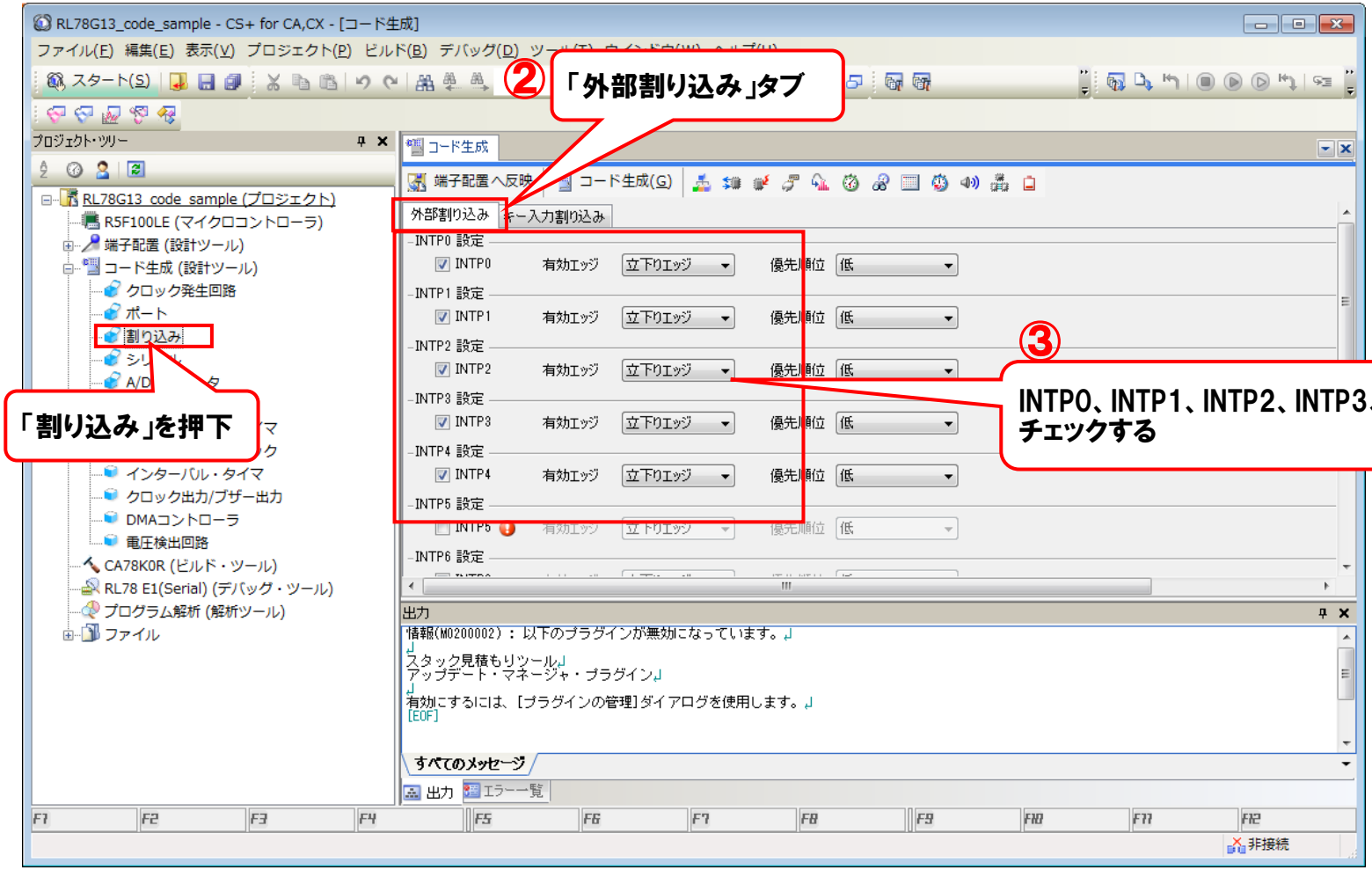
CS+ コード生成設定(システム)



CS+ コード生成設定（ウォッチドッグ・タイマ）

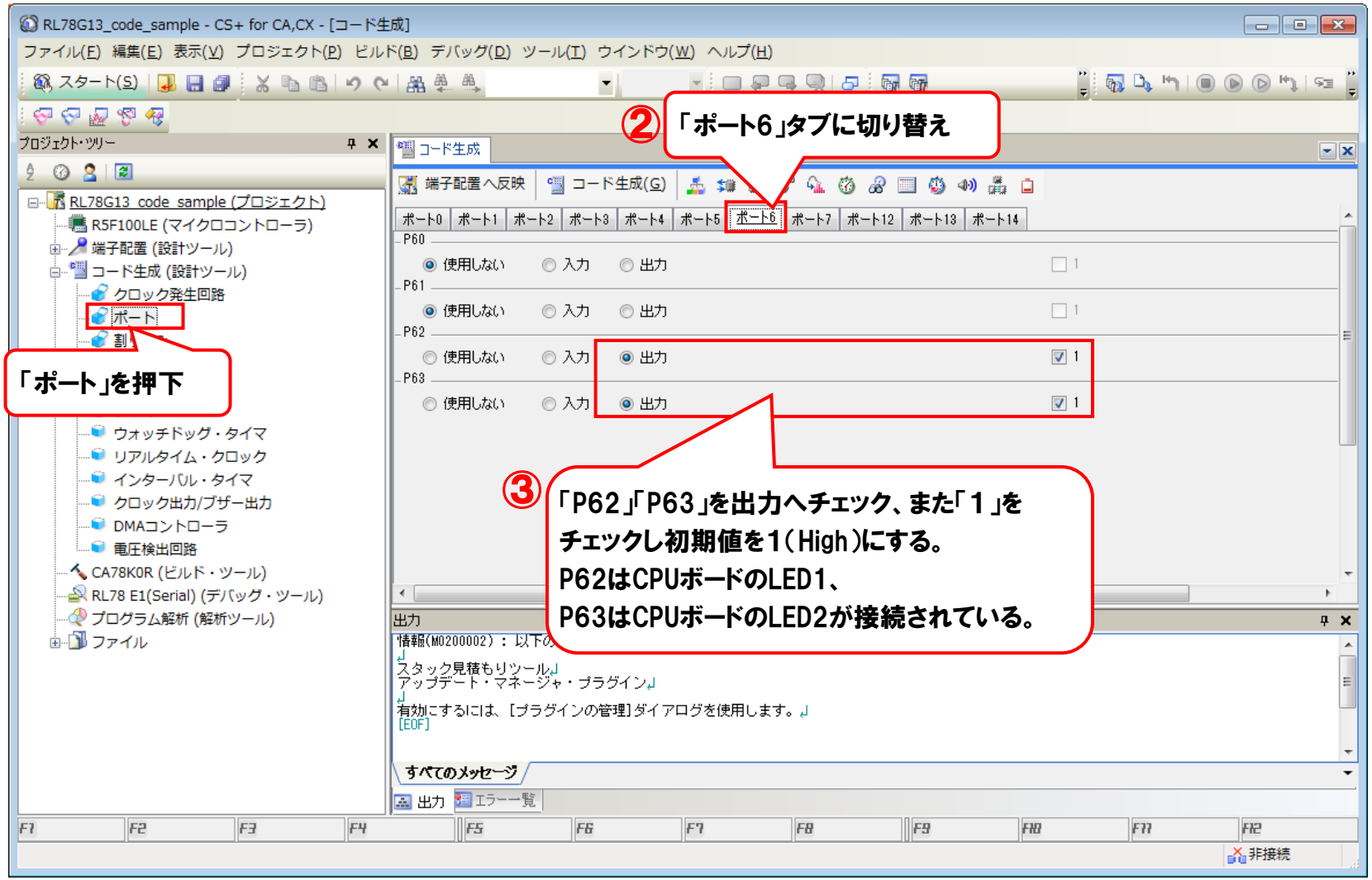


CS+ コード生成設定（割り込み）

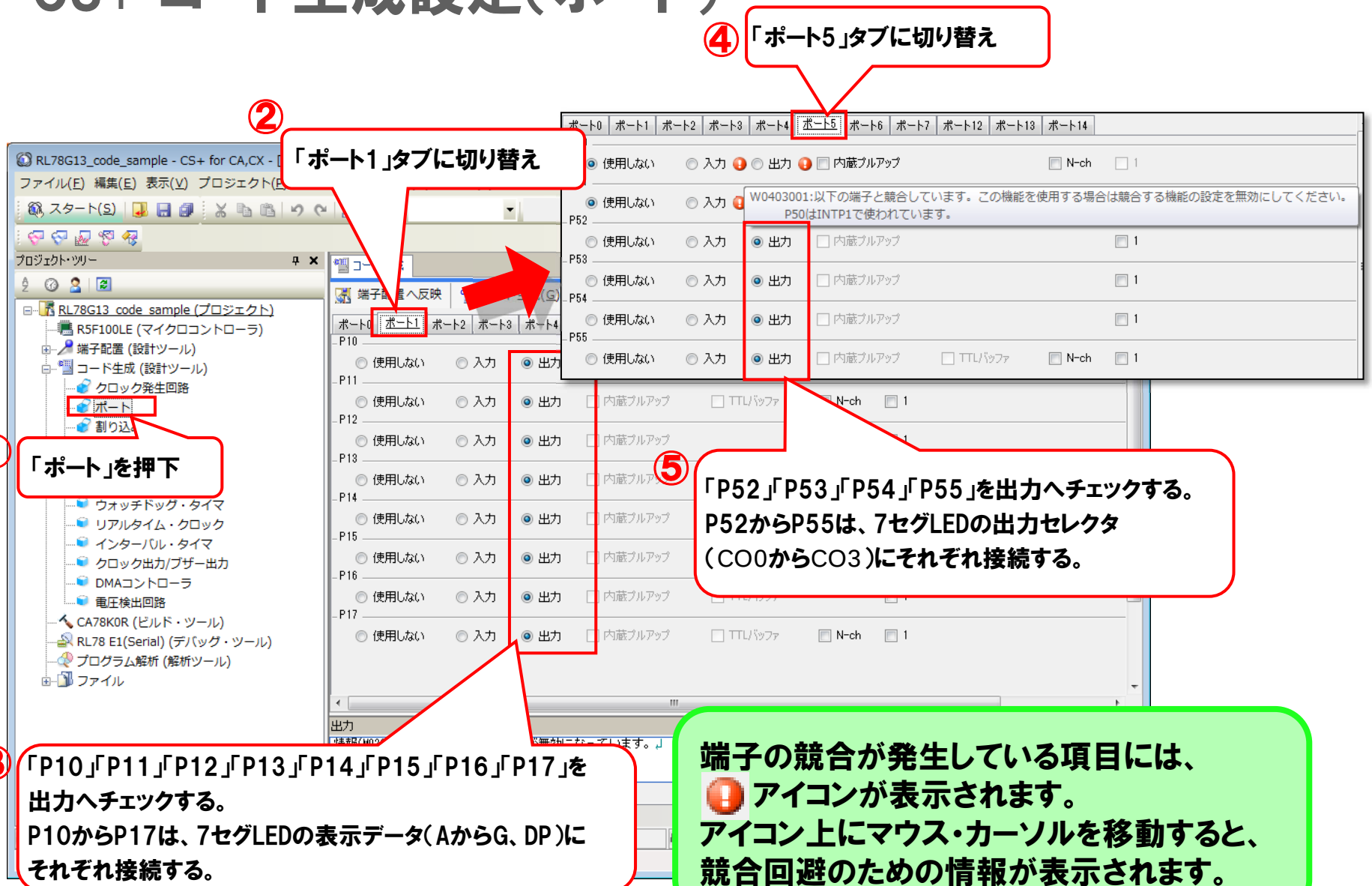


割り込み端子は、必要に応じて、ポートの内蔵プルアップ（該当するポートの内蔵プルアップにチェック）を使用してください。
（今回は、CPUボード、SWボードでプルアップされているので、設定不要です。）

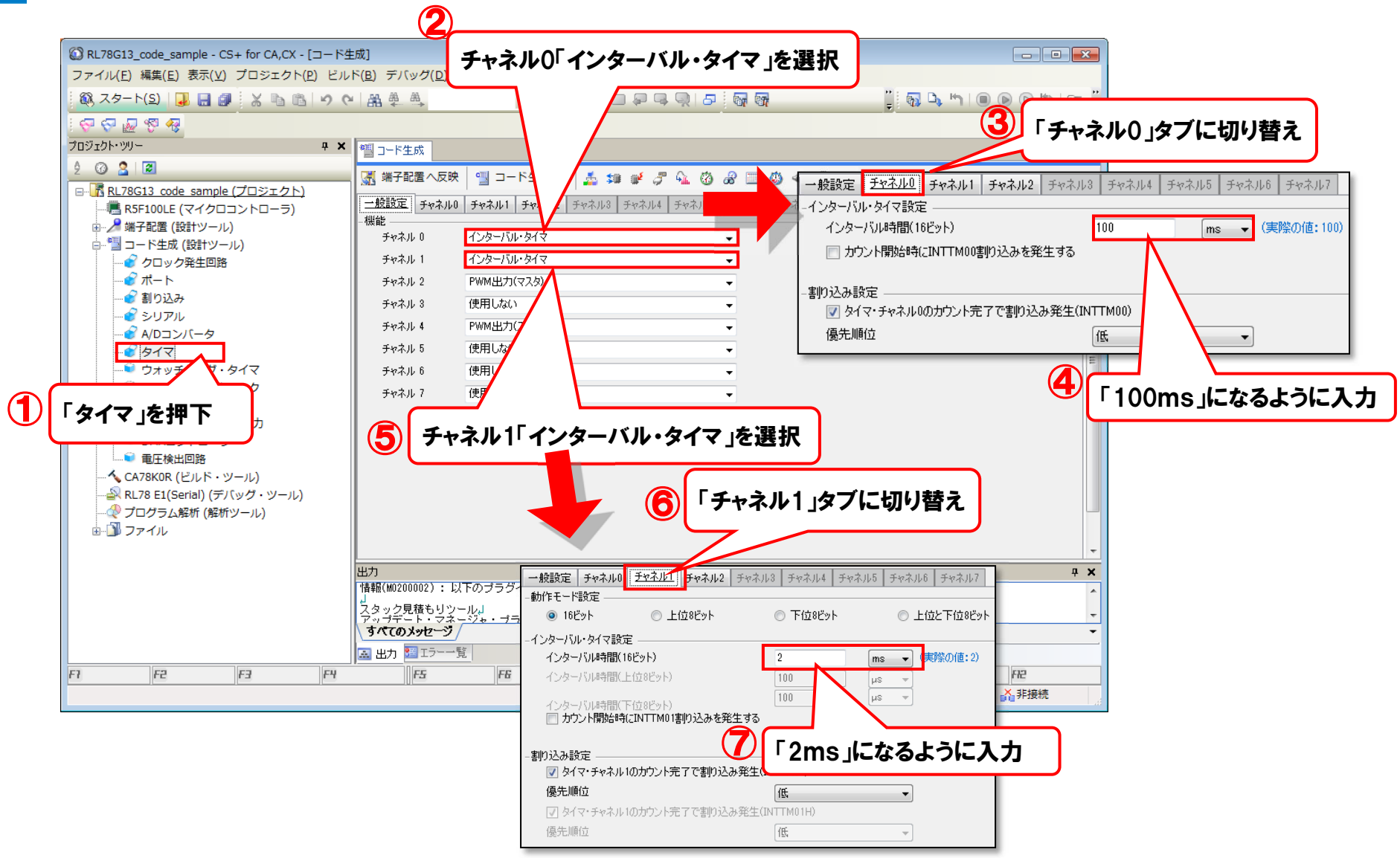
CS+ コード生成設定(ポート)



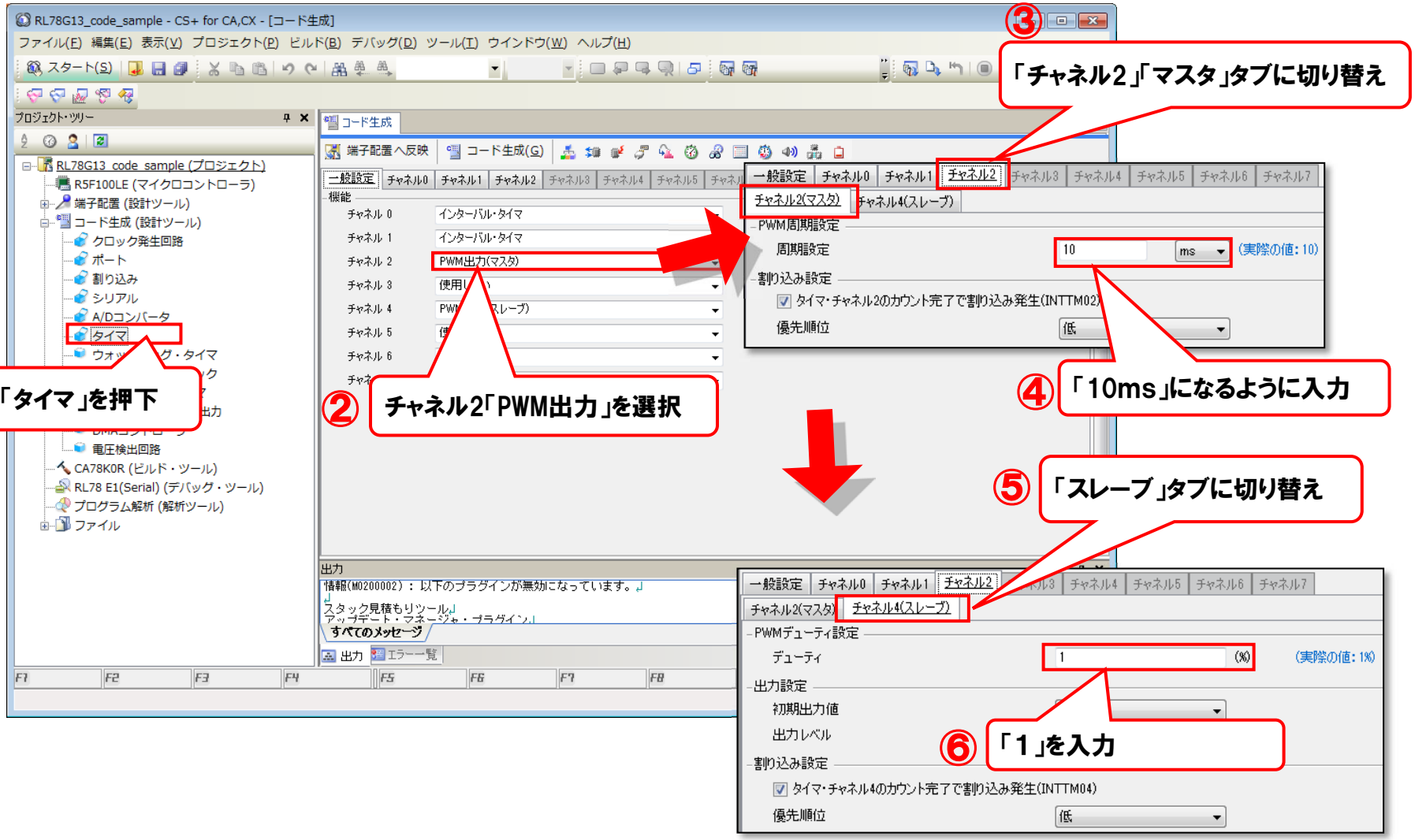
CS+ コード生成設定(ポート)



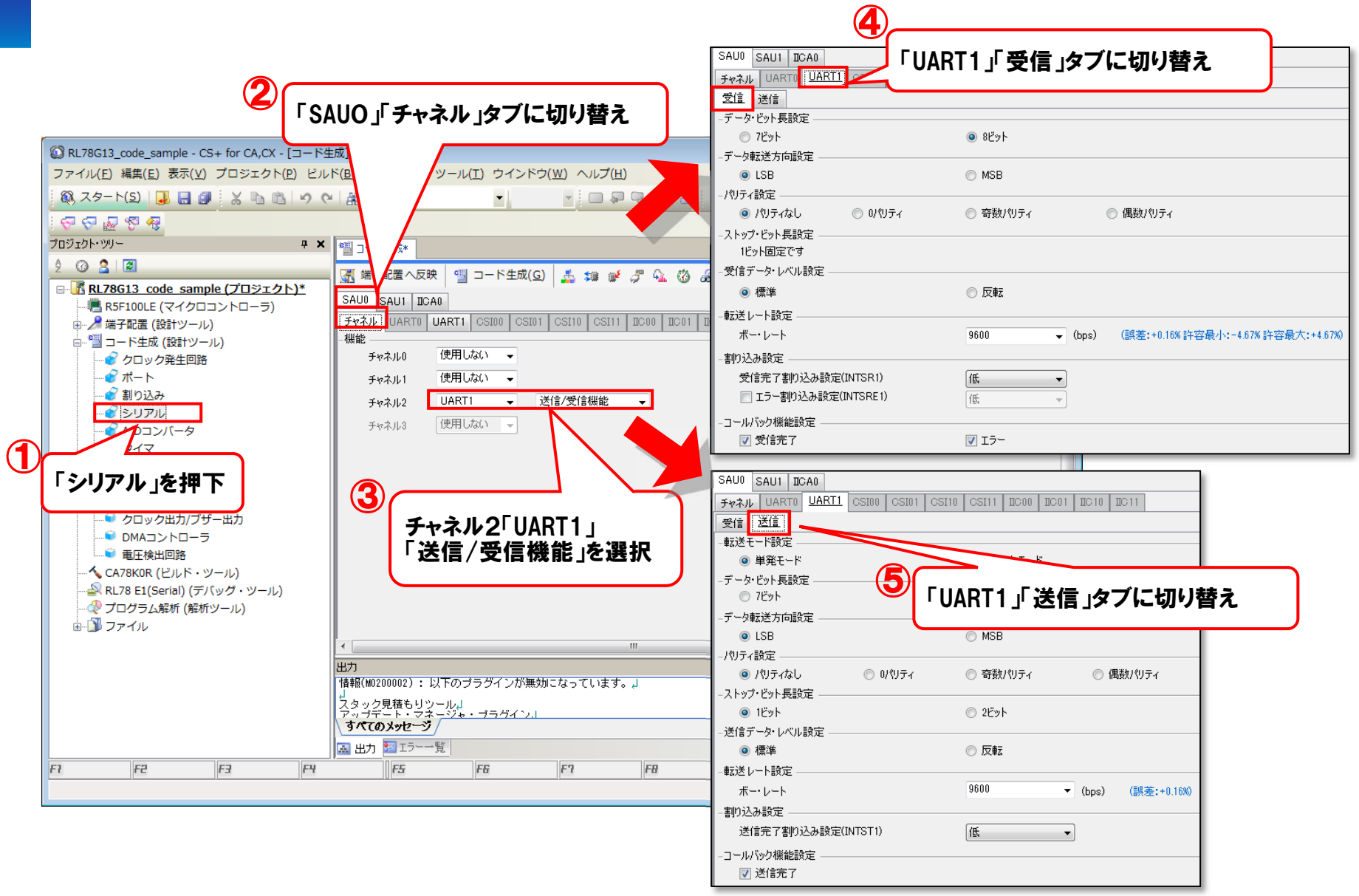
CS+ コード生成設定(タイマ)



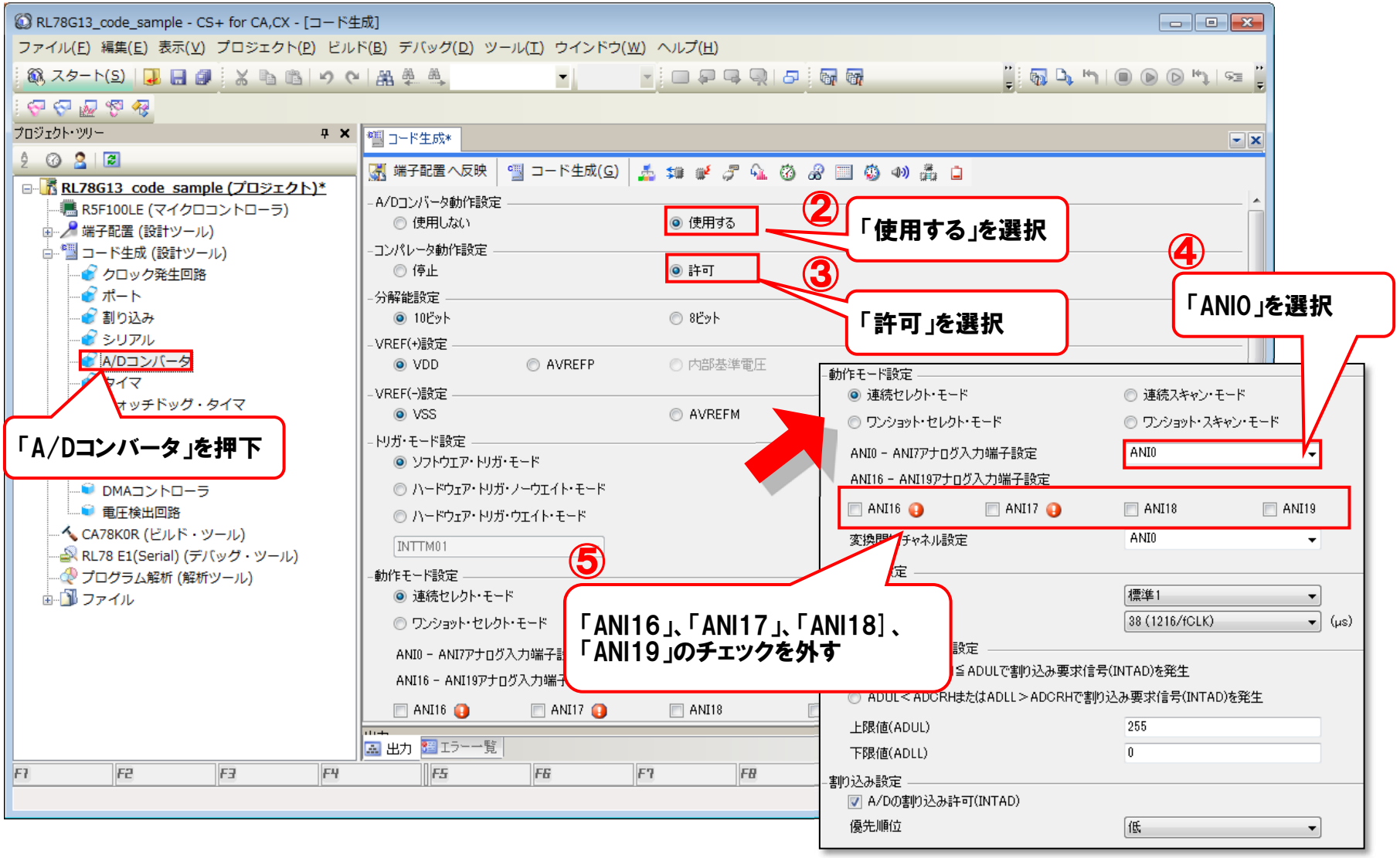
CS+ コード生成設定(タイマ)



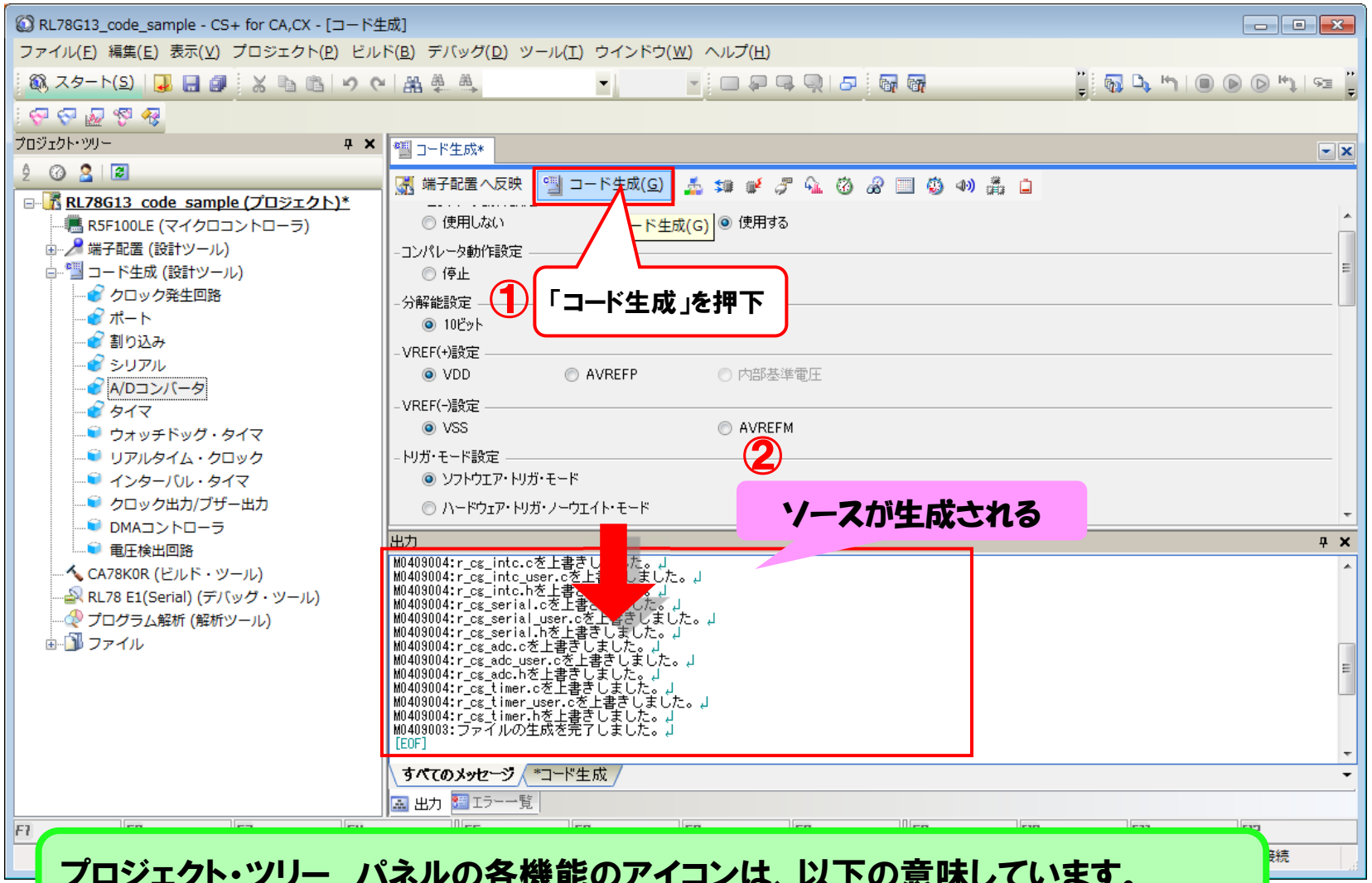
CS+ コード生成設定(シリアル)



CS+ コード生成設定(A/Dコンバータ)



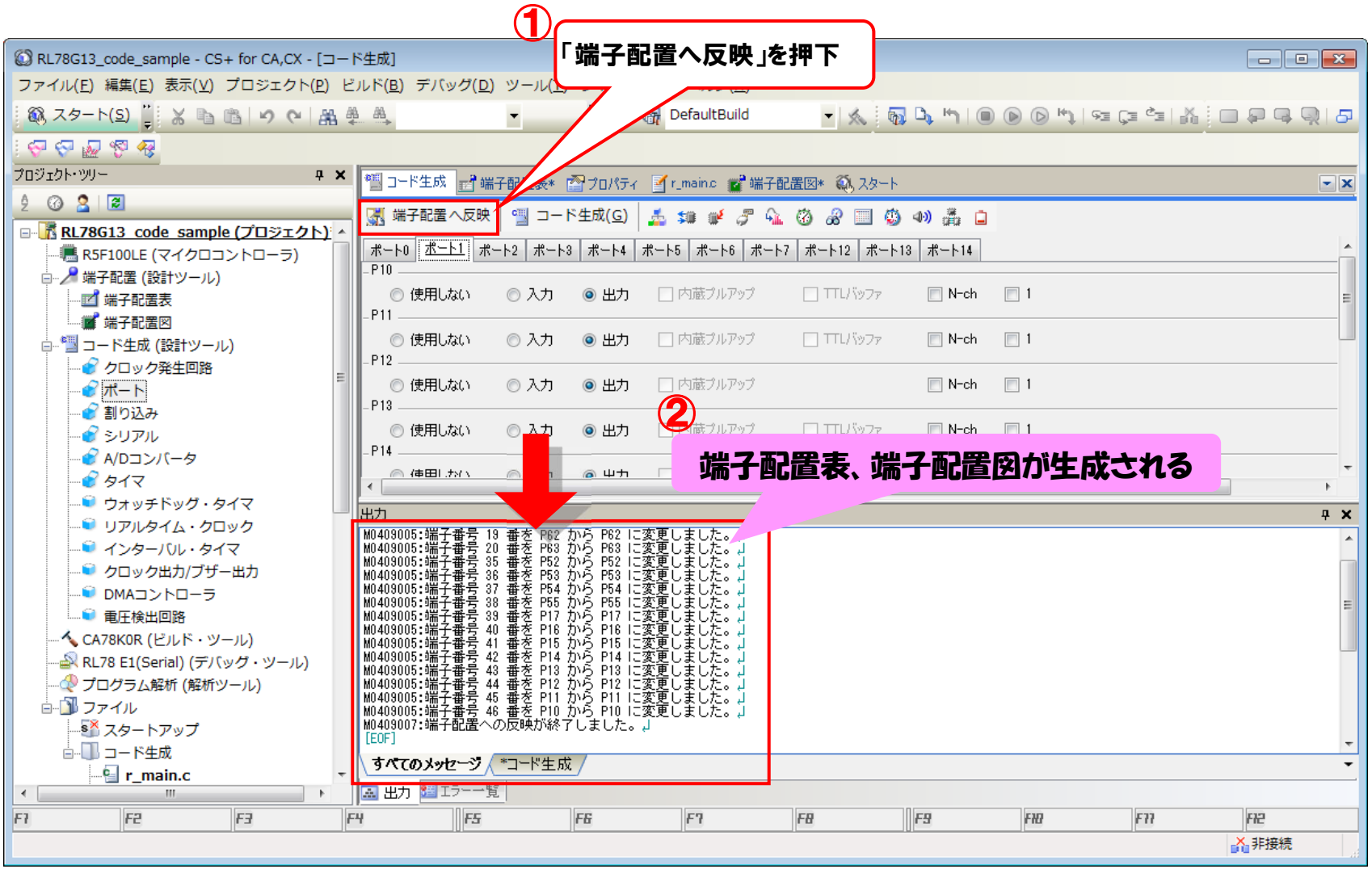
CS+ コード生成



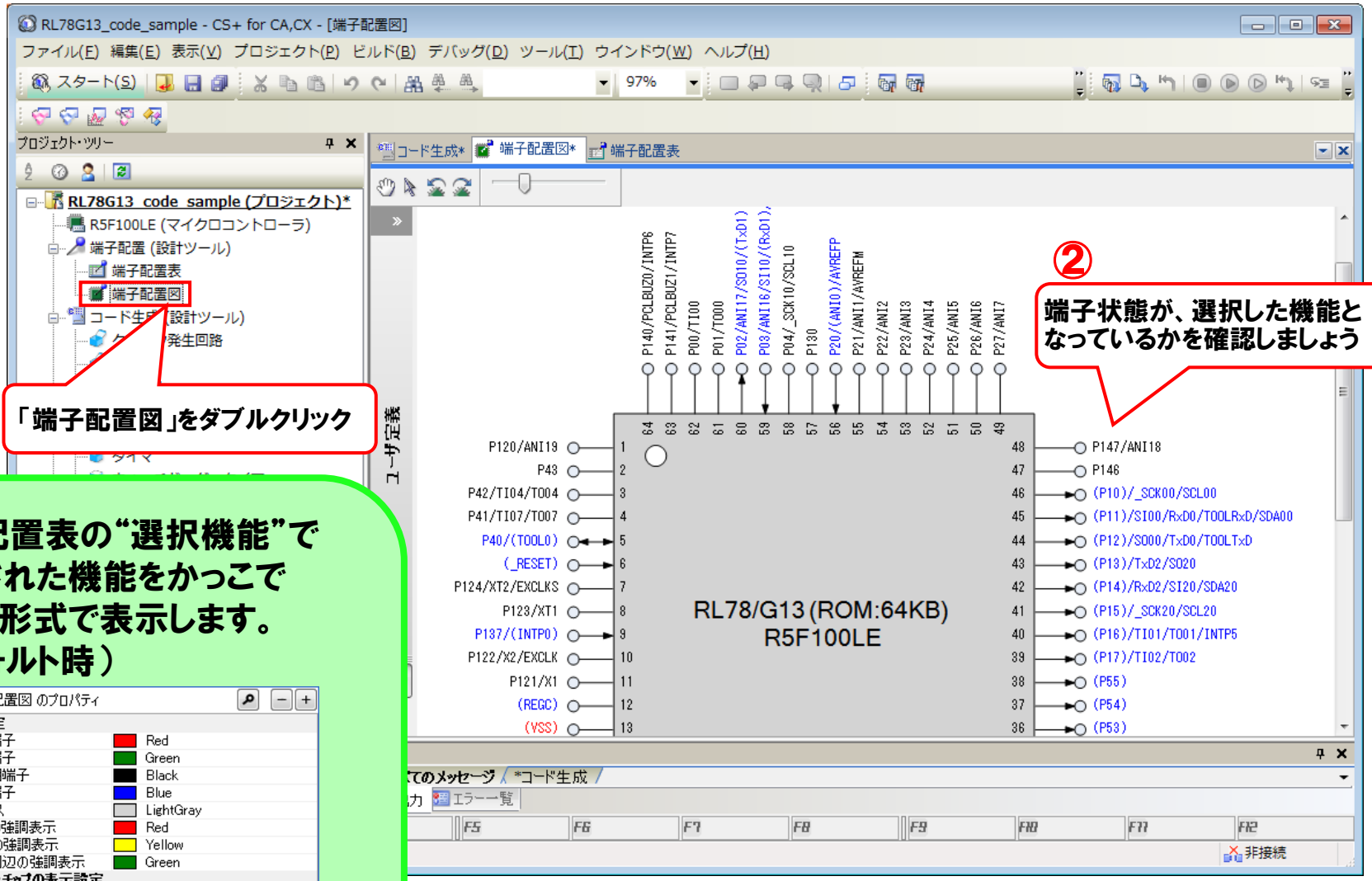
プロジェクト・ツリー　パネルの各機能のアイコンは、以下の意味しています。

:使用している機能　　:未使用の機能　　:設定内容に問題が発生

CS+ 端子配置へ反映


①
「端子配置へ反映」を押下
RL78G13_code_sample - CS+ for CA,CX - [コード生成]
端子配置へ反映
コード生成(G)
②
端子配置表、端子配置図が生成される
M0409005:端子番号 19 番を P62 から P62 に変更しました。
M0409005:端子番号 20 番を P63 から P63 に変更しました。
M0409005:端子番号 35 番を P52 から P52 に変更しました。
M0409005:端子番号 36 番を P53 から P53 に変更しました。
M0409005:端子番号 37 番を P54 から P54 に変更しました。
M0409005:端子番号 38 番を P55 から P55 に変更しました。
M0409005:端子番号 39 番を P17 から P17 に変更しました。
M0409005:端子番号 40 番を P16 から P16 に変更しました。
M0409005:端子番号 41 番を P15 から P15 に変更しました。
M0409005:端子番号 42 番を P14 から P14 に変更しました。
M0409005:端子番号 43 番を P13 から P13 に変更しました。
M0409005:端子番号 44 番を P12 から P12 に変更しました。
M0409005:端子番号 45 番を P11 から P11 に変更しました。
M0409005:端子番号 46 番を P10 から P10 に変更しました。
M0409007:端子配置への反映が終了しました。
[EOF]

CS+ 端子配置図

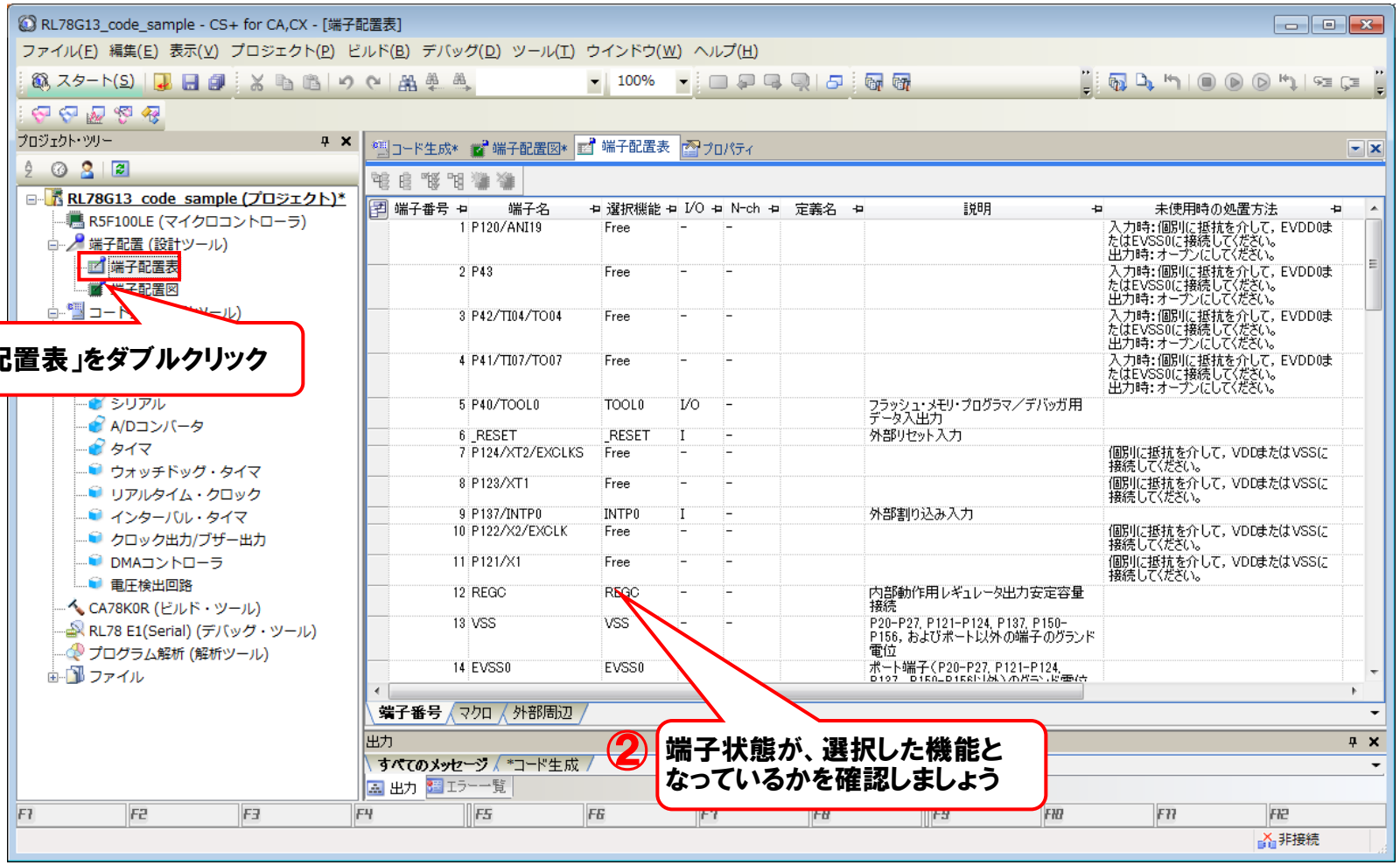


① 「端子配置図」をダブルクリック

② 端子状態が、選択した機能となっているかを確認しましょう

端子配置表の“選択機能”で選択された機能をかっこで括った形式で表示します。（デフォルト時）

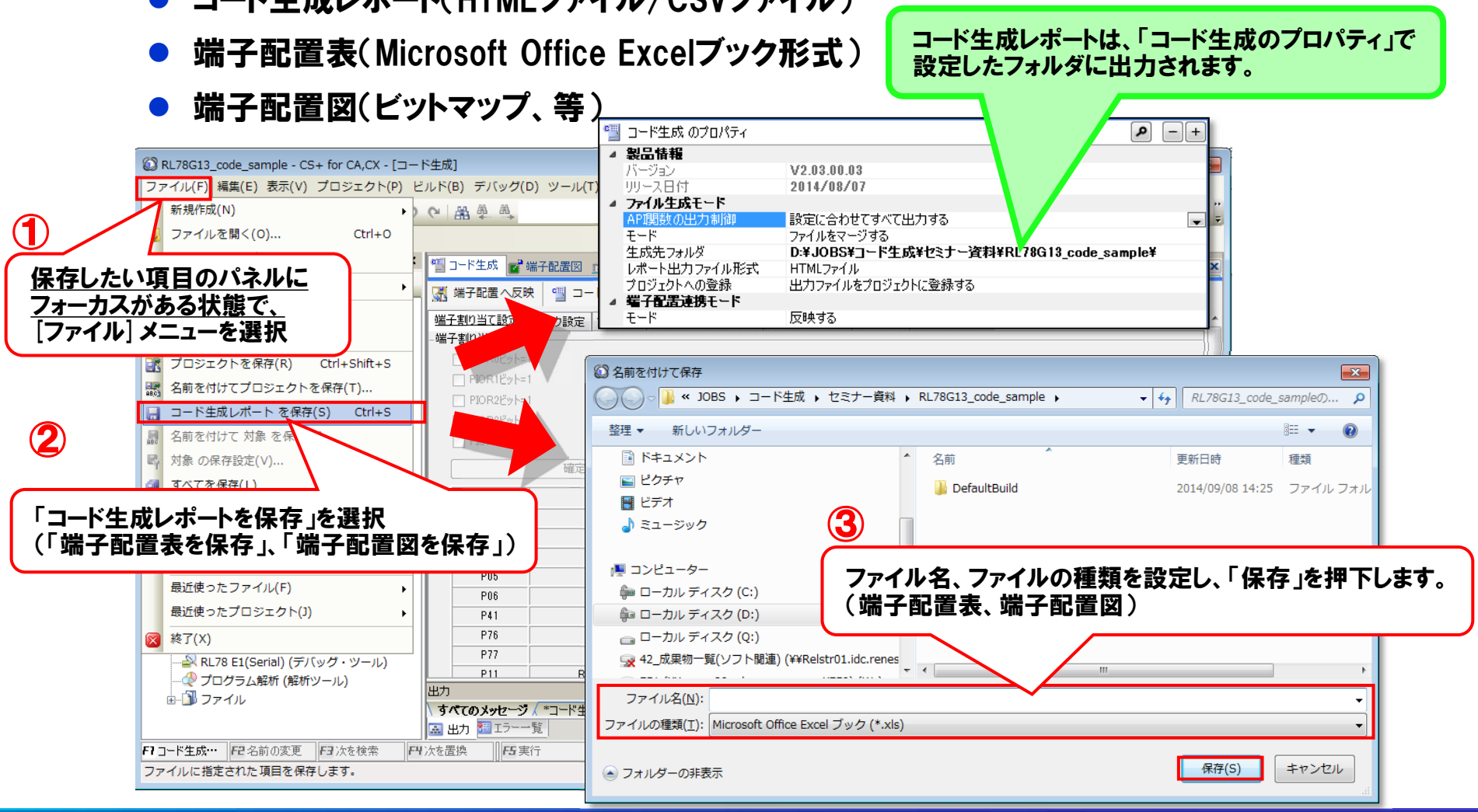
CS+ 端子配置表



端子配置表/端子配置図は、編集可能ですが、
その情報をコード生成、ソースに反映することはできません。

CS+ 設定した情報の保存

- **設定した情報は、ファイル出力可能です。**
  - **コード生成レポート（HTMLファイル／CSVファイル）**
  - **端子配置表（Microsoft Office Excelブック形式）**
  - **端子配置図（ビットマップ、等）**



CS+　コード生成で出力されたファイル

- **出力ファイル**

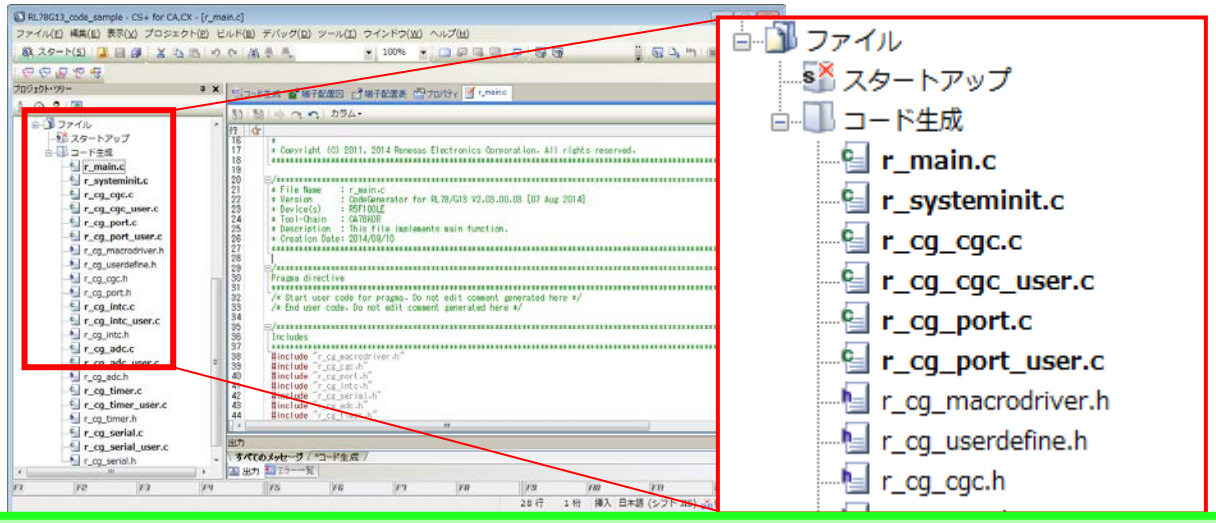
  GUIベースで設定した情報に応じたデバイス・ドライバ・プログラムを出力するとともに、main関数を含んだサンプル・プログラムやビルド環境一式も出力します。

  周辺機能は、以下のようなファイル名となります。

  r_cg_周辺機能名.c　　：初期化関数、API関数

  r_cg_周辺機能名_user.c　：割り込み関数、コールバック関数

  r_cg_周辺機能名.h　　：レジスタへの代入値マクロを定義



ユーザ・コードは、r_main.c、r_cg_周辺機能名_user.c、r_cg_userdefine.h に追記していきます。

CS+　コード生成で出力されたファイル

■ ユーザ記述コードの保護機能

コードを追記できる箇所には、ユーザ・コード記述用のコメントがあります。このコメント内にコードを記述すると、コード生成部で再度コード生成した場合にもユーザ追記の内容がマージ(保護)されます。ユーザ・コード記述用のコメントは、関数内、ファイル内の前方および後方に用意されています。

【ユーザ・コード記述用のコメント例】

```
/***********************************************************************************************************************
Global variables and functions
***********************************************************************************************************************/
/* Start user code for global. Do not edit comment generated here */
extern uint16_t gAdData;
/* End user code. Do not edit comment generated here */

/***********************************************************************************************************************
* Function Name: r_adc_interrupt
* Description  : This function is INTAD interrupt service routine.
* Arguments    : None
* Return Value : None
***********************************************************************************************************************/
__interrupt static void r_adc_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    R_ADC_Stop();
    R_ADC_Get_Result( &gAdData );
    /* 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 */
```

ユーザ・コード記述箇所

CS+ コード生成が出力するAPI関数(共通、周辺機能(初期化))

共通

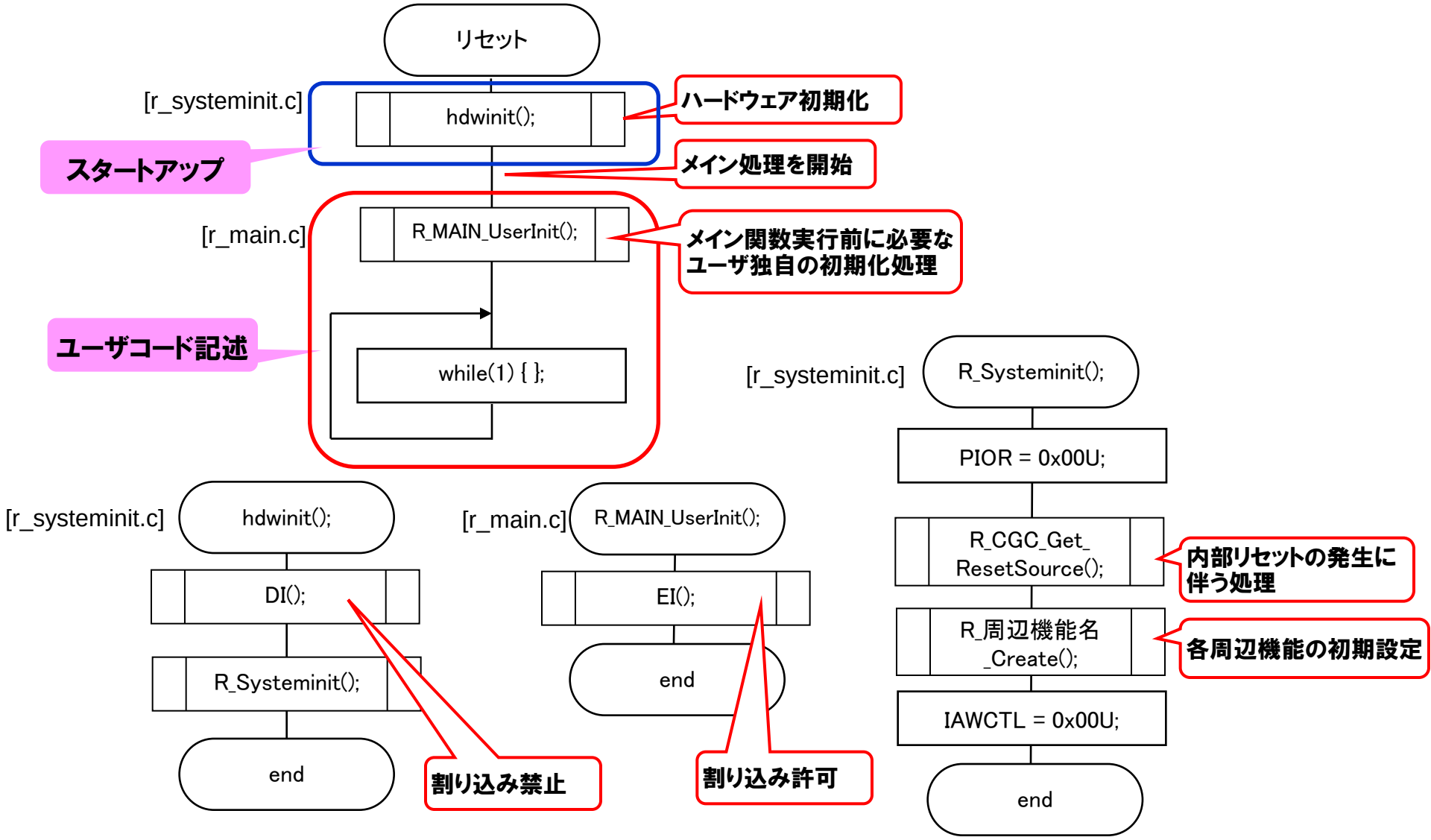
ファイル名	API関数名	機能概要
r_systemini.c	hdwinit	ルネサスエレクトロニクス製コンパイラ提供のスタートアップ・ルーチンから自動で呼ばれます。この関数内で R_Systeminit () を呼びます。
	R_Systeminit	hdwinit関数から自動で呼ばれます。この関数内で 、リセット要因を確認する関数、および各周辺機能の初期設定を行う関数 (R_周辺機能名_Create) を呼びます。
r_main.c	R_MAIN_UserInit	メイン関数実行前に必要なユーザ独自の初期化処理を行います。
	main	メイン関数
r_cg_macrodriver	-	全ソース・ファイルで共通使用するマクロを定義
r_cg_userdefine.h	-	空ファイル(ユーザ定義用)

周辺機能(初期化)

ファイル名	API関数名	機能概要
r_cg_周辺機能名.c	R_周辺機能名_Create	その機能を制御するうえで必要となる初期化処理を行います。 R_Systeminit 関数から自動で呼ばれます。
r_cg_周辺機能名_user.c	R_周辺機能名_Create_UserInit	その機能のユーザ独自の初期化処理を行います。
r_cg_周辺機能名.h	-	レジスタへの代入値マクロを定義

ファイル名、関数名は、コード生成の世代によって異なります。

CS+ コード生成が出力するAPI関数(共通)



CS+　コード生成が出力するAPI関数(周辺機能)

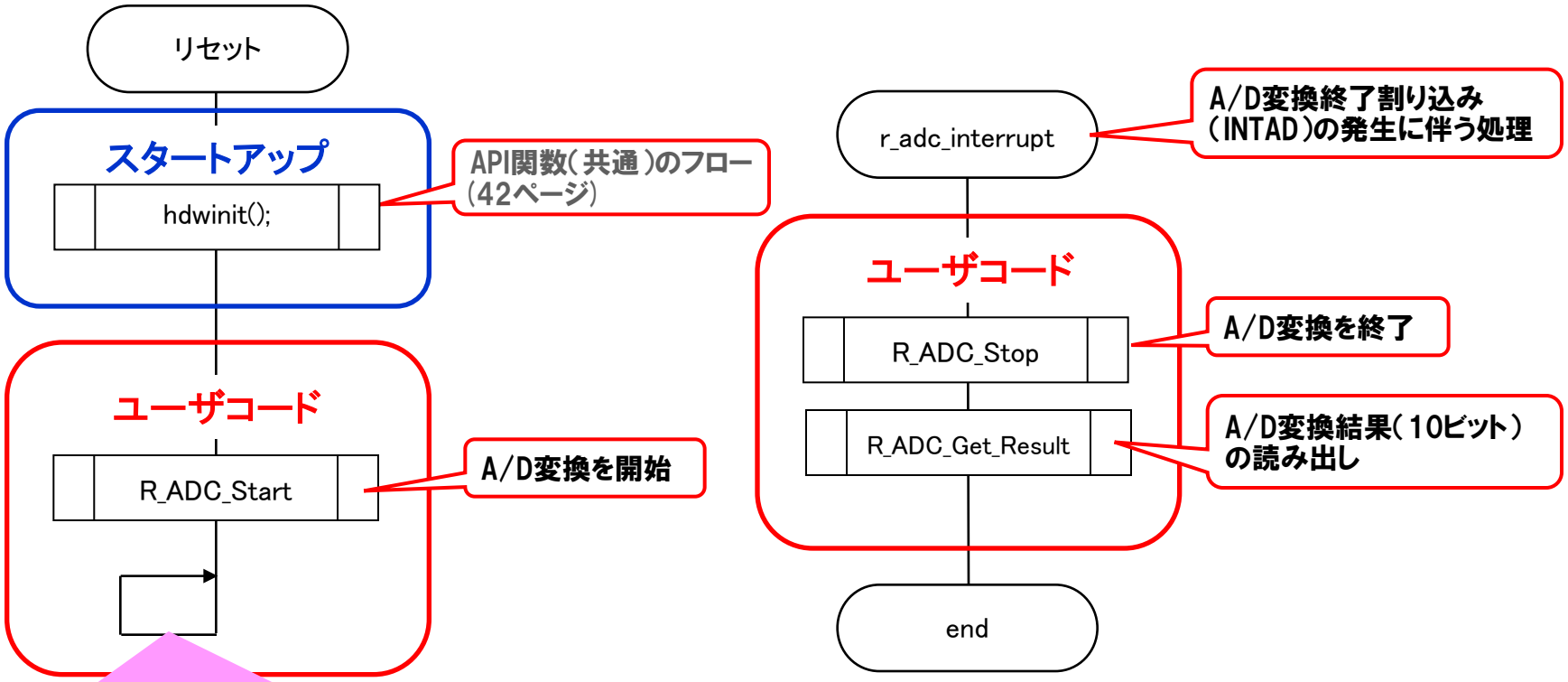
A/Dコンバータ　API関数一覧

ファイル名	API関数名	機能概要
r_cg_adc.c	R_ADC_Create	A/Dコンバータの機能を制御するうえで必要となる初期化処理を行います。
	R_ADC_Set_OperationOn	電圧コンパレータを動作許可状態に設定します。
	R_ADC_Set_OperationOff	電圧コンパレータを動作停止状態に設定します。
	R_ADC_Start	A/D変換を開始します。
	R_ADC_Stop	A/D変換を終了します。
	R_ADC_Set_PowerOff	A/Dコンバータに対するクロック供給を停止します。
	R_ADC_Set_ADChannel	A/D変換するアナログ電圧の入力端子を設定します。
	R_ADC_Set_SnoozeOn	STOPモードからSNOOZEモードへの切り替えを許可します。
	R_ADC_Set_SnoozeOff	STOPモードからSNOOZEモードへの切り替えを禁止します。
	R_ADC_Set_TestChannel	A/Dコンバータの動作モードを設定します。
	R_ADC_Get_Result	A/D変換結果(10ビット)を読み出します。
	R_ADC_Get_Result_8bit	A/D変換結果(8ビット:10ビット分解能の上位8ビット)を読み出します。
r_cg_adc_user.c	R_ADC_Create_UserInit	A/Dコンバータに関するユーザ独自の初期化処理を行います。
	r_adc_interrupt	A/D変換終了割り込み(INTAD)の発生に伴う処理を行います。

※　色塗り箇所は、サンプルプログラムで使用しているAPIを意味します。

CS+ コード生成が出力するAPI関数(周辺機能)

A/Dコンバータの使用例



A/D変換が終了すると割り込み(INTAD)が発生、
現在の処理(メイン関数)を中断し、割り込み処理を実行します。
割り込み処理が終了すると、再度メイン関数に戻ります。

コード生成が出力するAPI関数では、
割り込みを使用し、機能を実現して
います。

CS+　コード生成が出力するAPI関数(周辺機能)

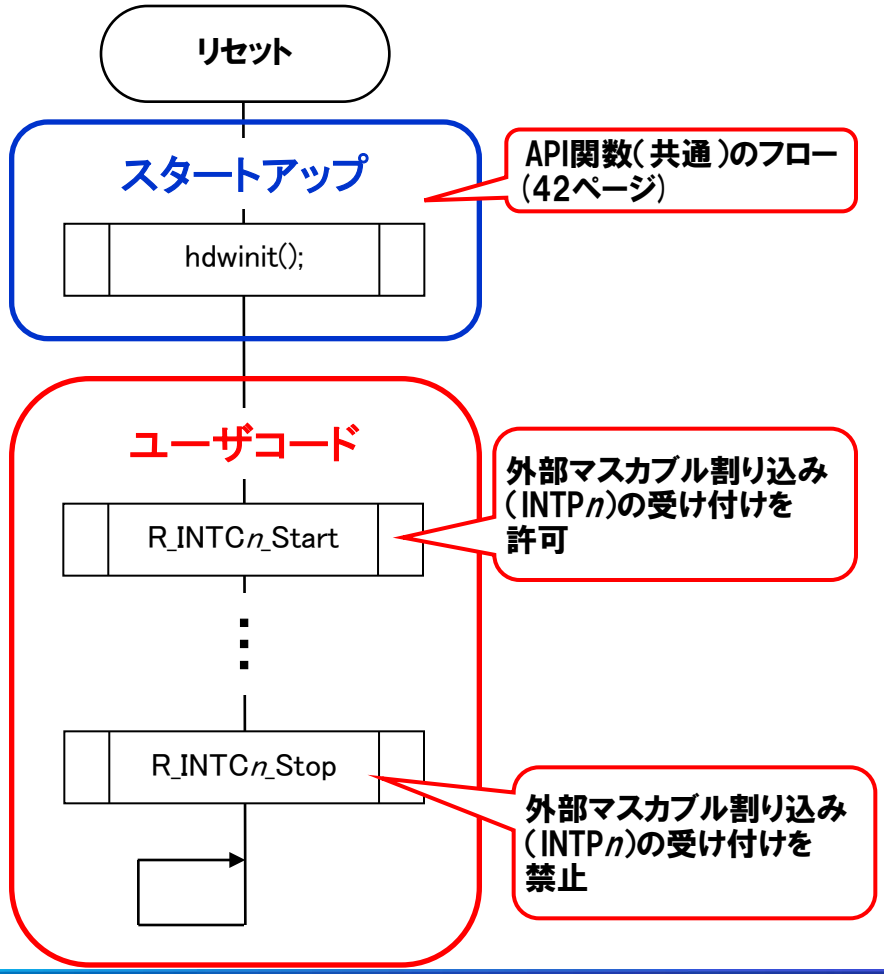
■ 外部マスカブル割り込み API関数一覧

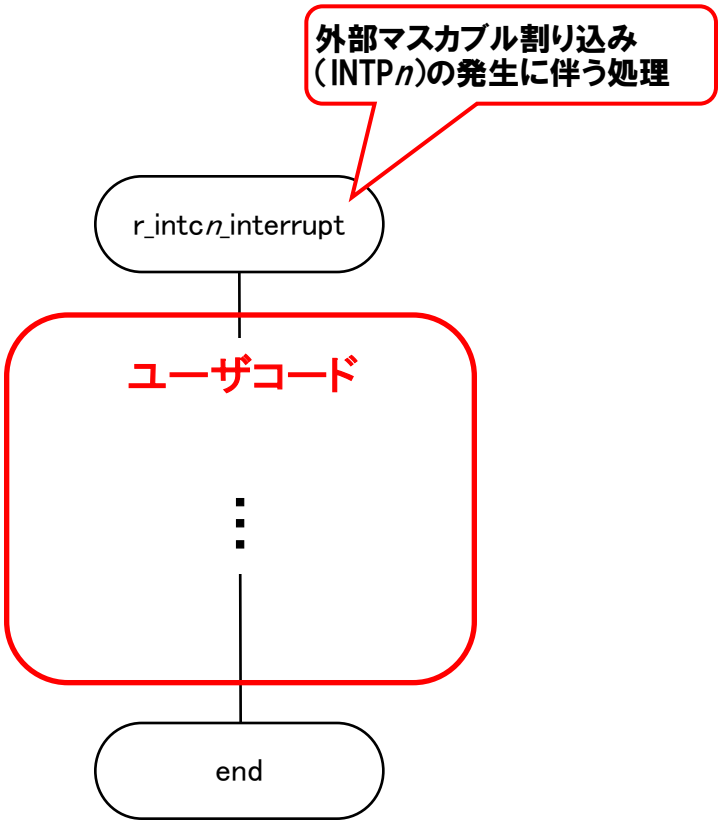
ファイル名	API関数名	機能概要
r_cg_intc.c	R_INTC_Create	外部マスカブル割り込み(INTP*n*／INTPLR*n*)の機能を制御するうえで必要となる初期化処理を行います。
	R_INTC*n*_Start	外部マスカブル割り込み(INTP*n*)の受け付けを許可します。
	R_INTC*n*_Stop	外部マスカブル割り込み(INTP*n*)の受け付けを禁止します。
	R_INTCLR*n*_Start	外部マスカブル割り込み(INTPLR*n*)の受け付けを許可します。
	R_INTCLR*n*_Stop	外部マスカブル割り込み(INTPLR*n*)の受け付けを禁止します。
r_cg_intc_user.c	R_INTC_Create_UserInit	外部マスカブル割り込み(INTP*n*／INTPLR*n*)に関するユーザ独自の初期化処理を行 います。
	r_intc*n*_interrupt	外部マスカブル割り込み(INTP*n*)の発生に伴う処理を行います。
	r_intclr*n*_interrupt	外部マスカブル割り込み(INTPLR*n*)の発生に伴う処理を行います。

※ 色塗り箇所は、サンプルプログラムで使用しているAPIを意味します。
※ *n* は、割り込み要因番号を意味します。

CS+ コード生成が出力するAPI関数(周辺機能)

外部マスカブル割り込みの使用例





※ *n* は、割り込み要因番号を意味します。

CS+　コード生成が出力するAPI関数（周辺機能）

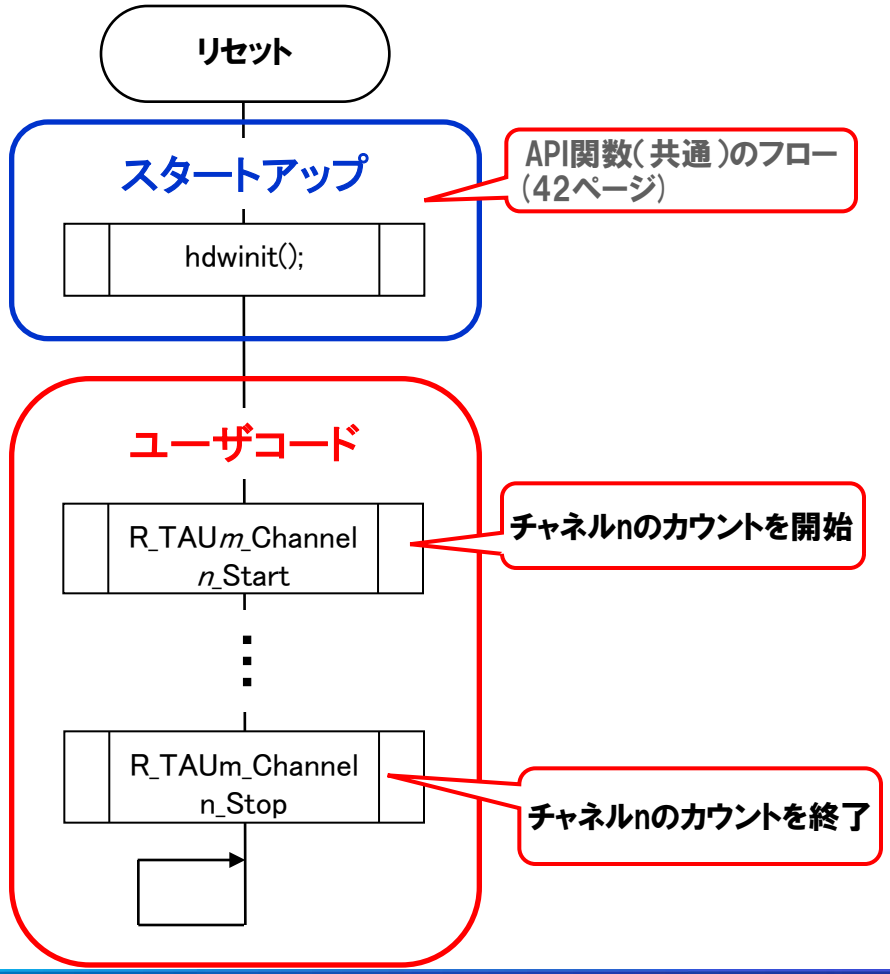
タイマ・アレイ・ユニット　API関数一覧

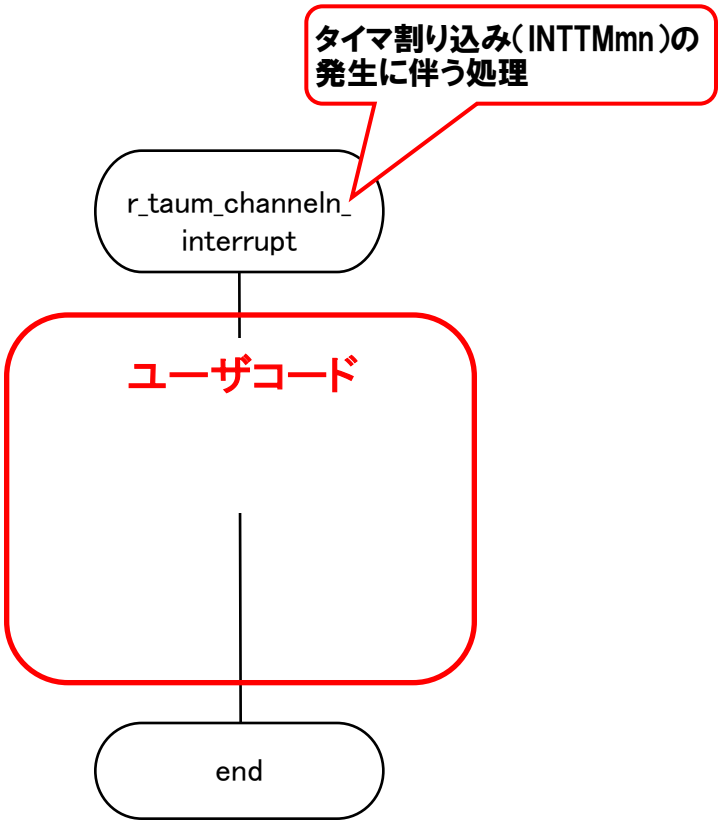
ファイル名	API関数名	機能概要
r_cg_timer.c	R_TAU*m*_Create	タイマ・アレイ・ユニットの機能を制御するうえで必要となる初期化処理を行います。
	R_TAU*m*_Channel*n*_Start	チャネル*n*のカウントを開始します。
	R_TAU*m*_Channel*n*_Higher8bits_Start	チャネル1，またはチャンネル3のカウント（上位8ビット）を開始します。
	R_TAU*m*_Channel*n*_Lower8bits_Start	チャネル1，またはチャンネル3のカウント（下位8ビット）を開始します。
	R_TAU*m*_Channel*n*_Stop	チャネル*n*のカウントを終了します。
	R_TAU*m*_Channel*n*_Higher8bits_Stop	チャネル1，またはチャンネル3のカウント（上位8ビット）を終了します。
	R_TAU*m*_Channel*n*_Lower8bits_Stop	チャネル1，またはチャンネル3のカウント（下位8ビット）を終了します。
	R_TAU*m*_Set_PowerOff	タイマ・アレイ・ユニットに対するクロック供給を停止します。
	R_TAU*m*_Channel*n*_Get_PulseWidth	TI*mn*端子に対する入力信号（入力パルス）のパルス間隔，またはハイ／ロウ・レベルの測定幅を獲得します。
	R_TAU*m*_Channel*n*_Set_SoftwareTriggerOn	ワンショット・パルス出力のためのトリガ（ソフトウエア・トリガ）を発生させます。
r_cg_timer_user.c	R_TAU*m*_Create_UserInit	タイマ・アレイ・ユニットに関するユーザ独自の初期化処理を行います。
	r_tau*m*_channel*n*_interrupt	タイマ割り込み（INTTM*mn*）の発生に伴う処理を行います。
	r_tau*m*_channel*n*_higher8bits_interrupt	タイマ割り込み（INTTM*mn*H）の発生に伴う処理を行います。

※　色塗り箇所は、サンプルプログラムで使用しているAPIを意味します。
※　*m* はユニット番号を、*n* はチャネル番号を意味します。

CS+ コード生成が出力するAPI関数(周辺機能)

タイマ・アレイ・ユニットの使用例





※ *m* はユニット番号を、*n* はチャネル番号を意味します。

CS+ コード生成が出力するAPI関数(周辺機能)

シリアル・アレイ・ユニット API関数一覧(1/2)

ファイル名	API関数名	機能概要
r_cg_serial.c	R_SAU*m*_Create	シリアル・アレイ・ユニット, およびシリアル・インタフェースの機能を制御するうえで必要となる初期化処理を行います。
	R_SAU*m*_Set_PowerOff	シリアル・アレイ・ユニットに対するクロック供給を停止します。
	R_SAU*m*_Set_SnoozeOn	STOPモードからSNOOZEモードへの切り替えを許可します。
	R_SAU*m*_Set_SnoozeOff	STOPモードからSNOOZEモードへの切り替えを禁止します。
	R_UART*n*_Create	シリアル・インタフェース(UART)用チャネルの初期化処理を行います。
	R_UART*n*_Start	UART通信を待機状態にします。
	R_UART*n*_Stop	UART通信を終了します。
	R_UART*n*_Send	データのUART送信を開始します。
	R_UART*n*_Receive	データのUART受信を開始します。
	R_CSI*mn*_Create	シリアル・インタフェース(CSI)用チャネルの初期化処理を行います。
	R_CSI*mn*_Start	CSI通信を待機状態にします。
	R_CSI*mn*_Stop	CSI通信を終了します。
	R_CSI*mn*_Send	データのCSI送信を開始します。
	R_CSI*mn*_Receive	データのCSI受信を開始します。
	R_CSI*mn*_Send_Receive	データのCSI送受信を開始します。
	R_IIC*mn*_Create	シリアル・インタフェース(簡易IIC)用チャネルの初期化処理を行います。
	R_IIC*mn*_StartCondition	スタート・コンディションを発生させます。
	R_IIC*mn*_StopCondition	ストップ・コンディションを発生させます。
	R_IIC*mn*_Stop	簡易IIC通信を終了します。
	R_IIC*mn*_Master_Send	簡易IICマスタ送信を開始します。
	R_IIC*mn*_Master_Receive	簡易IICマスタ受信を開始します。

※ 色塗り箇所は、サンプルプログラムで使用しているAPIを意味します。
※ *m* はユニット番号を、*n* はチャネル番号を意味します。

CS+ コード生成が出力するAPI関数(周辺機能)

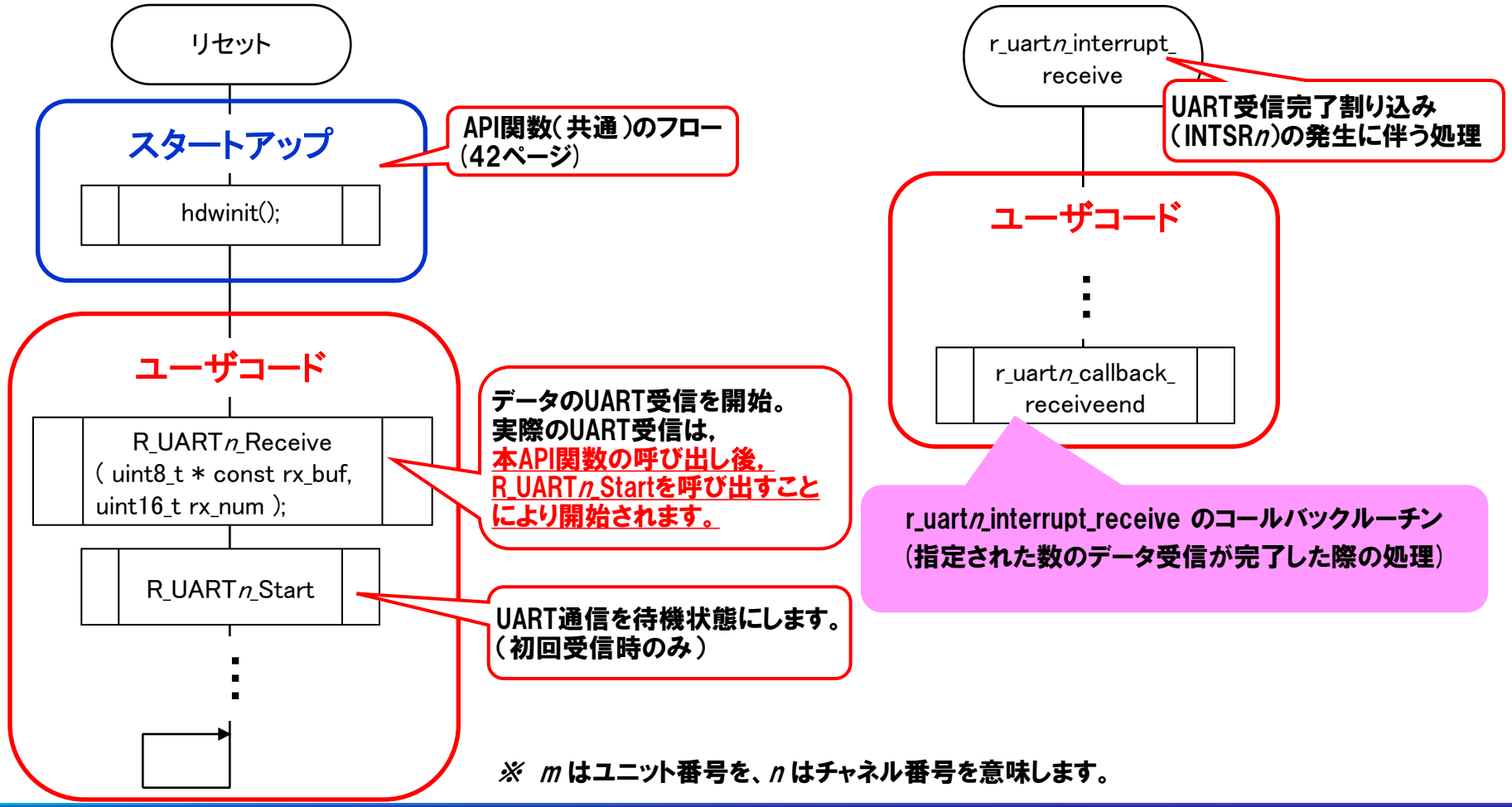
シリアル・アレイ・ユニット API関数一覧(2/2)

ファイル名	API関数名	機能概要
r_cg_serial_user.c	R_SAU*m*_Create_UserInit	シリアル・アレイ・ユニット, およびシリアル・インタフェースに関するユーザ独自の初期化処理を行います。
	r_uart*n*_interrupt_send	UART送信完了割り込み(INTST*n*)の発生に伴う処理を行います。
	r_uart*n*_interrupt_receive	UART受信完了割り込み(INTSR*n*)の発生に伴う処理を行います。
	r_uart*n*_interrupt_error	受信エラー割り込み(INTSRE*n*)の発生に伴う処理を行います。
	r_uart*n*_callback_sendend	UART送信完了割り込み(INTST*n*)の発生に伴う処理を行います。
	r_uart*n*_callback_receiveend	UART受信完了割り込み(INTSR*n*)の発生に伴う処理を行います。
	r_uart*n*_callback_error	UART受信エラー割り込み(INTSRE*n*)の発生に伴う処理を行います。
	r_uart*n*_callback_softwareoverrun	オーバラン・エラーの検出に伴う処理を行います。
	r_csi*mn*_interrupt	CSI通信完了割り込み(INTCSI*mn*)の発生に伴う処理を行います。
	r_csi*mn*_callback_sendend	CSI送信完了割り込み(INTCSI*mn*)の発生に伴う処理を行います。
	r_csi*mn*_callback_receiveend	CSI受信完了割り込み(INTCSI*mn*)の発生に伴う処理を行います。
	r_csi*mn*_callback_error	CSI受信エラー割り込み(INTSRE*n*)の発生に伴う処理を行います。
	r_iic*mn*_interrupt	簡易IIC通信完了割り込み(INTIIC*mn*)の発生に伴う処理を行います。
	r_iic*mn*_callback_master_sendend	簡易IICマスタ送信完了割り込み(INTIIC*mn*)の発生に伴う処理を行います。
	r_iic*mn*_callback_master_receiveend	簡易IICマスタ受信完了割り込み(INTIIC*mn*)の発生に伴う処理を行います。
	r_iic*mn*_callback_master_error	パリティ・エラー(ACKエラー)の検出に伴う処理を行います。

※ 色塗り箇所は、サンプルプログラムで使用しているAPIを意味します。
※ *m* はユニット番号を、*n* はチャネル番号を意味します。

CS+　コード生成が出力するAPI関数(周辺機能)

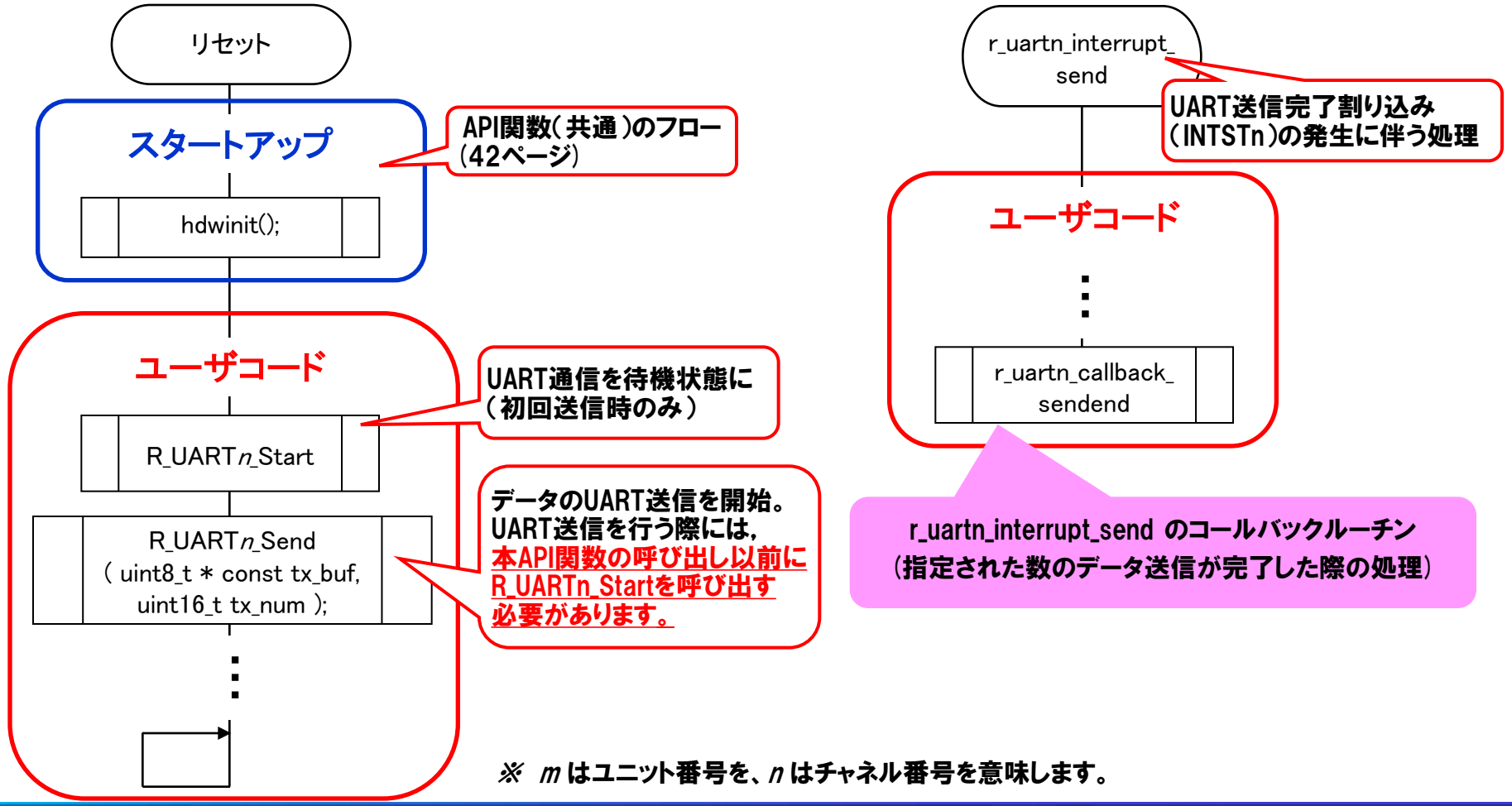
■ シリアル・アレイ・ユニット(シリアル・インタフェース(UART)受信)の使用例



※ *m* はユニット番号を、*n* はチャネル番号を意味します。

CS+　コード生成が出力するAPI関数(周辺機能)

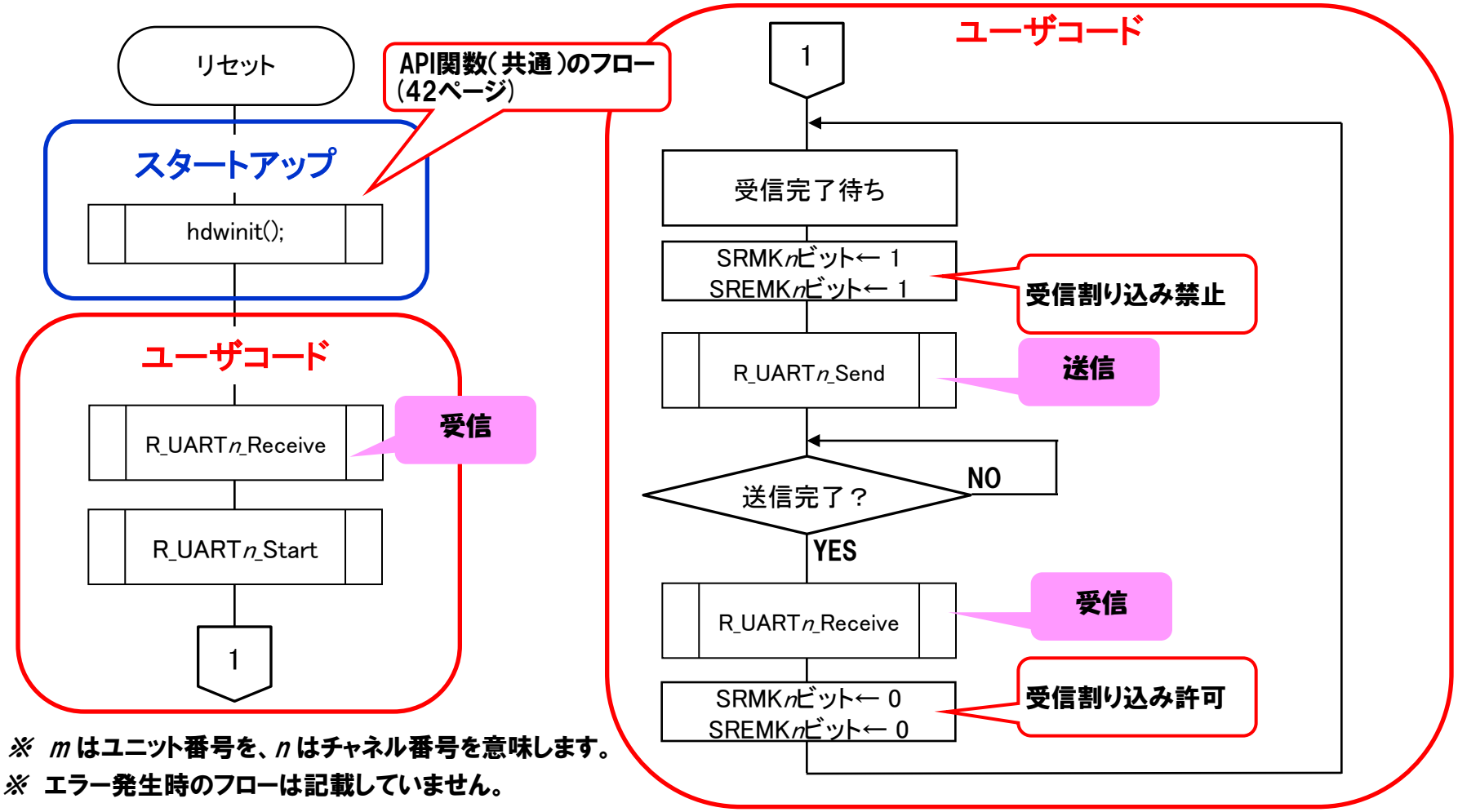
シリアル・アレイ・ユニット(シリアル・インタフェース(UART)送信)の使用例



※ *m* はユニット番号を、*n* はチャネル番号を意味します。

CS+　コード生成が出力するAPI関数(周辺機能)

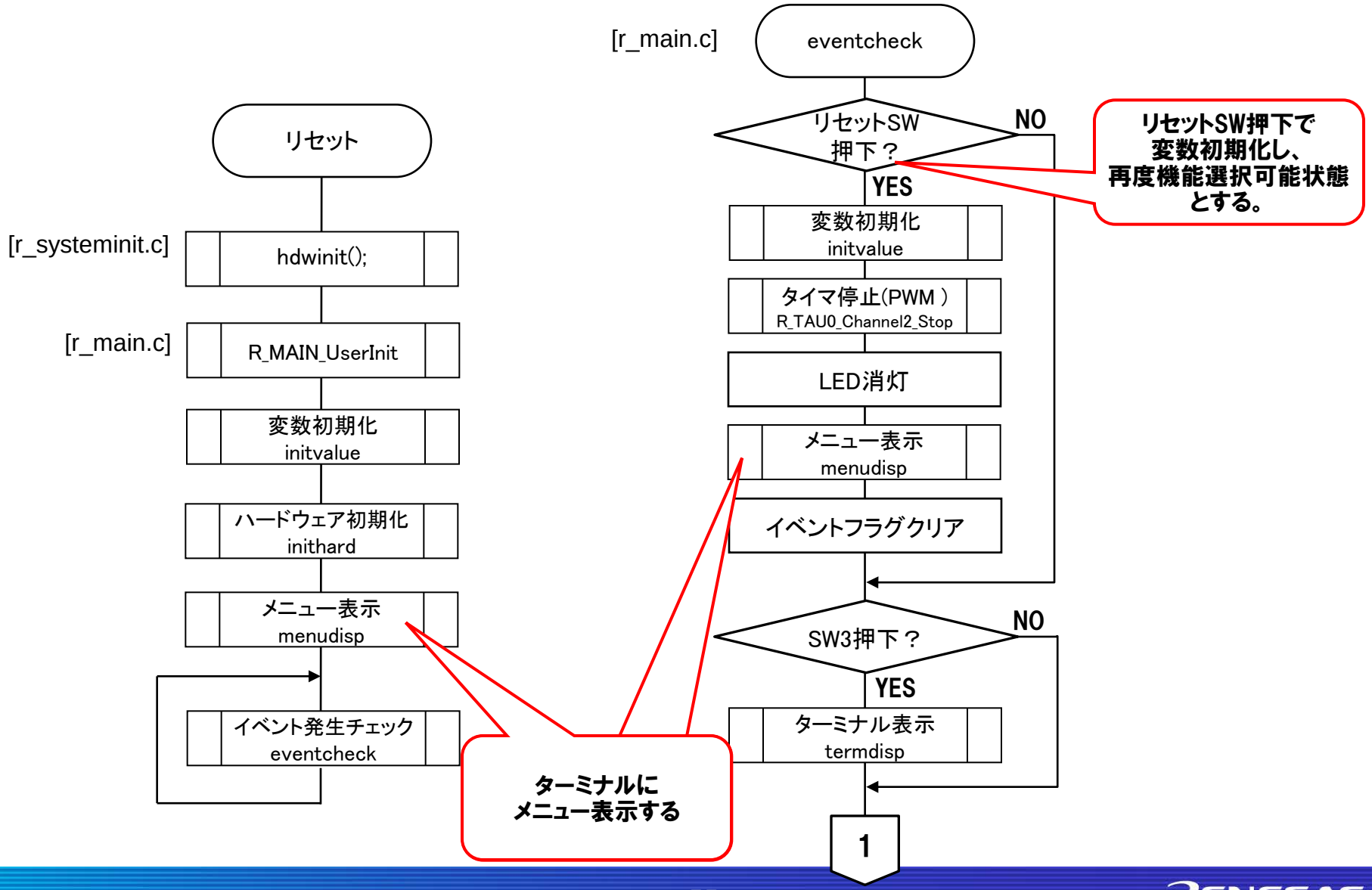
シリアル・アレイ・ユニット(シリアル・インタフェース(UART)送受信)使用例



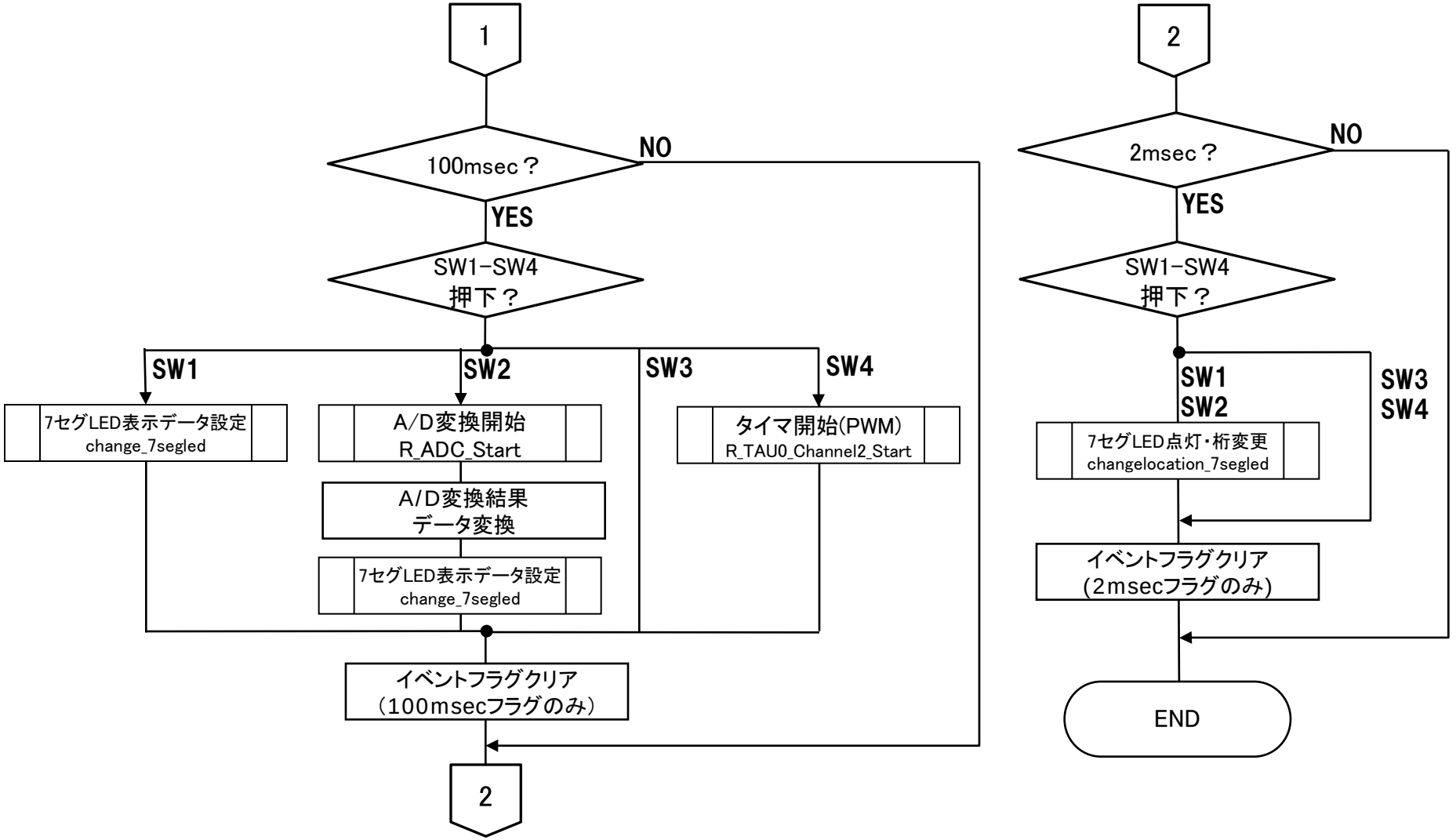
※ *m* はユニット番号を、*n* はチャネル番号を意味します。

※ エラー発生時のフローは記載していません。

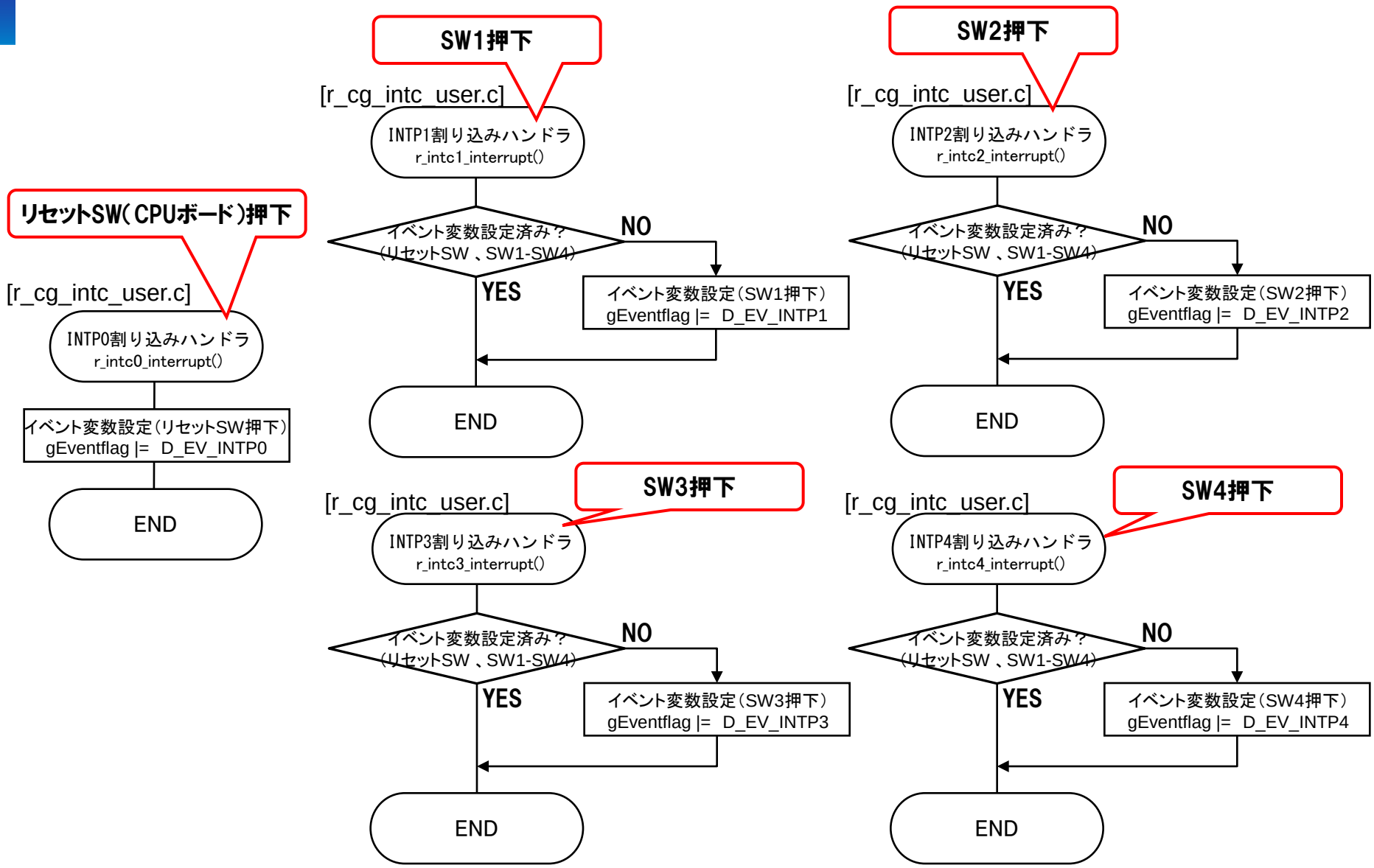
CS+ サンプルプログラムのフロー（1/9）



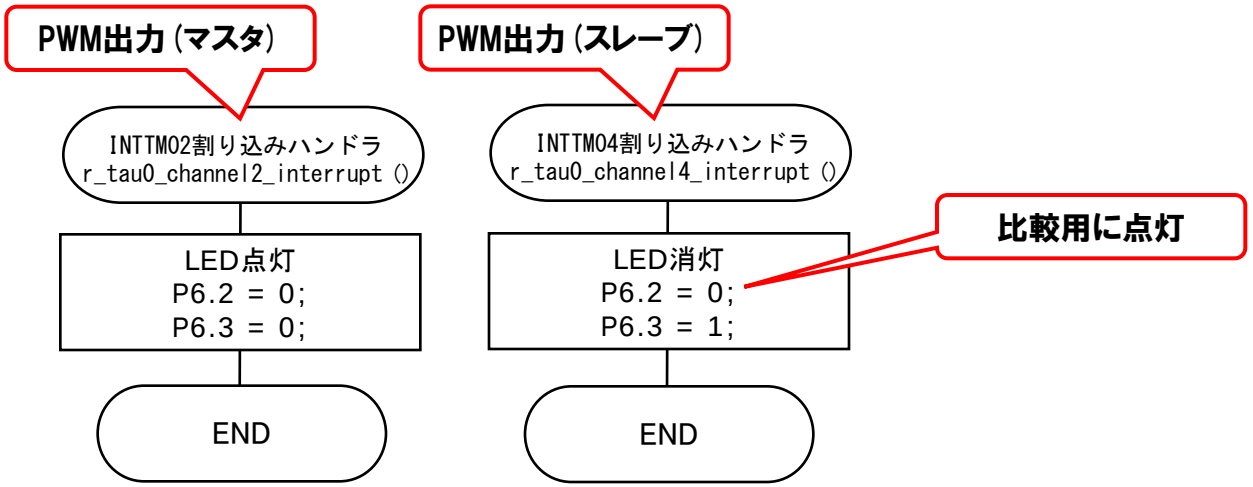
CS+ サンプルプログラムのフロー (2/9)



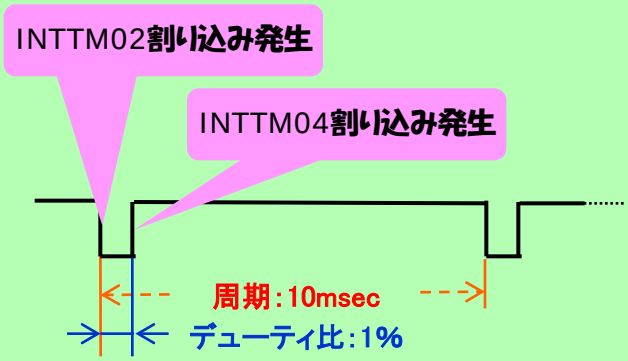
CS+ サンプルプログラムのフロー (3/9)



CS+ サンプルプログラムのフロー (4/9)

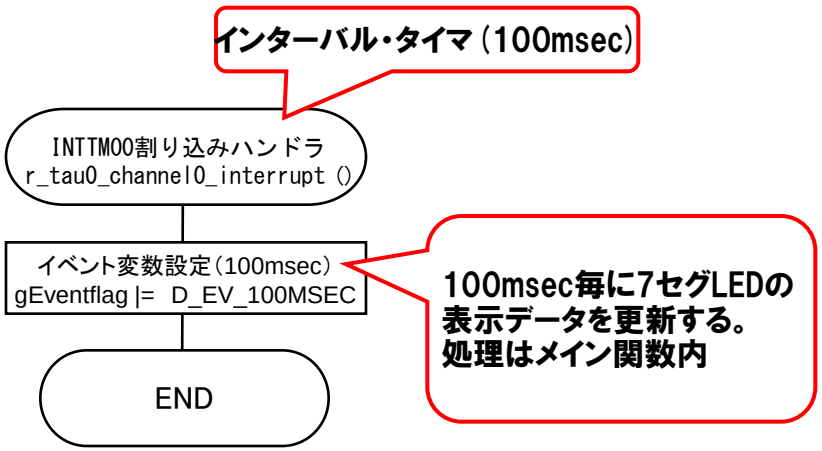


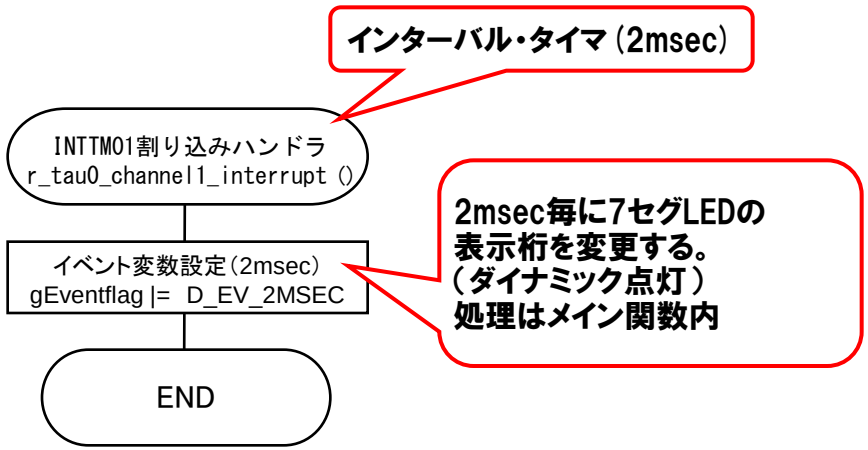
PWM制御:
高速で点灯/消灯を繰り返し、点灯しているように見せます。点灯期間の割合が小さくなる(デューティ比が小さくなる)と暗く見えます。

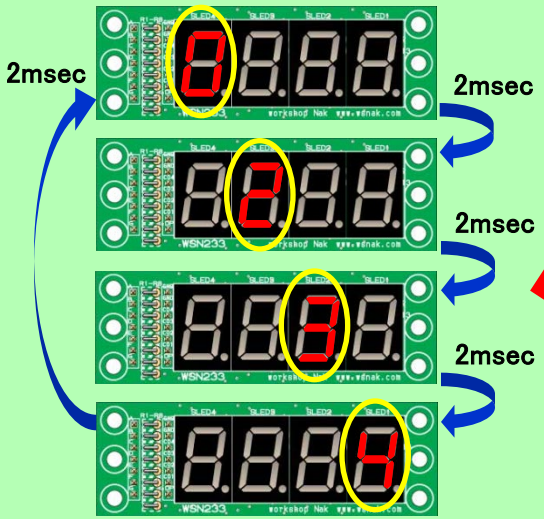


今回のセットでは、周期は20msec以下を推奨します。これ以上の値とすると点滅して見えます。
また、デューティ比を変化させると、明るさが変化することも確認してみましょう。
(コード生成の設定を変更して、ソースコードを修正します。)

CS+ サンプルプログラムのフロー (5/9)

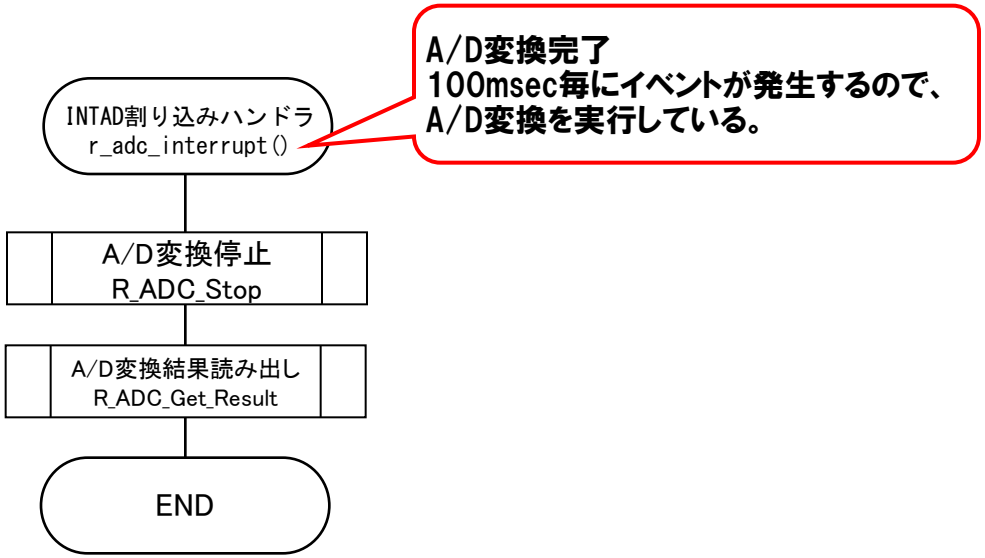






ダイナミック点灯:
7セグLEDの4つのLEDを1つのみ点灯し、点灯する桁の変更を高速で繰り返すことで、全ての桁を表示しているように見せます。
ポート制御が少なく済みますが、PWM制御と同様、消灯期間があるため、表示が暗くなります。

CS+ サンプルプログラムのフロー (6/9)

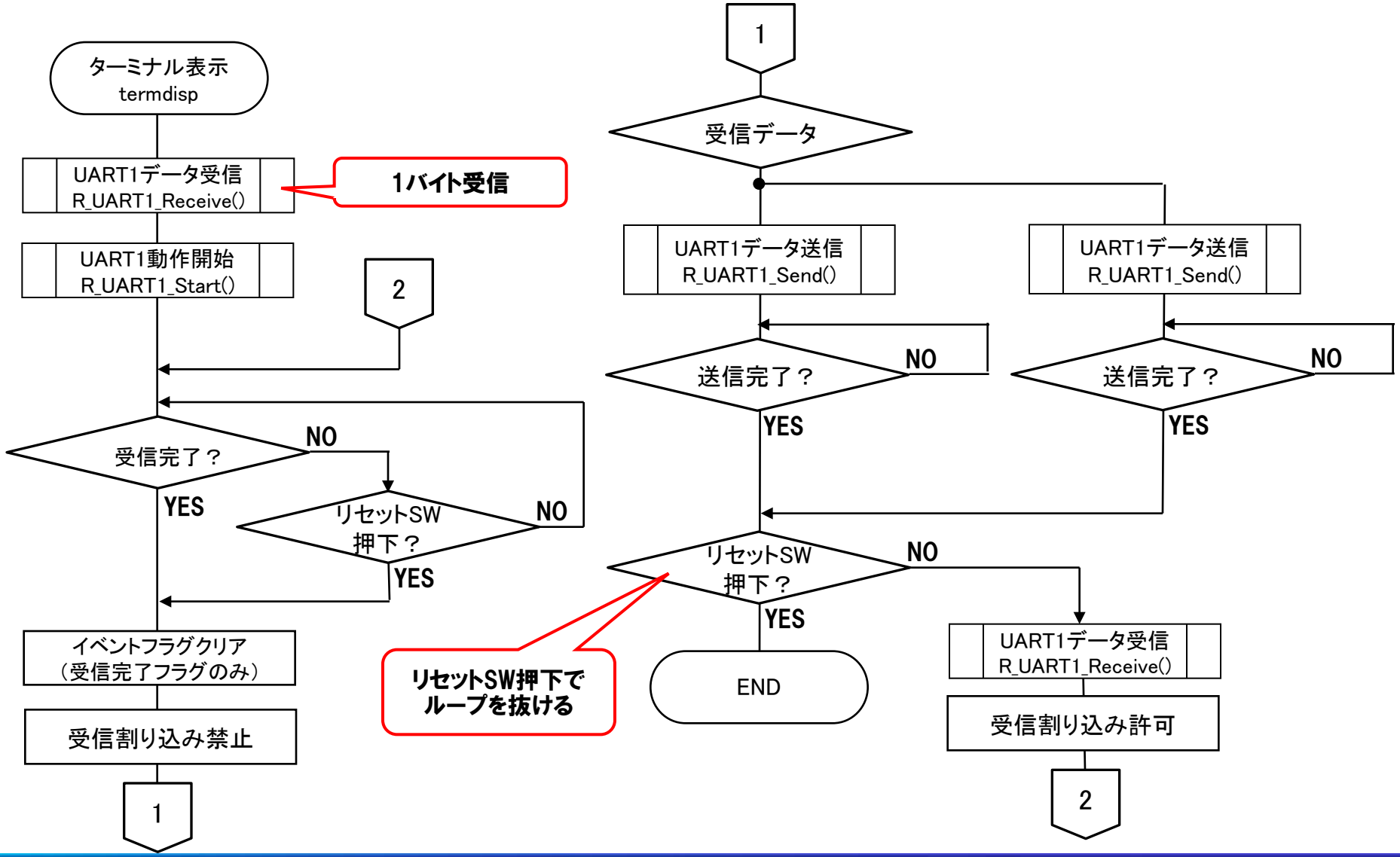


移動平均法:
このサンプルプログラムでは、A/D変換結果値をそのまま出力しています。このようにすると、(装置の微動な振動等のため)細かく値が変動していることがわかると思います。
このような細かな変化をそのまま使用すると、目障りとなる場合があります。そのため、移動平均法を用いた簡単なフィルタを利用することで不要な細かい変動を取り除くことができます。

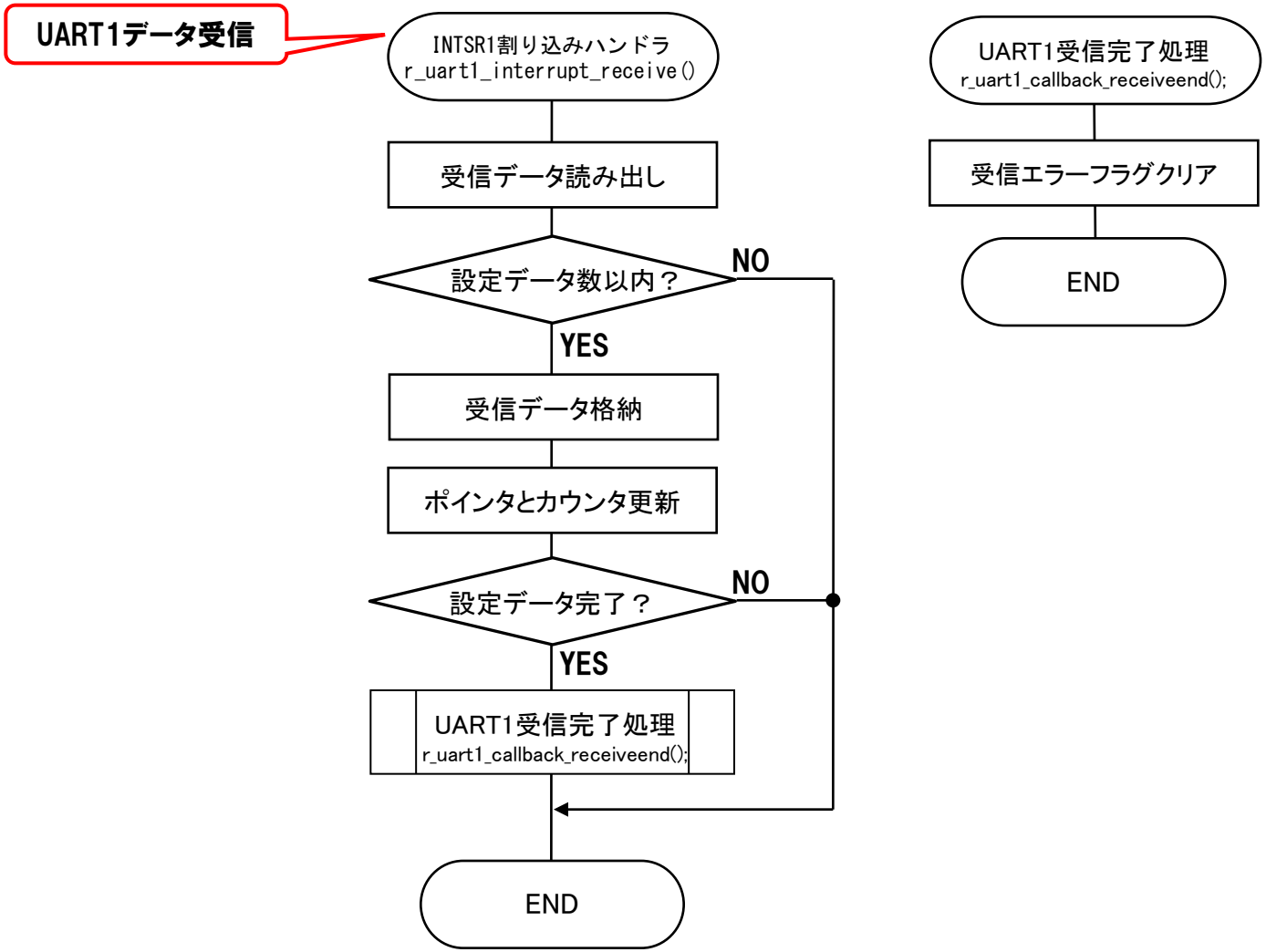
ex.
現在のA/D変換結果値をx[n]、1つ前の値をx[n-1]、その前の値をx[n-2]とします。この3つの値を足し合わせ、その値を1/3にし平均を取ります。この値を出力することで、不要な細かい変動を取り除きます。

$$(\text{7セグLEDの表示値}) = \frac{x[n] + x[n-1] + x[n-2]}{3}$$

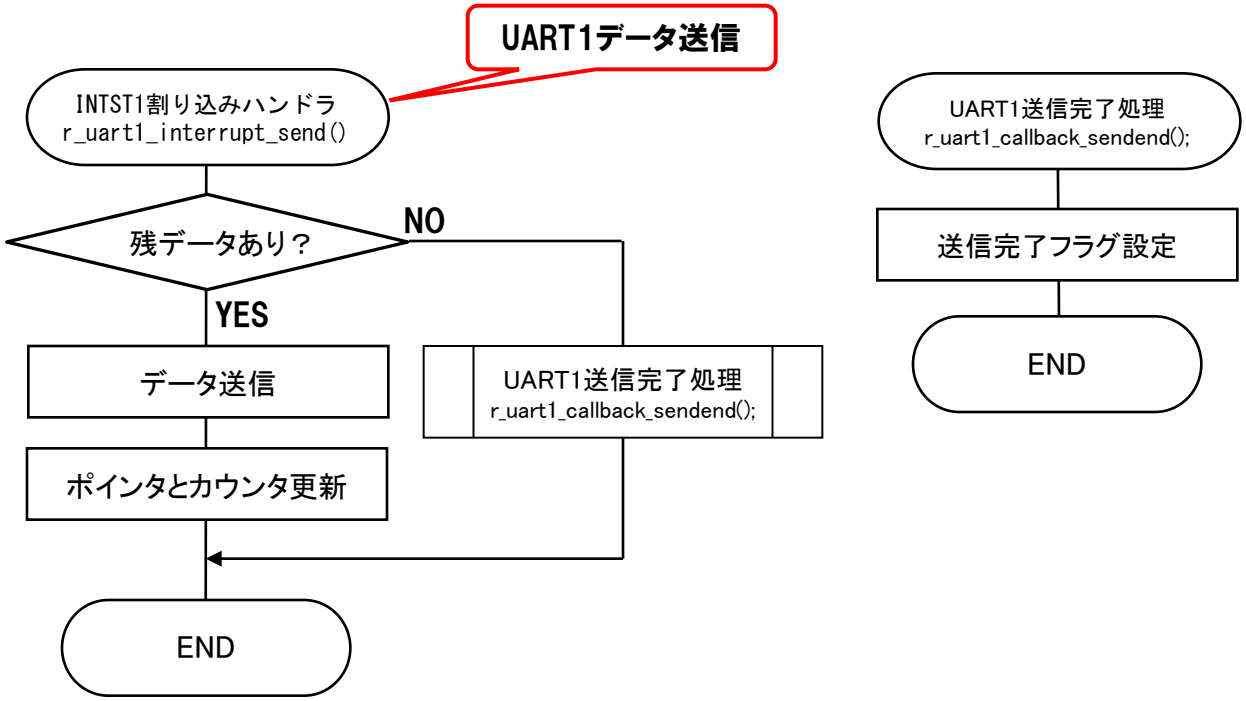
CS+ サンプルプログラムのフロー (7/9)



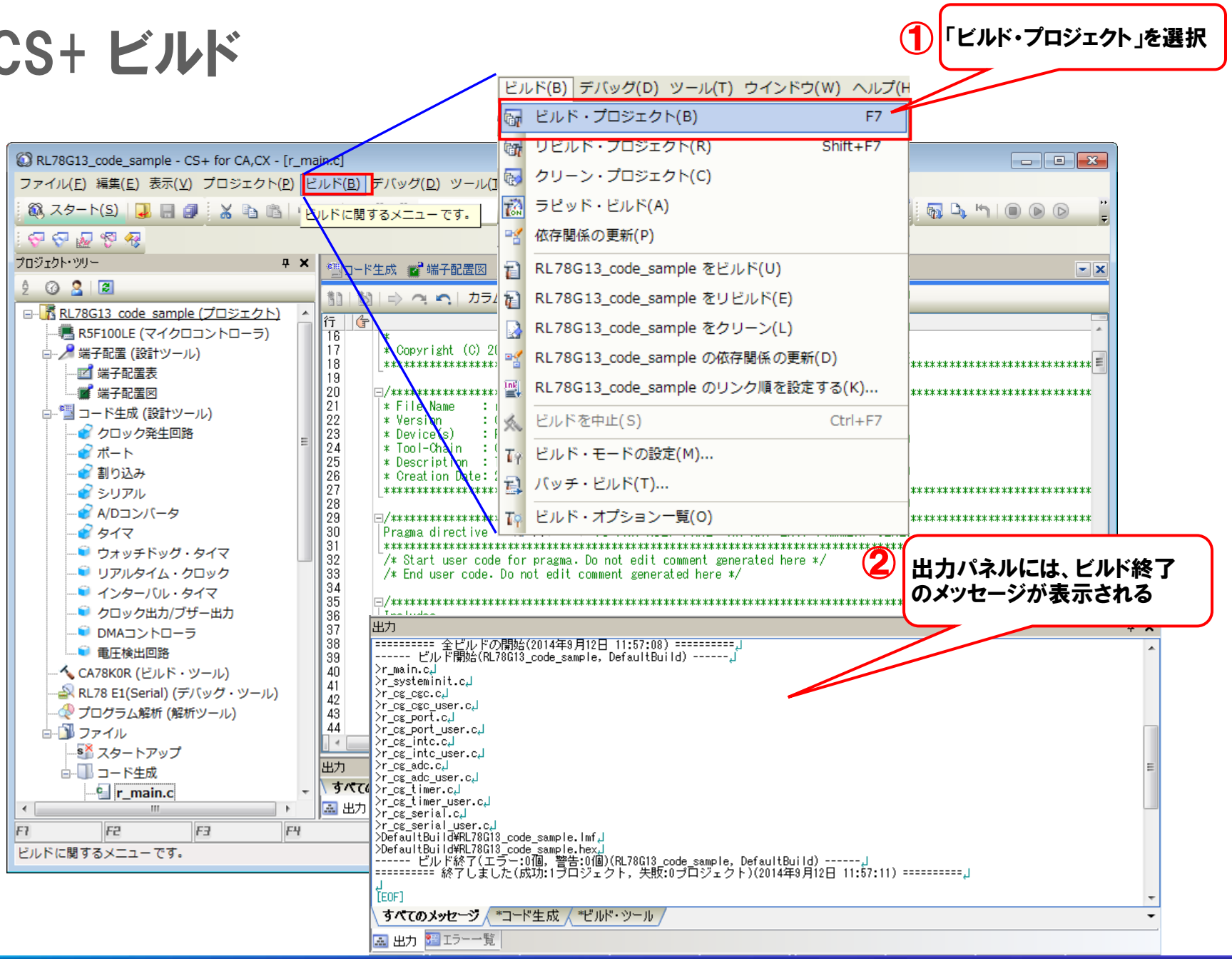
CS+ サンプルプログラムのフロー (8/9)



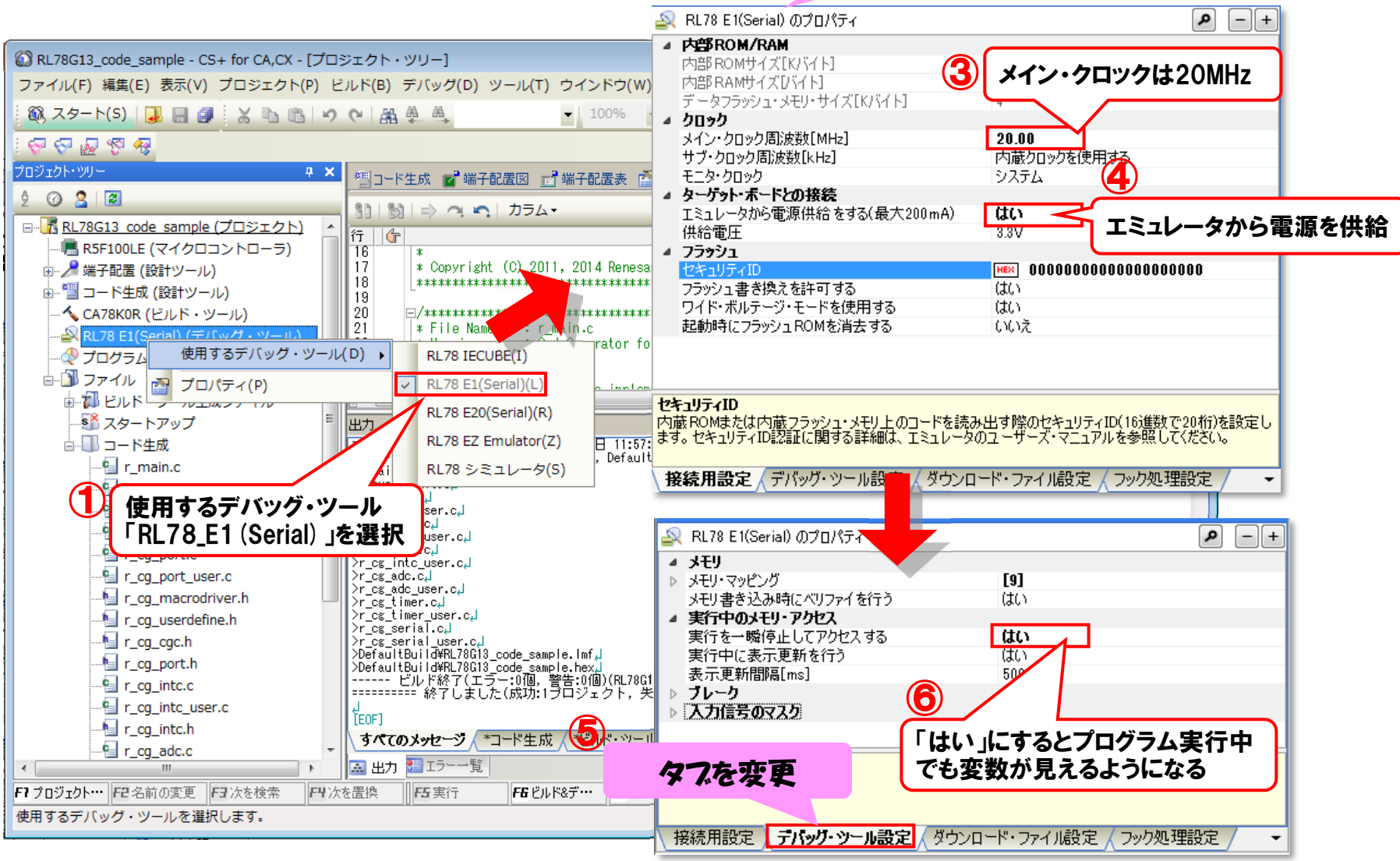
CS+ サンプルプログラムのフロー (9/9)



CS+ ビルド

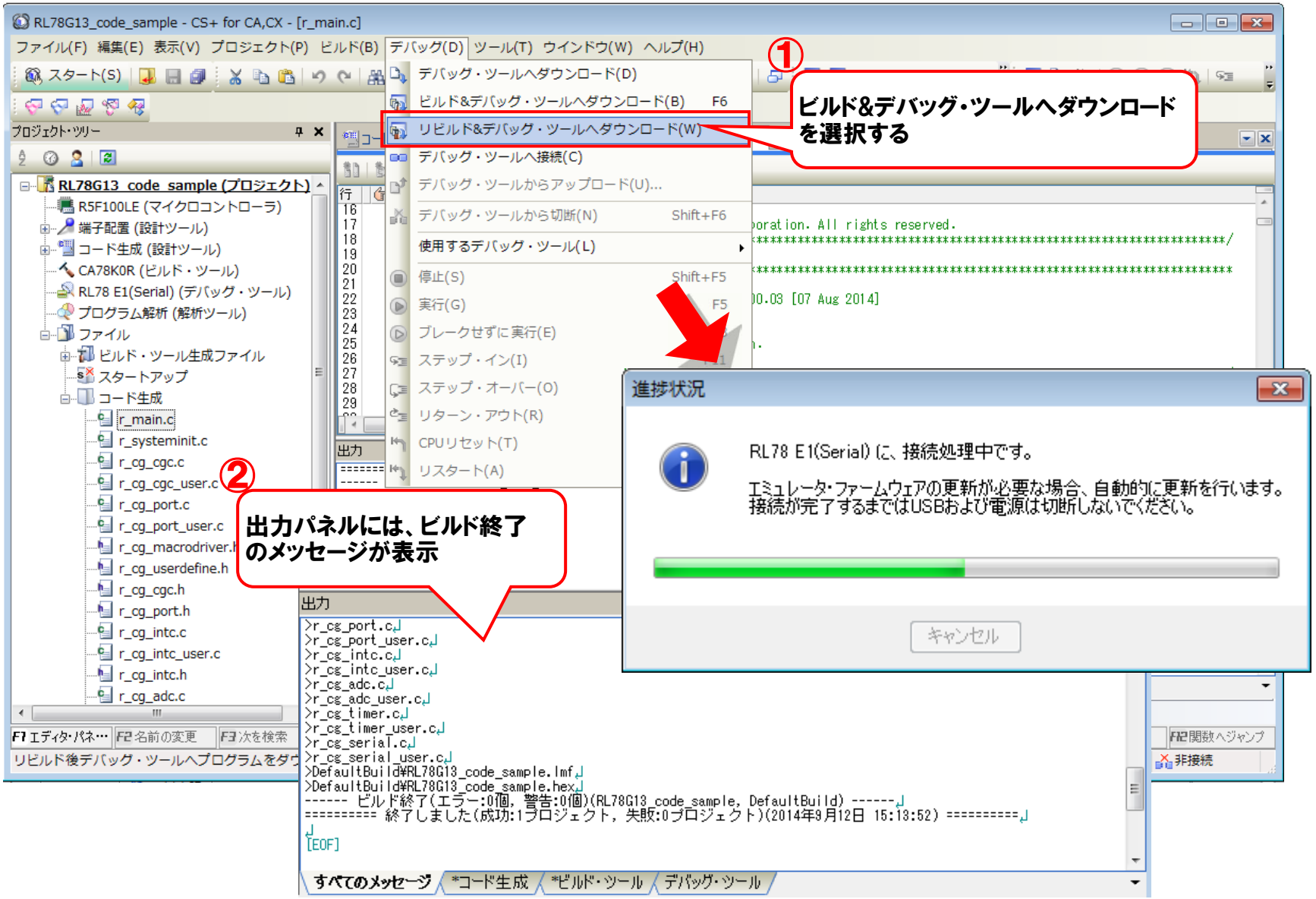

①「ビルド・プロジェクト」を選択
ビルド(B) デバッグ(D) ツール(T) ウインドウ(W) ヘルプ(H
ビルド・プロジェクト(B) F7
リビルド・プロジェクト(R) Shift+F7
クリーン・プロジェクト(C)
ラピッド・ビルド(A)
依存関係の更新(P)
RL78G13_code_sample をビルド(U)
RL78G13_code_sample をリビルド(E)
RL78G13_code_sample をクリーン(L)
RL78G13_code_sample の依存関係の更新(D)
RL78G13_code_sample のリンク順を設定する(K)...
ビルドを中止(S) Ctrl+F7
ビルド・モードの設定(M)...
バッチ・ビルド(T)...
ビルド・オプション一覧(O)
RL78G13_code_sample - CS+ for CA,CX - [r_main.c]
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) ツール(T)
スタート(S)
ビルドに関するメニューです。
プロジェクト・ツリー
RL78G13_code_sample (プロジェクト)
R5F100LE (マイクロコントローラ)
端子配置 (設計ツール)
端子配置表
端子配置図
コード生成 (設計ツール)
クロック発生回路
ポート
割り込み
シリアル
A/Dコンバータ
タイマ
ウォッチドッグ・タイマ
リアルタイム・クロック
インターバル・タイマ
クロック出力/ブザー出力
DMAコントローラ
電圧検出回路
CA78K0R (ビルド・ツール)
RL78 E1(Serial) (デバッグ・ツール)
プログラム解析 (解析ツール)
ファイル
スタートアップ
コード生成
r_main.c
コード生成 端子配置図
出力
========== 全ビルドの開始(2014年9月12日 11:57:08) ==========
------ ビルド開始(RL78G13_code_sample, DefaultBuild) ------
>r_main.c
>r_systeminit.c
>r_cg_cgc.c
>r_cg_cgc_user.c
>r_cg_port.c
>r_cg_port_user.c
>r_cg_intc.c
>r_cg_intc_user.c
>r_cg_adc.c
>r_cg_adc_user.c
>r_cg_timer.c
>r_cg_timer_user.c
>r_cg_serial.c
>r_cg_serial_user.c
>DefaultBuild¥RL78G13_code_sample.lmf
>DefaultBuild¥RL78G13_code_sample.hex
------ ビルド終了(エラー:0個, 警告:0個)(RL78G13_code_sample, DefaultBuild) ------
========== 終了しました(成功:1プロジェクト, 失敗:0プロジェクト)(2014年9月12日 11:57:11) ==========
[EOF]
すべてのメッセージ *コード生成 *ビルド・ツール
出力 エラー一覧
②出力パネルには、ビルド終了のメッセージが表示される

CS+ デバッグツール設定

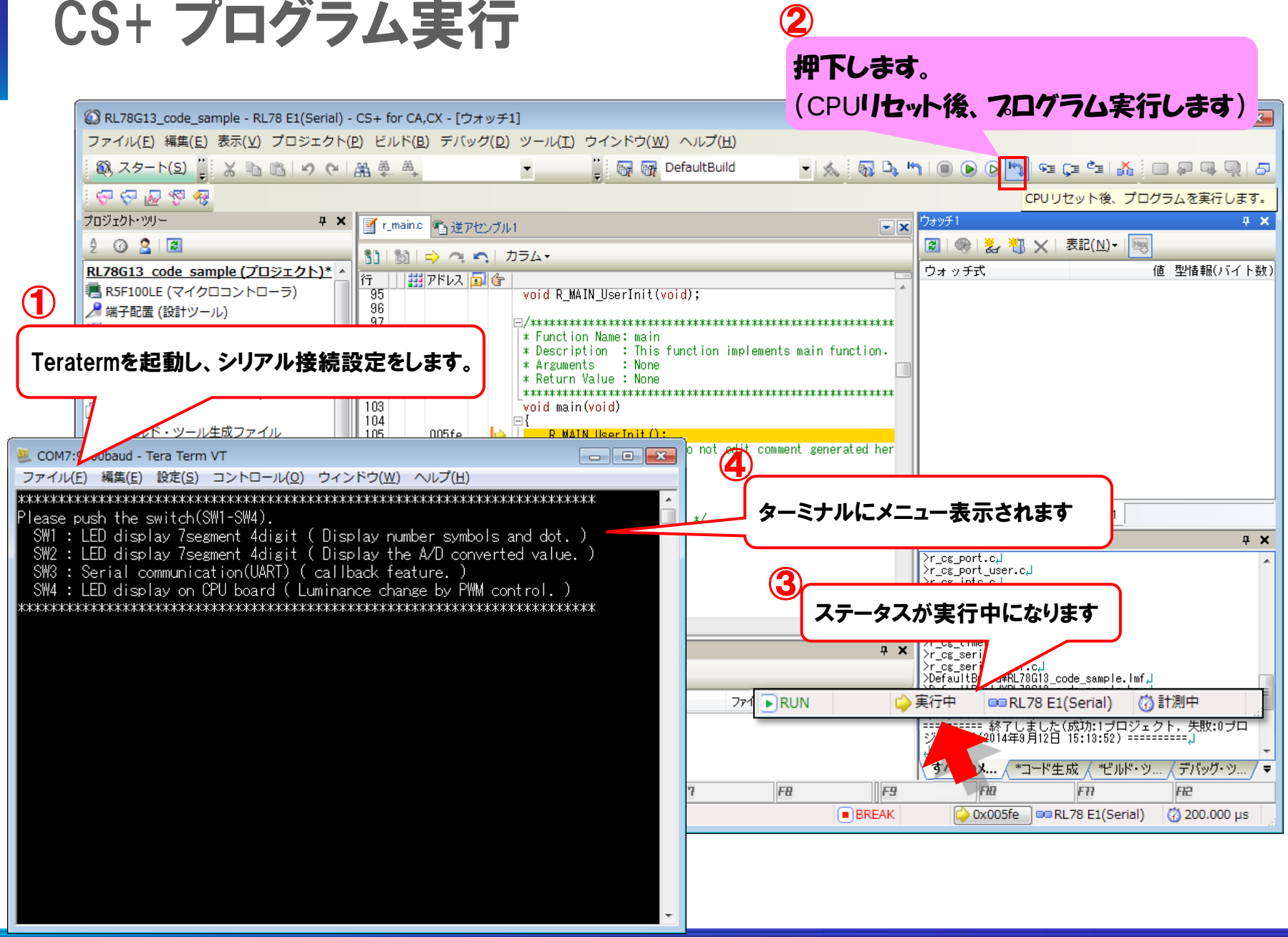


標準と異なるオプションを設定した場合、そのプロパティは太字表示されます。

CS+ デバッグ・ツールへダウンロード



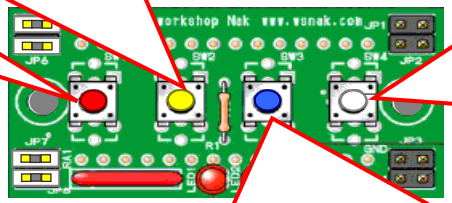
CS+ プログラム実行

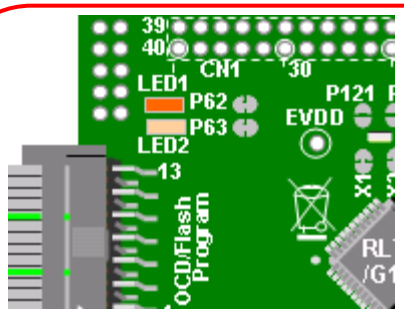


CS+ プログラム実行（サンプルプログラム動作確認）

SW1押下：
7セグLEDに0から9の数字と
.（ドット）が表示されます。

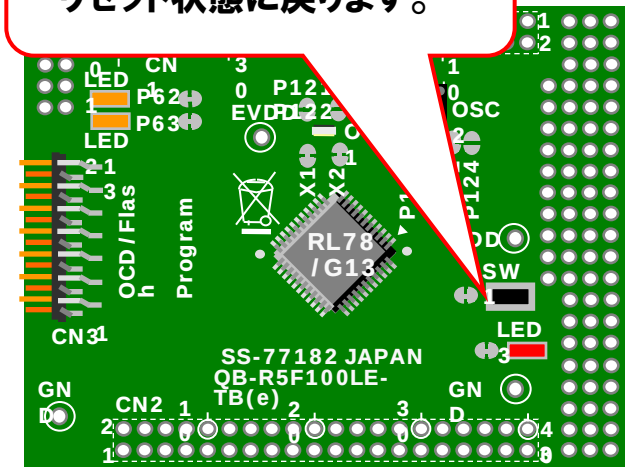
SW2押下：
ポテンションメータで設定した値の
A/D変換値（0～1023）が7セグLED
に表示されます。

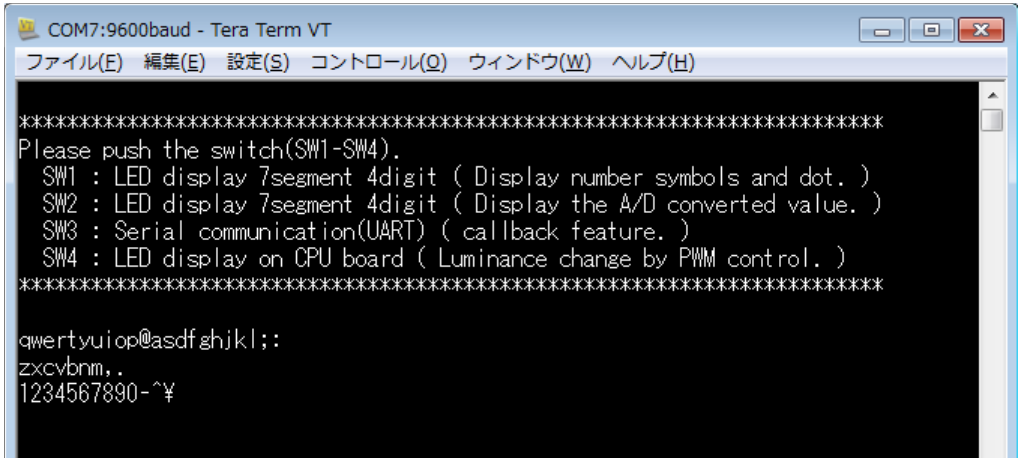




SW4押下：
PWMで設定した輝度でLED（P63）が
点灯します。
（LED（P62）は比較用に点灯します）

CPUボードのSW1押下：
リセット状態に戻ります。





SW3押下：
PCでキー入力したデータをターミナルに表示します。（コールバック）

おつかれさまでした。

ルネサス システムデザイン株式会社