

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

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ない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あ」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。

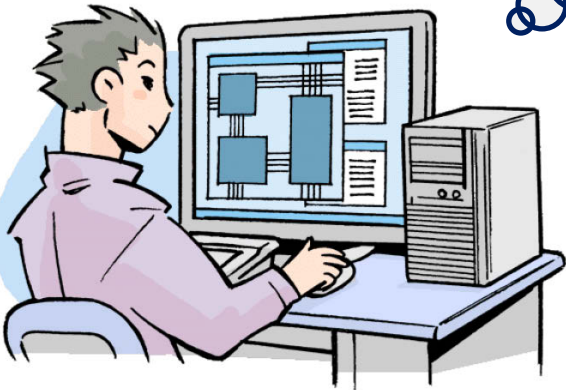


コード生成って？

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。

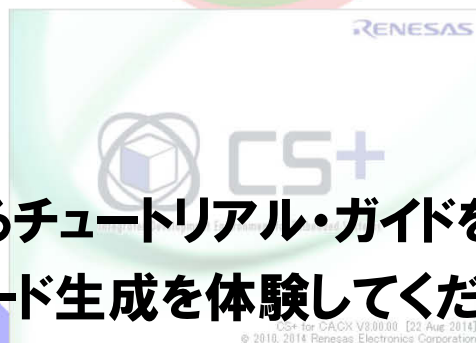
目的

- 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。

安心

簡単

快適



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。

The image shows a screenshot of the code generation tool's GUI. The main window displays various peripheral configuration options, such as clock settings, timer settings, and PWM settings. A callout box points to the 'コード生成 (設計ツール)' (Code Generation (Design Tool)) menu, which lists supported peripherals: クロック発生回路 (Clock Generator), ポート (Port), 割り込み (Interrupt), シリアル (Serial), A/Dコンバータ (A/D Converter), タイマ (Timer), ウォッチドッグ・タイマ (Watchdog Timer), リアルタイム・クロック (Real-time Clock), インターバル・タイマ (Interval Timer), クロック出力/ブザー出力 (Clock Output/Buzzer Output), DMAコントローラ (DMA Controller), and 電圧検出回路 (Voltage Detector). Another callout box points to the '直感的にわかる設定' (Intuitive Settings) section, which includes options for high-speed on-chip oscillator, high-speed system clock, and sub-system clock. A third callout box points to the 'PWMタイマも簡単設定' (PWM Timer Easy Setting) section, which shows the PWM configuration for Channel 2, including frequency, period, and duty cycle. A fourth callout box points to the 'サポートしている周辺機能一覧' (Supported Peripheral Function List) table, which shows the supported functions for each channel.

直感的にわかる設定

PWMタイマも簡単設定

サポートしている周辺機能一覧

機能	チャンネル0	チャンネル1	チャンネル2	チャンネル3	チャンネル4	チャンネル5	チャンネル6	チャンネル7
チャンネル0	インターバル・タイマ							
チャンネル1	インターバル・タイマ							
チャンネル2	PWM出力(マスタ)							
チャンネル3	使用しない							
チャンネル4	PWM出力(スレーブ)							
チャンネル5	使用しない							
チャンネル6	使用しない							
チャンネル7	使用しない							

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(G) ボタン押下でソースを作成

作成されたソースはビルド可能!

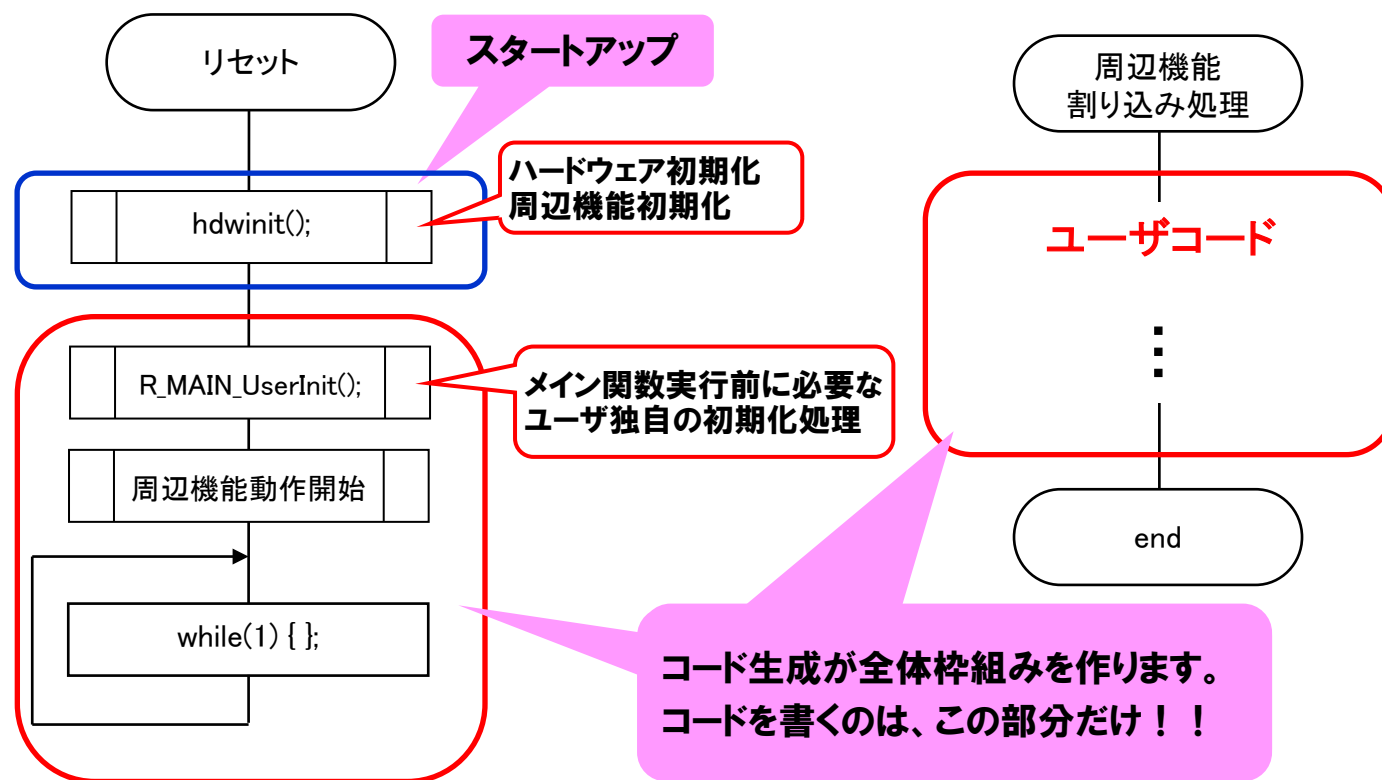
出力

```
===== 全ビルドの開始(2014年8月8日 14:25:24) =====  
----- 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8月8日 14:25:25) =====
```

テストするまで簡単!

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。

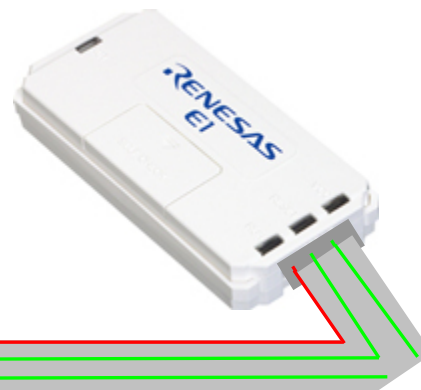
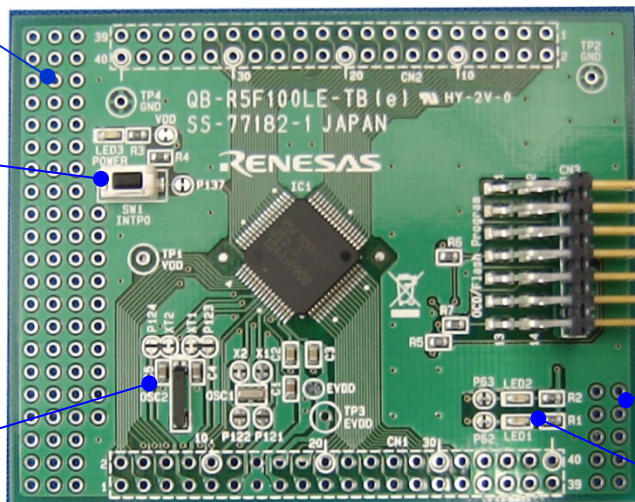
- **割り込み機能**を使用して、スイッチボードで作成したプログラムを選択できるようにしましょう。
 - ✓ SW1-4押下で、各機能を動作させます。
 - ✓ リセットSW(CPUボード上SW)押下で、再選択可能状態にします。

必要な機材(1/5) CPUボード

簡易ユニバーサル・エリア

スイッチ1個

クロック実装済み



鉛(Pb)フリー対応品

LED2個

※イメージ図

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・ドレイン: 0-4本)	
タイマ		
	16ビット・タイマ	8-16チャンネル
	ウォッチドッグ・タイマ	1チャンネル
	リアルタイム・クロック	1チャンネル
	インターバル・タイマ	1チャンネル
シリアル・インタフェース		
	CSI	2~8チャンネル
	UART/UART(LIN-bus対応)	2~4チャンネル
	I2C/簡易I2C	2~8チャンネル
8/10ビット分解能A/Dコンバータ	6-26チャンネル	
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。

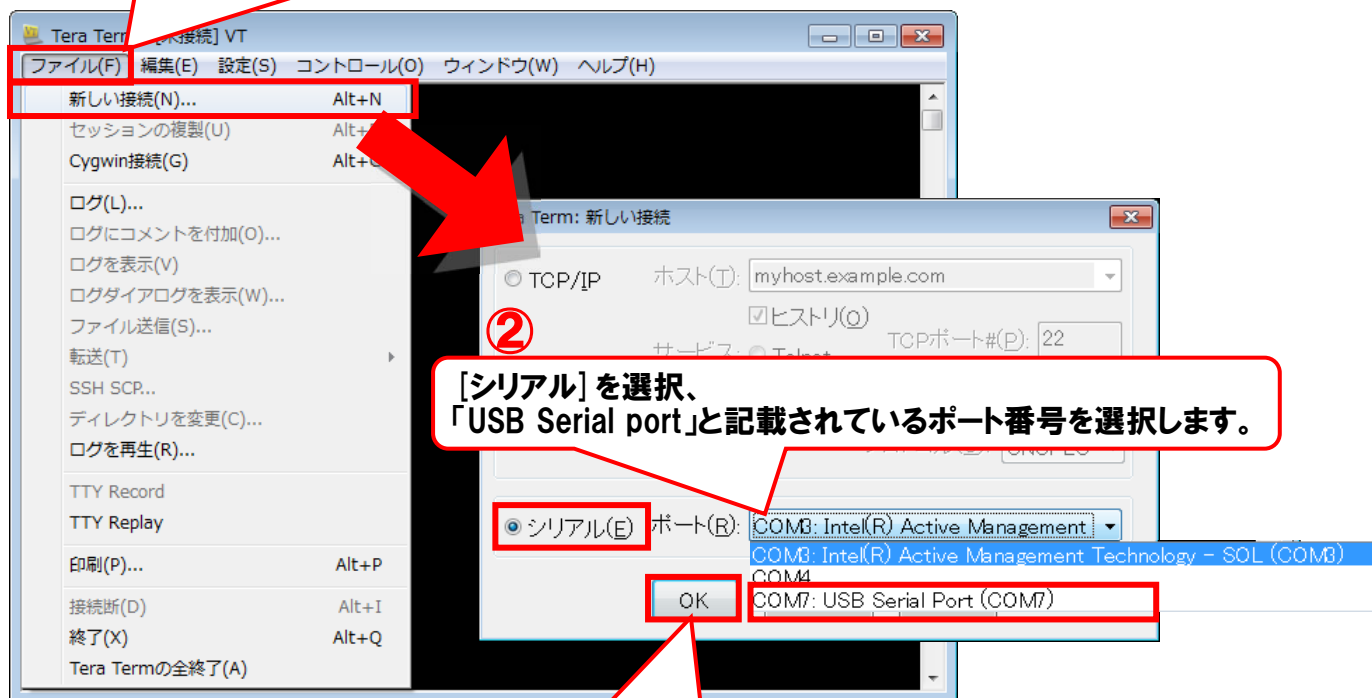
<http://sourceforge.jp/projects/ttssh2/>

- 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。

① メニューの [ファイル] から [新しい接続] を選択します。



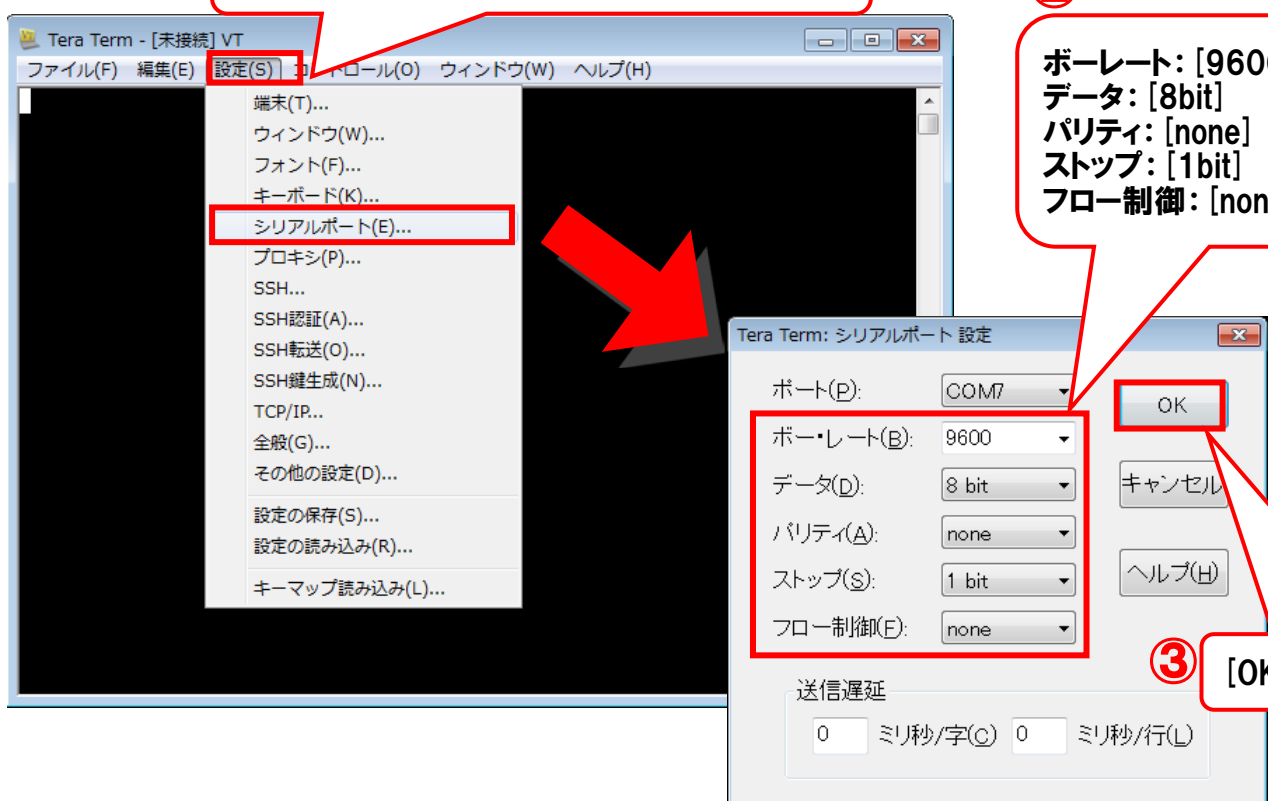
② [シリアル] を選択、
「USB Serial port」と記載されているポート番号を選択します。

③ [OK] ボタンを押下します。

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。

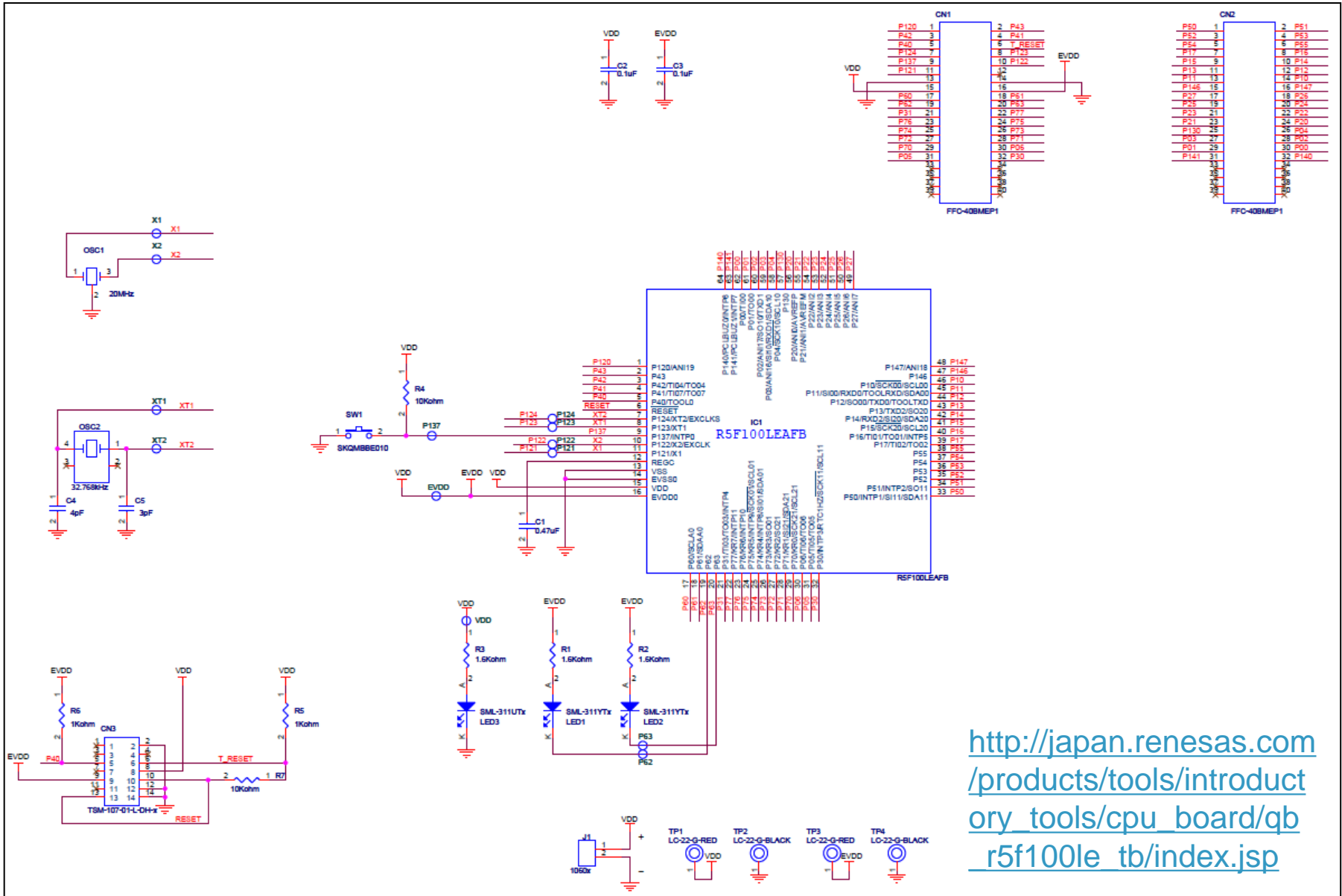
① メニューの [設定] から [シリアルポート] を選択



② ボーレート: [9600]
データ: [8bit]
パリティ: [none]
ストップ: [1bit]
フロー制御: [none] を選択

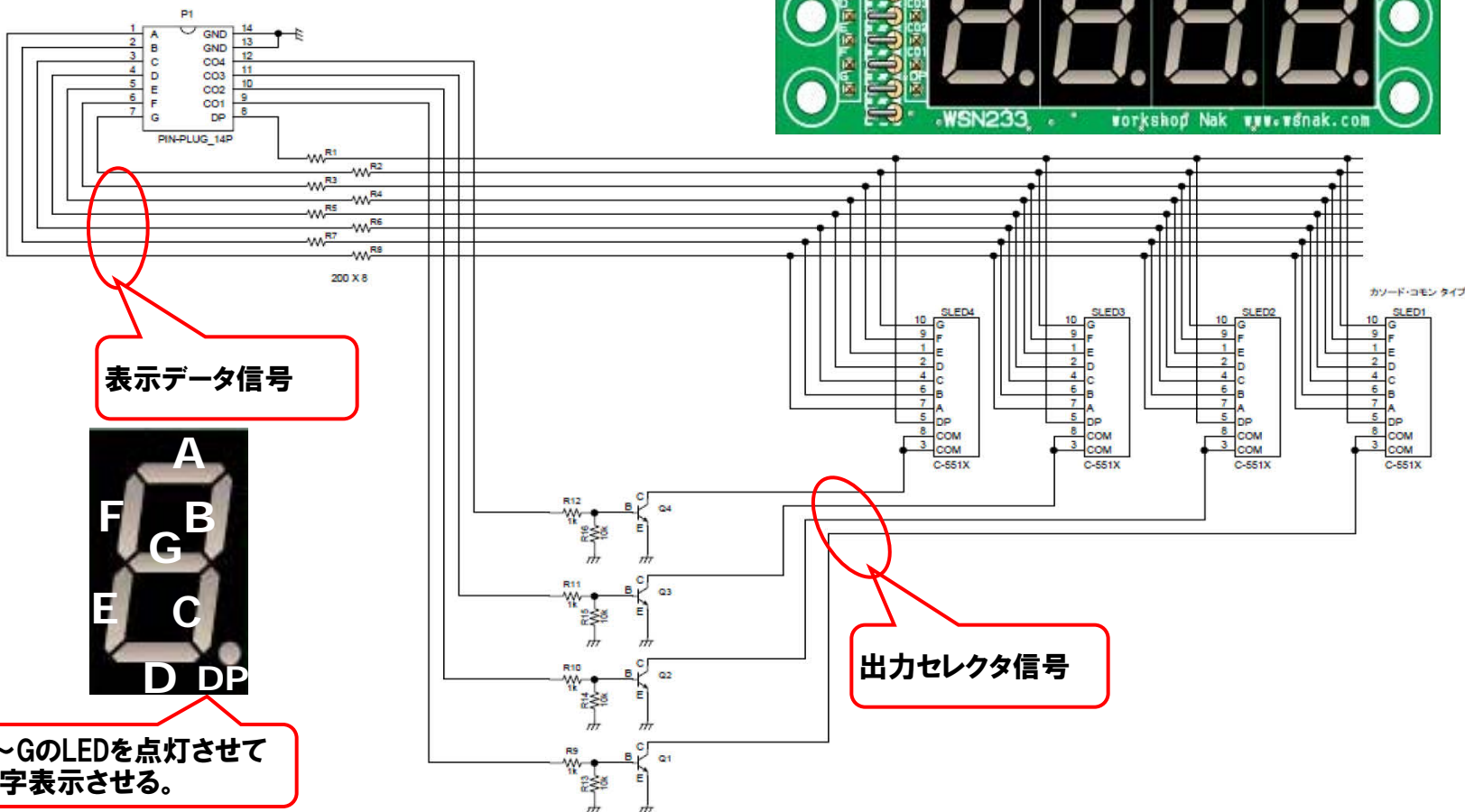
③ [OK] ボタンを押下します。

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/qbr5f100le_tb/index.jsp

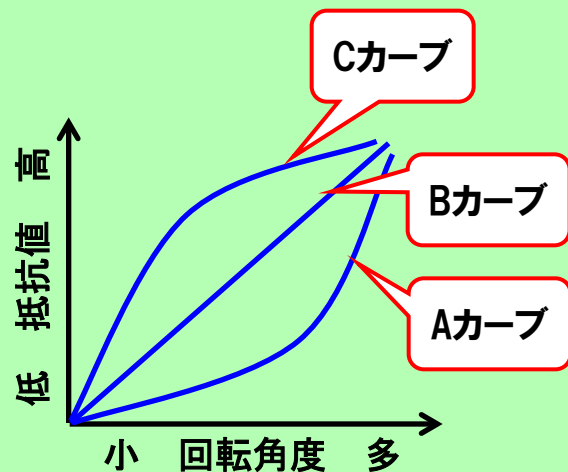
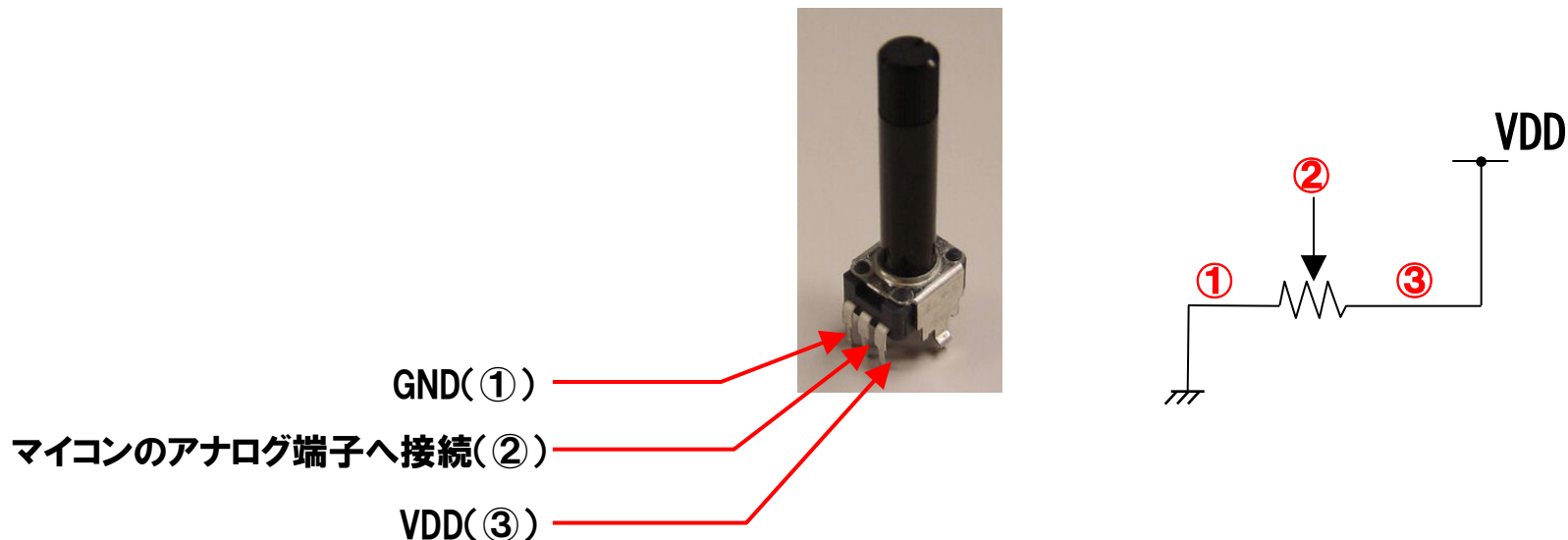
ハードウェア設計 回路図(4桁7セグメントLED)



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

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

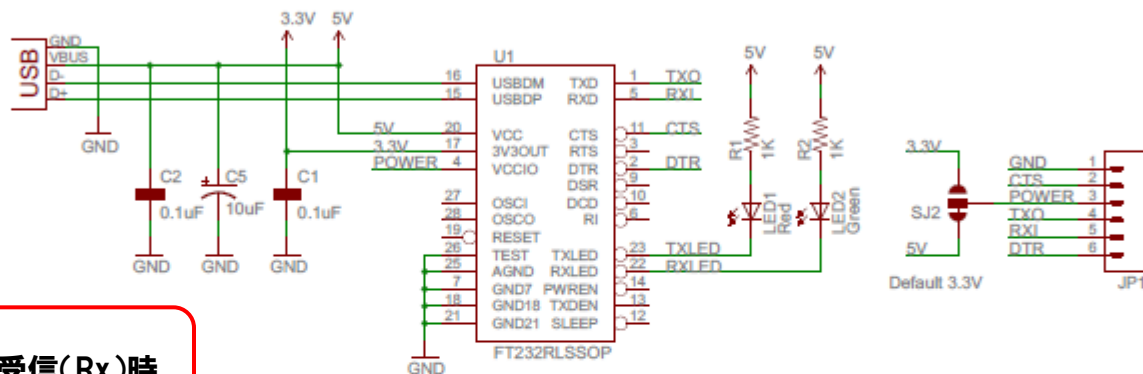
ハードウェア設計 回路図(ポテンションメータ)



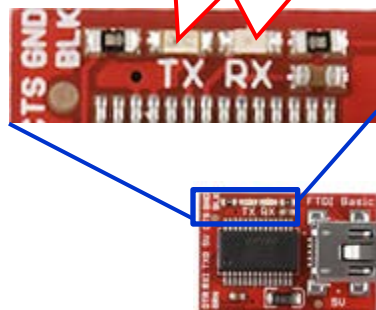
左図は、半固定抵抗器の特性を示したものです。半固定抵抗器は、一般的にポテンションメータと呼ばれます。

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)



LED
(送信(Tx)、受信(Rx)時
それぞれ点灯)



USBケーブル

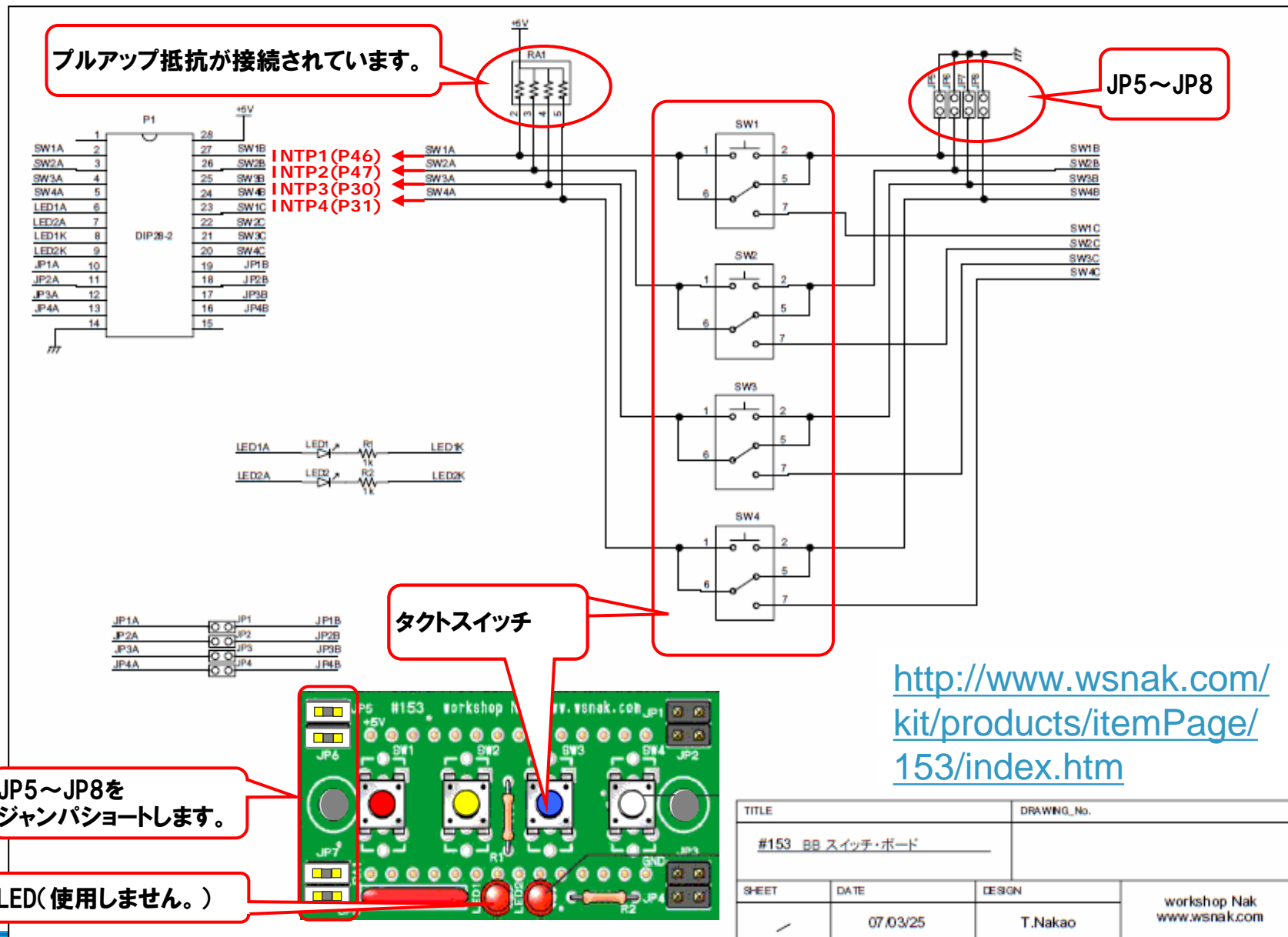


GND
(未接続)
(未接続)
RxD1へ
TxD1へ
(未接続)

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

⊗ ⊗ ⊗ ⊗			
TITLE: FTDI Basic-v21-5U		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AN10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チャンネル、チャンネル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。

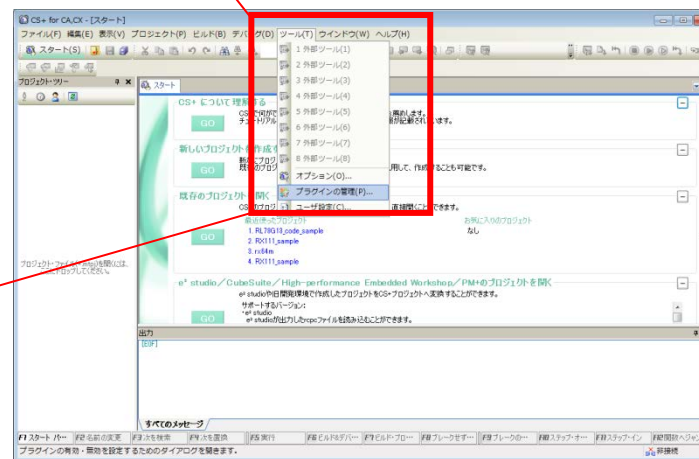
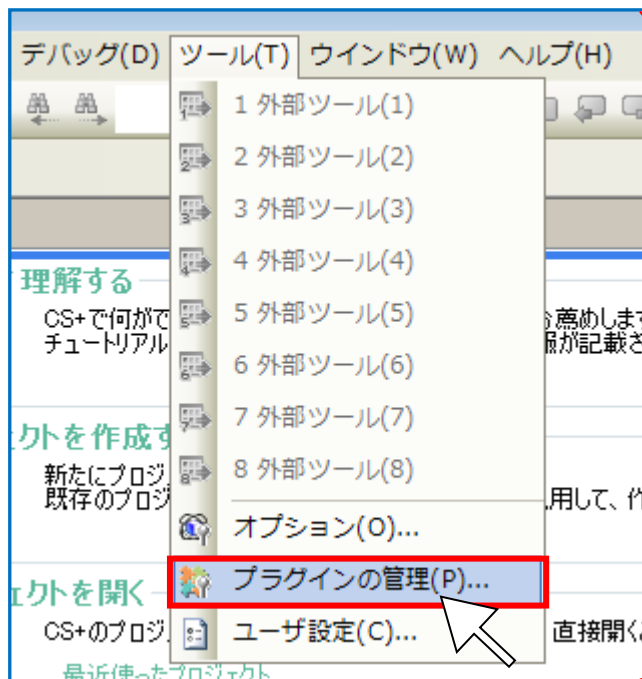
The image shows the CS+ for CA,CX software interface. On the left, the main window displays a 'GO' button for creating a new project. A red arrow points to this button, labeled with a circled '1'. On the right, the 'プロジェクト作成' (Project Creation) dialog box is open. It contains several fields and options, each with a numbered callout:

- 1**: 「新しいプロジェクト作成する」を選択 (Select 'Create new project') - points to the 'GO' button in the main window.
- 2**: マイコンコントローラはRL78 (Microcontroller is RL78) - points to the 'マイクロコントローラ(D):' dropdown menu.
- 3**: マイコンR5F100LE (64pin) 選択 (Select microcontroller R5F100LE (64pin)) - points to the selected item in the microcontroller list.
- 4**: プロジェクト名はRL78G13_sample (Project name is RL78G13_sample) - points to the 'プロジェクト名(N):' text field.
- 5**: フォルダはC: ¥ (Folder is C: ¥) - points to the '作成場所(L):' text field.
- 6**: 「プロジェクト名のフォルダを作成する」にチェック (Check 'Create folder with project name') - points to the checked checkbox.
- 7**: クリックして次へ (Click to next) - points to the '作成(C)' button.

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)

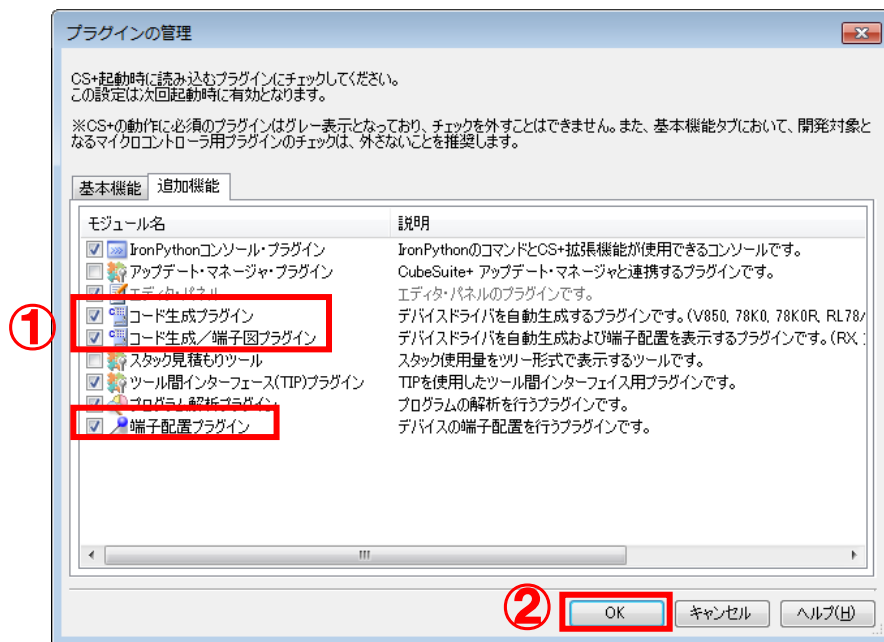
- メニューの [ツール] から [プラグイン管理] を選択してください。



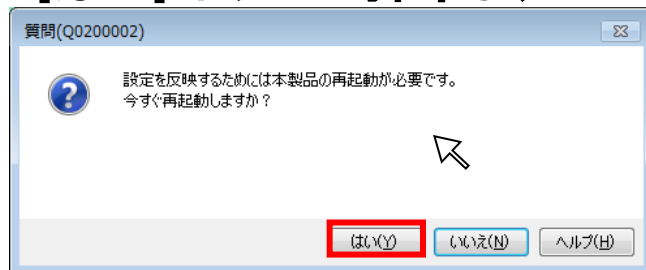
デフォルトの端子配置機能やコード生成機能はOFF状態になっていますので、プラグインをONにする必要があります。

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

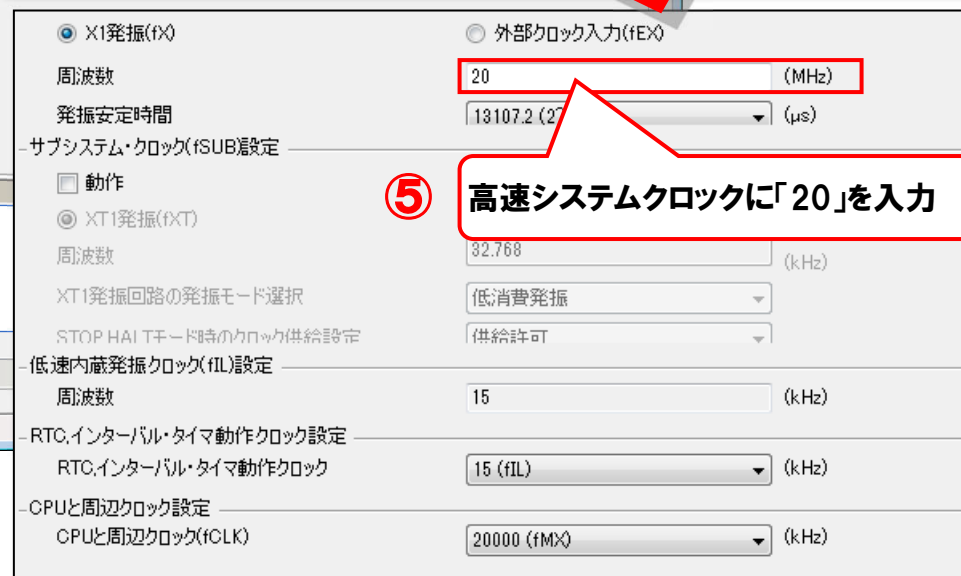
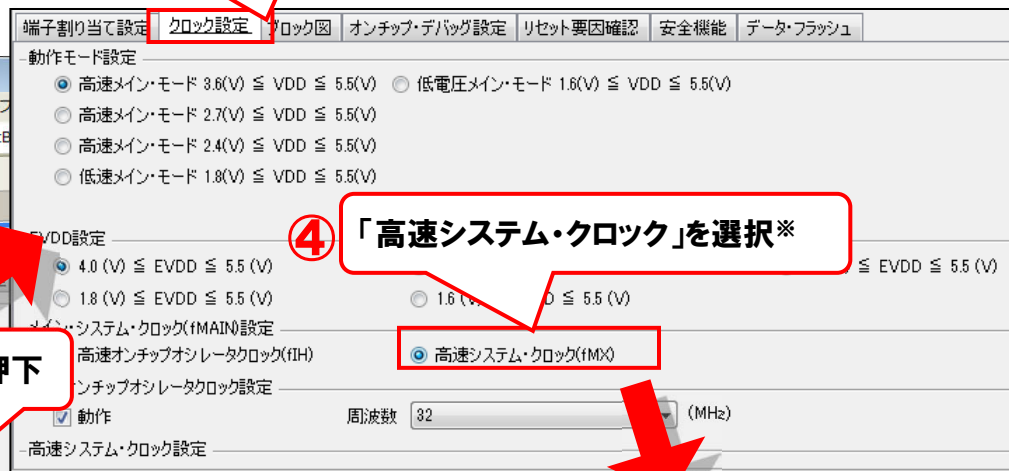
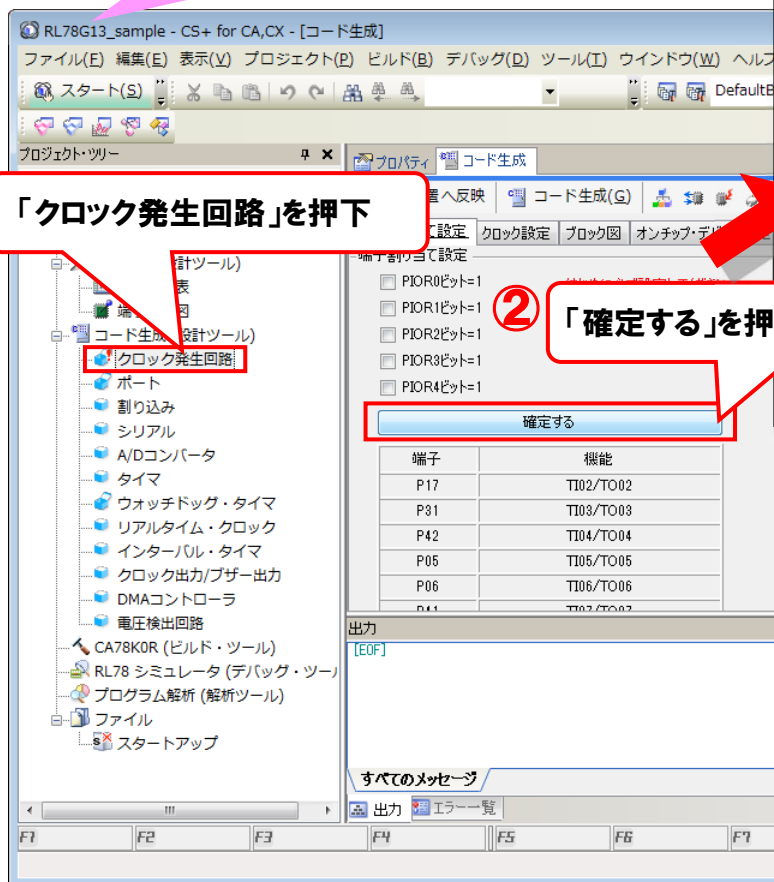
① 「クロック発生回路」を押下

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)

The screenshot shows the CS+ IDE interface for the 'RL78G13_code_sample' project. The 'コード生成' (Code Generation) window is open, and the 'オンチップ・デバッグ設定' (On-chip Debug Settings) tab is selected. Two red callouts provide instructions:

- ① 「オンチップ・デバッグ設定」タブに切り替え**: A red box highlights the 'オンチップ・デバッグ設定' tab in the top navigation bar.
- ② 「使用する」にチェック**: A red box highlights the '使用する' (Use) radio button under the 'オンチップ・デバッグ動作設定' (On-chip Debug Operation Settings) section.

The 'オンチップ・デバッグ動作設定' section includes the following options:

- オンチップ・デバッグ動作設定: 使用する, 使用しない
- RRM機能設定: 使用する, 使用しない
- セキュリティID設定: セキュリティIDを設定する. Security ID: 0x000000000000000000000000
- セキュリティID認証失敗時の設定: フラッシュ・メモリのデータを消去しない, フラッシュ・メモリのデータを消去する

The output window at the bottom shows the following message:

```
情報(M0200002) : 以下のプラグインが無効になっています。↓
スタック見積もリツール↓
アップデート・マネージャ・プラグイン↓
有効にするには、[プラグインの管理]ダイアログを使用します。↓
[EOF]
```

CS+ コード生成設定(ウォッチドッグ・タイマ)

① 「ウォッチドッグ・タイマ」を押下

② 「使用しない」にチェック

RL78G13_code_sample - CS+ for CA,CX - [コード生成]

ファイル(E) 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(I) ウィンドウ(W) ヘルプ(H)

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)
 - 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
 - アナログ・タイム・クロック
 - 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

コード生成

端子配置へ反映 コード生成(S)

ウォッチドッグ・タイマ動作設定

使用しない 使用する

HALT/STOP/SNOOZE時の動作設定

オーバーフロー時間 4369.07 (2¹⁶/fIL) (ms)

ウインドウ・オープン期間設定

ウインドウ・オープン期間 100 (%)

割り込み設定

オーバフロー時間の75% + 1/2fIL到達時にインターバル割り込みを発生する(INTWDT)

優先順位 低

出力

情報(M0200002) : 以下のプラグインが無効になっています。↓

スタック見積もりツール↓

アップデート・マネージャ・プラグイン↓

有効にするには、[プラグインの管理]ダイアログを使用します。↓

[EOF]

すべてのメッセージ

出力 エラー一覧

F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12

非接続

CS+ コード生成設定(割り込み)

① 「割り込み」を押下

② 「外部割り込み」タブ

③ INT0、INT1、INT2、INT3、INT4にチェックする

INTP0 設定	INTP1 設定	INTP2 設定	INTP3 設定	INTP4 設定	INTP5 設定	INTP6 設定
<input checked="" type="checkbox"/> INTP0	<input checked="" type="checkbox"/> INTP1	<input checked="" type="checkbox"/> INTP2	<input checked="" type="checkbox"/> INTP3	<input checked="" type="checkbox"/> INTP4	<input type="checkbox"/> INTP5	<input type="checkbox"/> INTP6

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)

① 「ポート」を押下

② 「ポート6」タブに切り替え

③ 「P62」「P63」を出力へチェック、また「1」をチェックし初期値を1(High)にする。
P62はCPUボードのLED1、
P63はCPUボードのLED2が接続されている。

ポート	ポート1	ポート2	ポート3	ポート4	ポート5	ポート6	ポート7	ポート12	ポート13	ポート14
P60										
P61										
P62						<input checked="" type="radio"/> 出力				
P63						<input checked="" type="radio"/> 出力				

CS+ コード生成設定(ポート)

① 「ポート」を押下

② 「ポート1」タブに切り替え

③ 「P10」「P11」「P12」「P13」「P14」「P15」「P16」「P17」を出力へチェックする。
P10からP17は、7セグLEDの表示データ(AからG、DP)にそれぞれ接続する。

④ 「ポート5」タブに切り替え

⑤ 「P52」「P53」「P54」「P55」を出力へチェックする。
P52からP55は、7セグLEDの出力セクタ(C00からC03)にそれぞれ接続する。

端子の競合が発生している項目には、
⚠ アイコンが表示されます。
アイコン上にマウス・カーソルを移動すると、
競合回避のための情報が表示されます。

ポート	ポート0	ポート1	ポート2	ポート3	ポート4	ポート5	ポート6	ポート7	ポート12	ポート13	ポート14
	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1					
P52	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1	W0403001:以下の端子と競合しています。この機能を使用する場合は競合する機能の設定を無効にしてください。 P50はINTP1で使われています。				
P53	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1					
P54	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1					
P55	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> TTIバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1				
P10	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> TTIバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1				
P11	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> TTIバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1				
P12	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> TTIバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1				
P13	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> TTIバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1				
P14	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> TTIバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1				
P15	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> TTIバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1				
P16	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> TTIバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1				
P17	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="radio"/> 内蔵プルアップ	<input type="checkbox"/> TTIバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1				

CS+ コード生成設定(タイマ)

① 「タイマ」を押下

② チャンネル0「インターバル・タイマ」を選択

③ 「チャンネル0」タブに切り替え

④ 「100ms」になるように入力

⑤ チャンネル1「インターバル・タイマ」を選択

⑥ 「チャンネル1」タブに切り替え

⑦ 「2ms」になるように入力

The screenshot shows the CS+ IDE interface for configuring a timer. The left sidebar shows the project structure with 'タイマ' (Timer) selected. The main window displays the 'コード生成' (Code Generation) settings for Channel 0 and Channel 1. Channel 0 is configured with an interval of 100ms, and Channel 1 is configured with an interval of 2ms. The '割り込み設定' (Interrupt Settings) section is also visible, showing that interrupts are enabled for both channels.

CS+ コード生成設定(タイマ)

The image shows a screenshot of the CS+ IDE interface with several callouts and arrows indicating the configuration steps for a timer. The main window displays the 'コード生成' (Code Generation) settings for various channels. A red arrow points from the 'タイマ' (Timer) option in the left-hand project tree to the 'チャンネル2' (Channel 2) tab. Another red arrow points from the 'PWM出力(マスタ)' (PWM Output (Master)) option in the channel list to the 'チャンネル2(マスタ)' (Channel 2 (Master)) sub-tab. A third red arrow points from the '10' value in the '周期設定' (Period Setting) field to a callout box. A fourth red arrow points from the 'チャンネル4(スレーブ)' (Channel 4 (Slave)) sub-tab to a callout box. A fifth red arrow points from the '1' value in the 'デューティ' (Duty) field to a callout box. A sixth red arrow points from the 'チャンネル4(スレーブ)' sub-tab to a callout box. The interface includes a menu bar, a toolbar, a project tree, and a main settings area with tabs for '一般設定' (General Settings) and individual channels.

① 「タイマ」を押下

② チャンネル2「PWM出力」を選択

③ 「チャンネル2」「マスタ」タブに切り替え

④ 「10ms」になるように入力

⑤ 「スレーブ」タブに切り替え

⑥ 「1」を入力

CS+ コード生成設定(シリアル)

The image shows the CS+ IDE interface for configuring serial code generation. The main window displays the project tree on the left and the configuration panel on the right. The configuration panel is divided into tabs for different channels (SAU0, SAU1, IICA0, UART0, UART1, etc.).

1 「シリアル」を押下

2 「SAU0」「チャンネル」タブに切り替え

3 チャンネル2「UART1」「送信/受信機能」を選択

4 「UART1」「受信」タブに切り替え

5 「UART1」「送信」タブに切り替え

The configuration panel shows the following settings for UART1:

- チャンネル: UART1
- 機能: 送信/受信機能
- データ・ビット長設定: 8ビット
- データ転送方向設定: LSB
- パリティ設定: パリティなし
- ストップ・ビット長設定: 1ビット固定です
- 受信データ・レベル設定: 標準
- 転送レート設定: ポーレート 9600 (bps)
- 割り込み設定: 受信完了割り込み設定(INTSR1) 低
- コールバック機能設定: 受信完了

CS+ コード生成設定(A/Dコンバータ)

The screenshot shows the CS+ IDE interface for the project 'RL78G13_code_sample'. The 'コード生成*' (Code Generation) window is open, displaying various configuration options for the A/D converter. The settings are as follows:

- A/Dコンバータ動作設定:** 使用する (Selected)
- コンパレータ動作設定:** 許可 (Selected)
- 分解能設定:** 10ビット (Selected)
- VREF(+)設定:** VDD (Selected)
- VREF(-)設定:** VSS (Selected)
- トリガ・モード設定:** ソフトウェア・トリガ・モード (Selected)
- 動作モード設定:** 連続セレクト・モード (Selected)
- 変換開始チャンネル設定:** ANI0 (Selected)
- 割り込み設定:** A/Dの割り込み許可(INTAD) (Selected)

Callout 1 points to the 'A/Dコンバータ' option in the left-hand project tree.

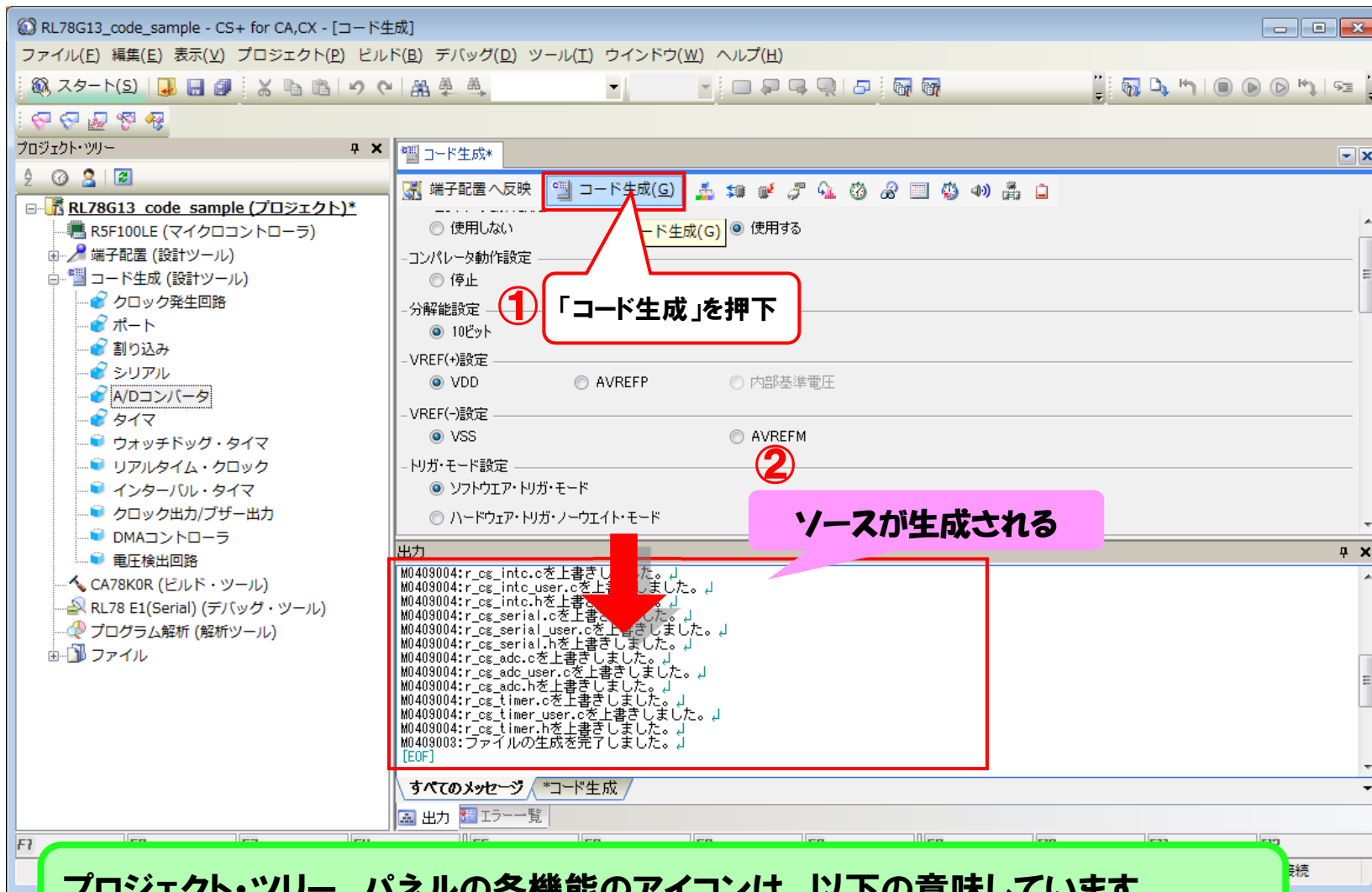
Callout 2 points to the '使用する' radio button.

Callout 3 points to the '許可' radio button.




Callout 4 points to the 'ANI0' dropdown menu in the '変換開始チャンネル設定' section.

Callout 5 points to the checkboxes for ANI16, ANI17, ANI18, and ANI19, which are currently unchecked.

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

② 端子配置表、端子配置図が生成される

出力

```
M0409005: 端子番号 19 を P62 から P62 に変更しました。
M0409005: 端子番号 20 を P63 から P63 に変更しました。
M0409005: 端子番号 35 を P52 から P52 に変更しました。
M0409005: 端子番号 36 を P53 から P53 に変更しました。
M0409005: 端子番号 37 を P54 から P54 に変更しました。
M0409005: 端子番号 38 を P55 から P55 に変更しました。
M0409005: 端子番号 39 を P17 から P17 に変更しました。
M0409005: 端子番号 40 を P16 から P16 に変更しました。
M0409005: 端子番号 41 を P15 から P15 に変更しました。
M0409005: 端子番号 42 を P14 から P14 に変更しました。
M0409005: 端子番号 43 を P13 から P13 に変更しました。
M0409005: 端子番号 44 を P12 から P12 に変更しました。
M0409005: 端子番号 45 を P11 から P11 に変更しました。
M0409005: 端子番号 46 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]
```

ポート	ポート1	ポート2	ポート3	ポート4	ポート5	ポート6	ポート7	ポート12	ポート13	ポート14
P10	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="checkbox"/> 内蔵プルアップ	<input type="checkbox"/> TTLバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1			
P11	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="checkbox"/> 内蔵プルアップ	<input type="checkbox"/> TTLバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1			
P12	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="checkbox"/> 内蔵プルアップ	<input type="checkbox"/> TTLバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1			
P13	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="checkbox"/> 内蔵プルアップ	<input type="checkbox"/> TTLバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1			
P14	<input type="radio"/> 使用しない	<input type="radio"/> 入力	<input checked="" type="radio"/> 出力	<input type="checkbox"/> 内蔵プルアップ	<input type="checkbox"/> TTLバッファ	<input type="checkbox"/> N-ch	<input type="checkbox"/> 1			

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)

端子配置図のプロパティ

色設定		
電源端子	Red	Red
特殊端子	Green	Green
未使用端子	Black	Black
使用端子	Blue	Blue
デバイス	LightGray	LightGray
端子の強調表示	Red	Red
マクロの強調表示	Yellow	Yellow
外部周辺の強調表示	Green	Green
ツールチップの表示設定		
ツールチップの表示設定	すべて表示	すべて表示
端子名表示設定		
定義名	表示する	表示する
兼用機能	すべて	すべて

CS+ 端子配置表

① 「端子配置表」をダブルクリック

端子番号	端子名	選択機能	I/O	N-ch	定義名	説明	未使用時の処置方法
1	P120/ANI19	Free	-	-			入力時:個別に抵抗を介して, EVDD0またはEVSS0に接続してください。 出力時:オープンにしてください。
2	P43	Free	-	-			入力時:個別に抵抗を介して, EVDD0またはEVSS0に接続してください。 出力時:オープンにしてください。
3	P42/TI04/T004	Free	-	-			入力時:個別に抵抗を介して, EVDD0またはEVSS0に接続してください。 出力時:オープンにしてください。
4	P41/TI07/T007	Free	-	-			入力時:個別に抵抗を介して, EVDD0またはEVSS0に接続してください。 出力時:オープンにしてください。
5	P40/TOOL0	TOOL0	I/O	-		フラッシュ・メモリ・プログラマ/デバッグ用 データ入出力	
6	_RESET	_RESET	I	-		外部リセット入力	
7	P124/X12/EXCLKS	Free	-	-			個別に抵抗を介して, VDDまたはVSSに 接続してください。
8	P123/X11	Free	-	-			個別に抵抗を介して, VDDまたはVSSに 接続してください。
9	P137/INTP0	INTP0	I	-		外部割り込み入力	
10	P122/X2/EXCLK	Free	-	-			個別に抵抗を介して, VDDまたはVSSに 接続してください。
11	P121/X1	Free	-	-			個別に抵抗を介して, VDDまたはVSSに 接続してください。
12	REGC	REGC	-	-		内部動作レギュレータ出力安定容量 接続	
13	VSS	VSS	-	-		P20-P27, P121-P124, P137, P150- P156, およびポート以外の端子のグラウンド 電位	
14	EVSS0	EVSS0	-	-		ポート端子(P20-P27, P121-P124, P137, P150-P156)のグラウンド電位	

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/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。

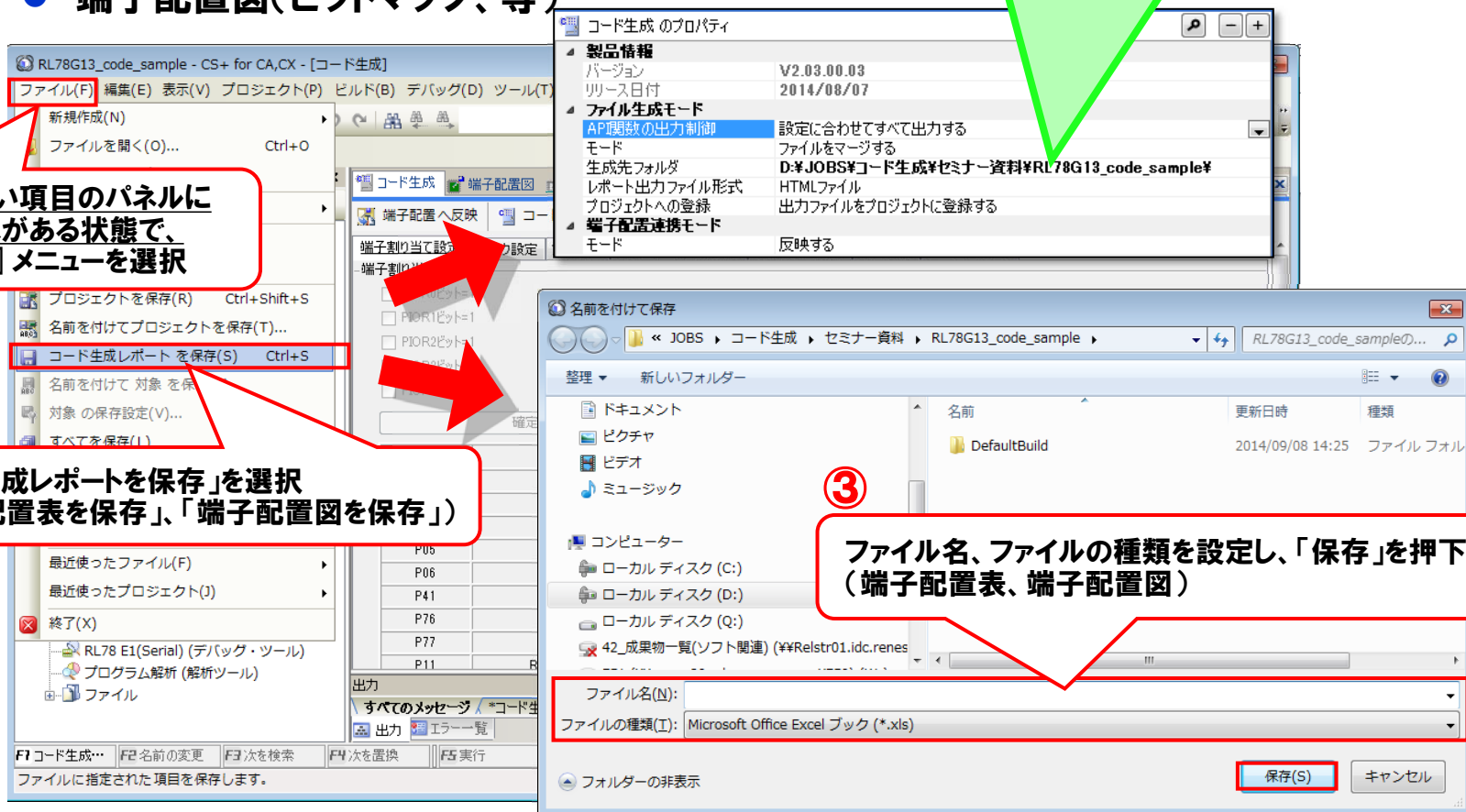
- コード生成レポート(HTMLファイル/CSVファイル)
- 端子配置表(Microsoft Office Excelブック形式)
- 端子配置図(ビットマップ、等)

コード生成レポートは、「コード生成のプロパティ」で設定したフォルダに出力されます。

① 保存したい項目のパネルにフォーカスがある状態で、「[ファイル]」メニューを選択

② 「コード生成レポートを保存」を選択
(「端子配置表を保存」、「端子配置図を保存」)

③ ファイル名、ファイルの種類を設定し、「保存」を押下します。
(端子配置表、端子配置図)



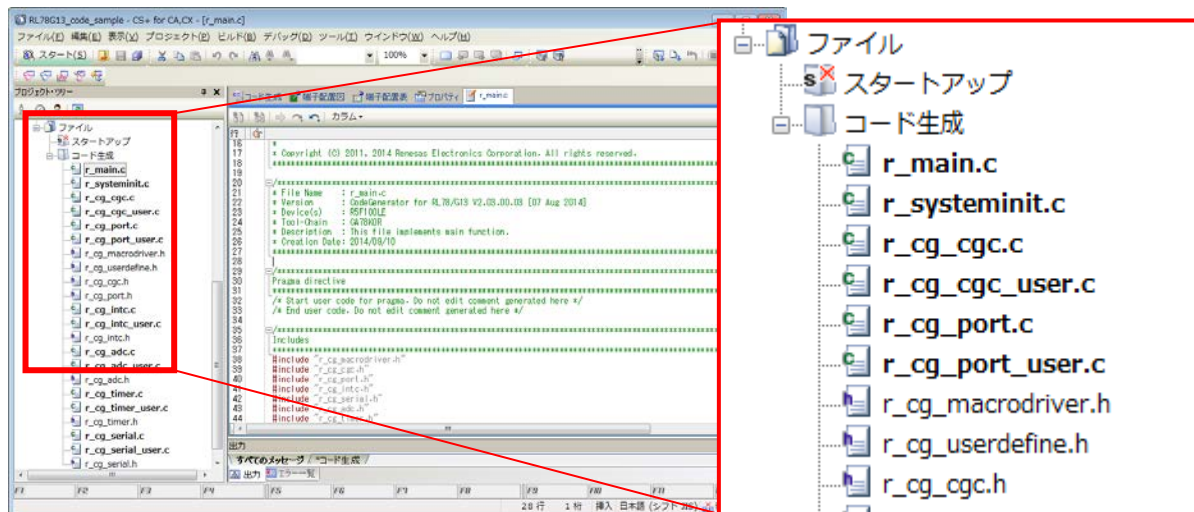
CS+ コード生成で出力されたファイル

■ 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

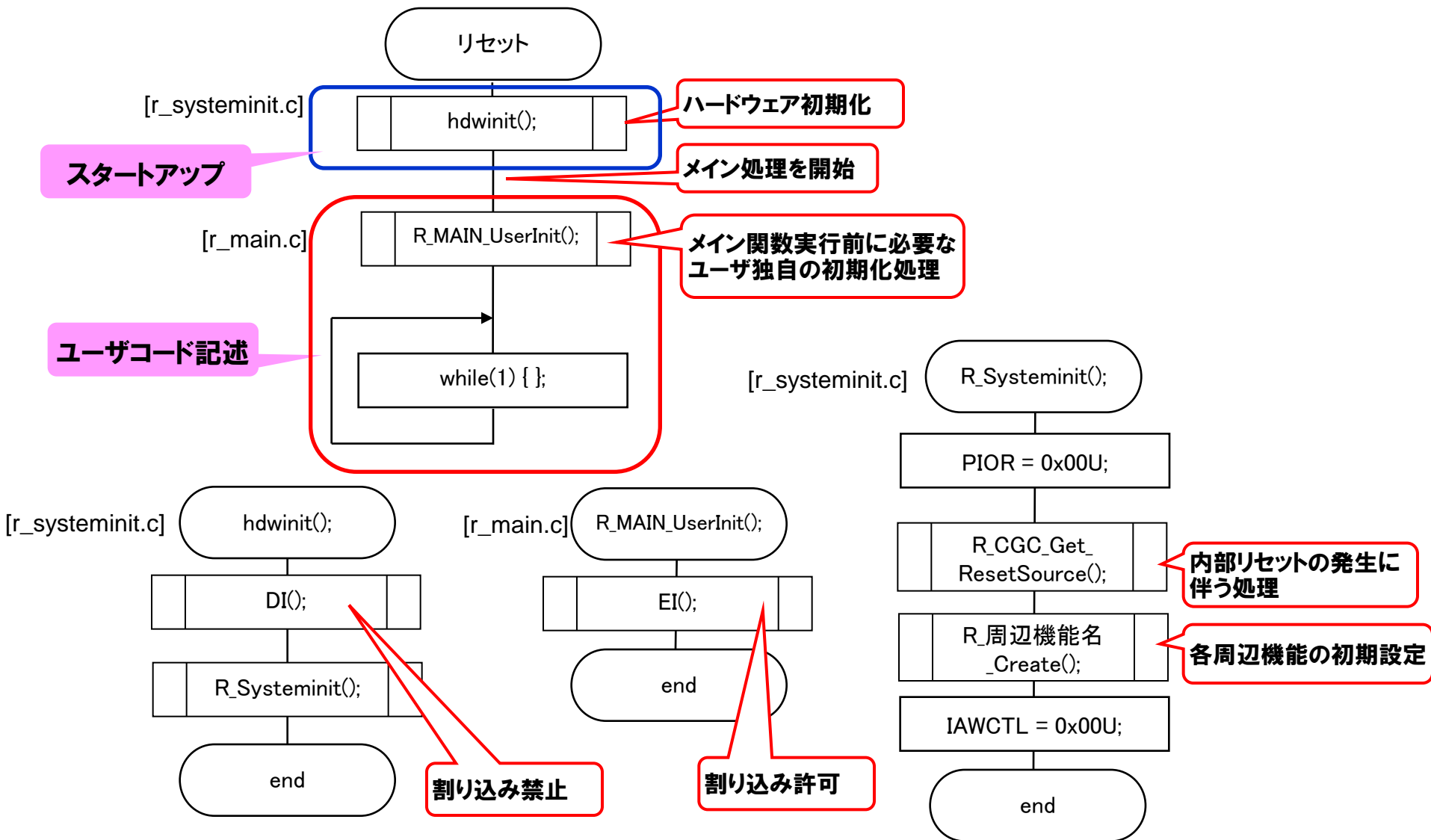
ファイル名	API関数名	機能概要
r_systemini.c	hdwinit	ルネサスエレクトロニクス製コンパイラ提供のスタートアップ・ルーチンから自動で呼ばれます。この関数内で R_Systeminit () を呼びます。
	R_Systeminit	hdwinit関数から自動で呼ばれます。この関数内で、リセット要因を確認する関数、および各周辺機能の初期設定を行う関数 (R_周辺機能名_Create) を呼びます。
r_main.c	R_MAIN_UserInit	メイン関数実行前に必要なユーザ独自の初期化処理を行います。
	main	メイン関数
r_cg_macrodriver	-	全ソ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

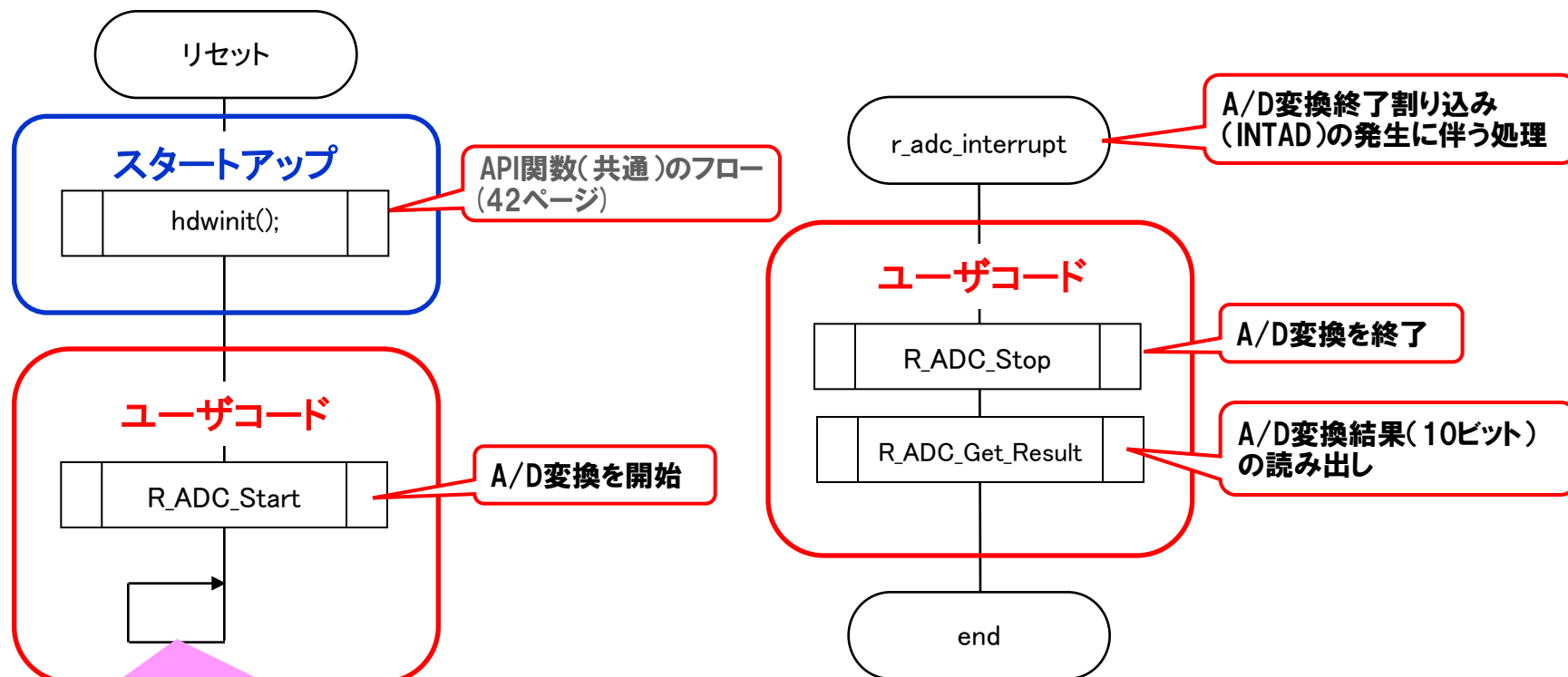
■ A/Dコンバータ API関数一覧

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

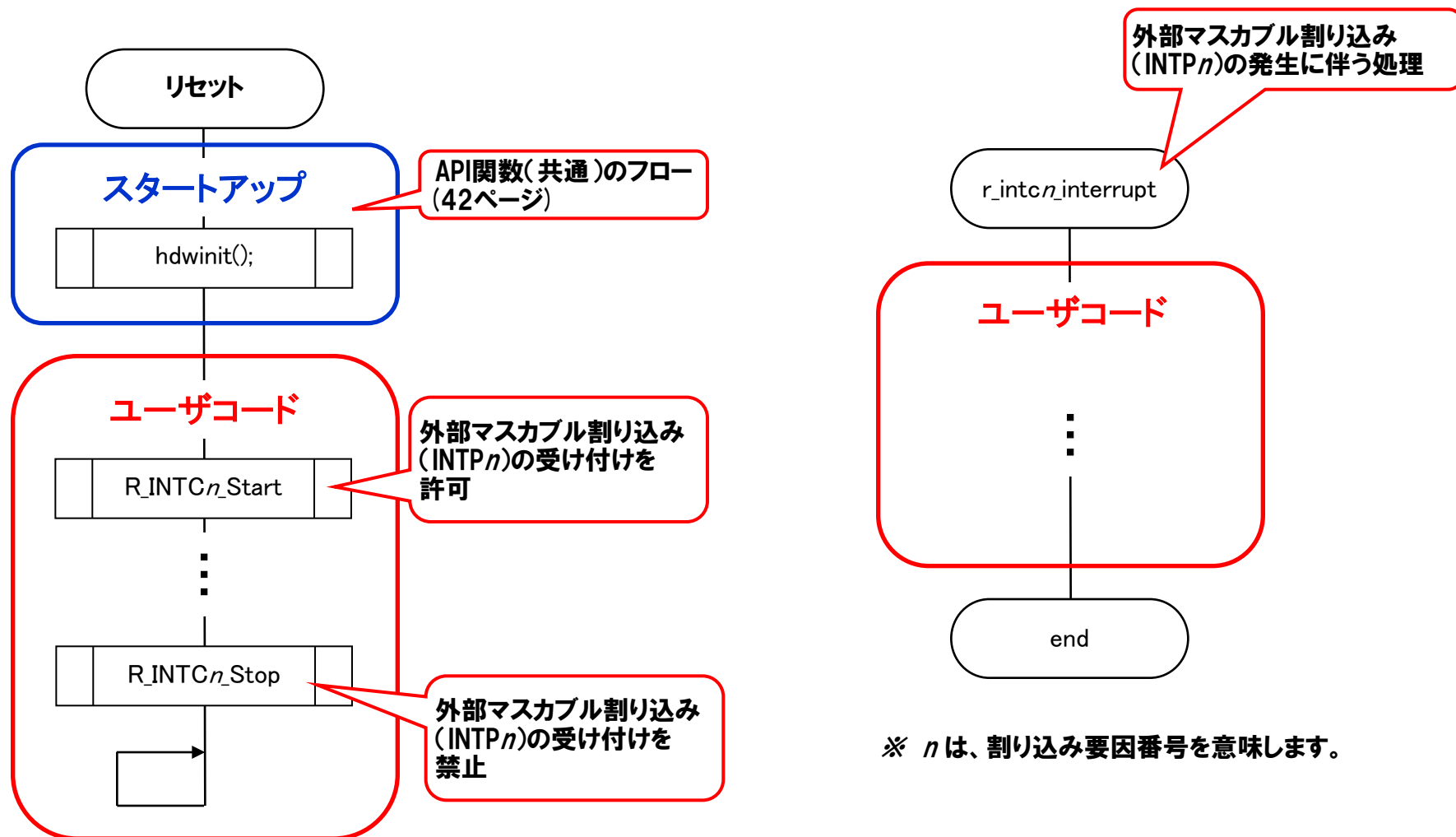
ファイル名	API関数名	機能概要
r_cg_intc.c	R_INTC_Create	外部マスカブル割り込み(INTP _n /INTPLR _n)の機能を制御するうえで必要となる初期化処理を行います。
	R_INTC _n _Start	外部マスカブル割り込み(INTP _n)の受け付けを許可します。
	R_INTC _n _Stop	外部マスカブル割り込み(INTP _n)の受け付けを禁止します。
	R_INTCLR _n _Start	外部マスカブル割り込み(INTPLR _n)の受け付けを許可します。
	R_INTCLR _n _Stop	外部マスカブル割り込み(INTPLR _n)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



CS+ コード生成が出力するAPI関数(周辺機能)

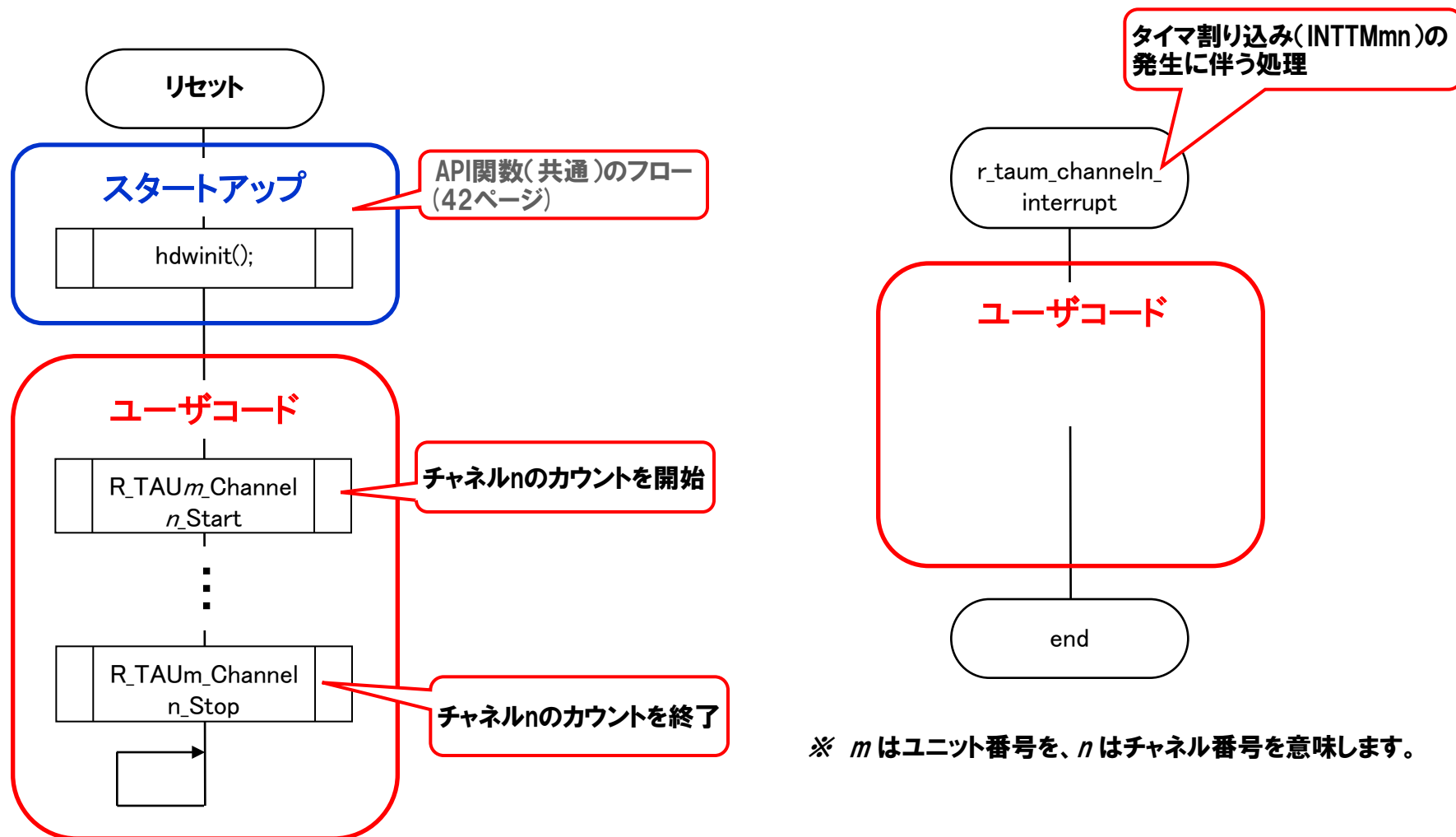
■ タイマ・アレイ・ユニット API関数一覧

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	チャンネル n のカウントを開始します。
	R_TAU m _Channel n _Higher8bits_Start	チャンネル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	チャンネル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	チャンネル n のカウントを終了します。
	R_TAU m _Channel n _Higher8bits_Stop	チャンネル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	チャンネル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	T1 m 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(ソフトウェア・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(INTT m n)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(INTT m n 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 はチャン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



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_SAUM_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_SAUM_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_SAUM_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_SAUM_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_UARTn_Create	シリアル・インタフェース(UART)用チャンネルの初期化処理を行います。
	R_UARTn_Start	UART通信を待機状態にします。
	R_UARTn_Stop	UART通信を終了します。
	R_UARTn_Send	データのUART送信を開始します。
	R_UARTn_Receive	データのUART受信を開始します。
	R_CSImn_Create	シリアル・インタフェース(CSI)用チャンネルの初期化処理を行います。
	R_CSImn_Start	CSI通信を待機状態にします。
	R_CSImn_Stop	CSI通信を終了します。
	R_CSImn_Send	データのCSI送信を開始します。
	R_CSImn_Receive	データのCSI受信を開始します。
	R_CSImn_Send_Receive	データのCSI送受信を開始します。
	R_IICmn_Create	シリアル・インタフェース(簡易IIC)用チャンネルの初期化処理を行います。
	R_IICmn_StartCondition	スタート・コンディションを発生させます。
	R_IICmn_StopCondition	ストップ・コンディションを発生させます。
	R_IICmn_Stop	簡易IIC通信を終了します。
	R_IICmn_Master_Send	簡易IICマスタ送信を開始します。
R_IICmn_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* はチャン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)

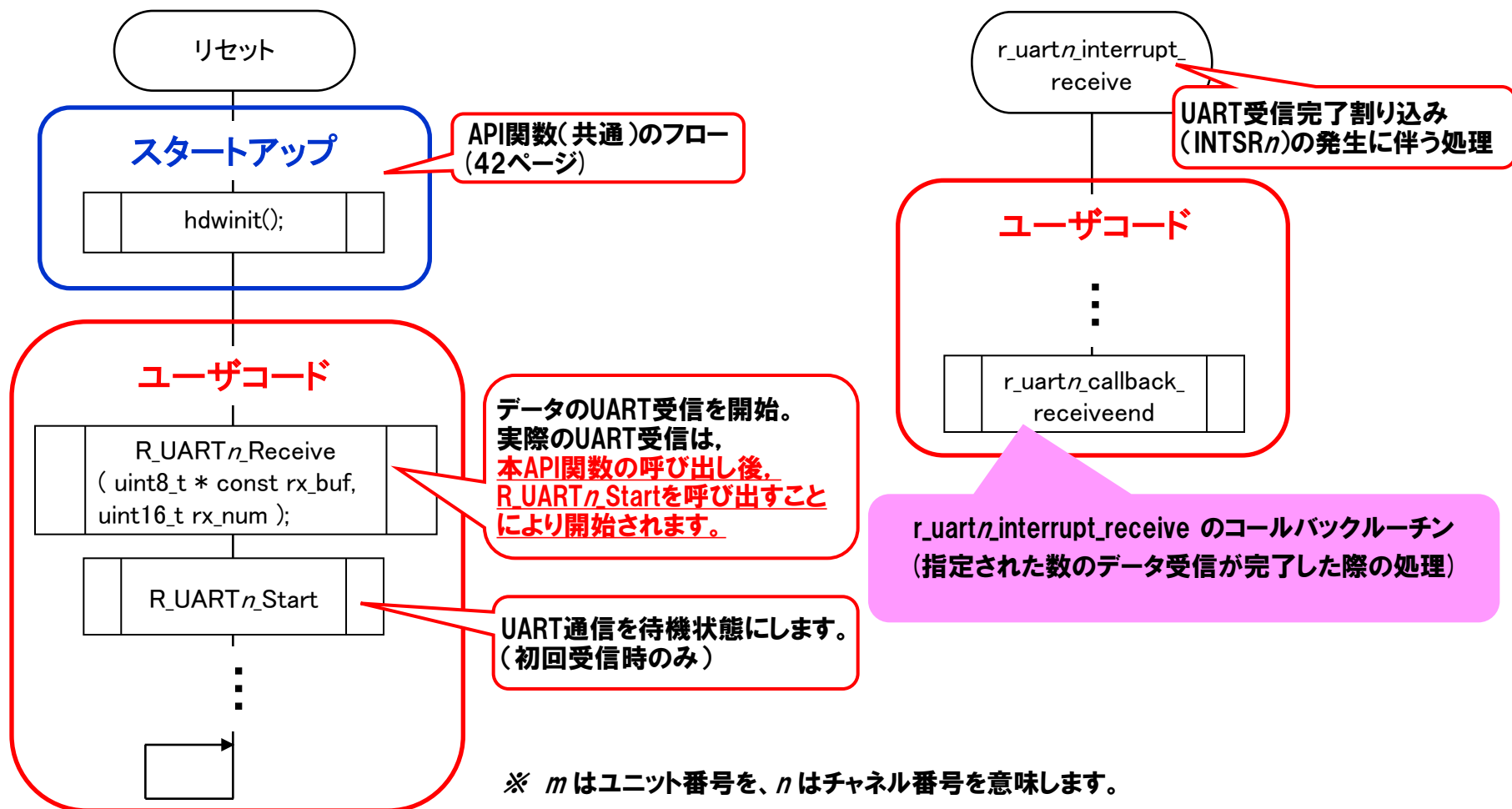
ファイル名	API関数名	機能概要
r_cg_serial_user.c	R_SAU m _Create_UserInit	シリアル・アレイ・ユニット, およびシリアル・インタフェースに関するユーザ独自の初期化処理を行います。
	r_uart n _interrupt_send	UART送信完了割り込み(INTST n)の発生に伴う処理を行います。
	r_uart n _interrupt_receive	UART受信完了割り込み(INTSR n)の発生に伴う処理を行います。
	r_uart n _interrupt_error	受信エラー割り込み(INTSRE n)の発生に伴う処理を行います。
	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_csim n _interrupt	CSI通信完了割り込み(INTCSI m n)の発生に伴う処理を行います。
	r_csim n _callback_sendend	CSI送信完了割り込み(INTCSI m n)の発生に伴う処理を行います。
	r_csim n _callback_receiveend	CSI受信完了割り込み(INTCSI m n)の発生に伴う処理を行います。
	r_csim n _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 m n _interrupt	簡易IIC通信完了割り込み(INTIIC m n)の発生に伴う処理を行います。
	r_iic m n _callback_master_sendend	簡易IICマスタ送信完了割り込み(INTIIC m n)の発生に伴う処理を行います。
r_iic m n _callback_master_receiveend	簡易IICマスタ受信完了割り込み(INTIIC m n)の発生に伴う処理を行います。	
r_iic m n _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 はチャンネル番号を意味します。

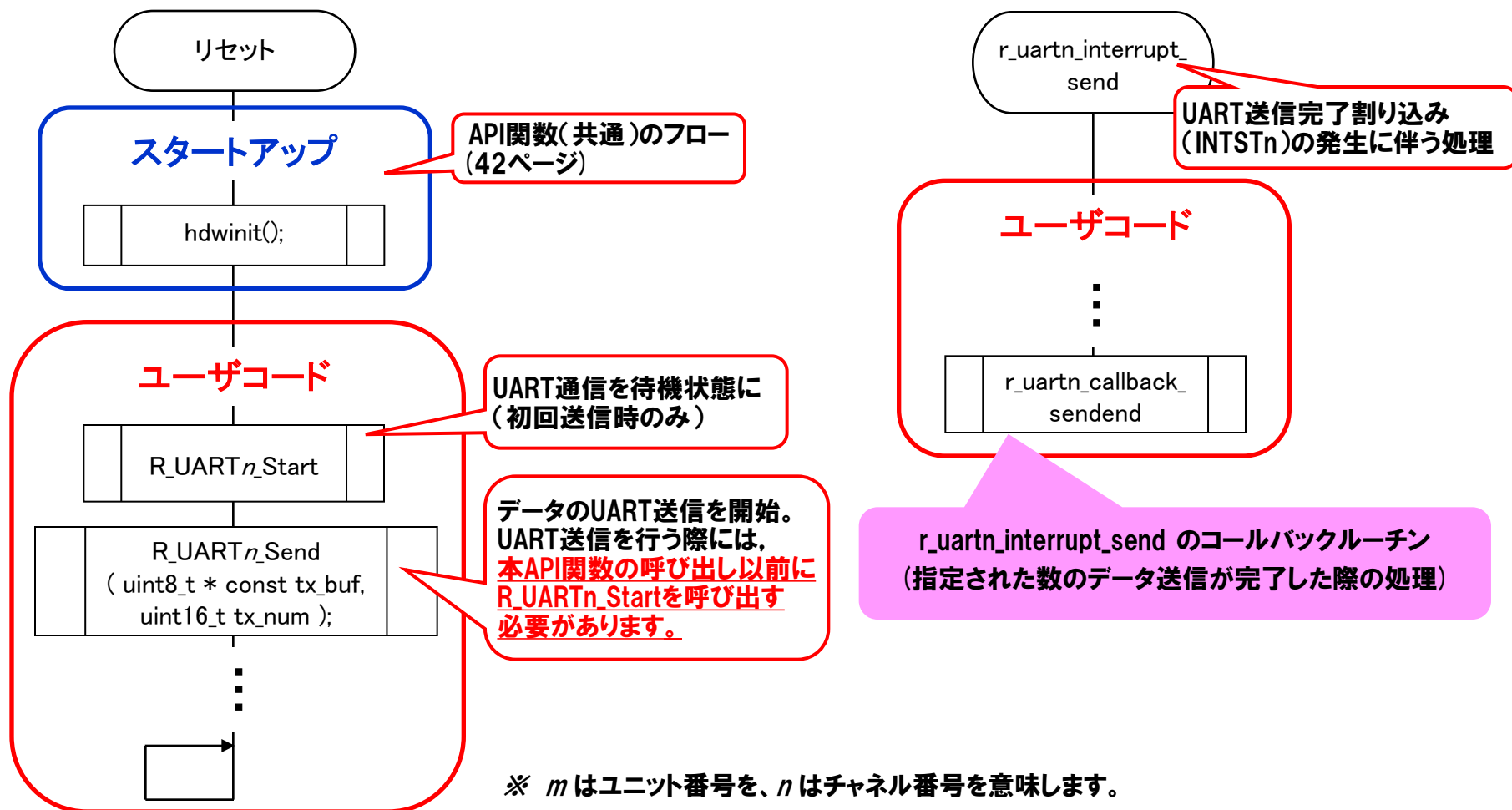
CS+ コード生成が出力するAPI関数(周辺機能)

■ 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



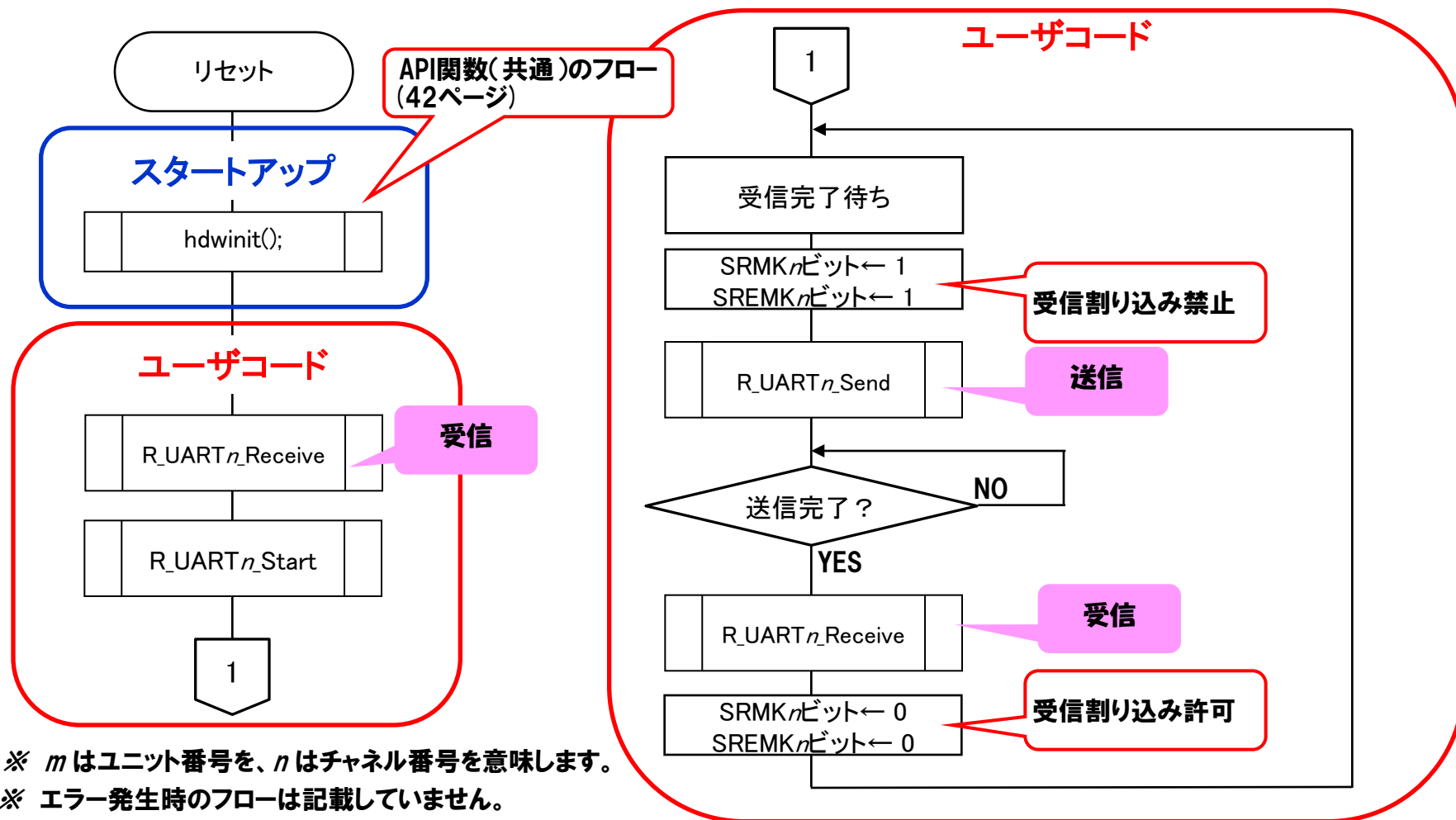
CS+ コード生成が出力するAPI関数(周辺機能)

■ 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

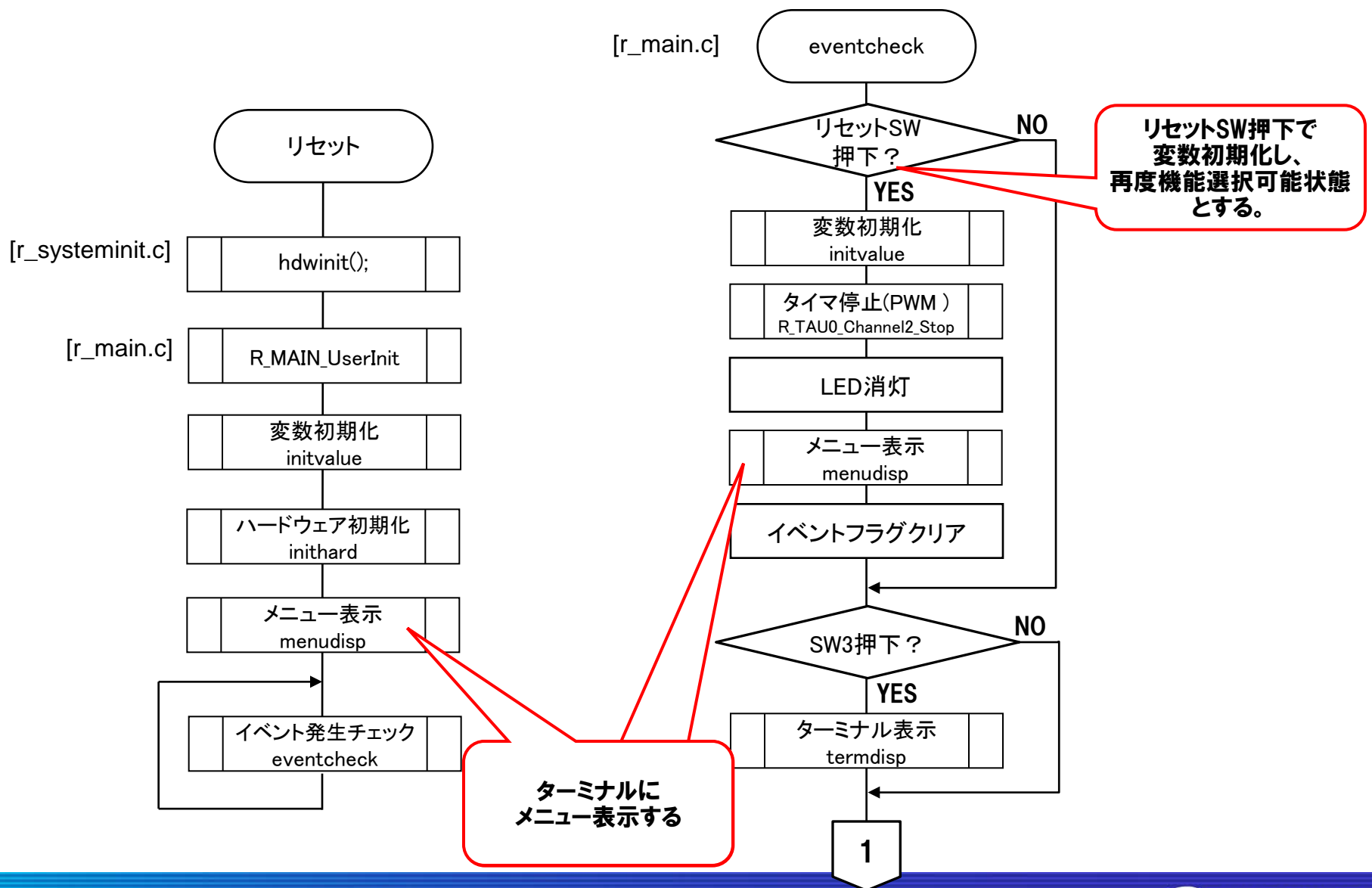


CS+ コード生成が出力するAPI関数(周辺機能)

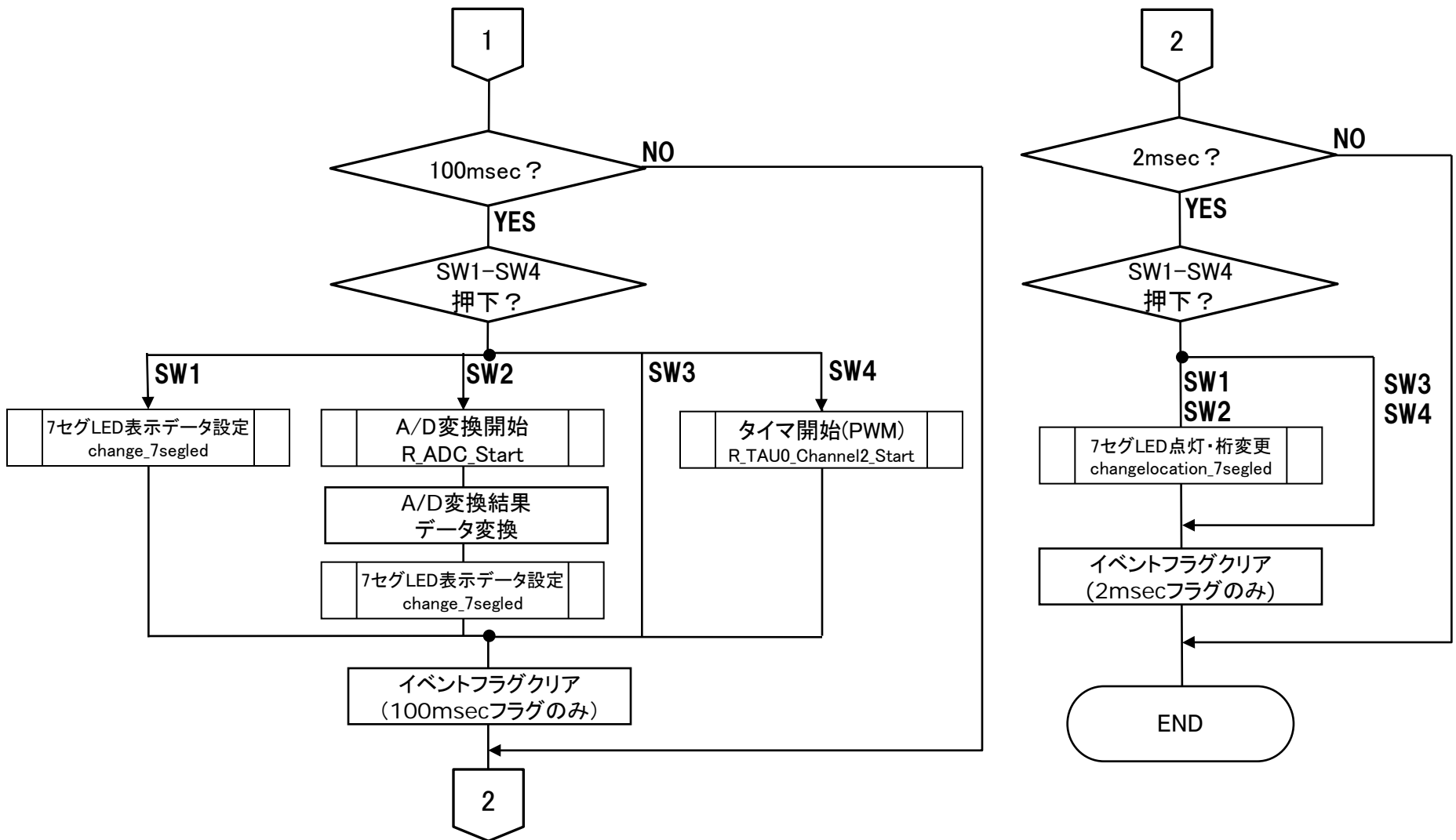
■ シリアル・アレイ・ユニット (シリアル・インタフェース(UART)送受信) 使用例



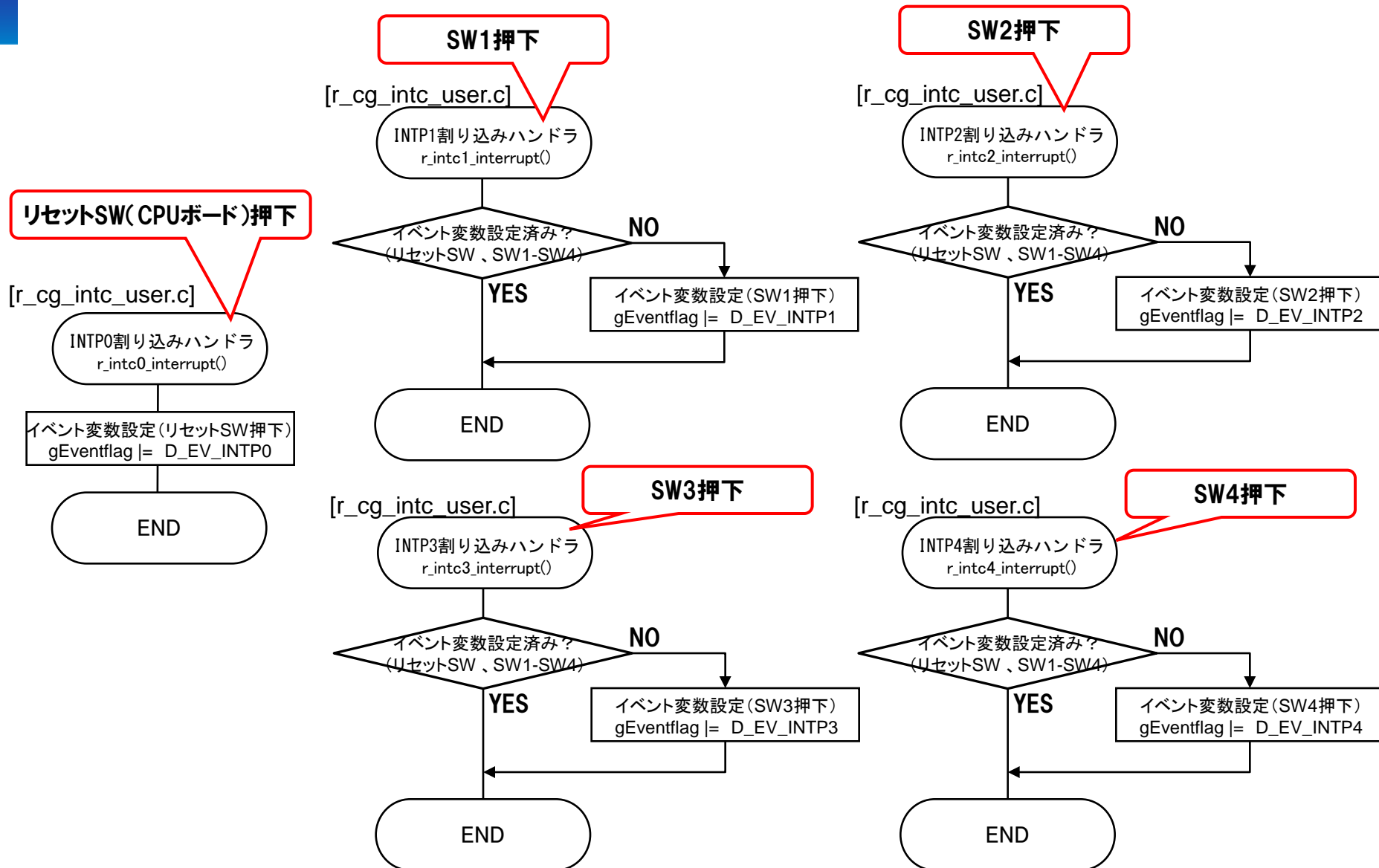
CS+ サンプルプログラムのフロー (1/9)



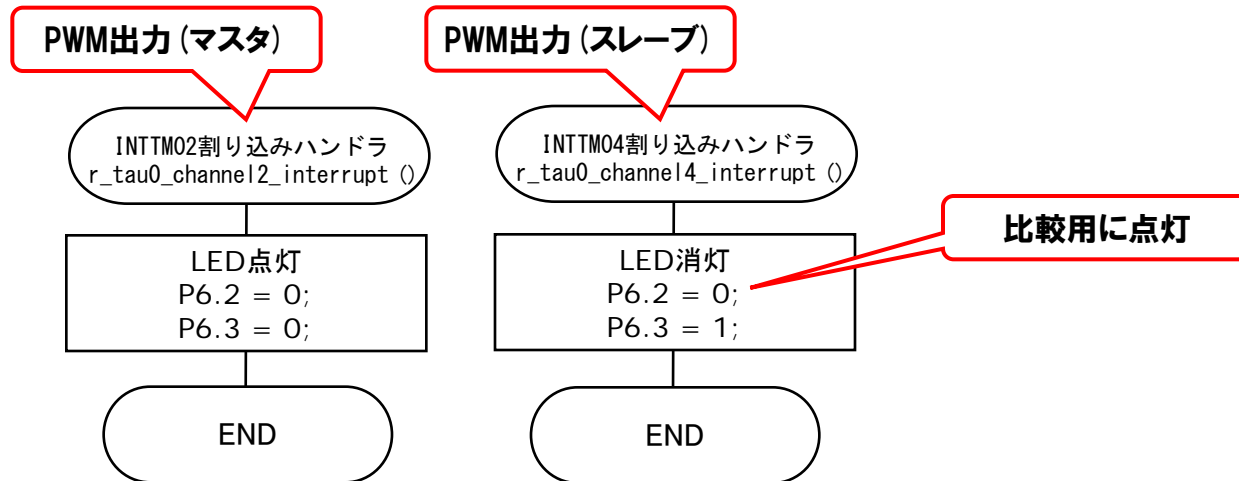
CS+ サンプルプログラムのフロー (2/9)



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。

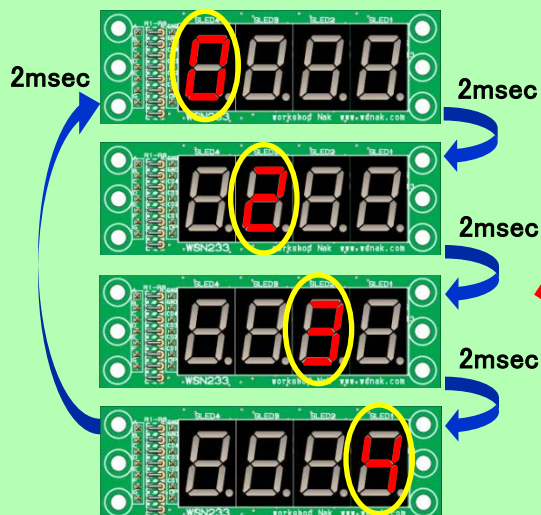
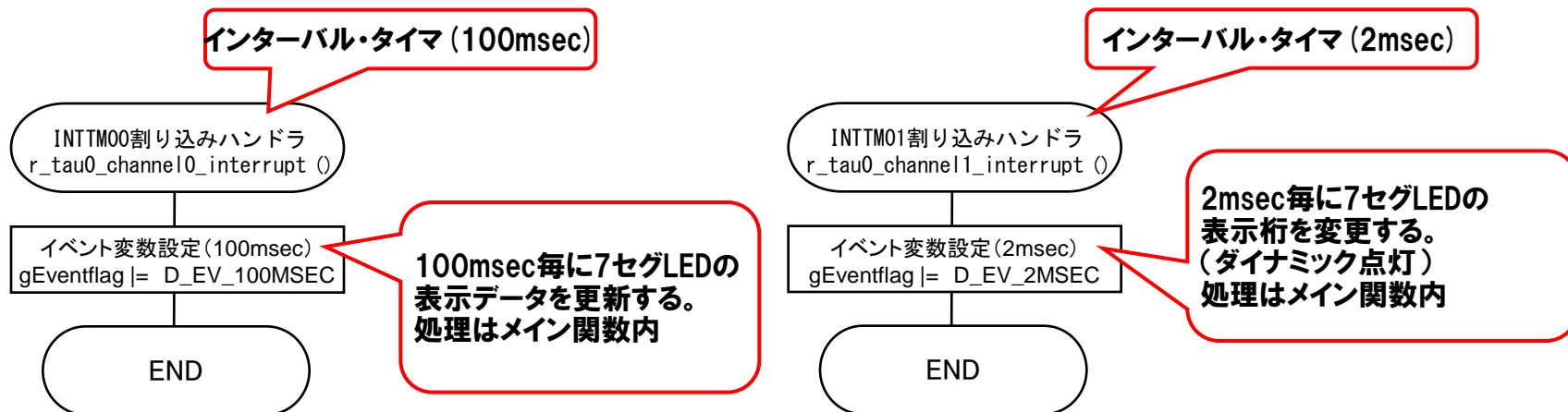
INTTM02割り込み発生

INTTM04割り込み発生



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)

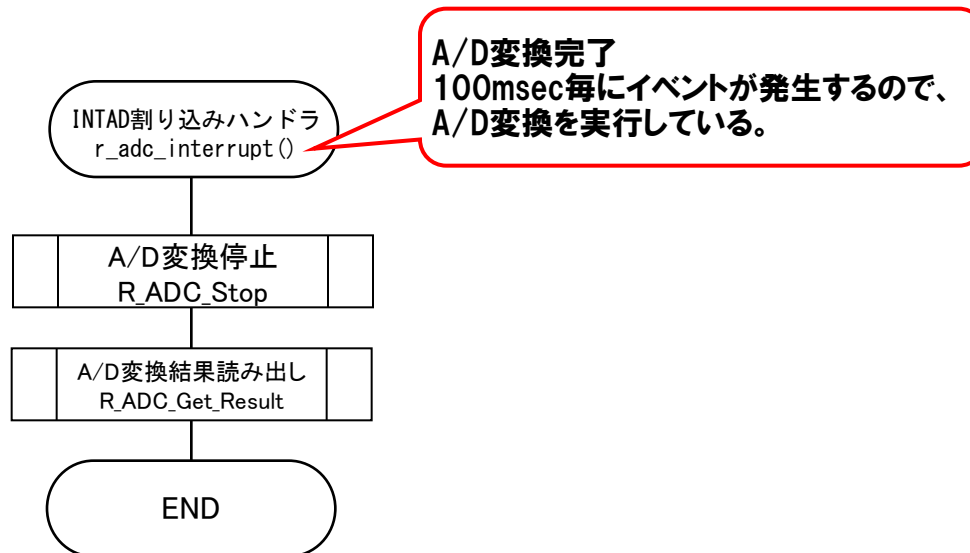


ダイナミック点灯:

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んと思います。

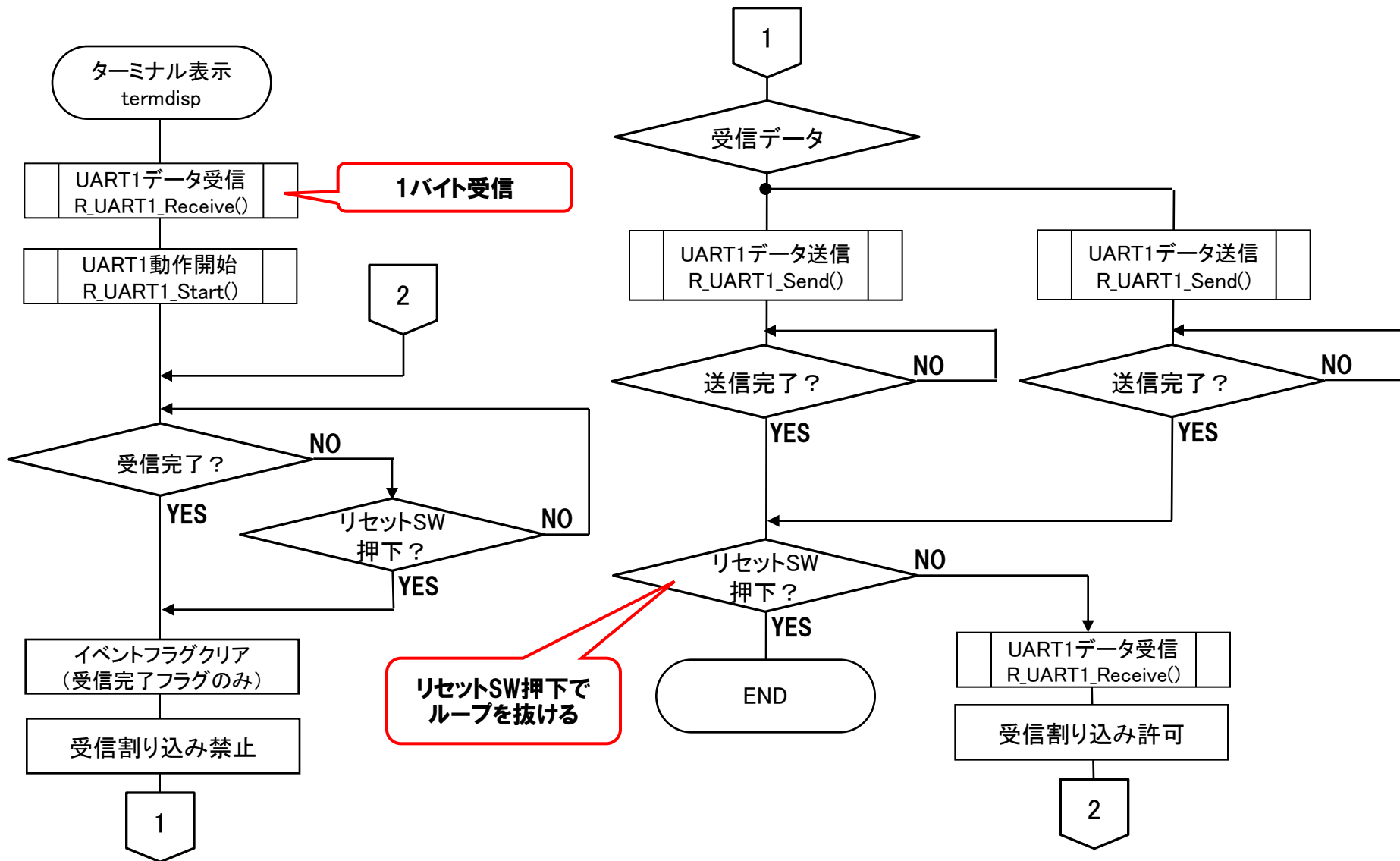
このような細かな変化をそのまま使用すると、目障りとなる場合があります。そのため、移動平均法を用いた簡単なフィルタを利用することで不要な細かい変動を取り除くことができます。

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。

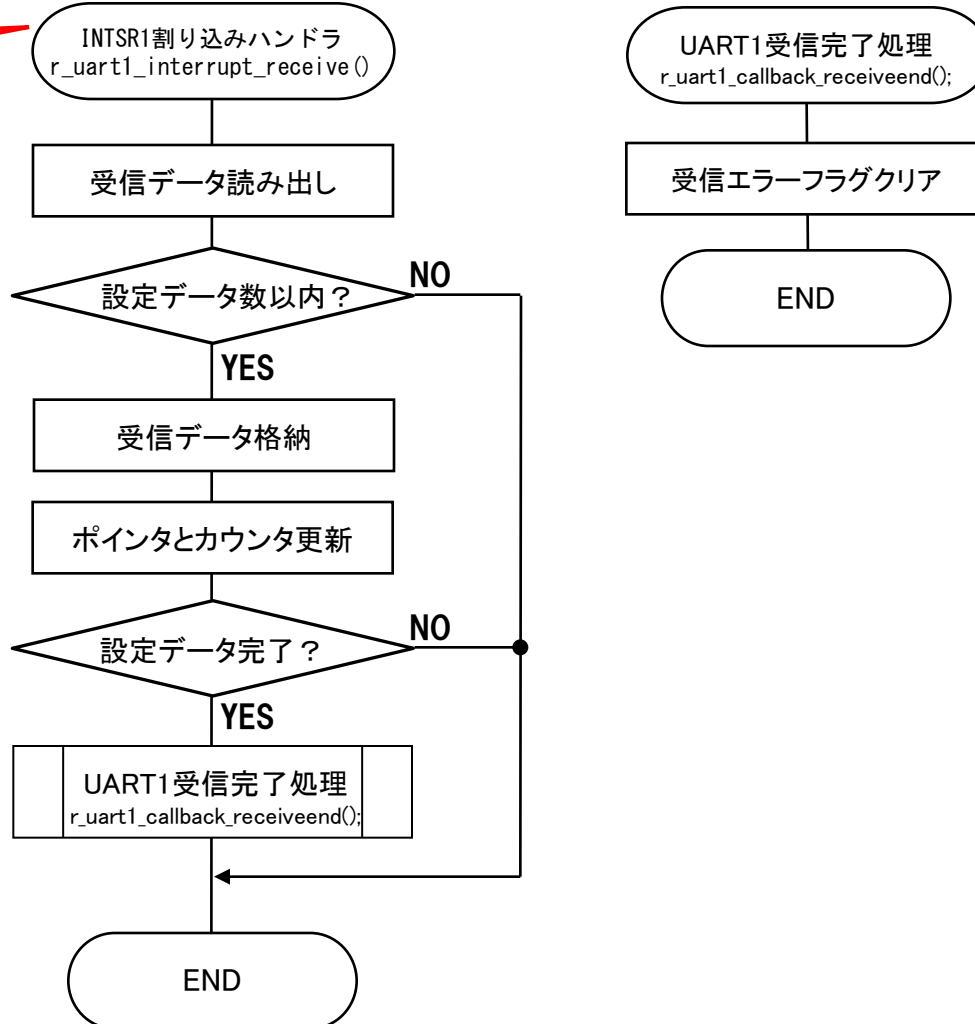
$$(7セグLEDの表示値) = \frac{x[n] + x[n-1] + x[n-2]}{3}$$

CS+ サンプルプログラムのフロー (7/9)

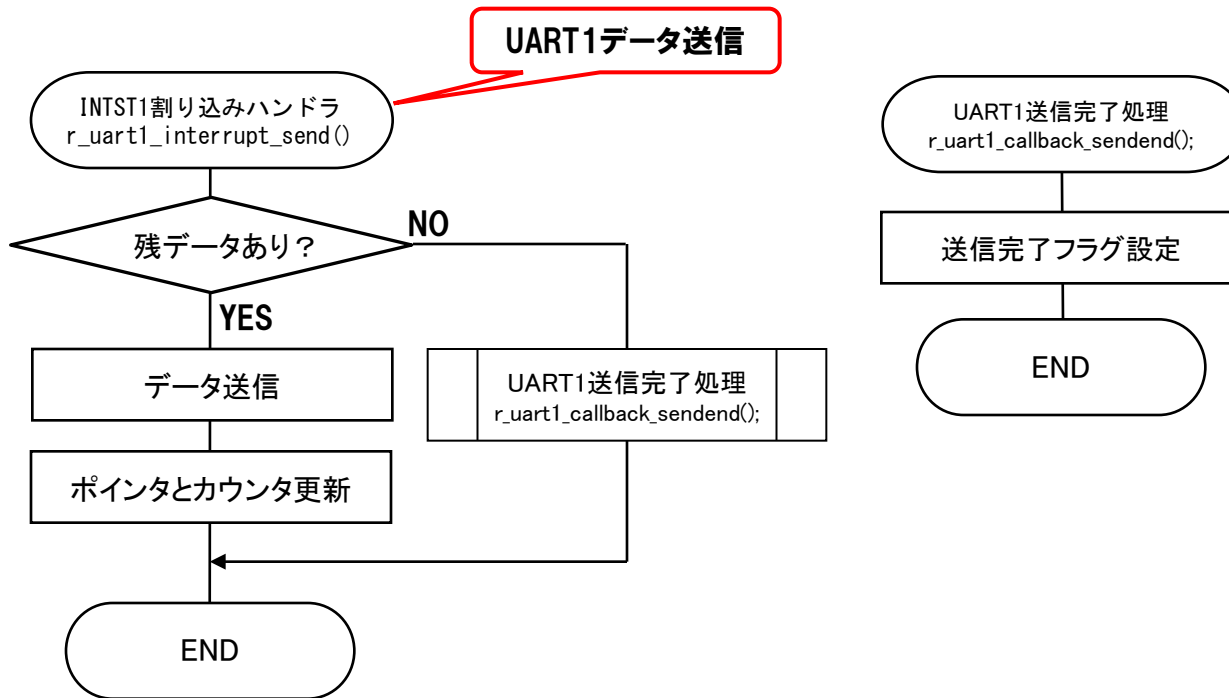


CS+ サンプルプログラムのフロー (8/9)

UART1データ受信



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

The screenshot displays the CS+ IDE interface. The 'ビルド(B)' menu is open, with 'ビルド・プロジェクト(B)' highlighted. A red box and arrow point to this menu item, with callout ①. The output window at the bottom shows the build process, including file compilation and linking. A red box and arrow point to the final build completion message in the output window, with callout ②.

ビルド(B) デバッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)

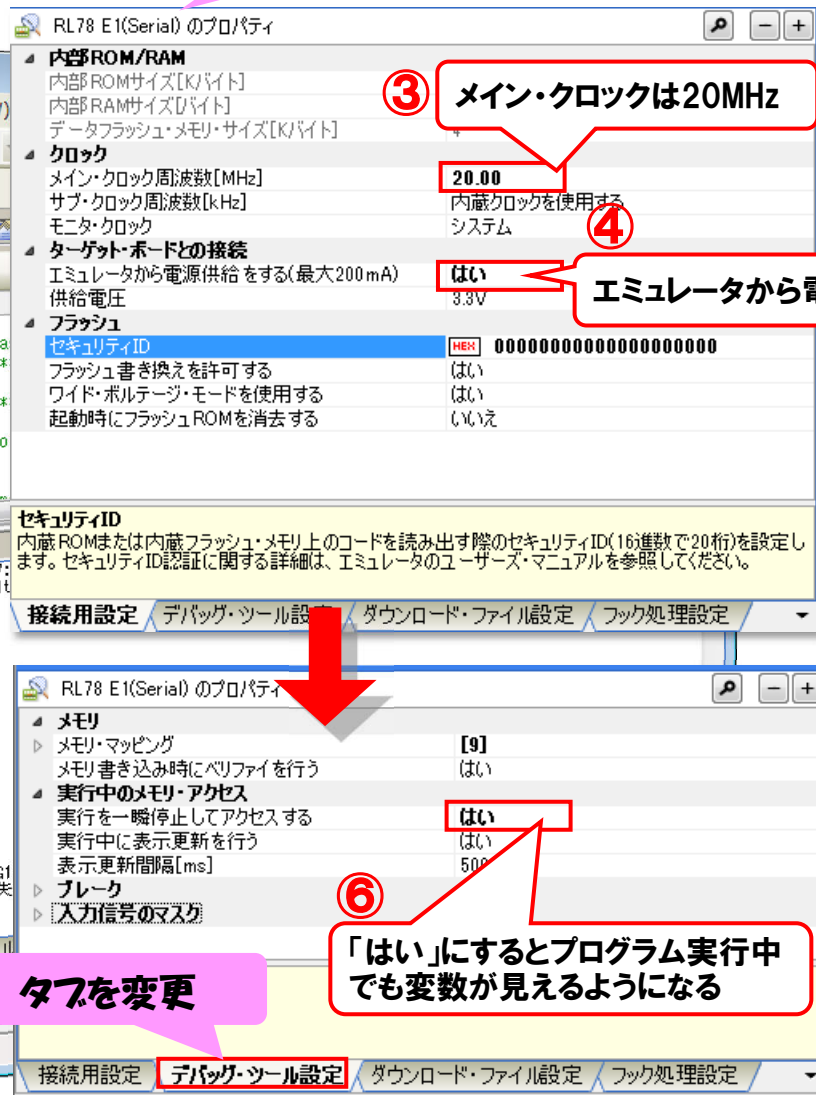
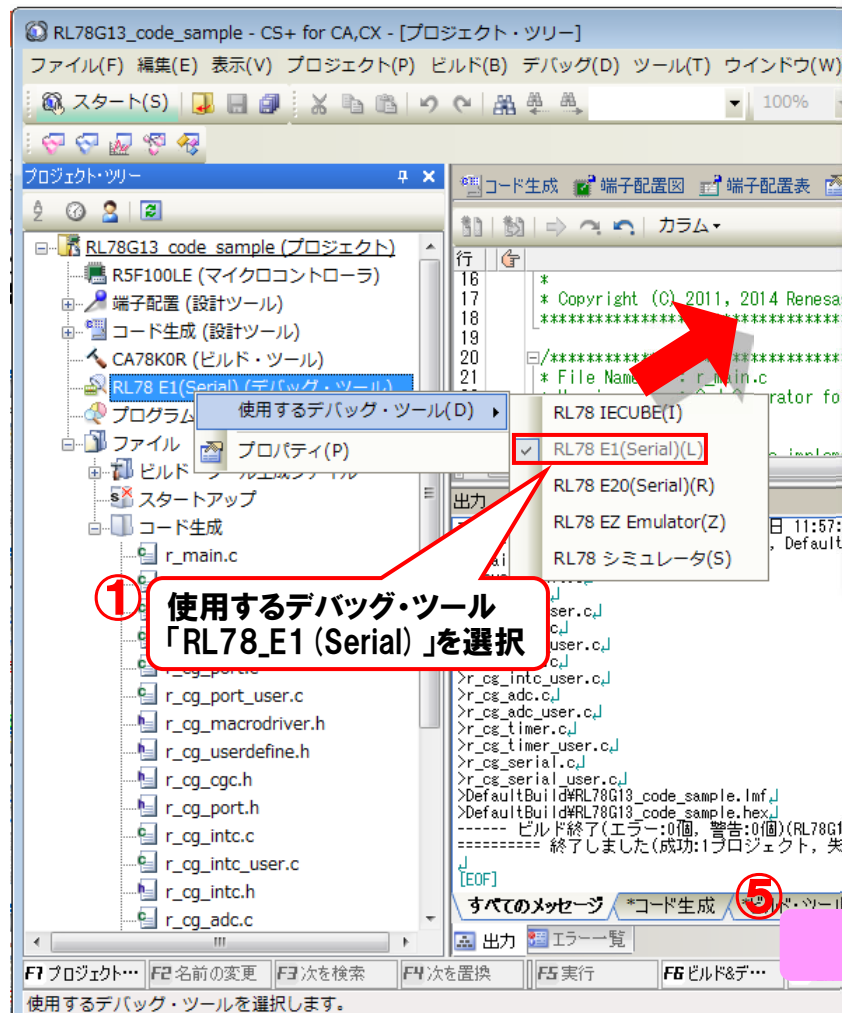
- ビルド・プロジェクト(B) F7
- リビルド・プロジェクト(R) Shift+F7
- クリーン・プロジェクト(C)
- ラビッド・ビルド(A)
- 依存関係の更新(P)
- RL78G13_code_sample をビルド(U)
- RL78G13_code_sample をリビルド(E)
- RL78G13_code_sample をクリーン(L)
- RL78G13_code_sample の依存関係の更新(D)
- RL78G13_code_sample のリンク順を設定する(K)...
- ビルドを中止(S) Ctrl+F7
- ビルド・モードの設定(M)...
- バッチ・ビルド(T)...
- ビルド・オプション一覧(O)

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]
```


CS+ デバッグツール設定

② RL78 E1 (Serial) のプロパティを開く



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・ツールヘダウンロード

The screenshot shows the CS+ IDE interface with the 'Build & Debug' menu open. A red box highlights the 'リビルド&デバッグ・ツールヘダウンロード(W)' option, with a callout bubble containing the text 'ビルド&デバッグ・ツールヘダウンロードを選択する'. A red arrow points to the '進捗状況' dialog box, which contains the message: 'RL78 E1(Serial) に、接続処理中です。エミュレータ・ファームウェアの更新が必要な場合、自動的に更新を行います。接続が完了するまではUSBおよび電源は切断しないでください。' and a 'キャンセル' button. A second callout bubble points to the output panel, stating '出力パネルには、ビルド終了のメッセージが表示'. The output panel shows the following text:

```
出力
>r_cg_port.c\j
>r_cg_port_user.c\j
>r_cg_intc.c\j
>r_cg_intc_user.c\j
>r_cg_adc.c\j
>r_cg_adc_user.c\j
>r_cg_timer.c\j
>r_cg_timer_user.c\j
>r_cg_serial.c\j
>r_cg_serial_user.c\j
>DefaultBuild#RL78G13_code_sample.lmf\j
>DefaultBuild#RL78G13_code_sample.hex\j
----- ビルド終了(エラー:0個, 警告:0個)(RL78G13_code sample, DefaultBuild) -----\j
===== 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 15:13:52) =====\j
\j
[EOF]
```

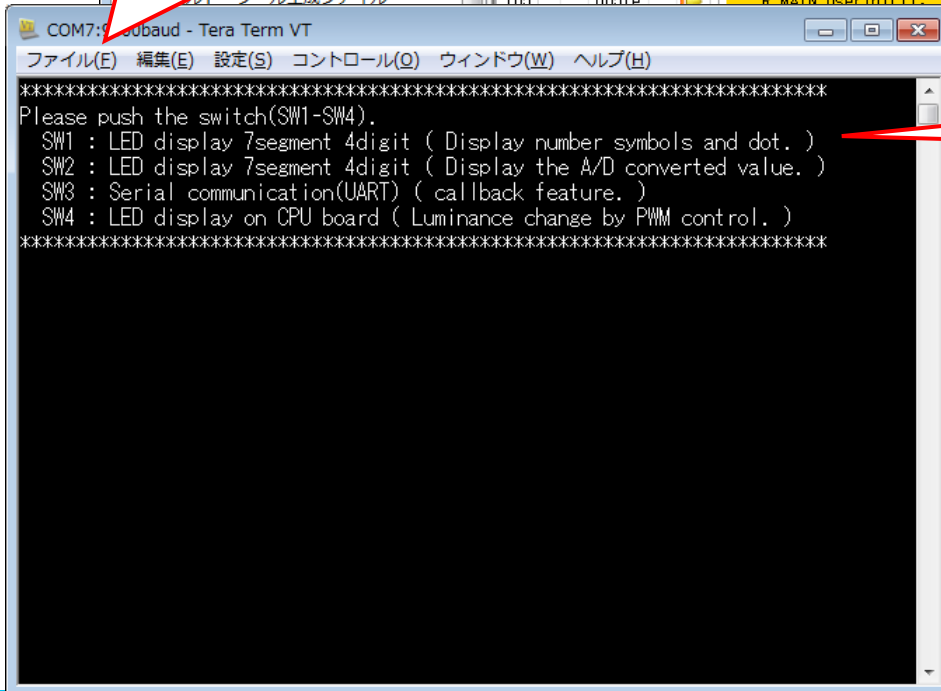
CS+ プログラム実行

②

押下します。
(CPUリセット後、プログラム実行します)

①

Teratermを起動し、シリアル接続設定をします。

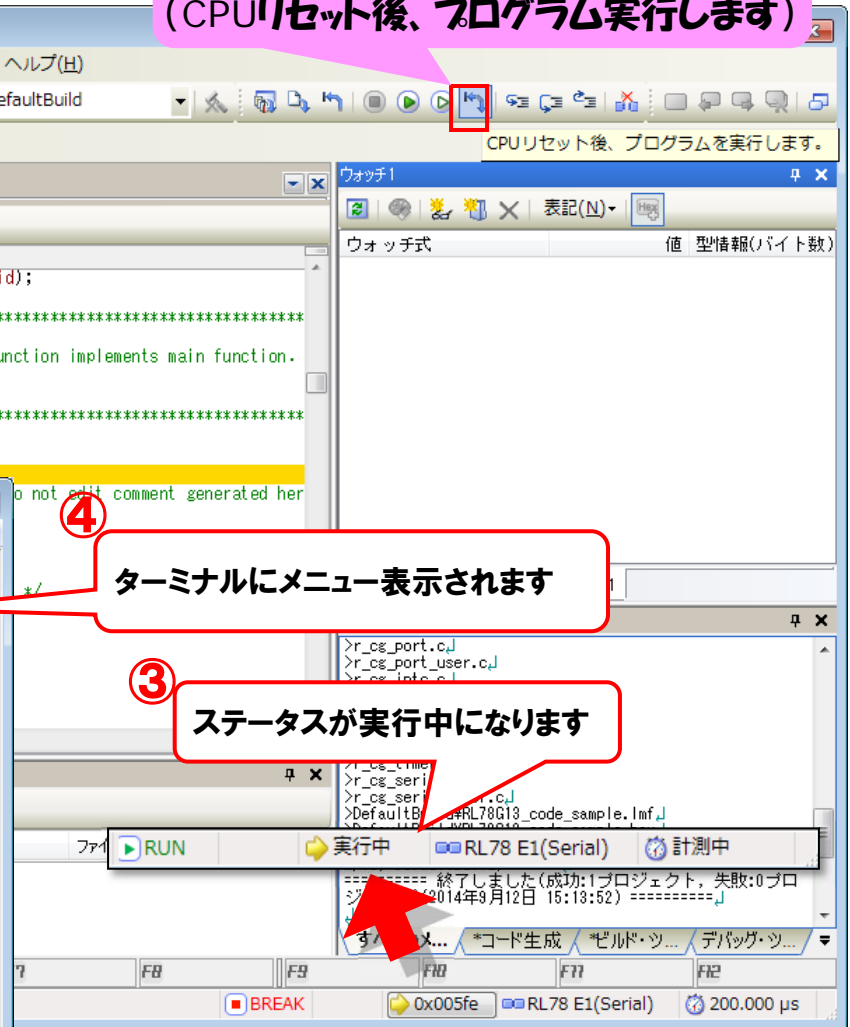


④

ターミナルにメニュー表示されます

③

ステータスが実行中になります



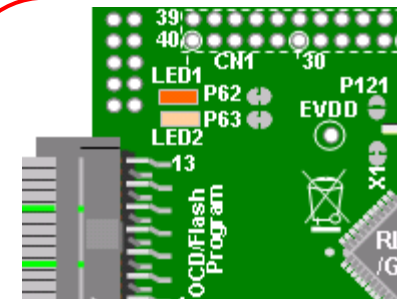
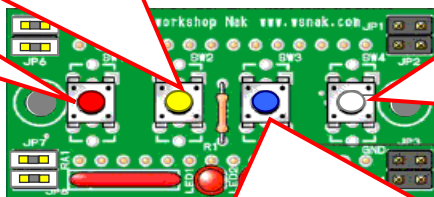
CS+ プログラム実行 (サンプルプログラム動作確認)



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。

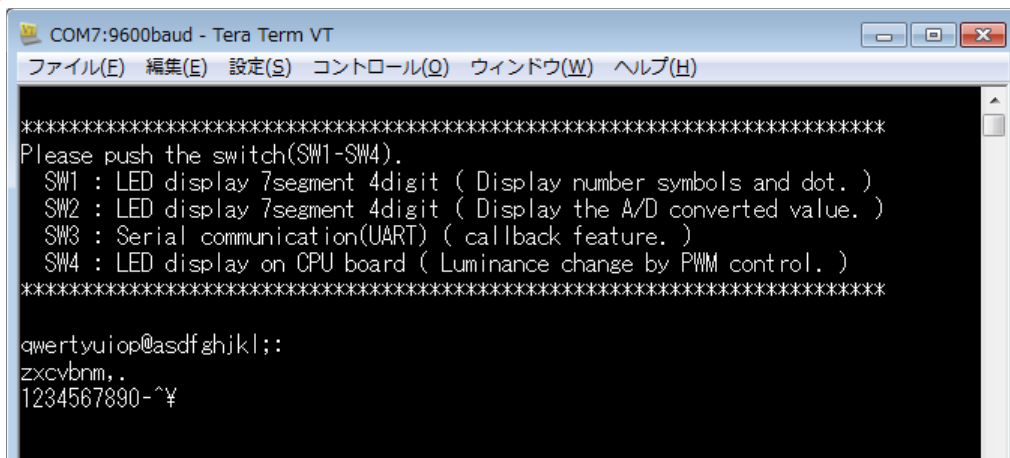
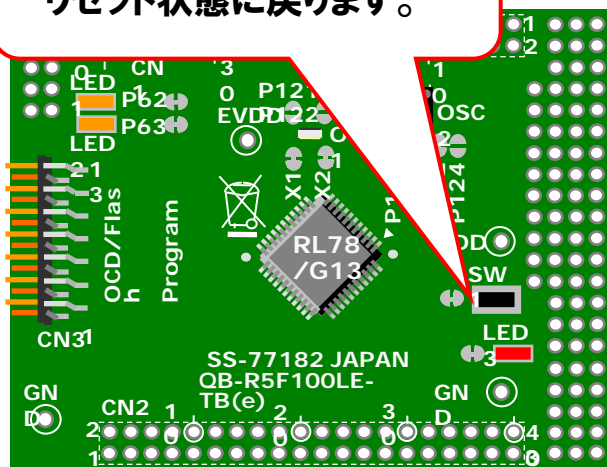


SW2押下:
ポテンションメータで設定した値の
A/D変換値 (0~1023) が7セグLED
に表示されます。



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

© 2014 Renesas System Design Co., Ltd.