RZ/T1 グループ
サンプルプログラムへのコード生成ツール適用ガイド

R01AN3860JJ0110
Rev.1.10
2018.04.11

要旨

本アプリケーションノートでは、マイコン内蔵周辺 I/O ドライバ自動生成ツール（以下、コード生成ツールと呼びます）で作成したマイコン周辺機能の制御プログラム（デバイス・ドライバ・プログラム）を RZ/T1 グループ 初期設定サンプルプログラムへ組み込む手順について説明します。

本サンプルプログラムでは、コンペアマッチタイマ（CMT）を周期カウント動作させ、コンペアマッチ割り込みにより LED を点灯 / 消灯させます。

対象デバイス

RZ/T1 グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

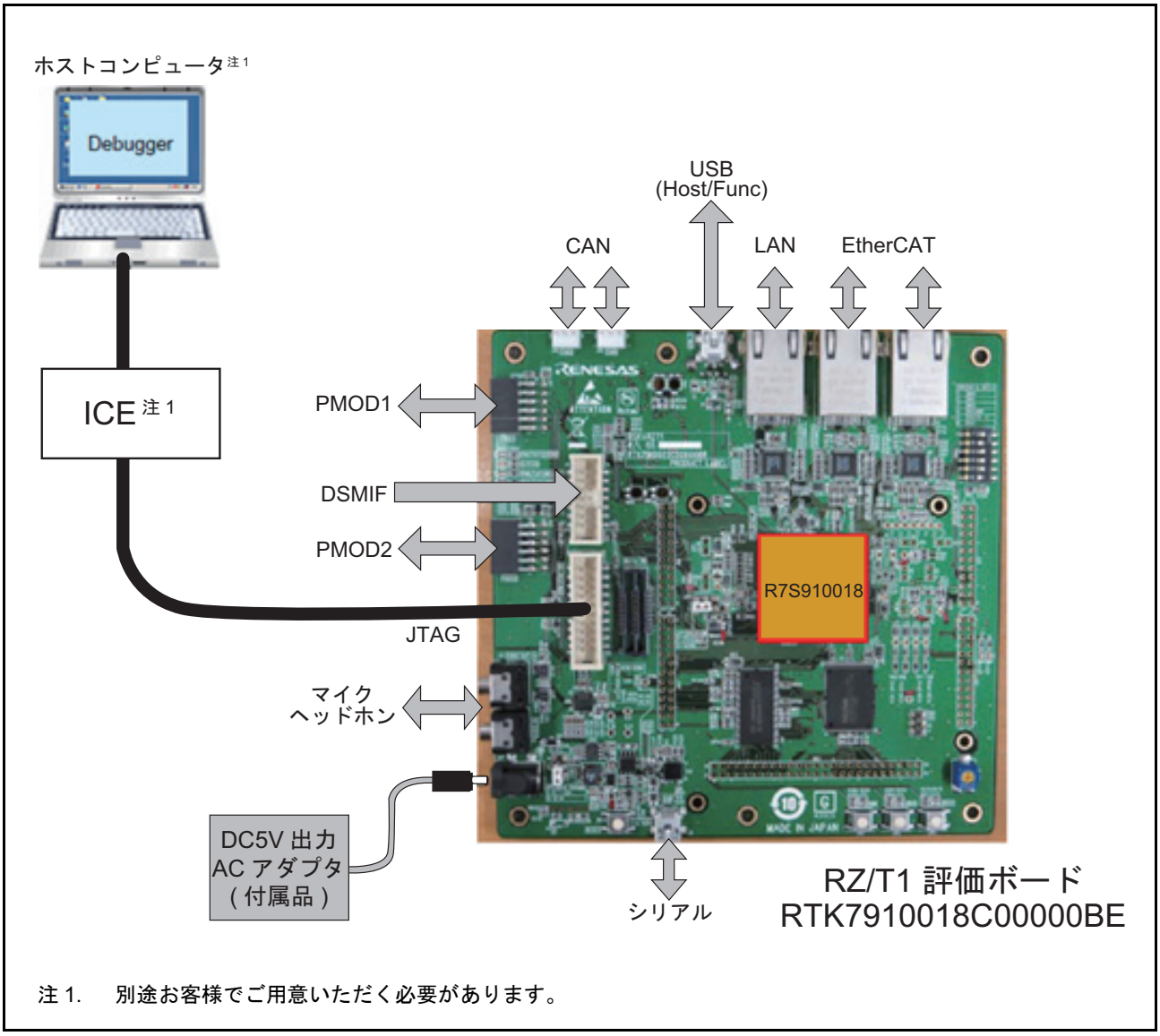
目　次

1.　仕様

表 1.1 に使用する周辺機能と用途を、図 1.1 に動作環境を示します。

表 1.1　使用する周辺機能と用途

周辺機能	用途
クロック発生回路（CPG）	CPU クロックおよび低速オンチップオシレータで使用
割り込みコントローラ（ICUA）	コンペアマッチ割り込み（CMI0）で使用
コンペアマッチタイマ（CMT）	コンペアマッチタイマの周期カウント動作で使用
バスステートコントローラ（BSC）	CS0、CS1 空間に NOR フラッシュメモリを接続、および、CS2、CS3 空間で SDRAM を接続するために使用
SPI マルチ I/O バスコントローラ（SPIBSC）	SPI マルチ I/O 空間にシリアルフラッシュメモリを接続するために使用
エラーコントロールモジュール（ECM）	ERROROUT#端子の初期化
汎用入出力ポート	LED の点灯および消灯のための端子制御に使用



注 1.　別途お客様でご用意いただく必要があります。

図 1.1　動作環境

2.　動作環境

本アプリケーションノートのサンプルプログラムは、下記の環境を想定しています。

表2.1　動作環境

項目	内容
使用マイコン	RZ/T1グループ
動作周波数	CPUCLK = 450MHz
動作電圧	3.3V
統合開発環境	IARシステムズ製 Embedded Workbench for ARM Version 7.80.2 ARM製 DS-5™ 5.25 RENESAS製 e²studio 5.2.0
コード生成ツール	RENESAS製 AP4 1.07 注．　e²studioは、同機能のAP4 1.04相当のコード生成プラグインが含まれています。
動作モード	SPIブートモード 16ビットバスブートモード
使用ボード	RZ/T1 評価ボード (RTK7910018C00000BE)
使用デバイス （ボード上で使用する機能）	• NORフラッシュメモリ（CS0、CS1空間に接続） メーカ名：Macronix International Co..、型名：MX29GL512FLT2I-10Q • SDRAM（CS2、CS3空間に接続） メーカ名：Integrated Silicon Solution Inc、型名：IS42S16320D-7TL • シリアルフラッシュメモリ メーカ名：Macronix International Co..、型名：MX25L51245G

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

- RZ/T1 グループ初期設定（R01AN2554JJ）
- RZ/T1 グループコンペアマッチタイマ（CMT）（R01AN2555JJ）

4.　周辺機能説明

動作モード、クロック発生回路（CPG）、コンペアマッチタイマ（CMT）、割り込みコントローラ（ICUA）、バスステートコントローラ（BSC）、SPI マルチ I/O コントローラ（SPIBSC）、エラーコントロールモジュール（ECM）、リセット、汎用入出力ポートについての基本的な内容は、RZ/T1 グループ・ユーザーズマニュアル ハードウェア編を参照してください。

5. ハードウェア説明

5.1 ハードウェア構成例

図 5.1 にハードウェア構成例を示します。

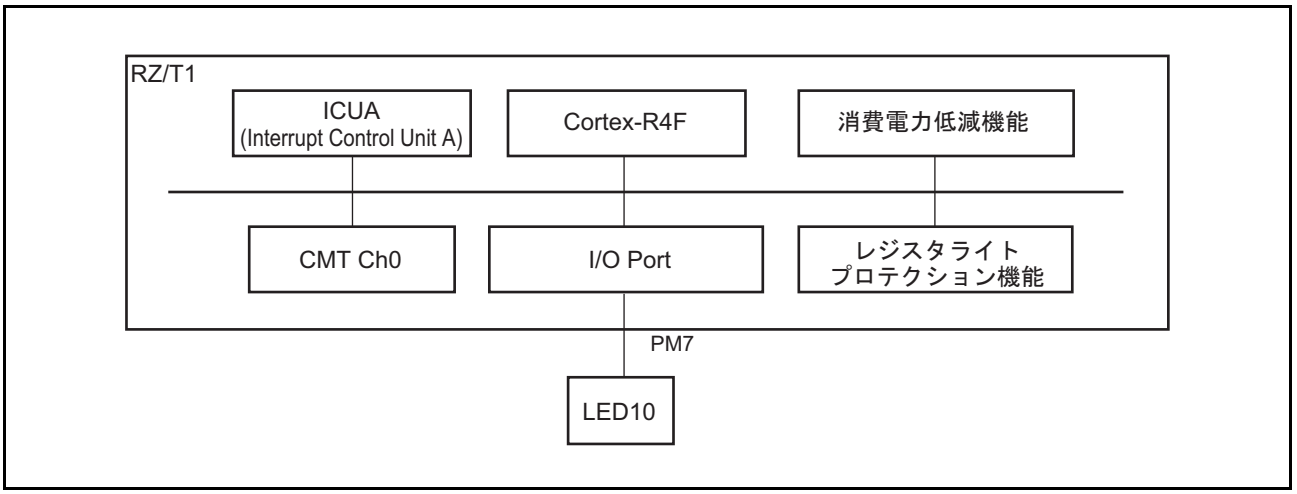


図 5.1　ハードウェア構成例

5.2　使用端子一覧

表 5.1 に使用端子と機能を示します。

表5.1　使用端子と機能

端子名	入出力	内容
A1～A25注1	出力	NORフラッシュメモリ、SDRAMへのアドレス信号出力
D0～D15注1	入出力	NORフラッシュメモリ、SDRAMのデータ信号入出力
CS0#注1	出力	CS0空間に接続されたNORフラッシュメモリへのデバイス選択信号出力
CS1#注1	出力	CS1空間に接続されたNORフラッシュメモリへのデバイス選択信号出力
CS2#注1	出力	CS2空間に接続されたSDRAMへのデバイス選択信号出力
CS3#注1	出力	CS3空間に接続されたSDRAMへのデバイス選択信号出力
RAS#注1	出力	SDRAMへのRAS#制御信号出力
CAS#注1	出力	SDRAMへのCAS#制御信号出力
RD/WR#注1	出力	SDRAMへのリード制御信号またはライト制御信号出力
CKE注1	出力	SDRAMへのCKイネーブル制御信号出力
RD#注1	出力	リード中を示すストローブ信号出力
BS#	出力	本サンプルプログラムでは、未使用
WE0#/DQMLL注1	出力	D15～D8に対するライトストローブ信号出力
WE1#/DQMLU注1	出力	D7～D0に対するライトストローブ信号出力
SPBSSL注1	出力	スレーブセレクト
SPBCLK注1	出力	クロック出力
SPBMO/SPBIO0注1	入出力	マスタ送出データ／データ0
SPBMI/SPBIO1注1	入出力	マスタ入力データ／データ1
SPBIO2注1	入出力	データ2
SPBIO3注1	入出力	データ3
MD0	入力	動作モードの選択 MD0 = “L”、MD1 = “L”、MD2 = “L”（SPIブートモード） MD0 = “L”、MD1 = “H”、MD2 = “L”（16ビットバスブートモード）
MD1	入力	
MD2	入力	
PM7注1	出力	LED10の点灯および消灯

注.　1.#は負論理（またはアクティブロー）を示す記号です。
注1.　各端子は、コード生成ツールで端子機能を設定。

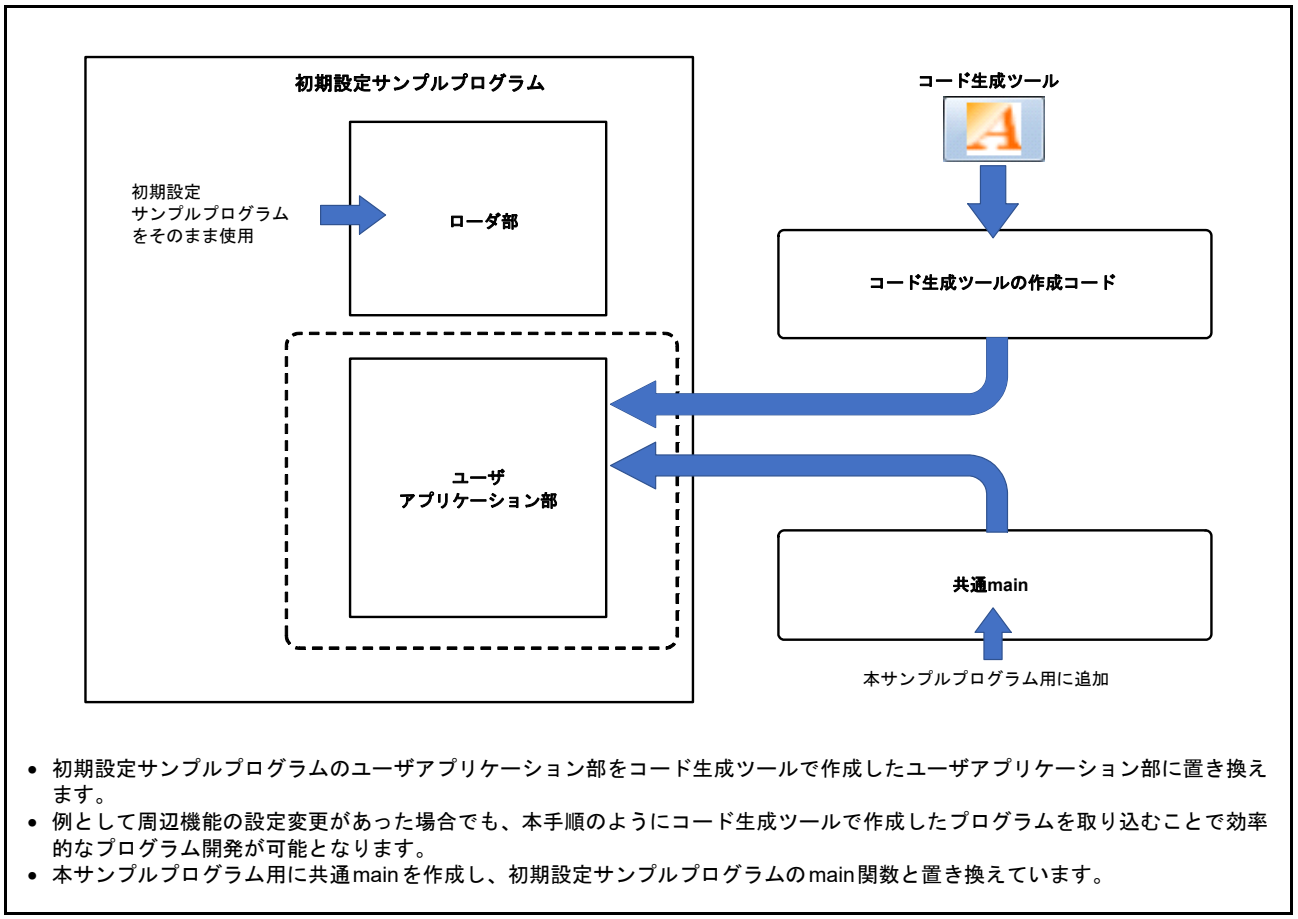
6.　ソフトウェア説明

6.1　動作概要

本サンプルプログラムは、RZ/T1 グループ初期設定サンプルプログラムをベースとし、ローダプログラム（ローダ）部は、初期設定サンプルプログラムのローダプログラムをそのまま利用し、ユーザアプリケーションプログラム（ユーザアプリケーション）部にはコード生成ツールで作成したコードと共通 main のコードを取り込んで使用します。実際の組み込み手順についても後述します。

RZ/T1 グループ初期設定サンプルプログラムの動作詳細については、RZ/T1 グループ初期設定アプリケーションノートを参照してください。

図 6.1 に本サンプルプログラムの構成概要を示します。



- 初期設定サンプルプログラムのユーザアプリケーション部をコード生成ツールで作成したユーザアプリケーション部に置き換えます。
- 例として周辺機能の設定変更があった場合でも、本手順のようにコード生成ツールで作成したプログラムを取り込むことで効率的なプログラム開発が可能となります。
- 本サンプルプログラム用に共通 main を作成し、初期設定サンプルプログラムの main 関数と置き換えています。

図 6.1　本サンプルプログラムの構成概要

図 6.2 にローダ部とユーザアプリケーション部の動作概要を示します。

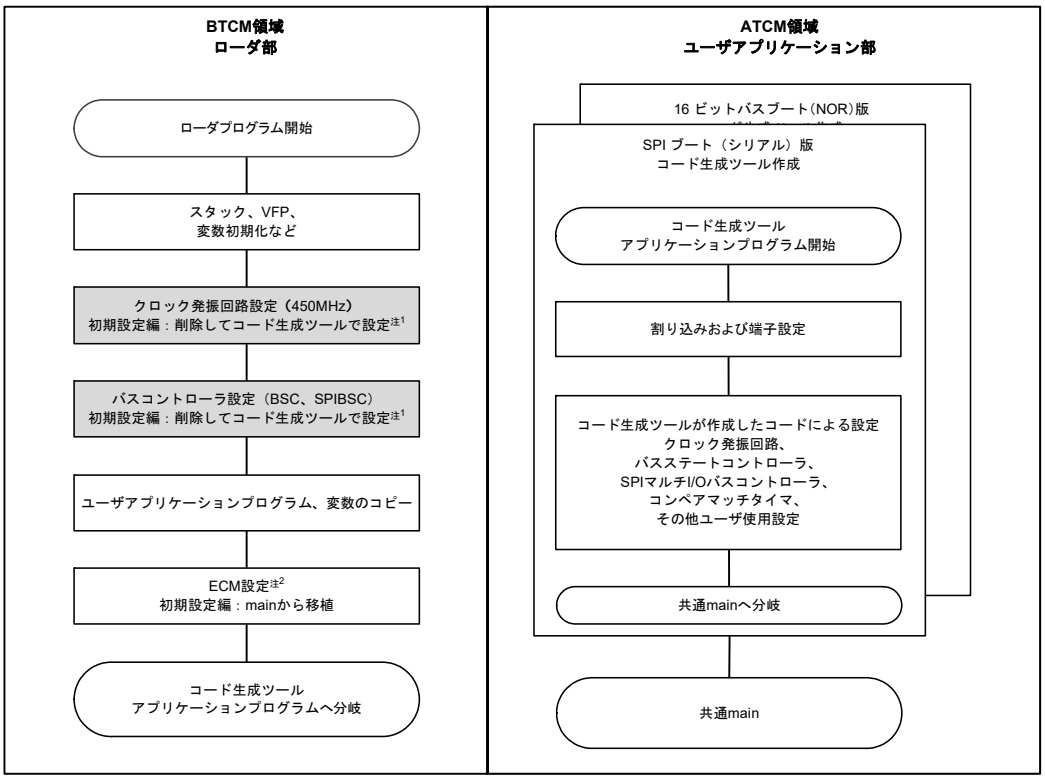
ローダ部では、ブート処理後にスタック設定やユーザアプリケーションプログラムのコピー、ユーザアプリケーションプログラムの先頭番地への分岐をします。

以下の図に示すようにクロック設定とバスコントローラ設定は、従来の初期設定サンプルプログラムのローダ部とコード生成されたユーザアプリケーション部のそれぞれで設定されています。このため本サンプルプログラムでは、ローダ部のクロック設定とバスコントローラ設定を無効化することで設定の競合を避けています。

ユーザアプリケーション部は、コード生成ツールで作成された割り込み、クロック、バス設定および各周辺機能の設定を行います。使用する動作モード（SPI ブート、16 ビットバスブート）毎に設定します。

また各動作モードによらない共通 main 関数として、コンペアマッチタイマ（CMT）を周期カウント動作させ、コンペアマッチ割り込みにより LED の点灯 / 消灯をする設定を行います。

本組み込み手順によって、初期設定サンプルプログラムのローダ部を利用してコード生成ツールでユーザが自由に設定したコードを取り込むことが可能となります。



注1.　ローダプログラムのクロック発振回路設定、バスコントローラ設定は、コード生成ツールとの設定が競合しないよう無効化しています。ただし、ローダプログラムでATCMへのコピー処理などを高速化したい場合は、ローダプログラム内で設定して構いません。

注2.　ECM設定では、ウエイト処理用にコンペアマッチタイマ（CMT）を使用し、処理後にCMTの初期化をしています。コード生成ツールとの競合を避けるため本サンプルプログラムでは、ECMの設定を初期設定サンプルプログラムのmain処理からローダプログラム側に移動しています。ECM設定の詳細については初期設定アプリケーションノートを参照してください。

図 6.2　　ブート処理後の動作概要

6.1.1　プロジェクト設定

本サンプルプログラムは以下の 3 種類のプロジェクトを含みます。

① RZ_T1_init_boot ― RZ_T1_init_nor_boot.eww：16ビットバスブート版
　　RZ_T1_init_serial_boot.eww：SPIブート版

② RZ_T1_init_ram ― RZ_T1_init.eww：RAM実行版

初期設定サンプルプログラムにコード生成ツールで作成したプログラムを組み込んだ際のフォルダ構成を表 6.1 に示します。本サンプルプログラムで組み込んだフォルダ、ファイルについては太字部で示します。太字部以外は初期設定サンプルプログラムと同じ構成です。

開発環境およびプロジェクト設定については、RZ/T1 グループ初期設定アプリケーションノートを参照してください。

表6.1　コード生成ツールで作成したコードを組み込んだ初期設定サンプルプログラムのフォルダ構成 (1/2)

主なフォルダ構成（EWARM）				
プロジェクトフォルダ	サブフォルダ			備考
RZ_T1_init_boot	inc			初期設定編のインクルードファイル格納フォルダ
	lib			EWARM用ライブラリフォルダ
	src	**cg_src_nor**	**cg_src**	**【コード生成ツールで作成したフォルダ】** **（16ビットバスブート版（NOR））** • **ソースコード出力フォルダ(cg_src)** • **コード生成ツールワークファイル（.cgp）** • **プロジェクト・コネクション・ファイル（.ipcf）**
			***.cgp**	
			***.ipcf**	
		cg_src_serial	**cg_src**	**【コード生成ツールで作成したフォルダ】** **（SPIブート版（Serial））** • **ソースコード出力フォルダ(cg_src)** • **コード生成ツールワークファイル（.cgp）** • **プロジェクト・コネクション・ファイル（.ipcf）**
			***.cgp**	
			***.ipcf**	
		common		初期設定編のサンプルプログラムフォルダ
		drv		ドライバフォルダ
		sample	**user_main.c**	初期設定編のmain用フォルダ **共通main** **(user_app_main)**
RZ_T1_init_ram	inc			RAM版初期設定編のインクルードファイル
	lib			EWARM用ライブラリフォルダ
	src	**cg_src_ram**	**cg_src**	**【コード生成ツール追加フォルダ】** **（RAM版）** • **ソースコード出力フォルダ(cg_src)** • **コード生成ツールワークファイル（.cgp）** • **プロジェクト・コネクション・ファイル（.ipcf）**
			***.cgp**	
			***.icpf**	
		common		初期設定編のサンプルプログラムフォルダ
		drv		ドライバフォルダ
		sample	**user_main.c**	RAM版初期設定編のmain用フォルダ **共通main** **(user_app_main)**

表6.2　　コード生成ツールで作成したコードを組み込んだ初期設定サンプルプログラムのフォルダ構成 (2/2)

主なフォルダ構成（e²studio）			備考
プロジェクトフォルダ	サブフォルダ		
RZ_T_nor_sample	.setting	**CodeGenerator**	**【コード生成ツールで作成したフォルダ】 （16ビットバスブート版（NOR））** • **コード生成ツールワークファイル（.cgp）**
	inc		初期設定編のインクルードファイル格納フォルダ
	src	**cg_src**	**【コード生成ツールで作成したフォルダ】 （16ビットバスブート版（NOR））** • **ソースコード出力フォルダ(cg_src)**
		common	初期設定編のサンプルプログラムフォルダ
		drv	ドライバフォルダ
		sample — **user_main.c**	初期設定編のmain用フォルダ **共通main (user_app_main)**
		iodefine.h[注1]	**Incフォルダから移動したe²studio用iodefine**
RZ_T_sflash_sample	.setting	**CodeGenerator**	**【コード生成ツールで作成したフォルダ】 （SPIブート版（Serial））** • **コード生成ツールワークファイル（.cgp）**
	inc		初期設定編のインクルードファイル格納フォルダ
	src	**cg_src**	**【コード生成ツールで作成したフォルダ】 （SPIブート版（Serial））** • **ソースコード出力フォルダ(cg_src)**
		common	初期設定編のサンプルプログラムフォルダ
		drv	ドライバフォルダ
		sample — **user_main.c**	初期設定編のmain用フォルダ **共通main (user_app_main)**
		iodefine.h[注1]	**Incフォルダから移動したe²studio用iodefine**
RZ_T_ram_sample	.setting	**CodeGenerator**	**【コード生成ツールで作成したフォルダ】 （RAM版）** • コード生成ツールワークファイル（.cgp）
	inc		RAM版初期設定編のインクルードファイル
	src	**cg_src**	**【コード生成ツールで作成したフォルダ】 （RAM版）** • ソースコード出力フォルダ(cg_src)
		common	**初期設定編のサンプルプログラムフォルダ**
		drv	**ドライバフォルダ**
		sample — **user_main.c**	RAM版初期設定編のmain用フォルダ **共通main (user_app_main)**
		iodefine.h[注1]	**Incフォルダから移動したe²studio用iodefine**

注1.　e²studio環境にコード生成ツールで作成したコードを組み込む際は、各プロジェクトのincフォルダにあるiodefine.hをプロジェクトフォルダの直下に移動してください。

6.1.2　使用準備

ご使用になられるプロジェクトによって、RZ/T1 評価ボード (RTK7910018C00000BE) 上にある SW4 の設定が異なります。**表 6.3** に SW4 の設定を示します。SW4 の各設定につきましては、RZ/T1 評価ボード RTK7910018C00000BE ユーザーズマニュアルに記載しています。詳細は、「**8.　参考ドキュメント**」を参照してください。

表6.3　SW4の設定

サンプルプログラム	SW4-1	SW4-2	SW4-3	SW4-4	SW4-5	SW4-6
16ビットバスブートモード版	ON	OFF	ON	ON	ON	OFF
SPI ブートモード版	ON	ON	ON	ON	ON	OFF
RAM実行版	上記いずれかの SW4 設定					

6.1.3　例外処理ベクタテーブル

RZ/T1 には 7 種類の例外処理（リセット、未定義命令、ソフトウェア割り込み、プリフェッチアボート、データアボート、IRQ、FIQ）があり、コード生成ツールを使用する場合 0000 0000h 番地から 34 バイトの領域（0000 0000h 番地～ 0000 0024h 番地）を使用します。例外処理ベクタテーブルには、各例外処理への分岐命令を記述します。

表 6.4 に本サンプルプログラムにおける例外処理ベクタテーブルの内容を示します。必要に応じて変更してください。

表6.4　例外処理ベクタテーブル

例外	ハンドラアドレス	備考
RESET例外	0000 0000h	自身へ分岐（暴走防止）
未定義命令例外	0000 0004h	自身へ分岐（ユーザで自由に定義）
ソフトウェア例外	0000 0008h	自身へ分岐（ユーザで自由に定義）
プリフェッチアボート例外	0000 000Ch	自身へ分岐（ユーザで自由に定義）
データアボート例外	0000 0010h	自身へ分岐（ユーザで自由に定義）
Reserved	0000 0014h	自身へ分岐（ユーザで自由に定義）
IRQ例外	0000 0018h	自身へ分岐（暴走防止）
FIQ例外	0000 001Ch	自身へ分岐（コード生成ツールで上書き）

注.　コード生成ツールが作成したコードにより、上記FIQ例外を含む0000 001Ch～0000 0024hまでをアプリケーションプログラム内で上書きするためご注意ください。

6.2　使用割り込み一覧

表 6.5 にサンプルプログラムで使用する割り込みを示します。

表6.5　サンプルプログラムで使用する割り込み

割り込み（要因ID）	優先度	処理概要
CMT0割り込み（CMI0）	15	コード生成ツールで指定したインターバル時間（100ms）によってコンペアマッチするたび、LED10が点灯／消灯を繰り返します。

6.3　コード生成ツール出力コードの組み込み手順

ここでは EWARM とコード生成ツールを使用した場合の手順例を示します（動作モードは SPI ブートモードです）。

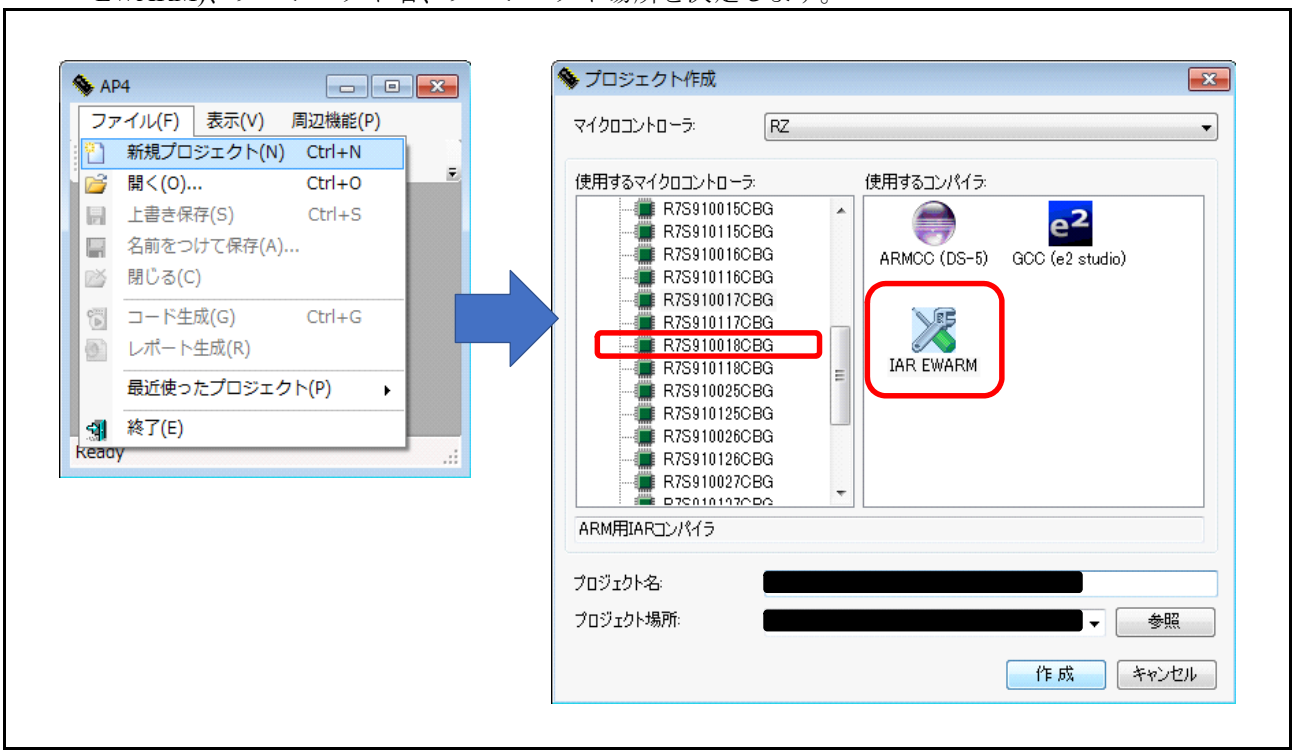
6.3.1　コード生成ツールを使用したコードの作成について

本サンプルでは、CMT0 を使用した周期カウント動作によるコンペアマッチ割り込みと LED10 の点灯 / 消灯を行うため PORTM7 の設定を行います。サンプル例を以下に示します。

(1) コード生成ツールを起動

(2) 新規プロジェクトを作成
新規プロジェクト作成する場合は、以下の設定が必要になります。

- マイクロコントローラ（R7S910018CBG を使用する場合）を選択、使用するコンパイラ (IAR EWARM)、プロジェクト名、プロジェクト場所を決定します。



- プロジェクト名とプロジェクト場所についての注意点
はじめにコード生成ツールを使用してプロジェクトを作成する場所は、ユーザの指定する任意の場所で構いませんが、本サンプルプログラムでは各動作モード毎にコード生成ツールの出力ファイルを格納する場所を分けています。
サンプル例)
16 ビットバスブートモード用フォルダ名：cg_src_nor
SPI ブートモード用フォルダ名：cg_src_serial
作成した各フォルダを初期設定サンプルプログラムへ組み込む方法については、「6.3.2 の（1）**コード生成ツールで作成した環境の移動（コピー）**」を参照ください。

(3) コンペアマッチタイマ（CMT）、ポートの設定

【コンペアマッチタイマ設定】

使用するコンペアマッチタイマ設定を下記のように設定してください。

使用チャネル：CMT0

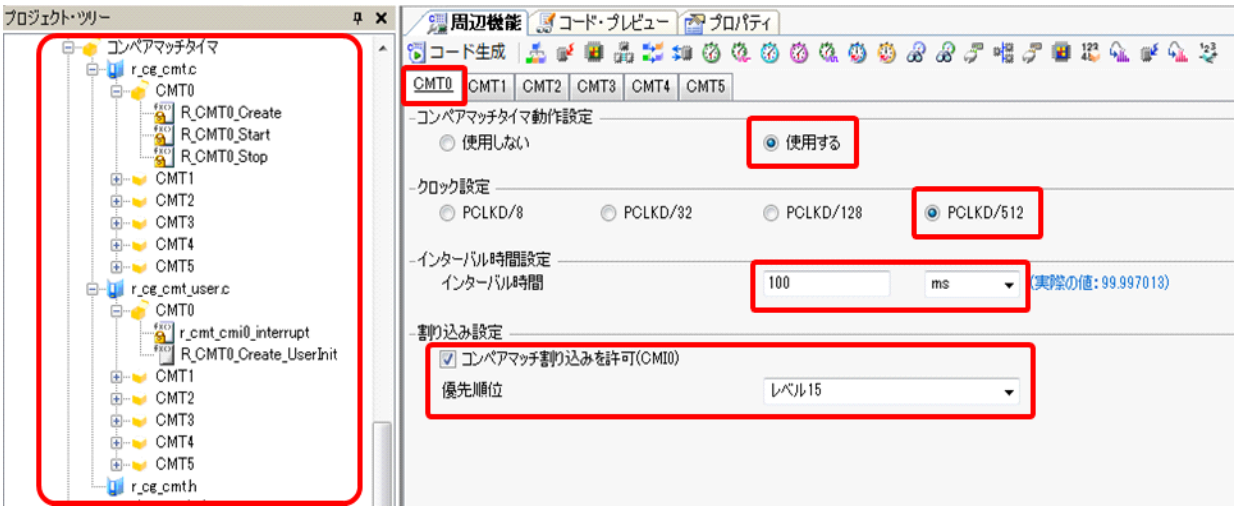
クロック設定：PCLKD/512

インターバル時間：100ms

コンペアマッチ割り込み許可 (CMI0)：使用

優先順位：レベル 15

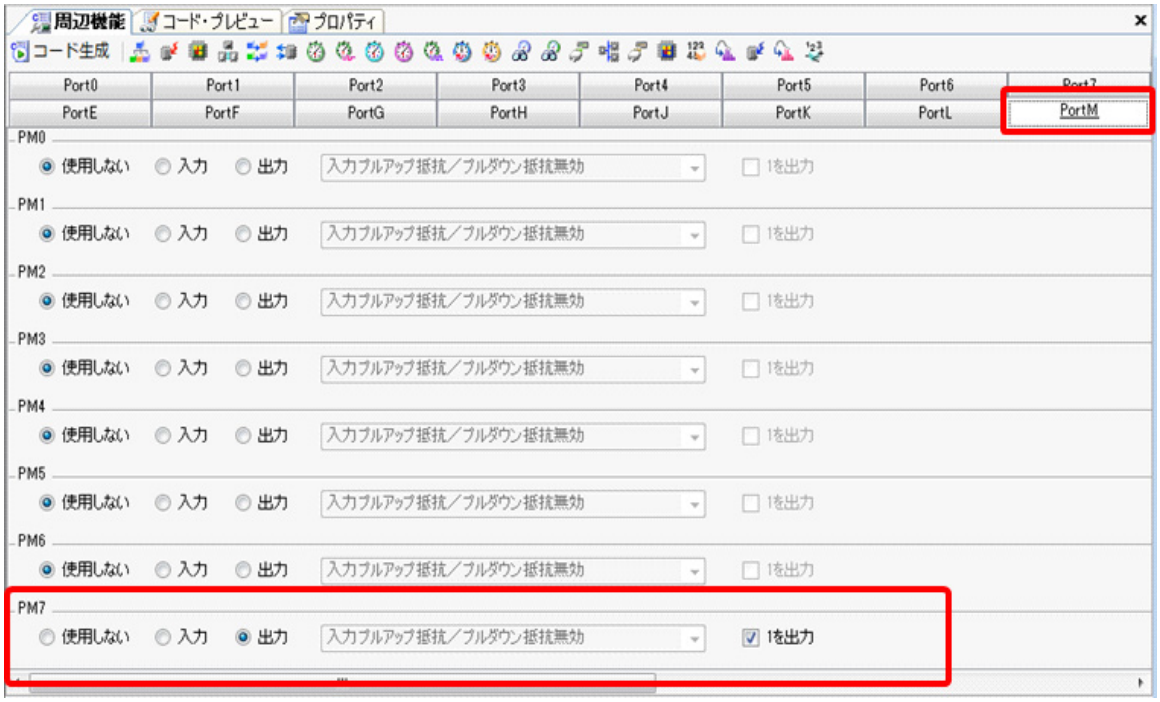
- 画面① CMT0 設定



- 各周辺機能で“使用する”を選択すると対応した機能の設定が出来ます。必要に応じて追加の設定をしてください。また、設定を行うと作成されるコードに対してプロジェクト・ツリー上で錠前マークが表示されます。

【ポート設定】
使用するポート設定を下記のように設定してください。
使用端子：PORTM7（LED10）
入出力：出力
出力設定：“1 を出力 ” に設定

- 画面② PORTM7 設定



- サンプル例としてLED10を点灯するためにPM7を出力ポートで“1を出力”に設定しています。

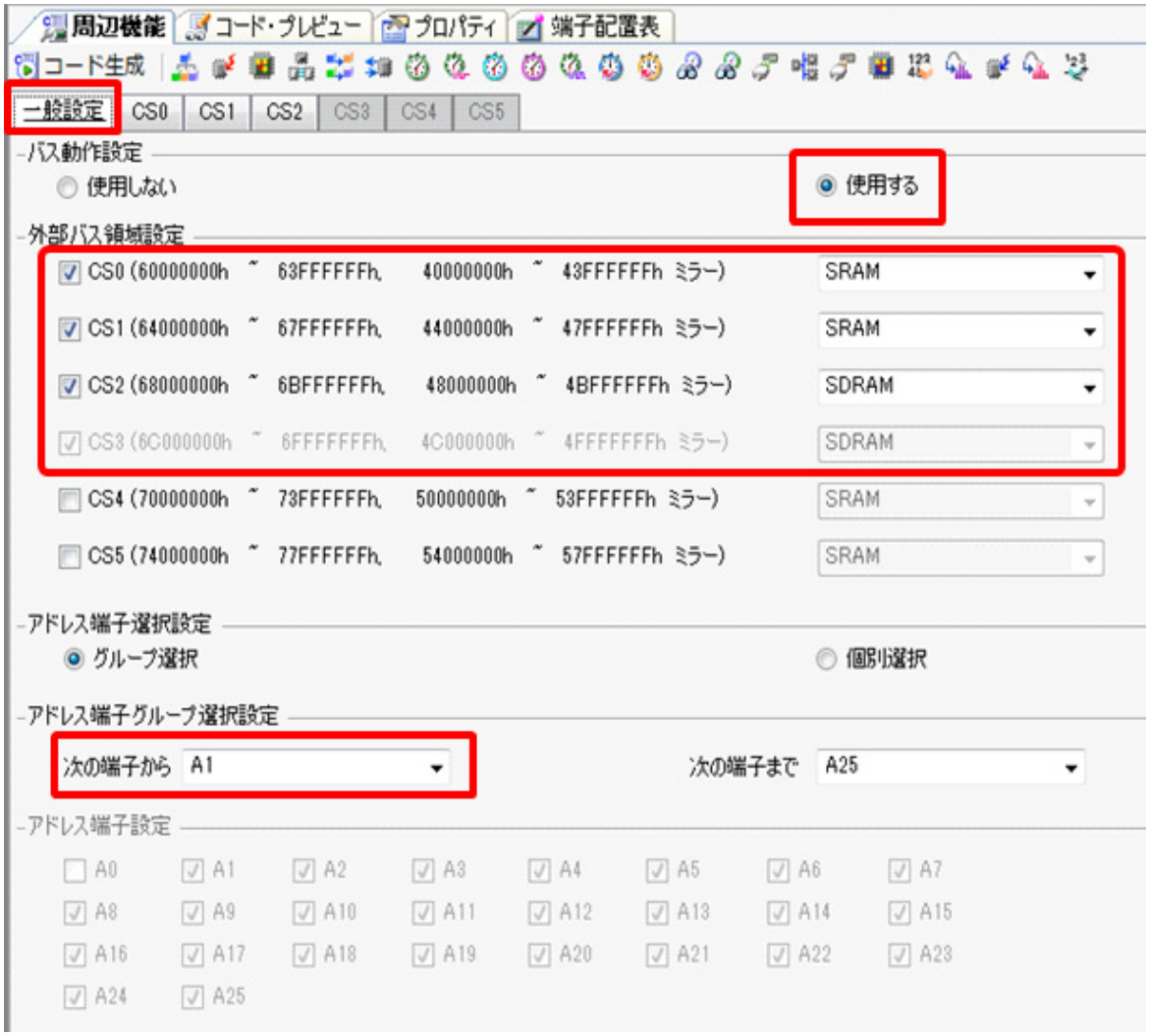
(4) クロック、バス機能の設定
本設定例は、初期設定サンプルプログラム（SPI ブートモード）と同じ設定となるよう例示しています。
【クロック発振回路設定】
下記のように設定してください。
ブートモード設定：SPI ブート
PLL1 回路設定：動作にチェック
低速オンチップオシレータ（LOCO）設定：動作にチェック
クロックソース：PLL1
CPU クロック（CPUCLK)：450（MHz）
外部バスクロック：75（MHz）

- 画面③クロック設定

注.　赤枠以外はコード生成ツールのデフォルト値となっています。

【バスステートコントローラ設定】
下記のように設定してください。
一般設定：バス動作設定を使用する
CS0 設定：SRAM
CS1 設定：SRAM
CS2 設定：SDRAM
CS3 設定：CS2 共通
アドレス端子選択設定：グループ選択
グループ選択設定：A1 ～ A25

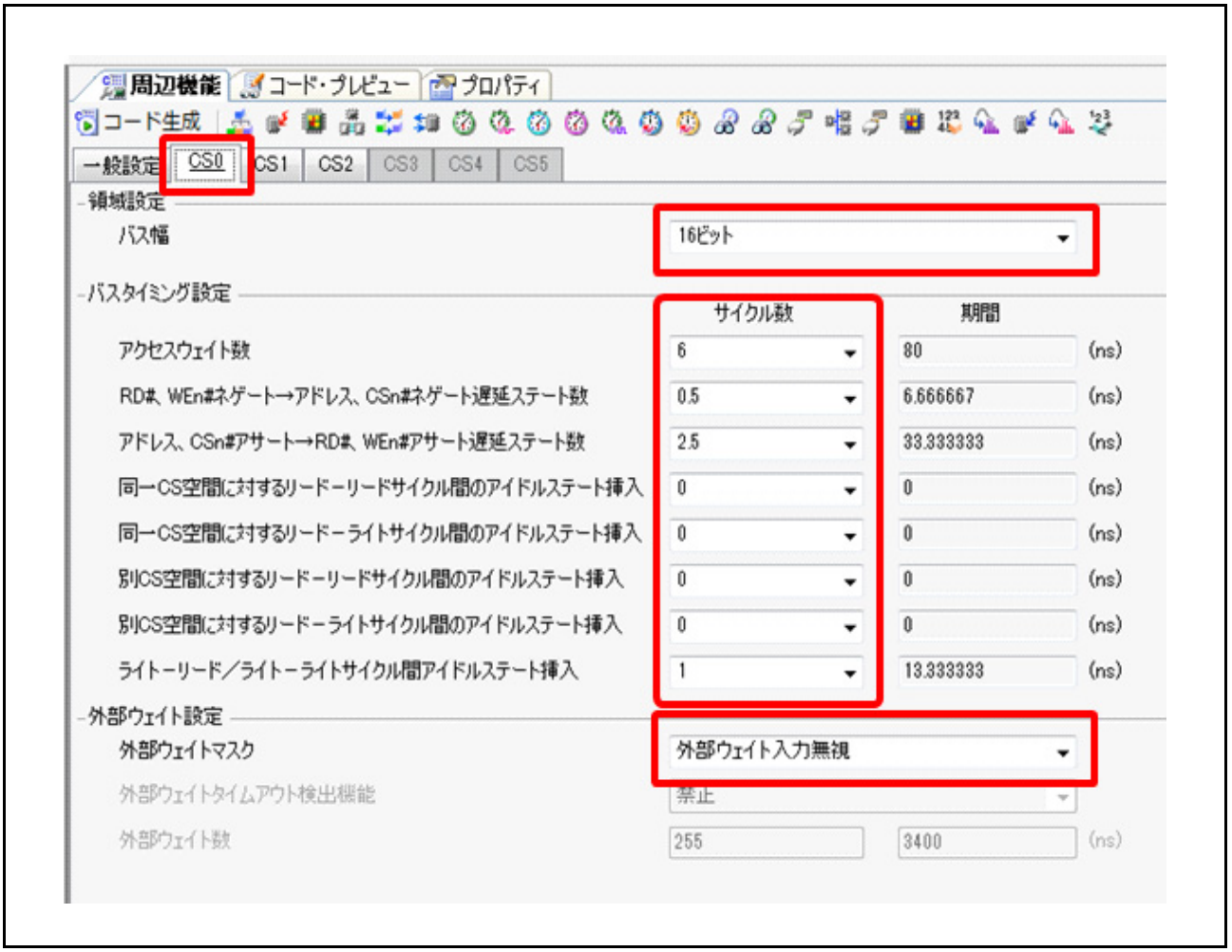
- 画面④バスステートコントローラ設定（一般設定）



注.　赤枠以外はコード生成ツールのデフォルト値となっています。

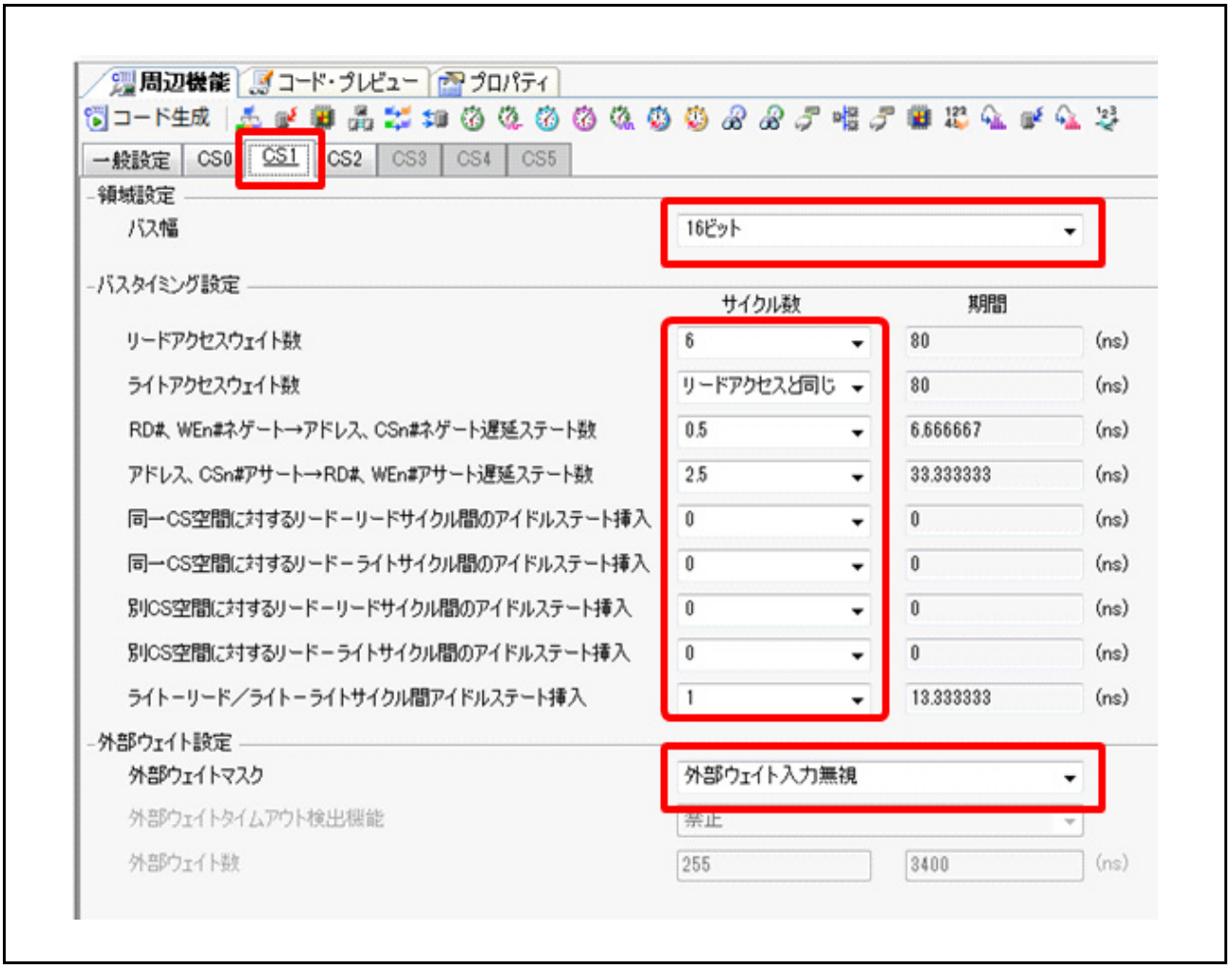
CS0：
領域設定：バス幅 16 ビット
バスタイミング設定：各サイクル数
外部ウエイト設定：外部ウエイトマスクの外部ウエイト入力無視

- 画面④バスステートコントローラ設定（CS0 設定）



CS1：
領域設定：バス幅 16 ビット
バスタイミング設定：各サイクル数
外部ウエイト設定：外部ウエイトマスクの外部ウエイト入力無視

- 画面④バスステートコントローラ設定（CS1 設定）



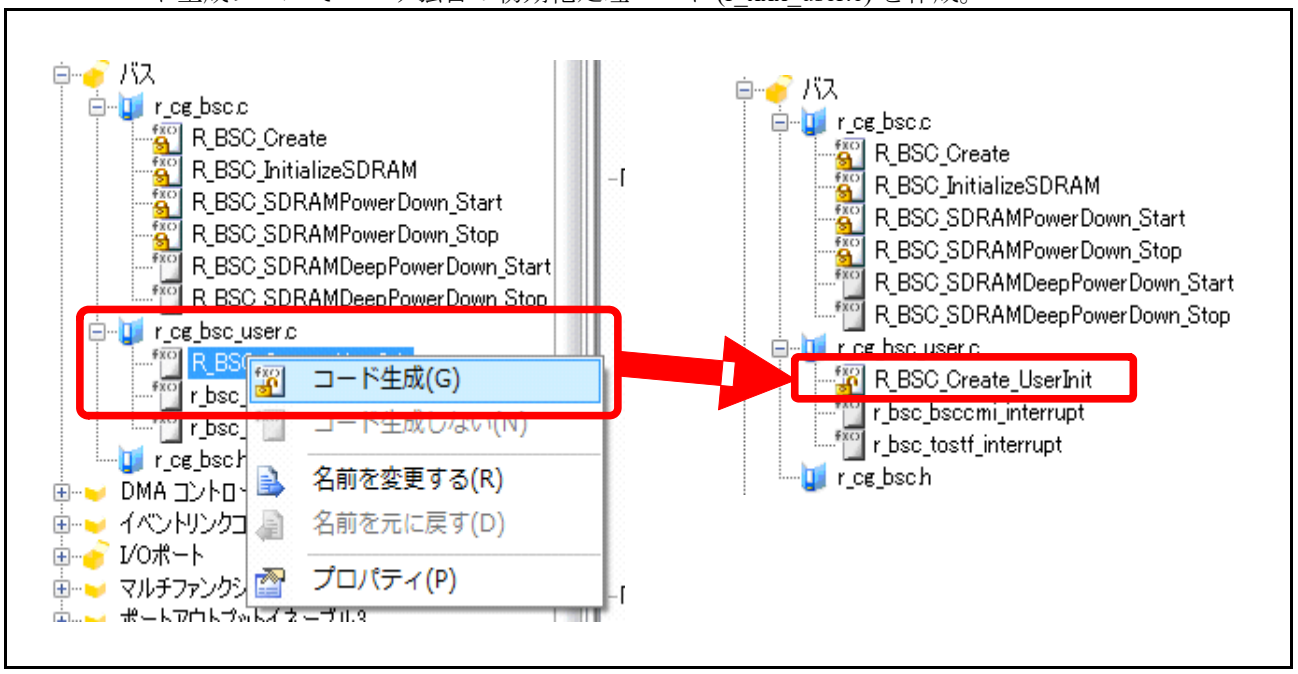
CS2：
領域設定：CS2 ／バス幅 16 ビット（デフォルト）
領域設定：CS3 ／バス幅 16 ビット（デフォルト）
種類：通常の SDRAM（デフォルト）
アドレスビット数：各エリアのロウ・カラムアドレスビット数
モード設定：各エリアのバーストリード／バーストライト
クロックセレクト：CKIO ／ 16
リフレッシュタイムコンスタントレジスタ値：36
コンペアマッチ割り込みを許可：チェック解除
バスタイミング設定：各サイクル数

- 画面④バスステートコントローラ設定（CS2 設定）

注.　赤枠以外はコード生成ツールのデフォルト値となっています。

初期設定サンプルプログラムと同じ設定となるようユーザ関数の出力設定を行います。

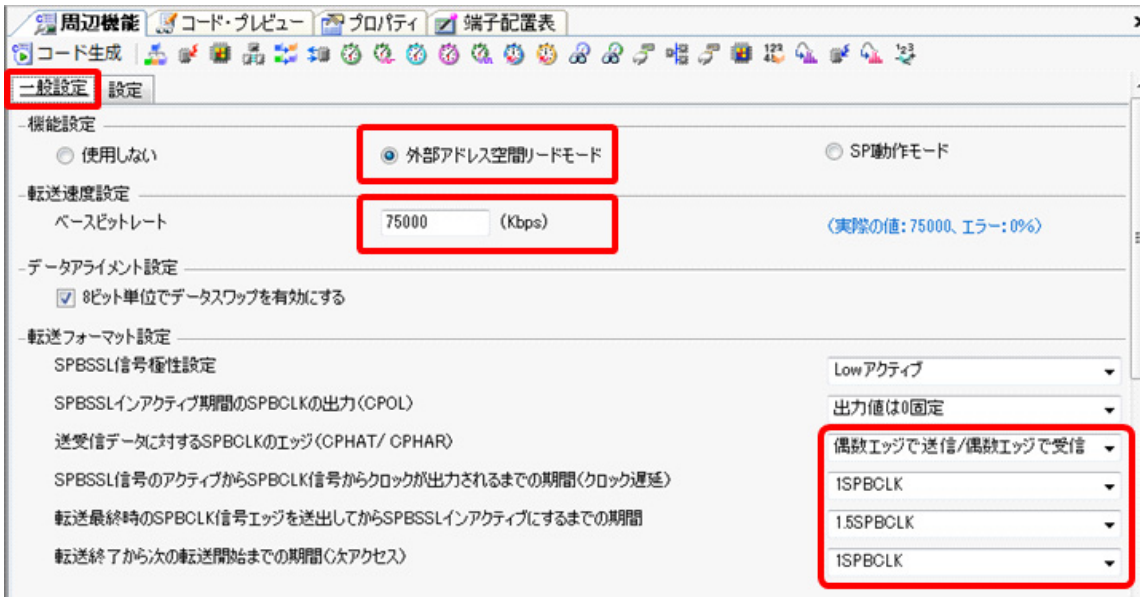
- コード生成ツールでユーザ独自の初期化処理コード (r_xxx_user.c) を作成。



コード生成ツールで錠前マークが付いていない API 関数はデフォルトではコード生成されません。ここでは R_BSC_Create_UserInit 関数を使用するための設定を行います。プロジェクト・ツリーのコード・プレビューで当該の API 関数を選択後に右クリックして「コード生成 (G)」を選択することで、実際のコード生成時に R_BSC_Create_UserInit 関数が r_cg_bsc_user.c ファイルと共に生成されるようになります。選択されると開いた錠前のマークが付きます。コード生成ツールの詳細な使用方法については、「AP4, Appliet3 ユーザーズマニュアル 共通操作編（R20UT3420JJ）」を参照ください。
作成されるファイル：r_cg_bsc_user.c
コードを生成した API 関数に設定する記述については、コード生成後の (6) で行います。

【SPI マルチ I/O バスコントローラ設定】
一般設定：機能設定を外部アドレス空間リードモード選択
転送速度設定：ベースビットレートを 75Mbps に設定
転送フォーマット設定：クロック遅延などを設定

- 画面⑤ SPI マルチ I/O バスコントローラ設定（一般設定）



注.　赤枠以外はコード生成ツールのデフォルト値となっています。

【注意点】
初期設定サンプルプログラムでは、シリアルフラシュを高速にするために一度 SPIBSC を SPI 動作モードに設定し、シリアルフラッシュを Quad I/O モードに設定します。その後外部アドレス空間リードモードに再設定をしています。
一方、本手順では、SPIBSC は外部アドレス空間リードモードで、シリアルフラッシュは Single I/O モードの前提で設定をしています。使用するコマンドは FAST READ4B（0x0C）コマンドです。

設定：
その他各種設定は、下図赤枠内を設定

- 画面⑤ SPI マルチ I/O バスコントローラ設定（設定）

注.　赤枠以外はコード生成ツールのデフォルト値となっています。

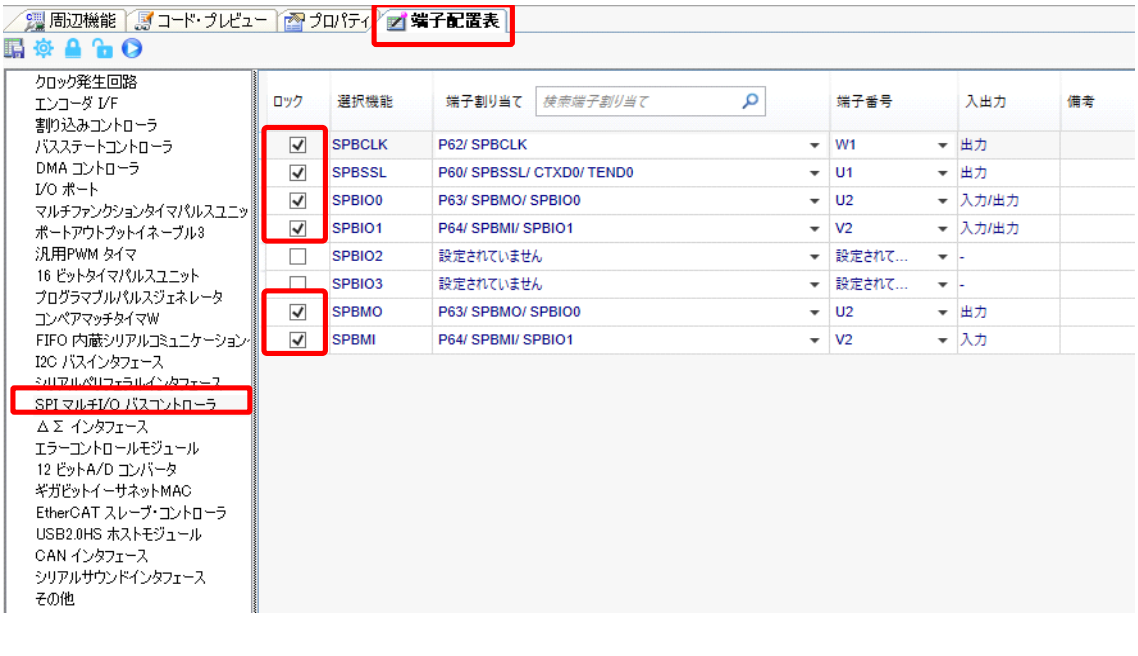
（5）端子機能設定

兼用機能として実際に使用する端子を設定してください。

サンプル例）端子配置表から端子機能の各設定画像と説明（1/3）

サンプル例）端子配置表から端子機能の各設定画像と説明 (2/3)

サンプル例）端子配置表から端子機能の各設定画像と説明 (3/3)



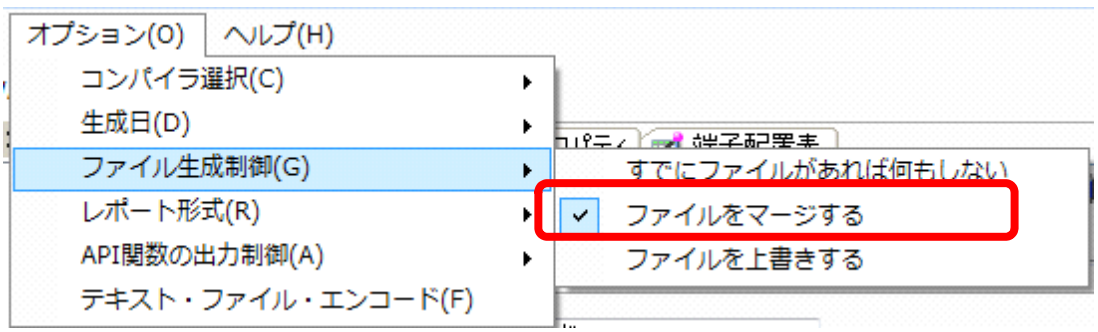
- ツールバーにある表示から端子配置表を選択、端子機能にあるバスステートコントローラを選択し、BS# 端子（PORT41）を設定してください。また、PORT10 の高駆動出力設定については、コード生成後の (6) で行います。

注 .　兼用端子機能のデフォルト設定はリセット解除後の設定です。実際に使用する端子が選択されているか確認してください。ロック選択していない端子は、別の設定で端子を使用した場合、変更される場合があります。不要な競合を避けるために確認後は、使用する端子をロック選択 (推奨) してください。また競合エラー等ないことを確認してください。

(6) コード生成とユーザ定義コードの編集

(1) ～ (5) までの設定が完了後、コード生成を行います。
コード生成するとコード生成ツールのプロジェクトを作成した cg_rc_serial フォルダ直下に cg_src フォルダとそのフォルダ内にソースコードとヘッダファイルが作成されます。

- コード生成したコードの編集について
  コード生成ツールでは、コード生成を行う度にコードが上書きされます。ユーザが書き換えたコードを保護するためにはファイル生成制御設定を行い、マージ用コメント行にコードを記述する必要があります。ファイル生成制御の設定は、以下のように［ファイルをマージ］がデフォルトで選択してあります。



マージ用コメント行は、以下のようにコード生成ツールが出力した各コード内に記述されています。
マージ用コメント行の間にある記述は、再度コード生成をしても上書きされずに既存ファイルとのマージを行うため、ユーザ・コードの保護が可能です。

```
/* Start user code. Do not edit comment generated here */
                                                    ←この間に記述
/* End user code. Do not edit comment generated here */
```

注.　マージ用コメントを編集または移動しないでください。編集または移動した場合、マージが正しく行われません。

【バスステートコントローラの API 関数設定】
(4) で作成したファイルを編集して以下のコードを記述します。
対象ファイル：r_cg_bsc_user.c
対象 API 関数：R_BSC_Create_UserInit

- PORT10 の高駆動出力設定
  バス用クロック（CKIO）に端子の機能を設定
- CS2 空間の SDRAM 用 WCR 設定（BSC_CS2WCR_1）
  e2studio 版（AP4 1.04 相当）を使用される場合は、ここで追加の設定を行う必要があります。
- SDRAM 初期化の API 関数設定（R_BSC_InitializeSDRAM）
  SDRAM 初期化の API 関数は、ユーザ独自の初期化処理コード内で関数コール

対象 API 関数内のマージ用コメント行へ以下のコードを記述します。

```
/* Start user code. Do not edit comment generated here */
/* Set PORT1 as high-drive output setting for connecting SDRAM */
PORT1.DSCR.WORD = 1;                                    ←PORT10の高駆動出力設定
/* Set wait control register of CS2 space */
BSC.CS2WCR.CS2WCR_1.LONG = 0x00000400;                  ←CS2空間のSDRAM用WCR設定（e2studio版のみ必要）
R_BSC_InitializeSDRAM();                                ←SDRAM初期化のAPI関数設定
/* End user code. Do not edit comment generated here */
```

【コンペアマッチタイマの API 関数設定】
(3) で作成したファイルを編集して割り込み処理に以下のコードを記述します。
対象ファイル：r_cg_cmt_user.c
対象 API 関数：r_cmt_cmi0_interrupt

- 周期カウント動作によるコンペアマッチ割り込み処理に LED10 点灯 / 消灯のコードを記述

対象 API 関数内のマージ用コメント行へ以下のコードを記述します。

```
/* Start user code. Do not edit comment generated here */
/* Toggle the PM7 output level(LED10) */
PORTM.PODR.BIT.B7 ^= 1;                                 ←割り込み処理（LED10点灯/消灯）
/* End user code. Do not edit comment generated here */
```

【ユーザアプリケーションプログラムの API 関数設定】
コード生成を行うと cg_src フォルダ下にコード生成ツールの main 処理がある r_cg_main.c ファイルが作成されます。ここでは、SPI ブートモード版の cg_src フォルダにある r_cg_main.c ファイルを編集します。
r_cg_main.c ファイルのユーザアプリケーション main 処理では、R_MAIN_UserInit 関数をコールしています。本サンプルプログラムでは、この関数内に 16 ビットバスブート版、SPI ブート（シリアル）版で共通となる main 処理の関数コールを記述します。
対象ファイル：r_cg_main.c
対象 API 関数：R_MAIN_UserInit
追加する共通の main 関数名：user_app_main
対象 API 関数内のマージ用コメント行へ以下のコードを記述します。

```
void R_MAIN_UserInit(void)
{
    /* Start user code. Do not edit comment generated here */
    user_app_main();                                          ←追加する共通のmain関数
    /* End user code. Do not edit comment generated here */
}
```

注 .　共通 main となる関数名は、ユーザ任意の名称で構いませんが、「6.3.2 の（4） 共通 main ファイルの作成」で作成する共通 main の関数名と同じ名称にしてください。

6.3.2　RZ/T1 グループ 初期設定サンプルプログラムへの取り込み

コード生成ツールで作成した生成コードを EWARM 版の初期設定サンプルプログラム環境に組み込みます。

サンプル例として「6.3.1　コード生成ツールを使用したコードの作成について」で作成したコードを使用した組み込みを以下に示します。

備考：以降の説明では、初期設定サンプルプログラム Rev1.30 をもとに行数などの説明をしています。組み込みをされる場合は、その際の最新の初期設定サンプルプログラムを利用してください。

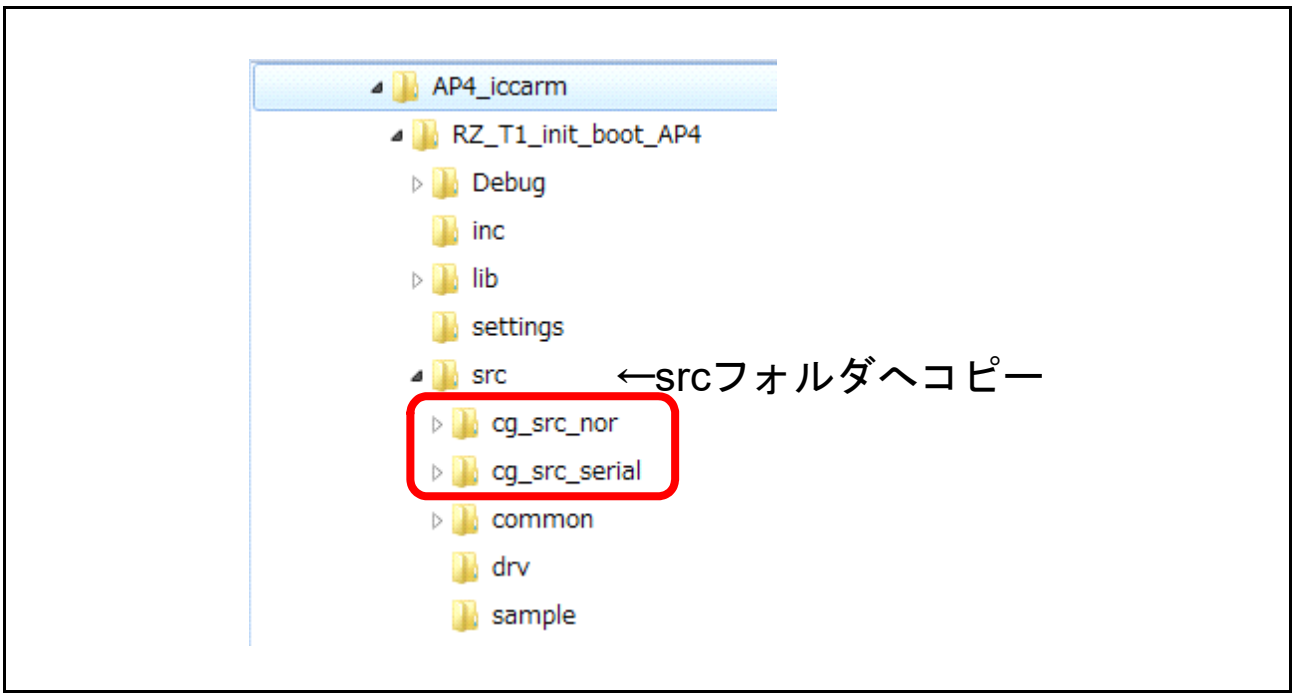
(1) コード生成ツールで作成した環境の移動（コピー）

初期設定サンプルプログラムのプロジェクトファイル内にあるソースファイル格納フォルダ (src) に「6.3.1 の（2）　新規プロジェクトを作成」で作成したコード生成ツールのプロジェクトフォルダごと移動（コピー）します。

サンプル例)

16 ビットバスブートモード用フォルダ名：cg_src_nor

SPI ブートモード用フォルダ名：cg_src_serial



注．　コード生成ツールが出力したコードを格納するフォルダ名が（cg_src）で固定のため、16 ビットバスブートモード版 (cg_src_nor) と SPI ブートモード版 (cg_src_serial) をそれぞれ別々にサンプルプログラムへ取り込めるように異なるプロジェクト名で作成してください。
コード生成されたそれぞれのプロジェクトフォルダ内には、(cg_src) フォルダ とワークスペースファイル (.cgp) が格納されています。

【補足事項】e²studio 環境にコード生成ツールで作成したコードを組み込む際は、各プロジェクトの inc フォルダにある iodefine.h をプロジェクトフォルダの直下に移動してください。詳細は、「6.1.1　プロジェクト設定」の表 6.2　コード生成ツールで作成したコードを組み込んだ初期設定サンプルプログラムのフォルダ構成 (2/2) を参照ください。

(2) 初期設定サンプルプログラムにあるローダプログラムの編集
初期設定サンプルプログラムで使用されている、クロック発振回路設定、バスステートコントローラ設定、SPI マルチ I/O バスコントローラ設定は、「6.3.1 の（4） **クロック、バス機能の設定**」からコード生成ツールで作成した設定に置き換えて使用します。そのため、本サンプルプログラムでは、ローダプログラムで指定しているこれらの設定を無効化（コメントアウト）します。

- 初期設定サンプルプログラムにあるプロジェクトフォルダ (RZ_T1_init_boot) 内、ソースファイル格納フォルダ (src) の common フォルダ以下にある、loader_init2.c ファイルを編集します。
  編集箇所は、ローダプログラム（loader_init2 関数）に 2 箇所あります。(111 行目、119 行目)

```
    /* Set CPU clock and LOCO clock */
//    cpg_init();                              ←クロック発振回路の初期化設定をコメントアウト注1
    ：中略
    /* Initialize the bus settings */
//    bus_init();                              ←バスコントローラ(BSC, SPIBSC)の初期化設定をコメントアウト注2
```

注 1. **ローダプログラムでクロック、バスの高速化を図りたい場合は、コメントアウトせずに各設定関数をコールすることも可能です。ただし、本サンプルプログラムでは以降のコード生成ツールによる初期化処理でクロック、バスの再設定がされるため注意が必要です。詳細については、「9. の (1)　bus_init() 関数を使用した場合の注意事項」を参照してください。**

注 2. **bus_init() 関数では、シリアルフラッシュメモリの設定を Single I/O モードから Quad I/O モードに変更します。一方、本手順でコード生成した場合、シリアルフラッシュメモリの設定は Signle I/O モードを前提としたリードコマンド（FAST READ4B) を用いる設定です。ローダプログラムでバス設定を行う目的で bus_init() 関数を使用する場合は Quad I/O モードに設定をするなど注意が必要となります。詳細は、「9.　(1) bus_init() 関数を使用した場合の注意事項」を参照してください。**

(3) エラーコントロールモジュール (ECM) の初期設定関数をローダプログラムへ移動
初期設定サンプルプログラムでは、main 処理中に ECM の設定を行っています。本プログラムでは、ユーザが作成する共通 main と初期設定サンプルプログラムの main を置き換えます。
そのため初期設定で実施している ERROROUT 端子の初期化処理を行う ECM の設定（ecm_init）は、ローダプログラムへ移動します。

- init_main.c から loader_init2.c へ ecm_init 関数を移植します。(DS-5 の場合は、cpu_init.c)
  init_main.c ファイルは、初期設定サンプルプログラムのプロジェクトファイル内にあるソースファイル格納フォルダ (src) の sample フォルダ以下にあります。
  ECM の拡張疑似エラー 35 の設定は不要なため削除します。
  同様に、loader_init2.c 内の ECM 拡張疑似エラー処理も削除します。

init_main.c ファイルの以下行を loader_init2.c ファイルへコピーします。

コピー元：init_main.c ファイル

- ①関数定義 86 行目：

```
void ecm_init(void);
```

- ② main 関数内にある ecm_init 関数コール 117, 118 行目：

```
    /* Initialize the ECM function */
    ecm_init();
```

- ③ ecm_init 関数 177 行目～ 202 行目

```
/***********************************************************************************
* Function Name: ecm_init
～中略～
***********************************************************************************/
void ecm_init(void)
{
～中略～
}
/***********************************************************************************
 End of function ecm_init
***********************************************************************************/
```

コピー先：loader_init2.c ファイル

- Private variables and functions へコピーした①関数定義を 98 行目へ追加

```
/***********************************************************************************
Private variables and functions
***********************************************************************************/
void loader_init2(void);
void reset_check(void);
void cpg_init(void);
void copy_to_atcm(void);
void copy_4byte(uint32_t *src, uint32_t *dst, uint32_t bytesize);
void ecm_init(void);                                          ←ecm_initの関数宣言追加
```

- loader_init2 処理内の set_low_vec 関数コール～ _main 関数コール間へ
  ② ecm_init 関数コールをコピーして 129 行目へ追加

```
void loader_init2(void)
{
～中略～
    /* Set RZ/T1 to Low-vector (SCTLR.V = 0) */
    set_low_vec();
    /* Wait for ensuring the wait setting of ATCM */
    asm("dmb");     /* Ensuring Context-changing */     ←ATCMのウエイト設定待ちのため追加注1
    /* Initialize the ECM function */
    ecm_init();                                         ←ecm_init関数コール追加
    /* Jump to _main() */
    _main();
}
```

注 1.　ecm_init 関数内の処理は、ATCM に配置されています。このため、直前で実行している R_ATCM_WaitSet() による ATCM のウエイト設定が確実に反映されるよう DMB 命令を追加しています。

- ファイルの行末に③ ecm_init 関数をコピーして追加（354 行目～ 359 行目の拡張疑似エラー設定を削除）

```
/***********************************************************************************
 End of function copy_4byte
***********************************************************************************/
/***********************************************************************************
* Function Name: ecm_init
～中略～
***********************************************************************************/
void ecm_init(void)                                          ←ecm_init関数を追加
{
    volatile uint8_t result;                                 ←使用しないresult定義を削除

    /* Initialize ECM function  */
    R_ECM_Init();

    /*****************************************/      ←以下からresultまでを削除
    /*      Set extended pseudo error 35       */
    /*****************************************/

    /* Enables internal reset configuration */
    result = R_ECM_Write_Reg32(ECM_COMMON, &(ECM.ECMIRCFG1.LONG), 0x00000004);

}
/***********************************************************************************
 End of function ecm_init
***********************************************************************************/
/* End of File */
```

- loader_init2 処理内の reset_check 関数の ECM 拡張疑似エラー処理を削除

```
/***********************************************************************************
* Function Name : reset_check
* Description   : Check the reset source and execute the each sequence.
*                 When error source number 35 is generated, set P77 pin to High.
* Arguments     : none
* Return Value : none
***********************************************************************************/
void reset_check(void)
{
    volatile uint8_t result;                    ←使用しないresult定義を削除
    volatile uint32_t dummy;                    ←使用しないdummy定義を削除

    /* Check the reset status flag and execute the each sequence */
    if (RST_SOURCE_ECM == SYSTEM.RSTSR0.LONG) // ECM reset is generated
    {
        /* Clear reset status flag */
        R_RST_WriteEnable();              // Enable writing to the RSTSR0 register
        SYSTEM.RSTSR0.LONG = 0x00000000;  // Clear reset factor flag
        R_RST_WriteDisable();             // Disable writing to the RSTSR0 register

        /* Check the ECM error source */       ←以下のif文とif文内を削除
        if (1 == ECMM.ECMMESSTR1.BIT.ECMMSSE102) // Error source number 35 is generated
        {
        ～中略～
        }

    ～以下略～

/***********************************************************************************
 End of function reset_check
***********************************************************************************/
}
```

（4）共通 main ファイルの作成

コード生成ツールが作成したユーザアプリケーションプログラムは、16 ビットバスブートモード版、SPI ブートモード版で異なるファイルを使用しています。

16 ビットバスブートモード版、SPI ブートモード版の共通となる main 処理を共通 main として作成します。

本サンプルプログラムでは、共通の main 関数のため、init_main.c ファイルと同じ初期設定サンプルプログラムのプロジェクトファイル内にあるソースファイル格納フォルダ (src) の sample フォルダに作成します。

ファイル名称については、ユーザ任意のファイル名称で構いません。本サンプルプログラムでは、作成する関数名をコード生成ツールが作成したユーザアプリケーションプログラムからコールするため、以下の名称としています。

サンプル例)

作成する共通の main 関数：user_app_main

作成ファイル：user_main.c

以下は、本サンプルプログラムの user_main.c ファイルを参考にして作成してください。

```
～
/***********************************************************************************
Includes <System Includes> , "Project Includes"
***********************************************************************************/
#include "r_cg_cmt.h"                                          ←CMT0関数用インクルード定義
#include "iodefine.h"
#include "r_system.h"
/***********************************************************************************
～中略～
/***********************************************************************************
Private variables and functions
***********************************************************************************/
void user_app_main(void);                                      ←user_app_mainの関数宣言

/***********************************************************************************
* Outline      : user main processing
* Function Name: user_app_main
～中略～
***********************************************************************************/
void user_app_main (void)                                      ←user_app_mainの関数
{
    R_CMT0_Start();                                            ←サンプル使用CMT0の開始
}
/***********************************************************************************
 End of function main
***********************************************************************************/
/* End of File */
```

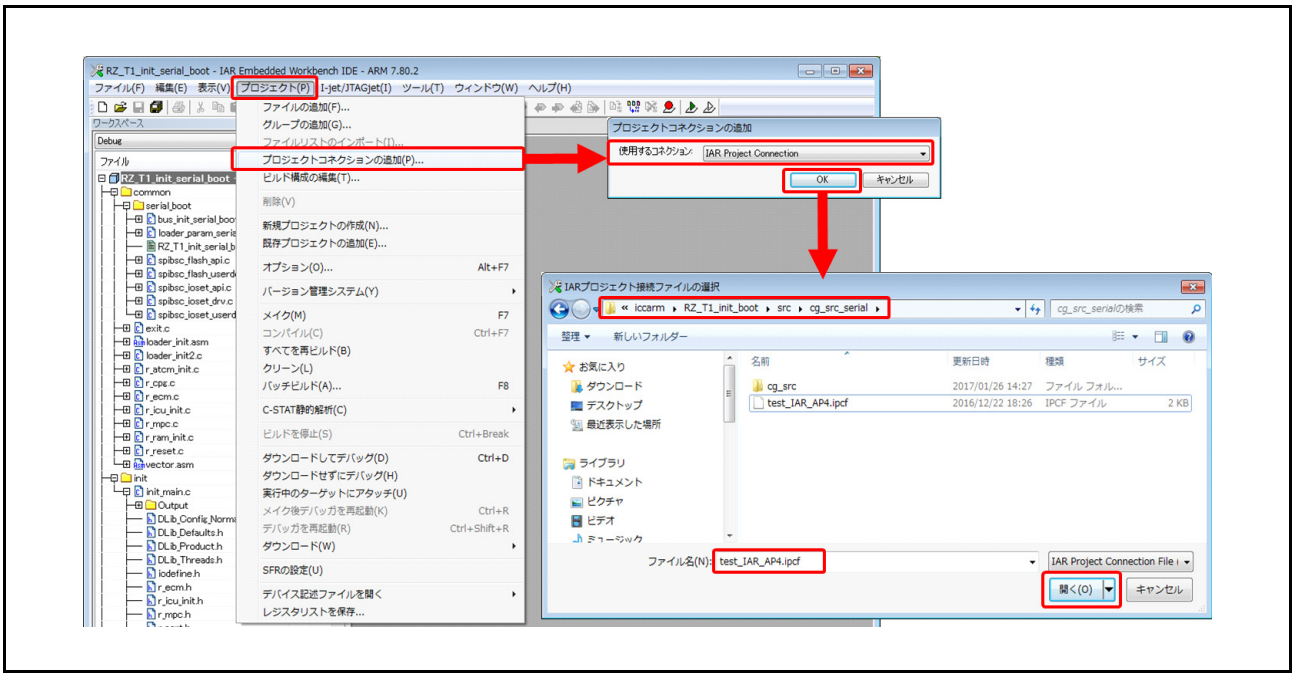
(5) コード生成ツールで作成したソースコードのコンパイル対象設定

EWARM 版の初期設定サンプルプルグラムのコンパイル対象にコード生成ツールで作成したソースコードを以下のように設定します。

【プロジェクトコネクションの追加】

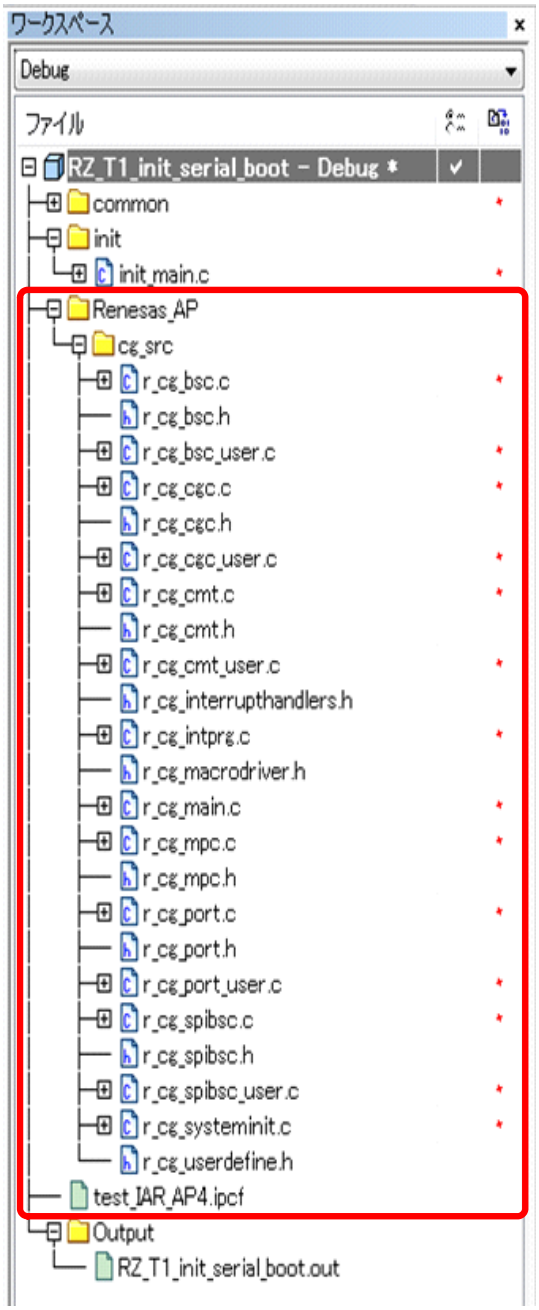
- ツールバーの［プロジェクト］メニューから［プロジェクトコネクションの追加］を選択します。
  ［プロジェクトコネクションの追加］のダイアログが表示されます。
  使用するコネクションに “IAR Project Connection” を選択し “OK” ボタンを押します。

- 画像⑤ -1 プロジェクトコネクションの追加



- ［IAR プロジェクト接続ファイルの選択］ダイアログが表示されます。プロジェクト・コネクション・ファイル（.ipcf）を選択して［開く］ボタンをクリックします。
  プロジェクト・コネクション・ファイルには、ソースファイルの登録情報が含まれています。
  ここで選択するプロジェクト・コネクション・ファイルは、コード生成ツールで作成した以下のフォルダ（推奨）内にコード生成すると作成されます。
  参照先のフォルダサンプル例)
  16 ビットバスブートモード版フォルダ名：cg_src_nor
  SPI ブートモード版フォルダ名：cg_src_serial
  完了すると以下のようにコンパイル対象となるファイルが追加されます。

- 画像⑤ -2 コンパイル対象となるファイルの追加例

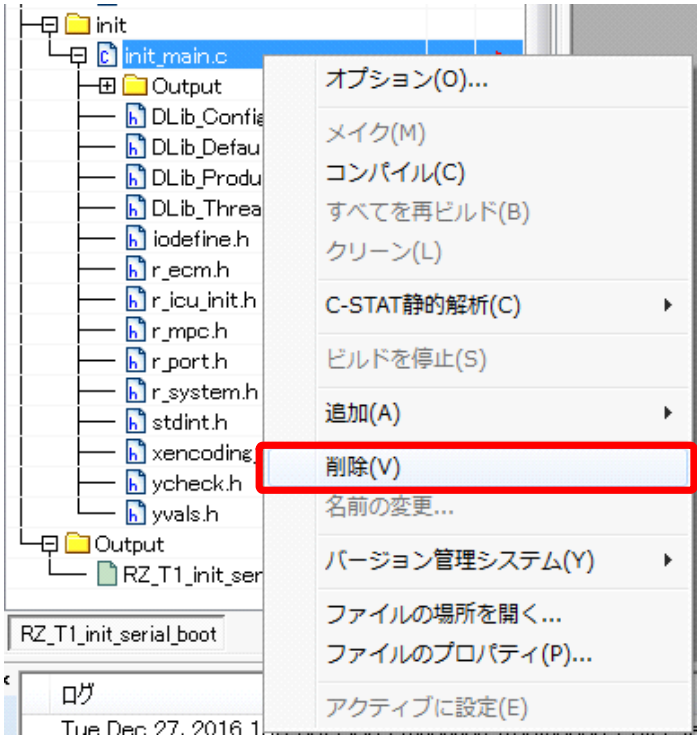


注.　［プロジェクトコネクションの追加］の使用方法については、「AP4, Applet3 ユーザーズマニュアル 共通操作編（R20UT3420JJ）」の「3.12.1 IAR Embedded Workbench の出力ソース・コードの取り込み方法」を参照ください。

備考：DS-5、e²studio では、コード生成ツールが作成したフォルダにファイルを追加すると自動でプロジェクトがファイルを認識します。

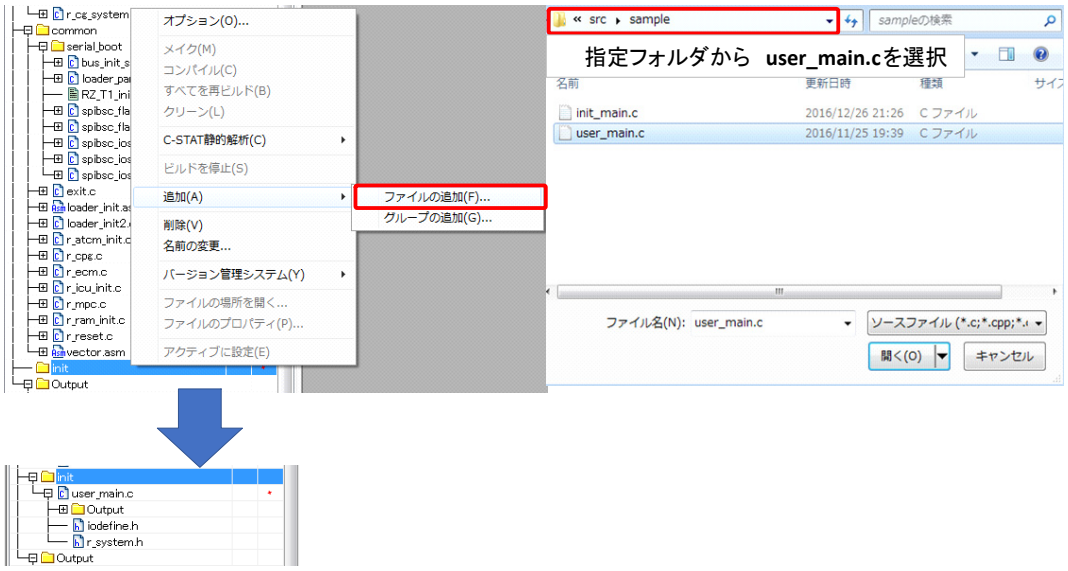
（6）初期設定で使用している main 関数のコンパイル対象の除外設定

- ワークスペースにある init_main ファイルを選択して右クリック
  [削除] を選択してコンパイル対象の除外設定



（7）共通 main 関数のコンパイル対象設定

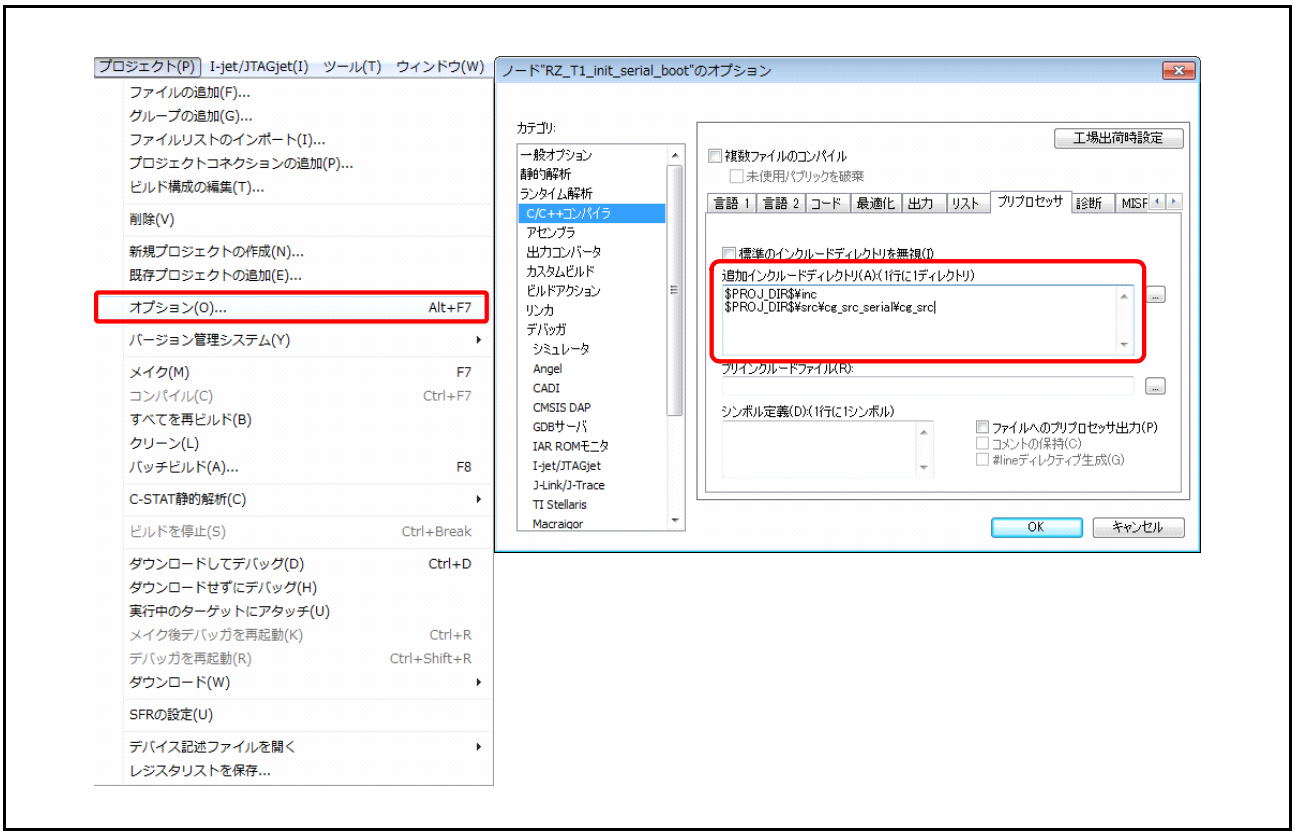
- init フォルダにコンパイル対象となる user_main.c を追加



(8) 新規追加したコード生成ツール出力の生成コードで使用するインクルードパスを設定

- ツールバーにあるプロジェクトからオプションを選択し表示されたオプション設定より［C/C++ コンパイラ］→［プリプロセッサ］を選択
  追加インクルードディレクトリ設定にコード生成ツールで作成したフォルダを以下のサンプル例のようにワークファイルパスで追加
  サンプル例)

```
$PROJ_DIR$\src\cg_src_serial\cg_src
```

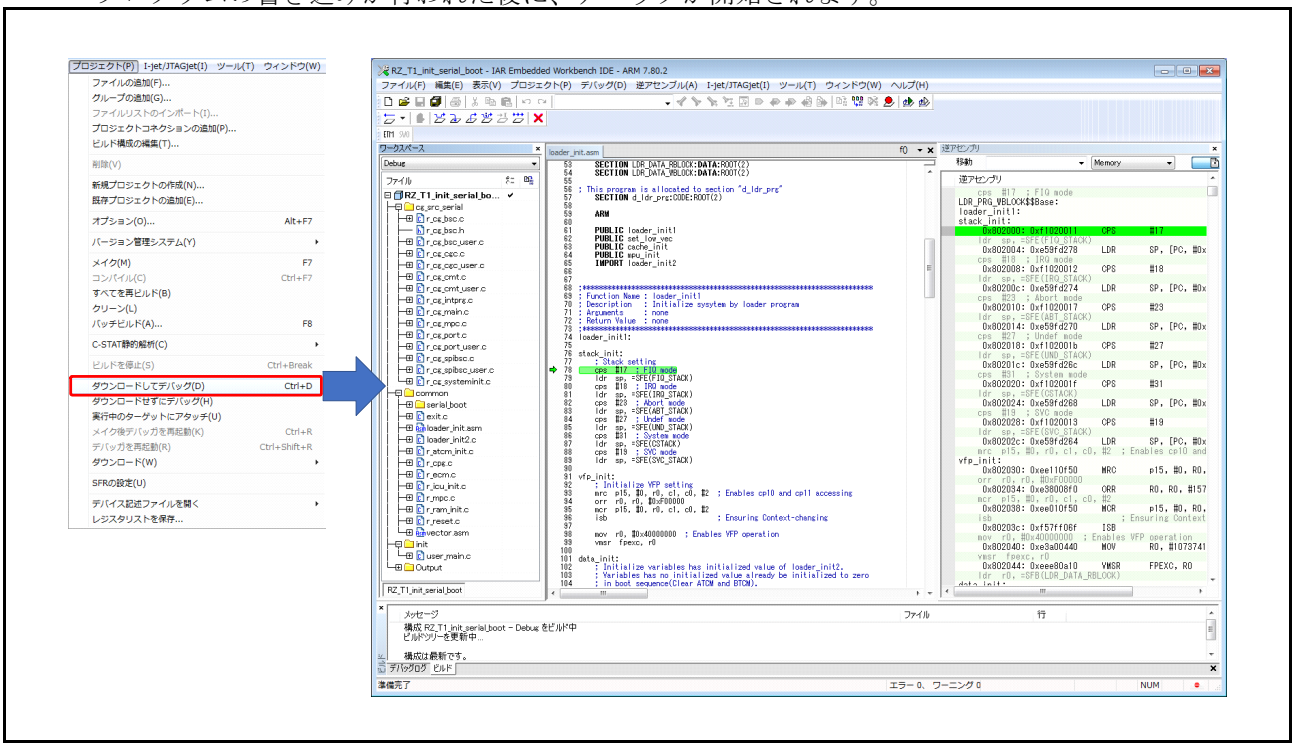


(9) すべてを再ビルド

- ツールバーにあるプロジェクトから以下のように［すべてを再ビルド］を選択します。
  選択すると再ビルドが開始されます。エラーがないことを確認ください。

(10)ダウンロードしてデバッグを実行

- ツールバーにあるプロジェクトから以下のように［ダウンロードしてデバッグ］を選択します。
  エミュレータ接続後、専用フラッシュダウンローダ機能により外付けシリアルフラッシュメモリへプログラムの書き込みが行われた後に、デバッグが開始されます。



6.4　固定幅整数一覧

表 6.6 にサンプルプログラムで使用する固定幅整数を示します。

表6.6　サンプルプログラムで使用する固定幅整数

シンボル	内容
int8_t	8ビット整数、符号あり（標準ライブラリにて定義）
int16_t	16ビット整数、符号あり（標準ライブラリにて定義）
int32_t	32ビット整数、符号あり（標準ライブラリにて定義）
uint8_t	8ビット整数、符号なし（標準ライブラリにて定義）
uint16_t	16ビット整数、符号なし（標準ライブラリにて定義）
uint32_t	32ビット整数、符号なし（標準ライブラリにて定義）

6.5　関数一覧

初期設定編のサンプルプログラム内で使用している関数については、RZ/T1 グループ初期設定アプリケーションノートを参照してください。本一覧については、初期設定編に追加する関数について表 6.7 に示します。

表6.7　関数一覧

関数名
user_app_main

6.6　フローチャート

6.6.1　ローダプログラム処理

図 6.3 にローダプログラム処理のフローチャートを示します。

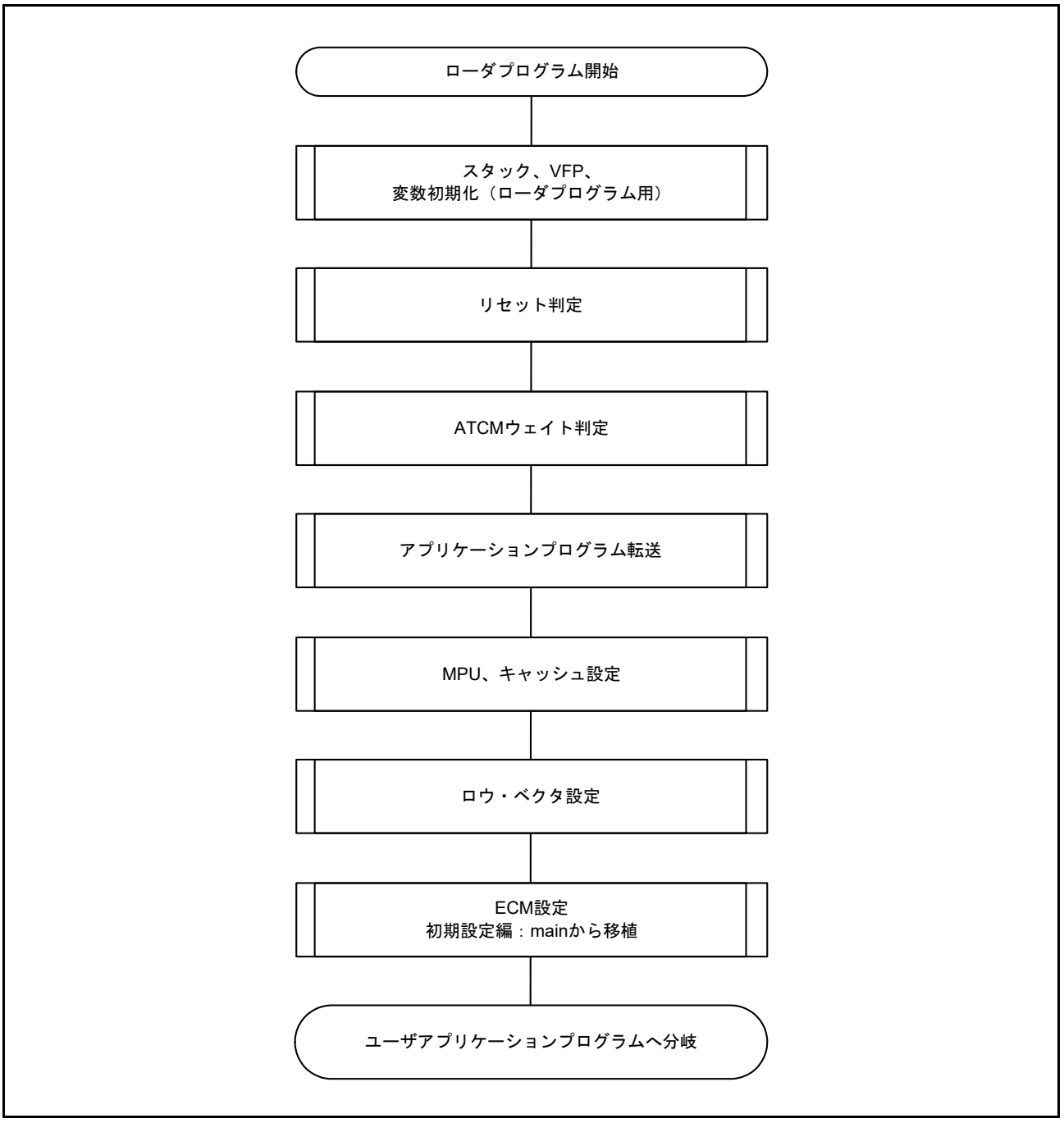
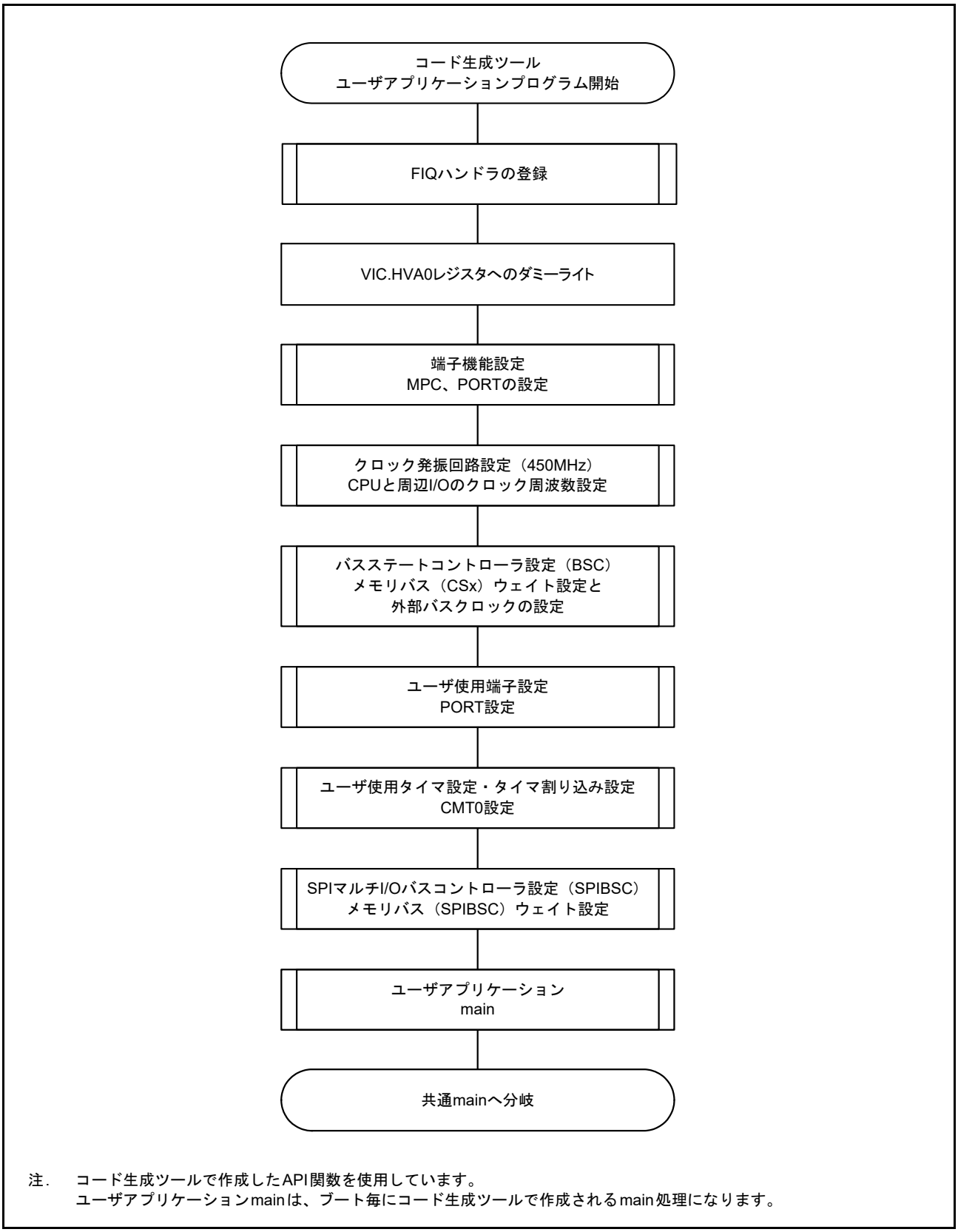


図 6.3　ローダプログラム処理

ローダプログラム部のフロー詳細については、RZ/T1 グループ初期設定アプリケーションノートを参照してください。

6.6.2　コード生成ツール作成アプリケーションプログラム処理

図 6.4 にコード生成ツール作成アプリケーションプログラムのフローチャートを示します。



注.　コード生成ツールで作成したAPI関数を使用しています。
　　ユーザアプリケーションmainは、ブート毎にコード生成ツールで作成されるmain処理になります。

図 6.4　コード生成ツール作成アプリケーションプログラム

6.6.3　共通 main 処理

図 6.5 に共通 main のフローチャートを示します。

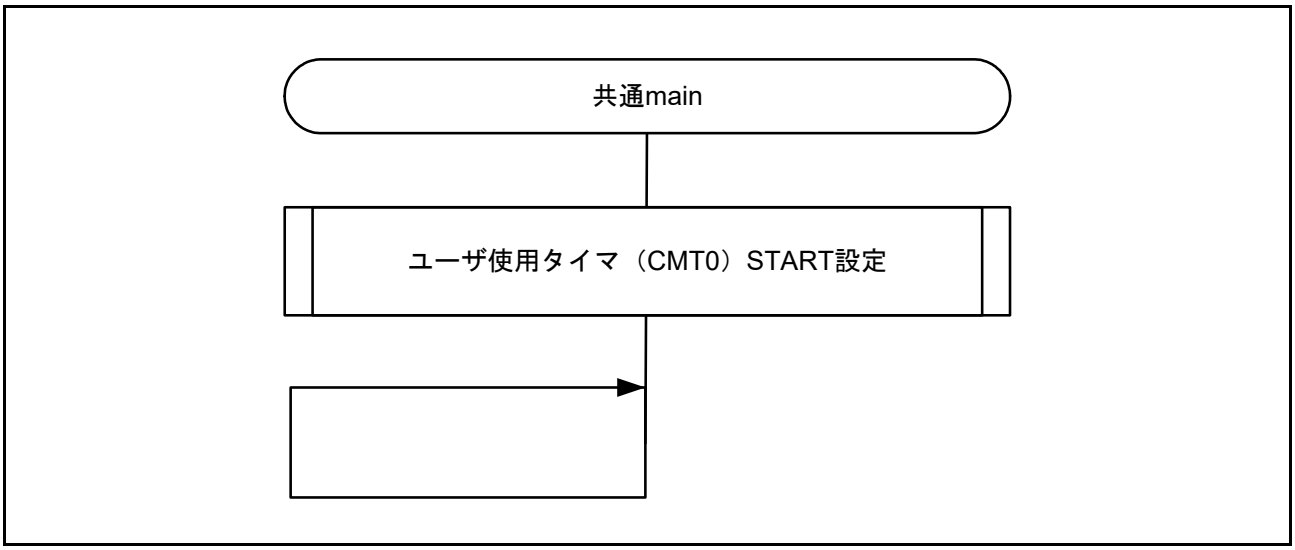


図 6.5　共通 main 処理

6.6.4　ユーザ使用タイマ（CMT0）割り込み処理

図 6.6 にユーザ使用タイマ（CMT0）割り込み処理のフローチャートを示します。

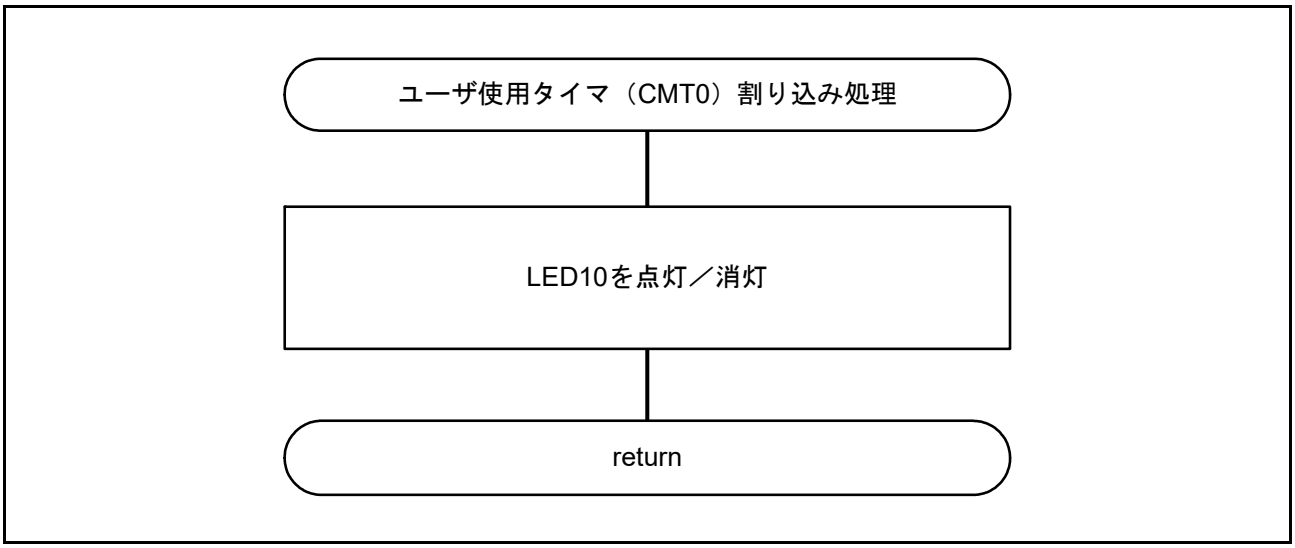


図 6.6　ユーザ使用タイマ（CMT0）割り込み処理

7.　サンプルプログラム

サンプルプログラムは、ルネサス エレクトロニクスホームページから入手してください。

8. 参考ドキュメント

- ユーザーズマニュアル：ハードウェア
  RZ/T1 グループ ユーザーズマニュアル ハードウェア編
  （最新版をルネサス エレクトロニクスホームページから入手してください。）

  RZ/T1 評価ボード RTK7910018C00000BE ユーザーズマニュアル
  （最新版をルネサス エレクトロニクスホームページから入手してください。）

- テクニカルアップデート／テクニカルニュース
  （最新の情報をルネサス エレクトロニクスホームページから入手してください。）

- ユーザーズマニュアル：開発環境
  IAR 統合開発環境（IAR Embedded Workbench® for ARM）に関しては、最新版を IAR ホームページから入手してください。
  ARM 統合開発環境（Development Studio 5™）に関しては、最新版を ARM ホームページから入手してください。
  ルネサス エレクトロニクス統合開発環境（e² studio）に関しては、最新版をルネサスエレクトロニクスホームページから入手してください。

9.　注意事項

(1)　bus_init() 関数を使用した場合の注意事項

本サンプルプログラムではベースとなる初期設定編サンプルプログラムの bus_init() 関数をコメントアウトして使用していません。ローダプログラムにてバス設定を行うため、bus_init() 関数を使用することも可能ですが、その際は以下の点にご注意ください。

bus_init() 関数では、シリアルフラッシュの設定を Single I/O モードから Quad I/O モードに設定しています。一方、本サンプルプログラムでは Single I/O モードを前提とした設定を行っているため、シリアルフラッシュのリード処理が正常に行えません。bus_init() 関数を使用される場合は、「6.3.1 の（4） **クロック、バス機能の設定**」の**【SPI マルチ I/O バスコントローラ設定】**手順にてコマンドに 0xEC（4READ4B）を設定するなど Quad I/O モードに対応した設定となるよう変更を行ってください。

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://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録	サンプルプログラムへのコード生成ツール適用ガイド アプリケーションノート

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2017.06.30	—	初版発行
1.10	2018.04.11	2. 動作環境	
		4	表2.1 動作環境　コード生成ツール：RENESAS製AP4のバージョンを変更、注を変更
		3. 関連アプリケーションノート	
		5	アプリケーションノートのドキュメント番号を追加
		6. ソフトウェア説明	
		13	6.1.3 例外処理ベクタテーブル　34バイト領域の番地を変更
		13	表6.4 例外処理ベクタテーブル　注：番地を変更
		23	6.3.1 コード生成ツールを使用したコードの作成について　コード生成ツールの説明を変更
		24	「画面⑤ SPIマルチI/Oバスコントローラ設定(一般設定)」の画面を変更、【注意点】を変更
		25	「画面⑤ SPIマルチI/Oバスコントローラ設定(設定)」の画面を変更
		27	「サンプル例) 端子配置表から端子機能の各設定画像と説明(3/3)」の画面を変更
		29	6.3.1, 【パスステートコントローラのAPI関数設定】の説明を変更、コードの記述を変更
		31	6.3.2 RZ/T1グループ 初期設定サンプルプログラムへの取り込み　備考：初期設定サンプルプログラムのRev.を変更
		32	6.3.2, (2) 初期設定サンプルプログラムにあるローダプログラムの編集　注1を変更、注2を追加
		33	6.3.2, (3): In the loader_init2 処理: set_low_vec(); と /* Initialize the ECM function */ の間に、ATCMのウエイト設定待ちのための関数を追加、注1を追加
		9. 注意事項	
		49	9. 注意事項　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. 未使用端子の処理
   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】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. リザーブアドレス（予約領域）のアクセス禁止
   【注意】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、将来の機能拡張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。
4. クロックについて
   【注意】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。
5. 製品間の相違について
   【注意】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、内部 ROM、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、ノイズ輻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)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/