

RX210 グループ

MTU2 を用いた相補 PWM モードの波形出力

要旨

本サンプルコードでは、MTU2 を用いて相補 PWM モードの波形を出力する方法について説明します。

対象デバイス

- RX210

内容

1.	仕様	3
2.	動作確認条件	3
3.	ハードウェア説明.....	4
3.1	使用端子一覧.....	4
4.	ソフトウェア説明.....	5
4.1	動作概要	5
4.1.1	出力する PWM 波形パターン.....	6
4.1.2	PWM 波形切り替えタイミング図	10
4.2	ファイル構成.....	15
4.3	オプション設定メモリ.....	16
4.4	定数一覧	16
4.5	変数一覧	17
4.6	関数一覧	18
4.7	関数仕様	19
4.8	作成する関数のフローチャート.....	20
4.8.1	初期設定.....	20
4.8.2	メイン処理.....	21
4.8.3	割り込み関数.....	22
5.	PDG の設定	23
5.1	SYSTEM 設定	25
5.2	MTU2 の設定	26
5.3	SYSTEM の端子設定	30
5.4	ソースの生成.....	31
5.5	CS+への登録.....	32
6.	CS+のプロジェクトに PDG のソースファイルを登録する際の設定.....	33
7.	デバッグについて.....	35
7.1	ライブラリの内部処理をデバッグする準備.....	35
7.2	デバッグ	38
8.	動作確認方法	41
9.	参考ドキュメント.....	41

1. 仕様

MTU2（マルチファンクションタイマパルスユニット 2）を使用して、相補 PWM モードの波形を出力します。

2. 動作確認条件

本サンプルコードは、表 2.1 の条件で動作を確認しています。

表 2.1 動作確認条件

項目	内容
使用マイコン	R5F5210BBDFP（RX210 グループ）
動作周波数	・メインクロック：20MHz ・システムクロック (ICLK)：20MHz（メインクロック 1 分周） ・周辺モジュールクロック B(PCLKB)：20MHz（メインクロック 1 分周）
ボード電源電圧	5V
マイコン動作電圧	5V
エンディアン	リトルエンディアン
動作モード	シングルチップモード
プロセッサモード	スーパバイザモード
統合開発環境	ルネサスエレクトロニクス製品 CS+ for CC-RL V5.00.00
エミュレータ	ルネサスエレクトロニクス製 E1 エミュレータ
使用ボード	北斗電子製評価ボード HSBRX210-100B (R5F5210BBDFP)

3. ハードウェア説明

3.1 使用端子一覧

表 3.1 に使用端子と機能を示します。

表 3.1 使用端子と機能

端子名	入出力	内容
P14	出力	MTIOC3A (PWM 周期に同期したトグル出力)
P16	出力	MTIOC3D (PWM 出力 1 逆相出力)
P17	出力	MIIOC3B (PWM 出力 1 正相出力)
P24	出力	MTIOC4A (PWM 出力 2 正相出力)
P25	出力	MTIOC4C (PWM 出力 2 逆相出力)
P30	出力	MTIOC4B (PWM 出力 3 正相出力)
P31	出力	MTIOC4D (PWM 出力 3 逆相出力)

4. ソフトウェア説明

4.1 動作概要

MTU2 の相補 PWM モードを使用して、相補 PWM 波形と 1/2 周期ごとに反転する波形を出力する方法を説明します。

MTU2 から PWM 波形を正相 3 本、逆相 3 本、PWM の 1/2 周期の反転出力波形 1 本、計 7 本の波形を出力します。正相 3 本、逆相 3 本はそれぞれ同じ信号を出力し、正相と逆相のペア 3 相はそれぞれノンオーバーラップの関係にあります。ノンオーバーラップ時間をデッドタイムと呼びます。

また、PWM 波形は一定周期毎に異なる波形へ切り替えます。切り替える波形は 4.1.1 に示す PWM 波形 1、PWM 波形 2、PWM 波形 3、PWM 波形 4 です。各波形を 10 周期ずつ出力しながら、PWM 波形 1 → PWM 波形 2 → PWM 波形 3 → PWM 波形 4 → PWM 波形 1 → … という順番で出力します。波形を切り替える際の動作については 4.1.2 を参照してください。

〈MTU2 チャンネル 3, チャンネル 4〉

MTU2 を以下に設定します。相補 PWM モードにすると、チャンネル 3、チャンネル 4 を組み合わせて PWM 波形を 3 相出力します。合わせて「5.2 MTU2 の設定」も参照ください。

【初期設定】

- 相補 PWM モード
- MTIOC3B、MTIOC3D、MTIOC4A、MTIOC4C、MTIOC4B、MTIOC4D 端子の出力許可
- 初期出力を High、アクティブレベルを Low
- PWM 同期出力のトグル出力を許可
- PWM 周期を 350us
- デッドタイムを 25us
- バッファ動作を許可

4.1.1 出力する PWM 波形パターン

図 4.1 に PWM 波形 1 を示します。

正相出力：非アクティブレベル H 期間 ($50\ \mu\text{s}$) → アクティブレベル L 期間 ($250\ \mu\text{s}$)

→ 非アクティブレベル H 期間 ($50\ \mu\text{s}$)

逆相出力：アクティブレベル L 期間 ($25\ \mu\text{s}$) → 短絡防止時間(デッドタイム) ($25\ \mu\text{s}$)

→ 非アクティブレベル H 期間 ($250\ \mu\text{s}$)

→ 短絡防止時間(デッドタイム) ($25\ \mu\text{s}$) → アクティブレベル L 期間 ($25\ \mu\text{s}$)

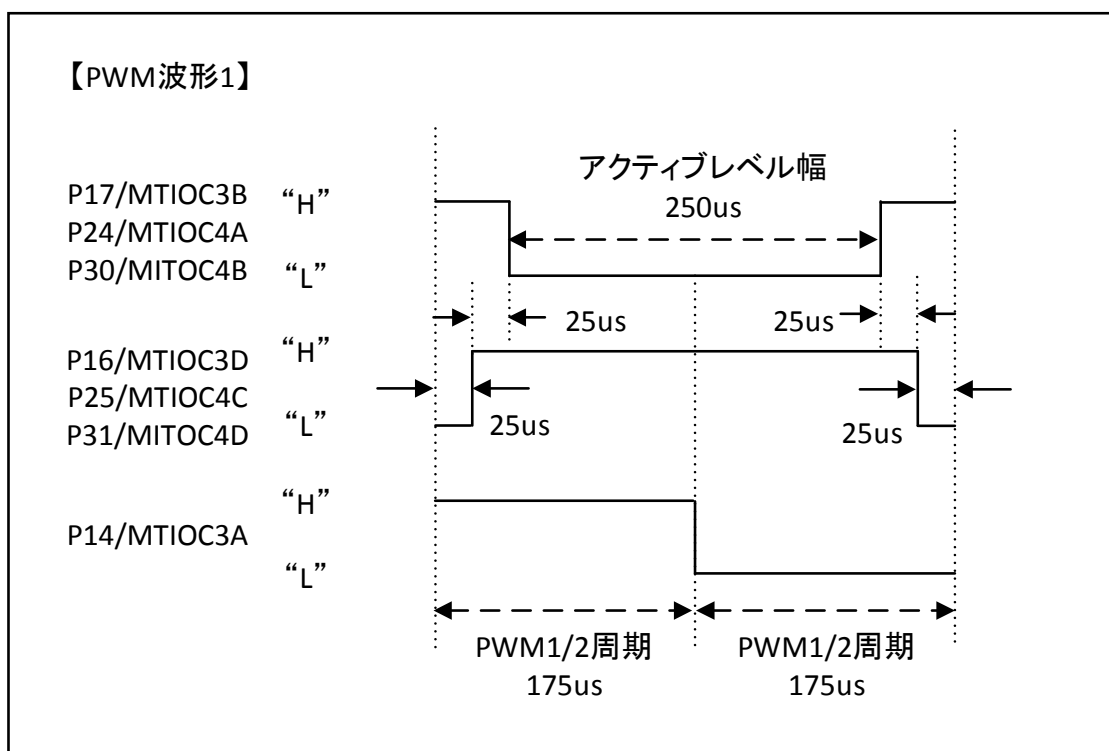


図 4.1 PWM 波形 1

図 4.2 に PWM 波形 2 を示します。

正相出力：非アクティブレベル H 期間 (125 μ s) \rightarrow アクティブレベル L 期間 (100 μ s)

\rightarrow 非アクティブレベル H 期間 (125 μ s)

逆相出力：アクティブレベル L 期間 (100 μ s) \rightarrow 短絡防止時間 (デッドタイム) (25 μ s)

\rightarrow 非アクティブレベル H 期間 (100 μ s)

\rightarrow 短絡防止時間 (デッドタイム) (25 μ s) \rightarrow アクティブレベル L 期間 (100 μ s)

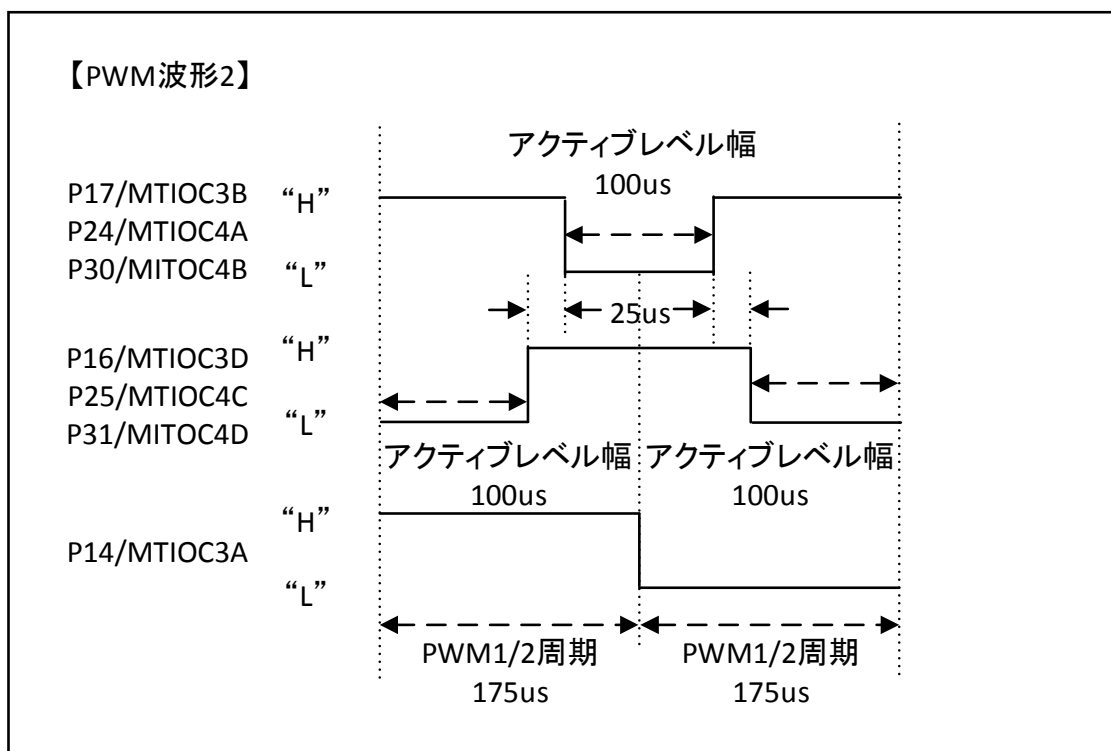


図 4.2 PWM 波形 2

図 4.3 に PWM 波形 3 を示します。
正相出力：アクティブレベル L 期間 (350 μ s)
逆相出力：非アクティブレベル H 期間 (350 μ s)

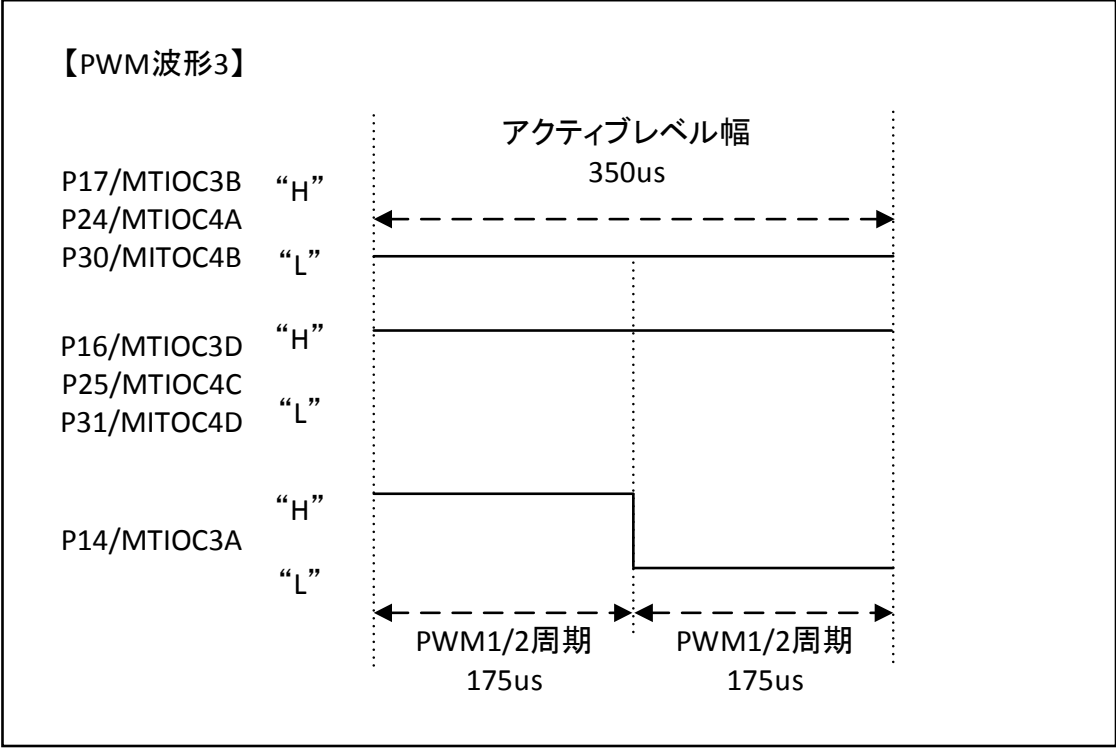


図 4.3 PWM 波形 3

図 4.4 に PWM 波形 4 を示します。
正相出力：非アクティブレベル H 期間 (350 μ s)
逆相出力：アクティブレベル L 期間 (350 μ s)

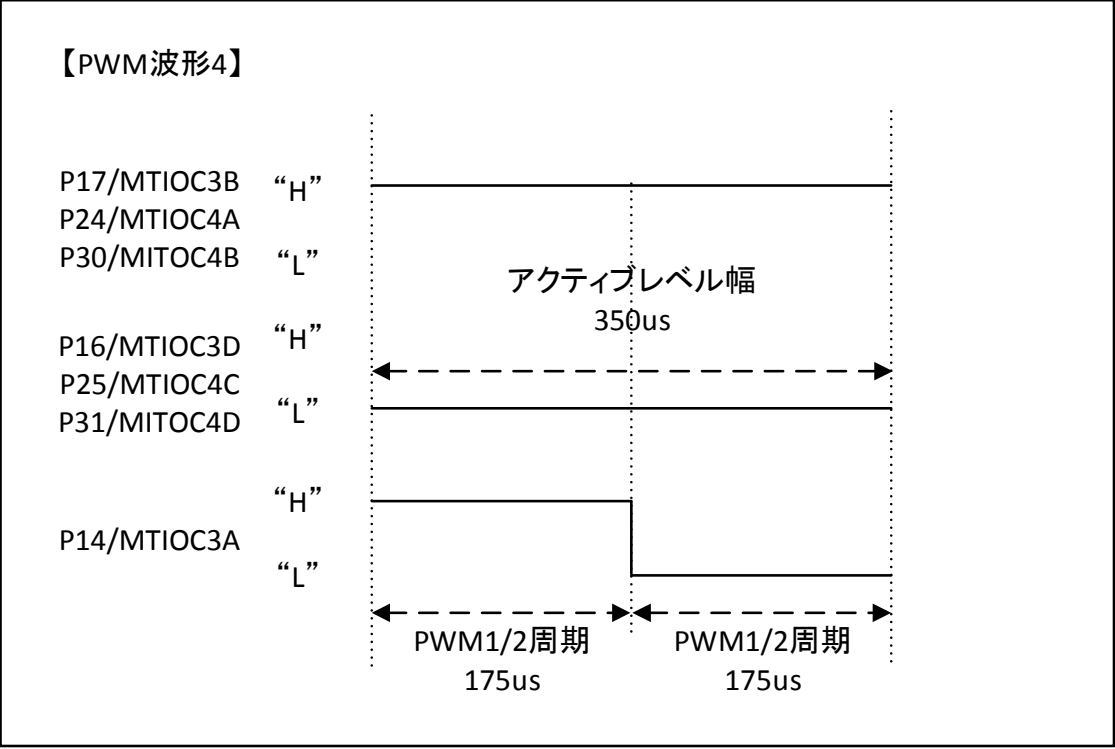


図 4.4 PWM 波形 4

4.1.2 PWM 波形切り替えタイミング図

各 PWM 波形を切り替えるタイミングを以下に示します。

図 4.5 に初期出力から PWM 波形 1 の出力開始（タイマスタート）を示します。

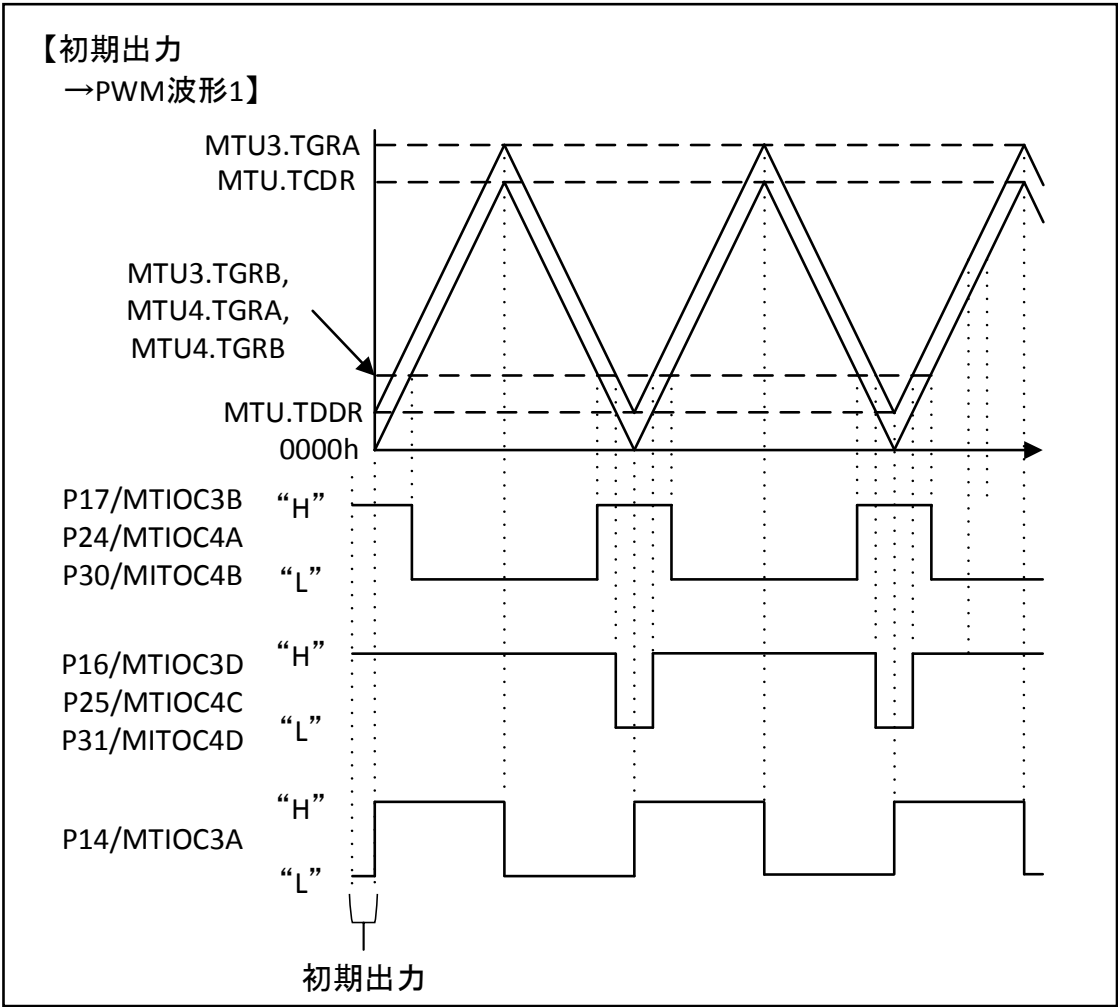


図 4.5 初期出力 → PWM 波形 1

図 4.6 に PWM 波形 1 から PWM 波形 2 の切り替えタイミングを示します。カウンタの谷でバッファの値を更新し、カウンタの谷でバッファ転送に設定しているため、カウンタの谷以降で出力が切り替わります。

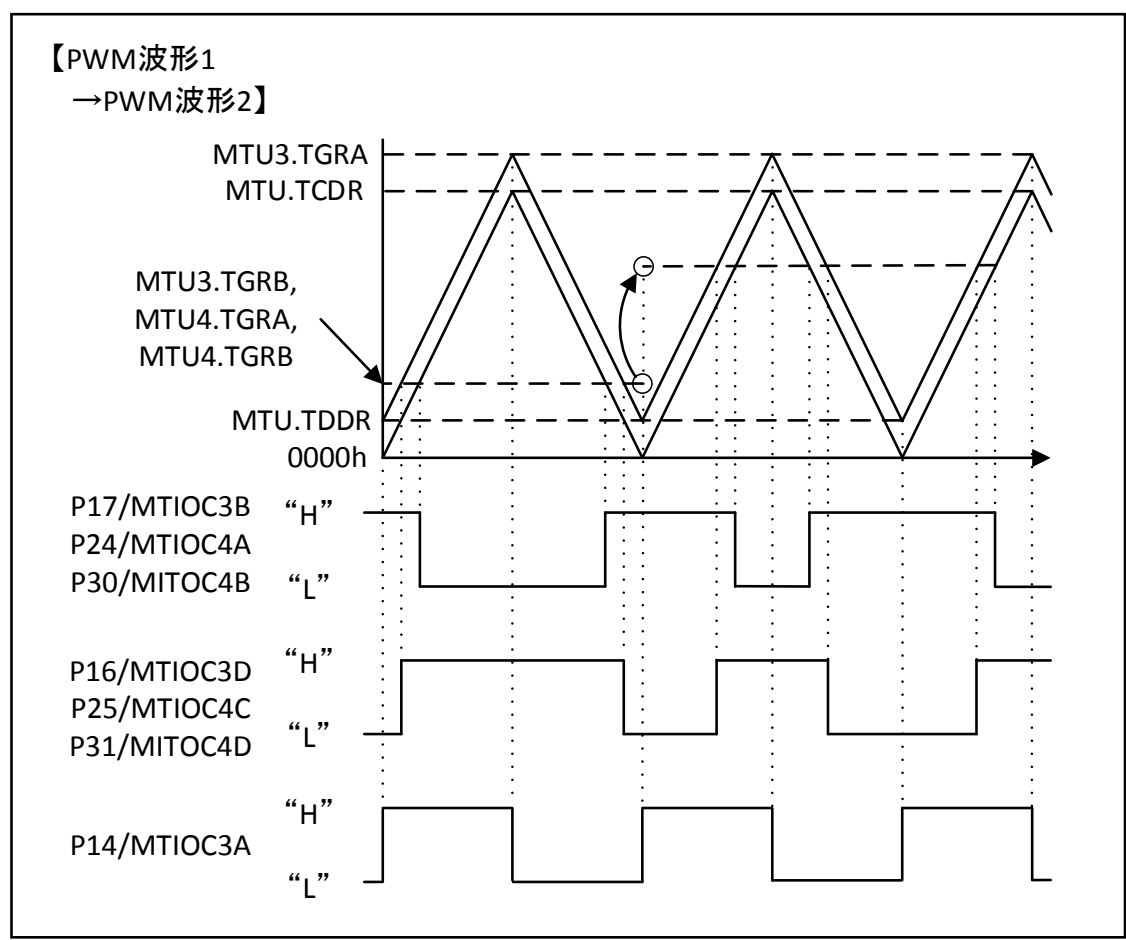


図 4.6 PWM 波形 1 → PWM 波形 2

図 4.7 に PWM 波形 2 から PWM 波形 3 の切り替えタイミングを示します。正相出力をデューティ比 100%（アクティブレベル L）に設定するため、MTU3.TGRB、MTU4.TGRA、MTU4.TGRB に 0000h を設定します。

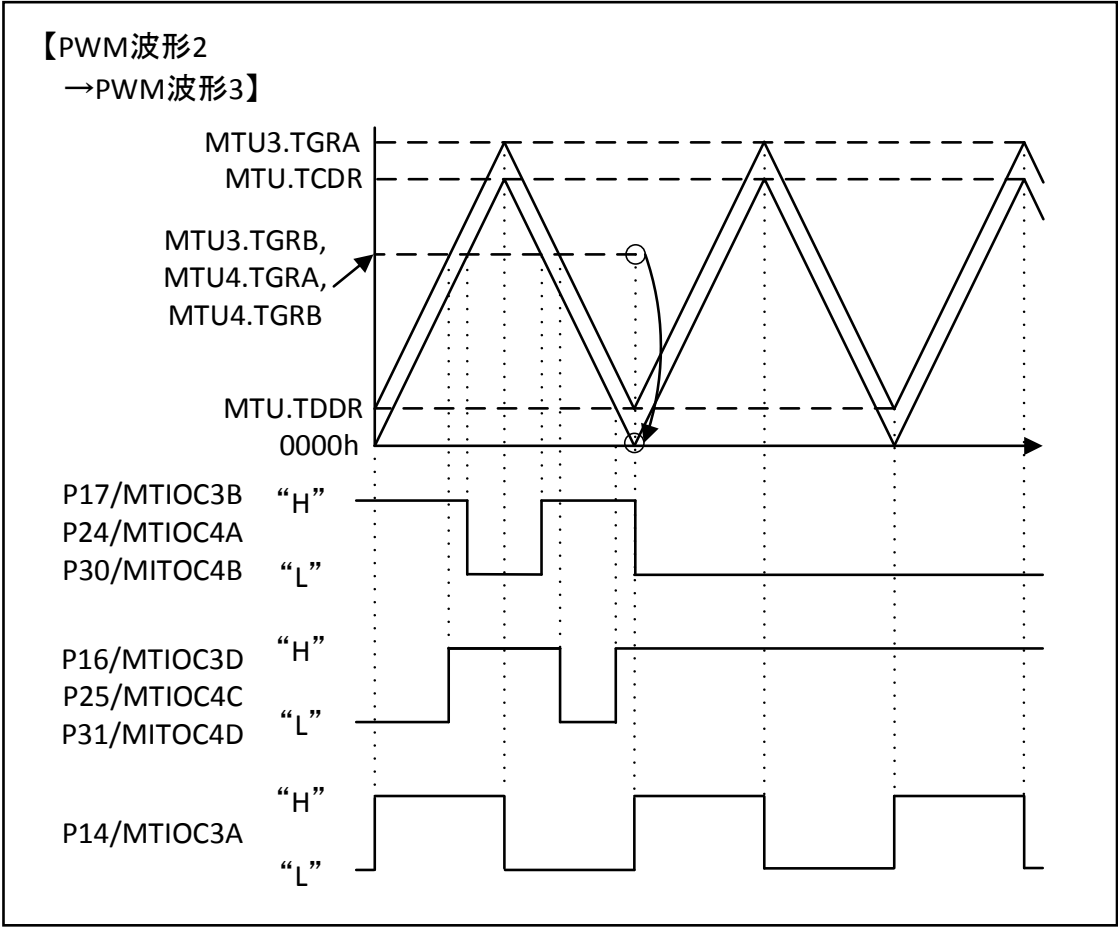


図 4.7 PWM 波形 2 → PWM 波形 3

図 4.8 に PWM 波形 3 から PWM 波形 4 の切り替えタイミングを示します。正相出力をデューティ比 0%（非アクティブレベル H）に設定するため、MTU3.TGRB、MTU4.TGRA、MTU4.TGRB に MTU3.TGRA と同値を設定します。

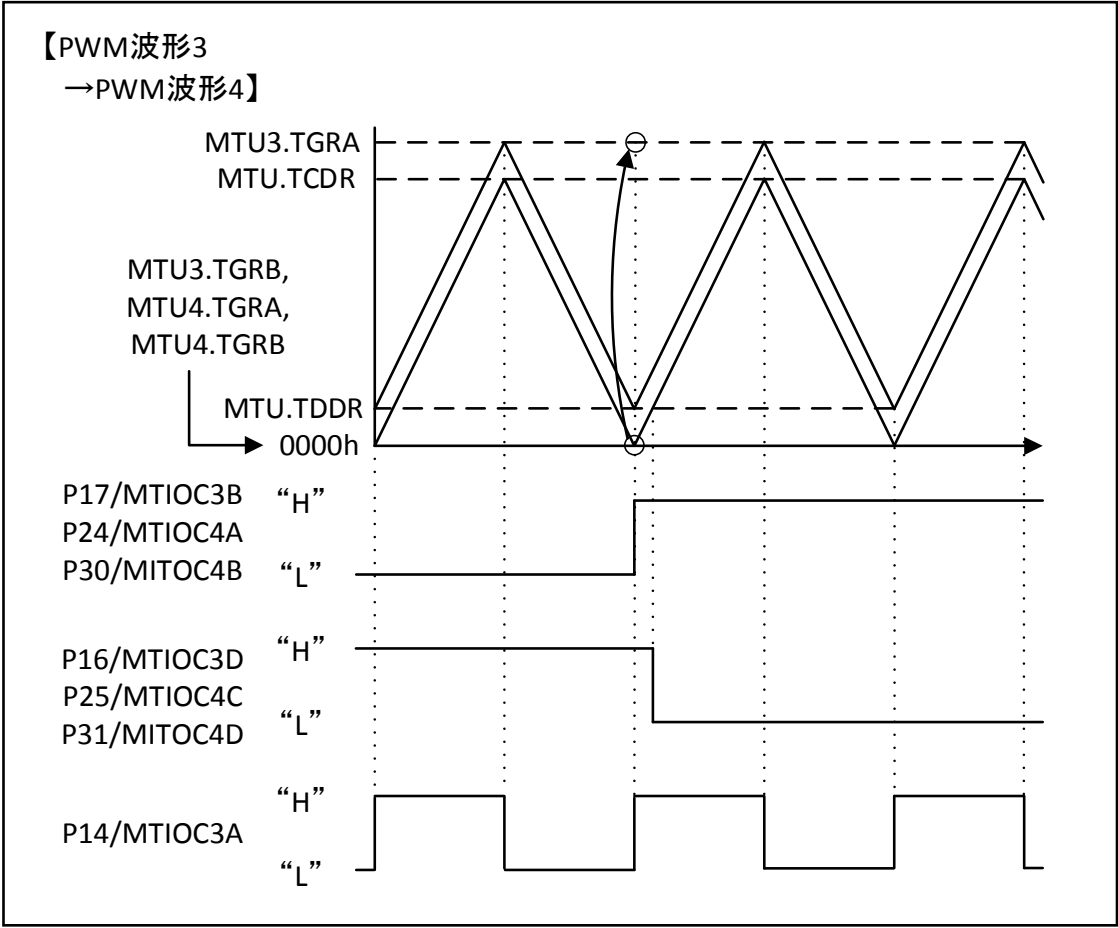


図 4.8 PWM 波形 3 → PWM 波形 4

図 4.9 に PWM 波形 4 から PWM 波形 1 の切り替えタイミングを示します。

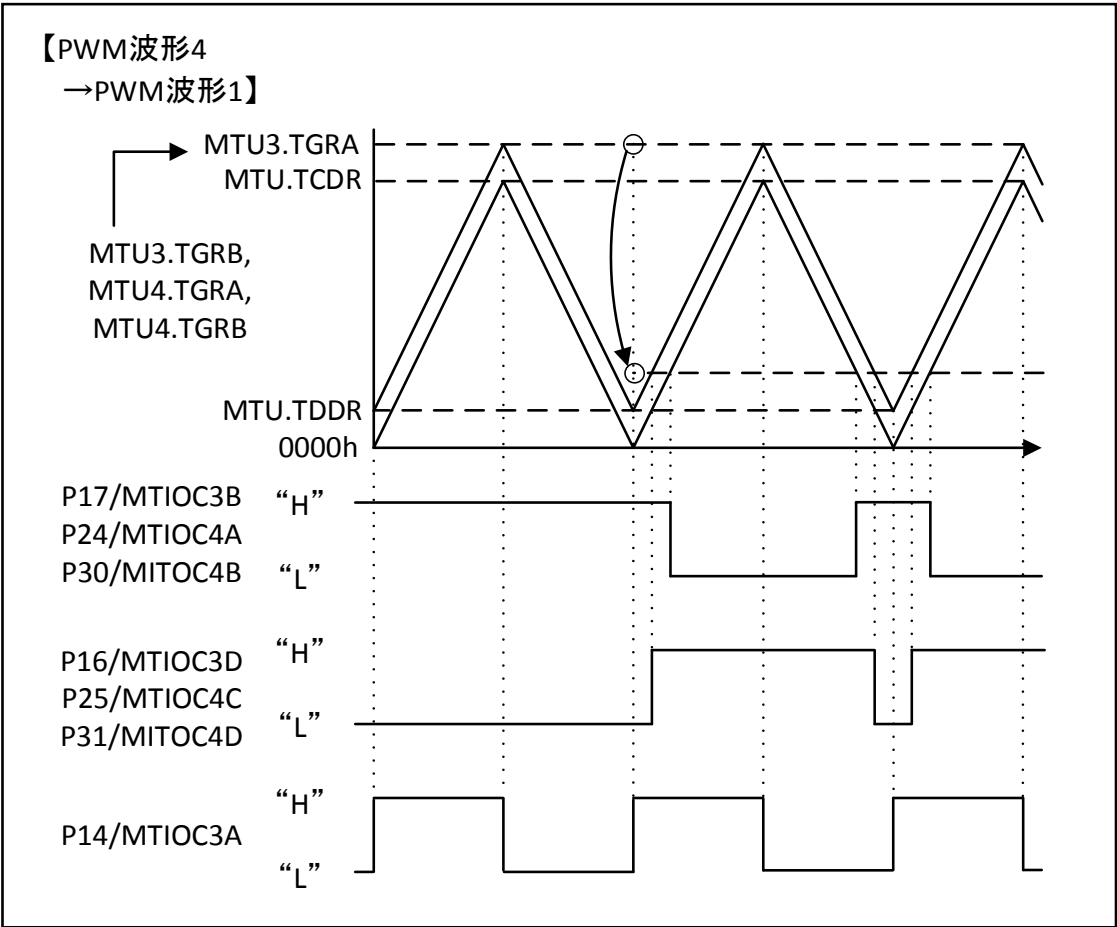


図 4.9 PWM 波形 4 → PWM 波形 1

4.2 ファイル構成

本サンプルコードを作成するにあたり、編集したファイルを表 4.1 に示します。統合開発環境で自動生成されて編集していないファイル、および 5. PDG の設定で生成されるファイルに関しましては割愛します。

表 4.1 ファイル名一覧

ファイル名	概要	備考
ComplementaryPWM_RX210.c	メインファイル ・PWM 波形パターンの切り替え ・オプション設定メモリ	
hwsetup.c	初期設定 ・存在しない端子の処理 ・クロックの設定 ・MTU2 の初期設定	
resetprg.c	リセット例外処理	HardwareSetup(); のコメントアウトを解除 しました

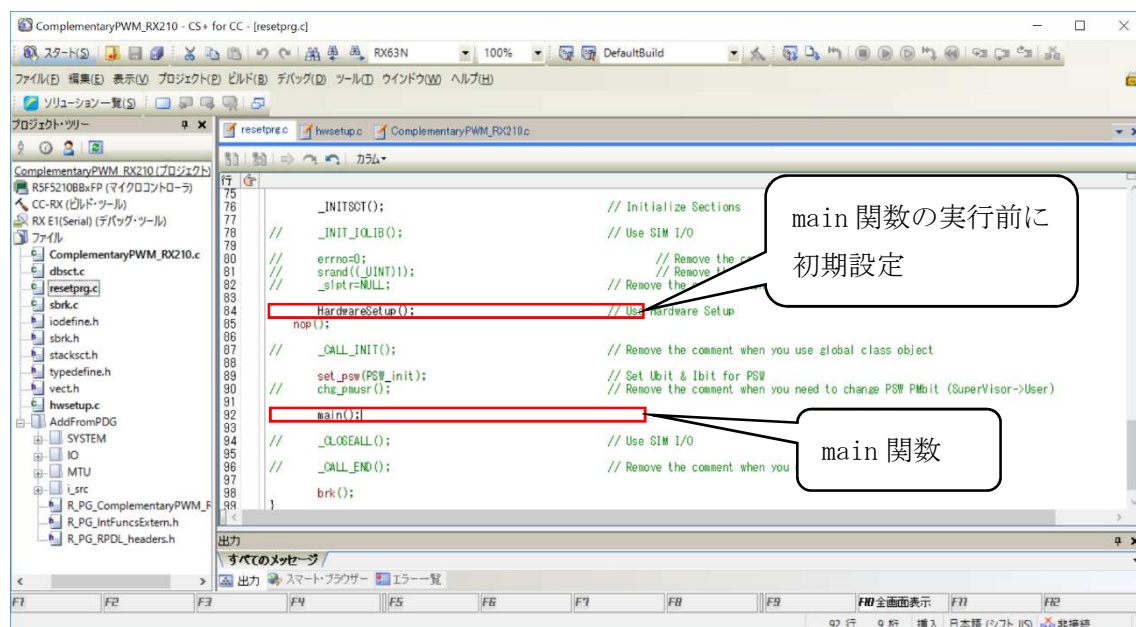


図 4.10 resetprg.c

4.3 オプション設定メモリ

表 4.2 に本サンプルコードで使用するオプション設定メモリの状態を示します。

表 4.2 オプション設定メモリー一覧

シンボル	アドレス	設定値	内容
OFS0	FFFF FF8Fh～FFFF FF8Ch	FFFF FFFFh	リセット後、IWDT は停止 リセット後、WDT は停止
OFS1	FFFF FF8Bh～FFFF FF88h	FFFF FFFFh	リセット後、 電圧監視 0 リセット無効 H0C0(高速オンチップオシレー タ)発振が無効
MDES	FFFF FF83h～FFFF FF80h	FFFF FFFFh	リトルエンディアン

OFS0 と OFS1 はメインファイルの最後尾に記載しています。

MDES については vecttbl.c ファイル(プロジェクト作成時に自動生成されるファイル)に定義されています。

4.4 定数一覧

表 4.3 に本サンプルコードで使用する定数、表 4.4 に const 型定数を示します。

表 4.3 サンプルコードで使用する定数

定数名	設定値	内容
NUM_WAVEFORM	4	出力させる相補 PWM 波形のパターン数
CYCLE_TIMES	10	PWM 波形を切り替えるタイミングの PWM 出力周期数

表 4.4 サンプルコードで使用する const 型定数

型	変数名	内容	使用関数
const uint16_t	duty_ratio[NUM_WAVEFORM]	各相補 PWM 波形における デューティ比の設定値	main

4.5 変数一覧

表 4.5 に本サンプルコードで使用する変数を示します。

表 4.5 サンプルコードで使用する変数

型	変数名	内容	使用関数
volatile uint8_t	int_count	PWM 出力カウンタ ※現在出力している PWM 波形パターンの出力周期数	main Mtu3IntFunc_A
volatile uint8_t	current_wave	現在出力している PWM 波形パターンを表す変数	main

4.6 関数一覧

表 4.6 に関数一覧を掲載します。本サンプルコードで新規作成、もしくは編集した関数のみ記載しています。PDG の設定は「5. PDG の設定」を参照ください。サンプルコードで使用している PDG で生成された関数に関しましては、「RX210 グループ Peripheral Driver Generator リファレンスマニュアル」を参照ください。

表 4.6 関数一覧

関数名	概要
main	メイン処理
Mtu3IntFunc_A	MTU2 チャネル 3 タイマカウンタと TGRA のコンペアマッチ 割り込み処理

4.7 関数仕様

本サンプルコードで作成、もしくは編集した関数仕様を示します。

main

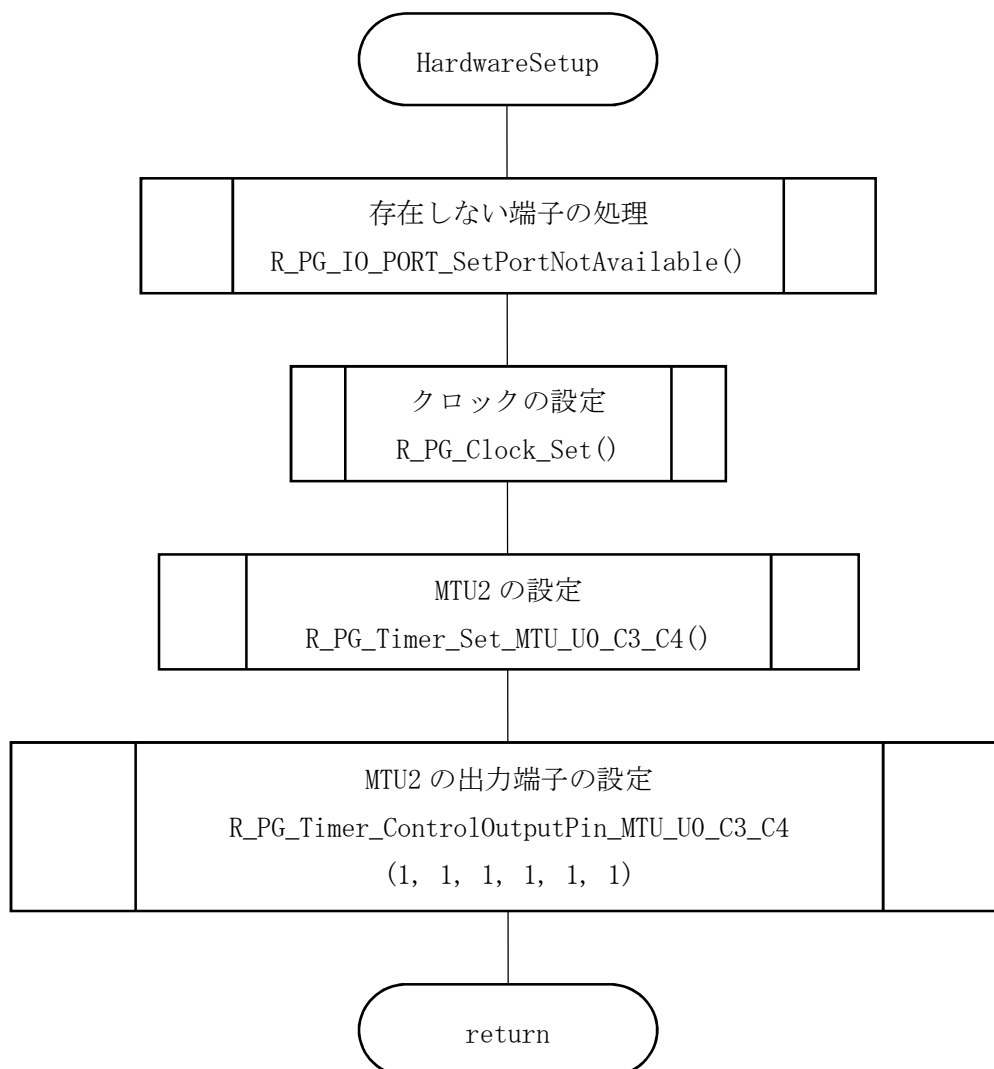
概要	メイン処理
ヘッダ	なし
宣言	void main(void)
説明	PWM 波形切り替えタイミングで PWM 波形 1 → PWM 波形 2 → PWM 波形 3 → PWM 波形 4 → PWM 波形 1 → … という順番で出力する
引数	なし
リターン値	なし

Mtu3IntFunc_A

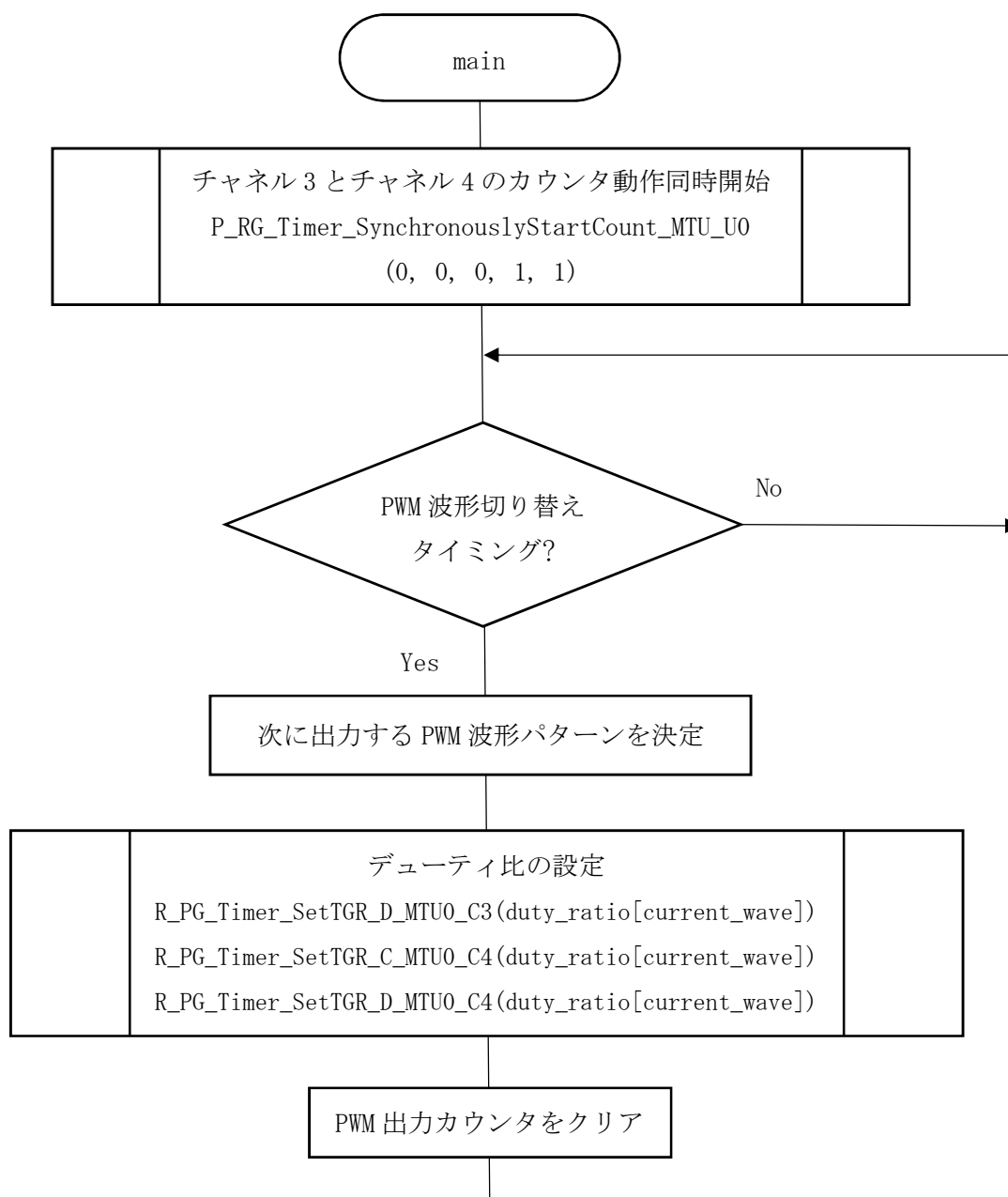
概要	MTU2 チャンネル 3 タイマカウンタと TGRA のコンペアマッチ割り込み処理
ヘッダ	なし
宣言	void Mtu3IntFunc_A (void)
説明	タイマカウンタの山の数进行を数える (PWM 波形切り替えタイミングに使用)
引数	なし
リターン値	なし

4.8 作成する関数のフローチャート

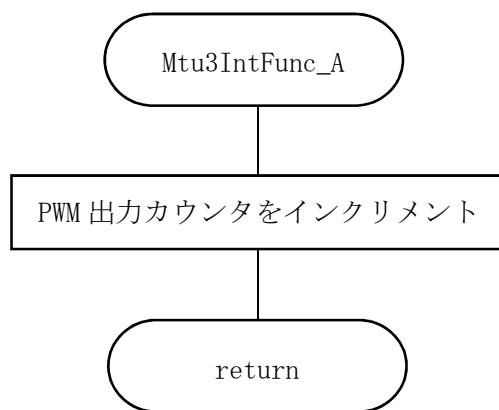
4.8.1 初期設定



4.8.2 メイン処理



4.8.3 割り込み関数



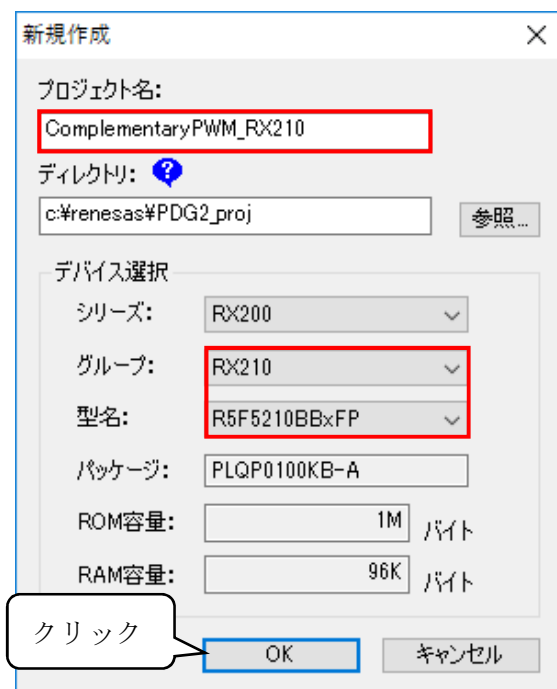
5. PDG の設定

本サンプルコードにおける PDG の設定を以下に説明します。本設定において生成されるソースファイルの詳細は”RX210 グループ Peripheral Driver Generator リファレンスマニュアル”を参照ください。

Peripheral Driver Generator 2 を起動します。



メニューバーのファイル->プロジェクトの新規作成 をクリックすると、以下のウィンドウが表示されます。プロジェクト名、マイコンのグループ、型を入力し、「OK」をクリックすると、プロジェクトが作成されます。



新規作成

プロジェクト名:
ComplementaryPWM_RX210

ディレクトリ: ?
c:\renesas\PDG2_proj 参照...

デバイス選択

シリーズ: RX200

グループ: RX210

型名: R5F5210BBxFP

パッケージ: PLQP0100KB-A

ROM容量: 1M バイト

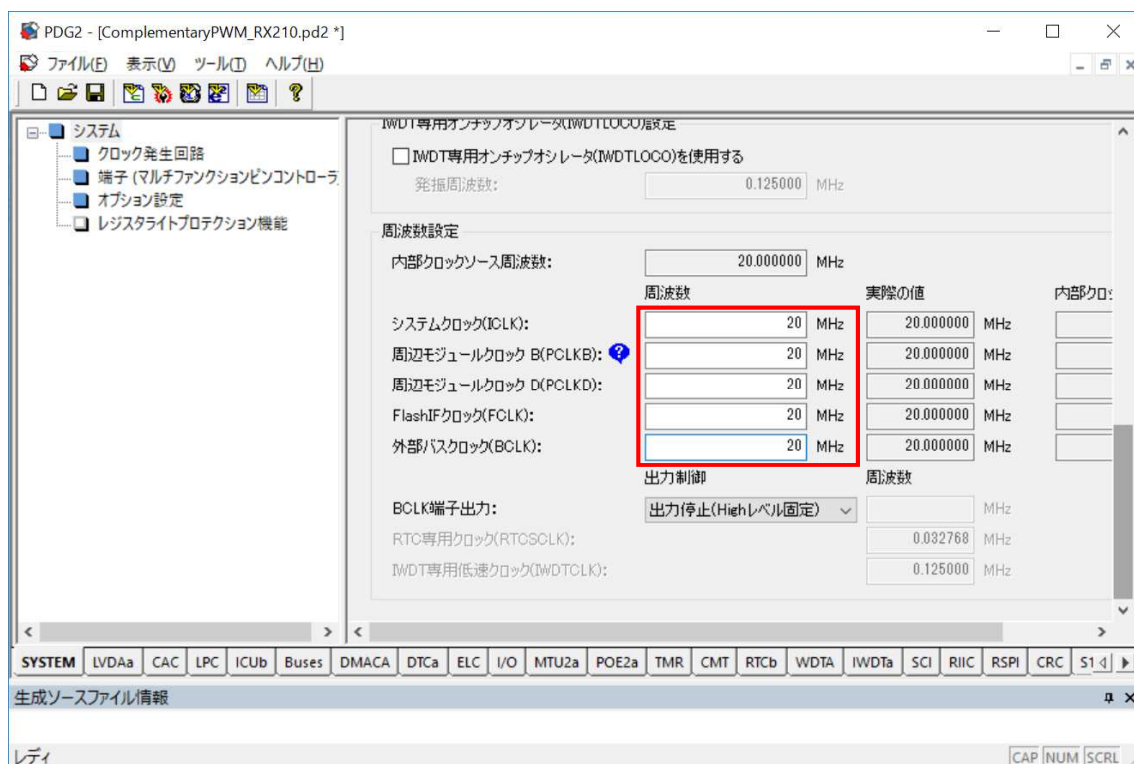
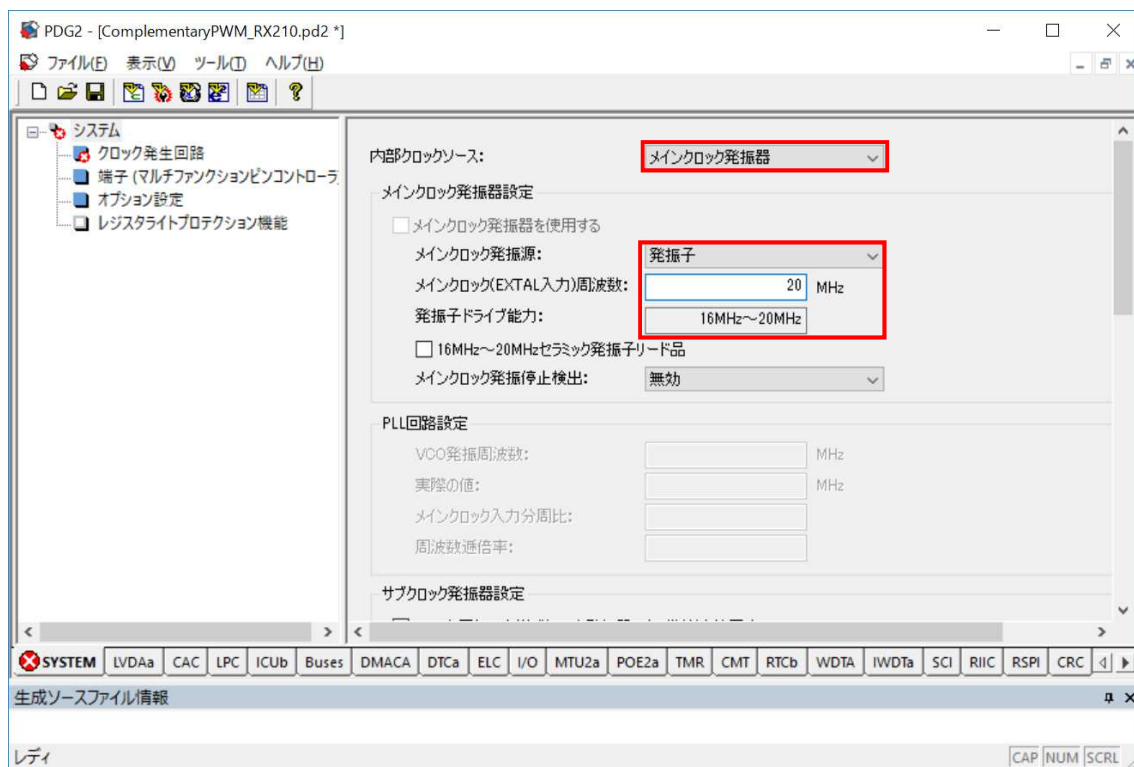
RAM容量: 96K バイト

クリック

OK キャンセル

5.1 SYSTEM 設定

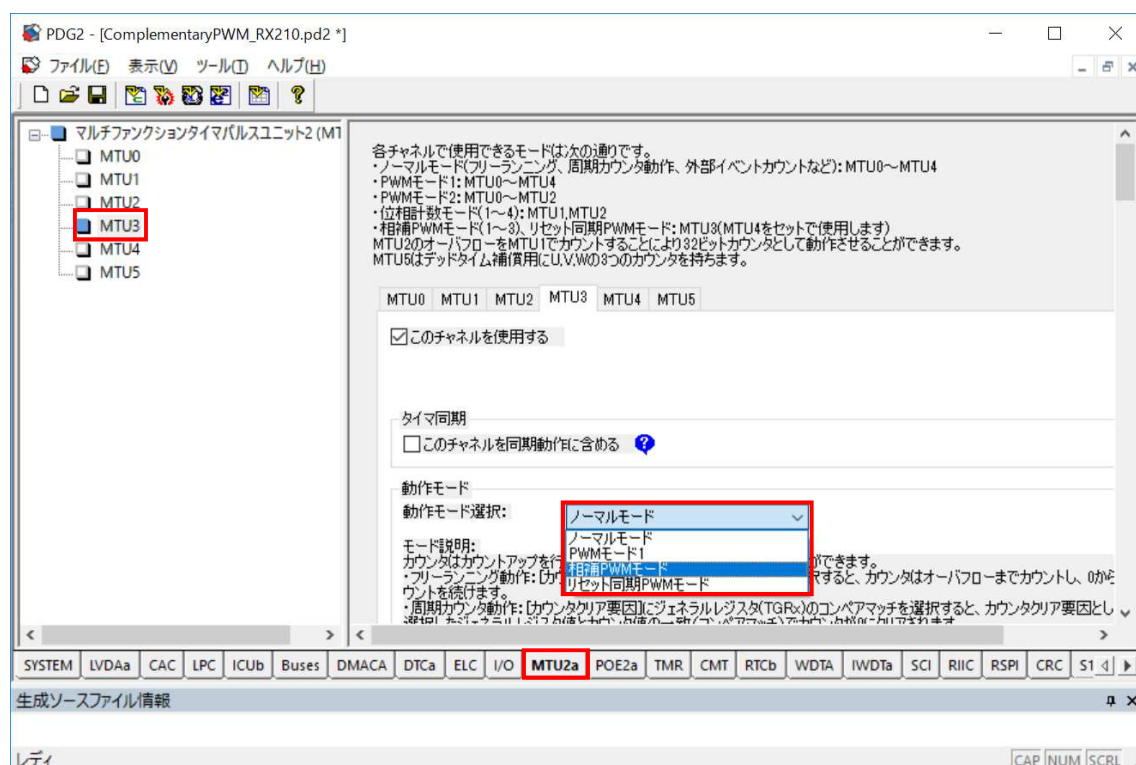
システムタブのクロック発生回路の設定を以下に示します。



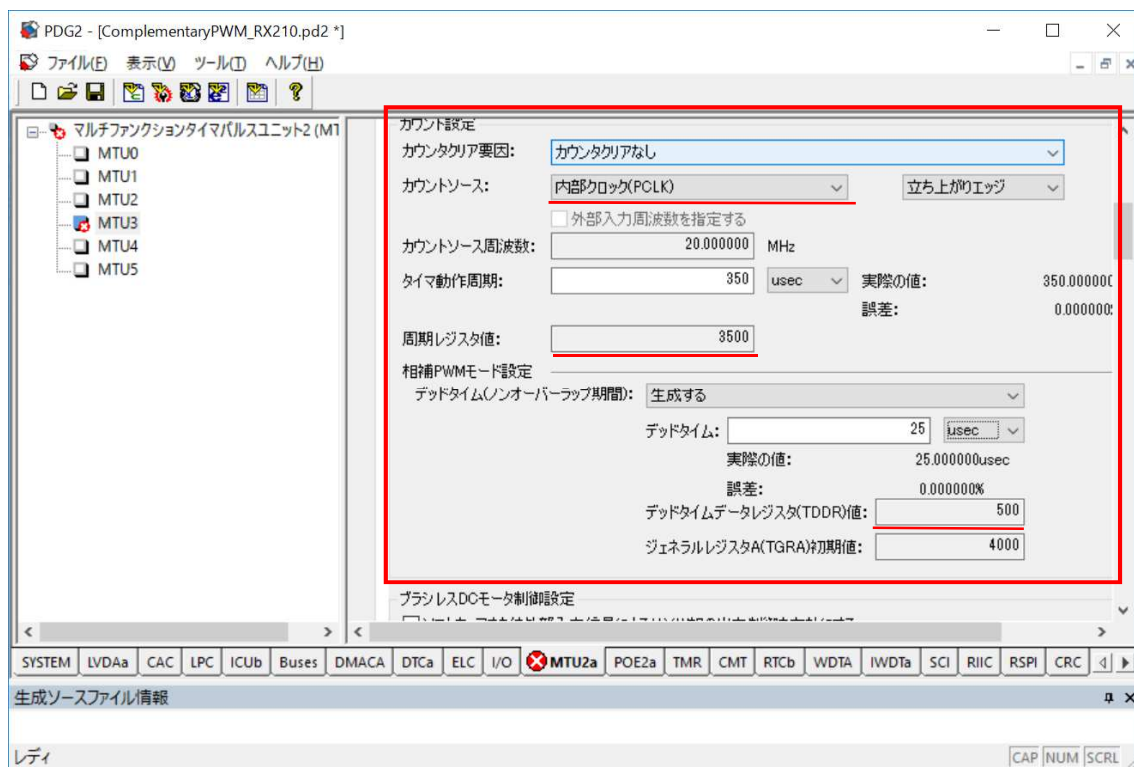
5.2 MTU2 の設定

MTU2 の設定を以下のように行います。

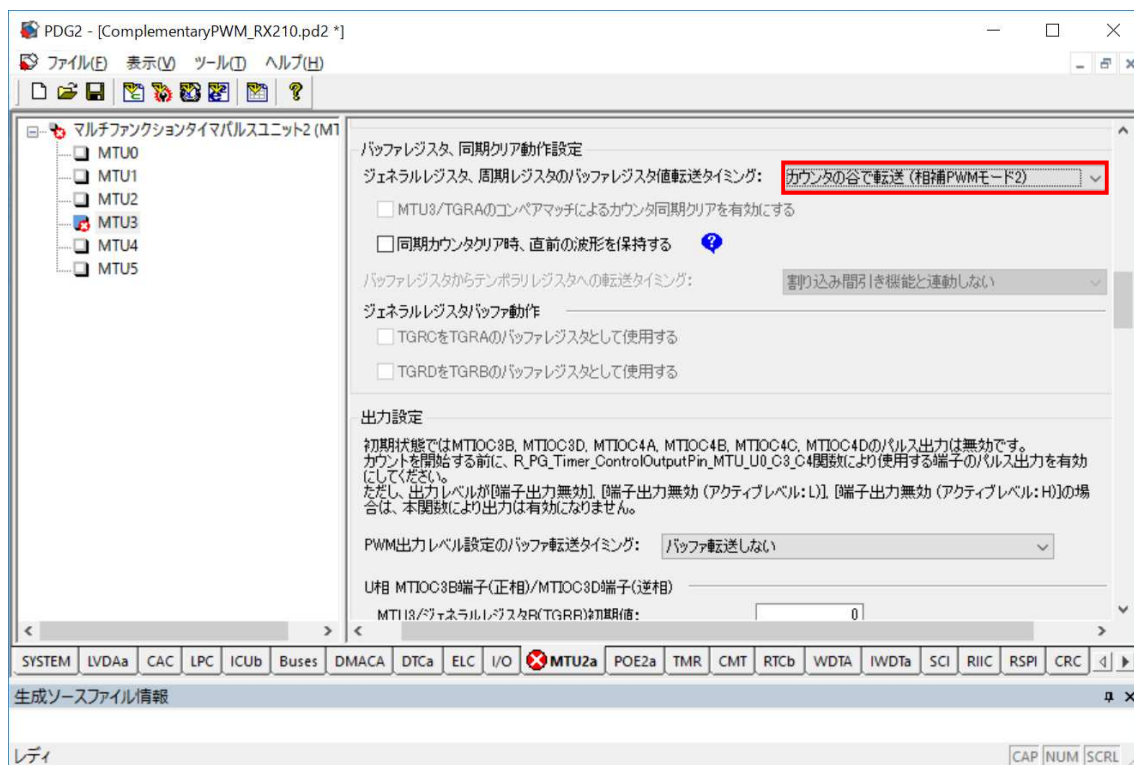
下部のタブから MTU2a を選択し、左部のツリーから MTU3 を選択します。その後、「このチャネルを使用する」にチェックを入れ、動作モードを「相補 PWM モード」に設定します。相補 PWM モードはチャンネル 3 とチャンネル 4 を使用しますが、PDG2 で相補 PWM モードを選択した場合、チャンネル 3 での設定が自動的にチャンネル 4 にも反映されます（チャンネル 4 に関して、PDG2 上で直接設定を指定する必要はありません）。



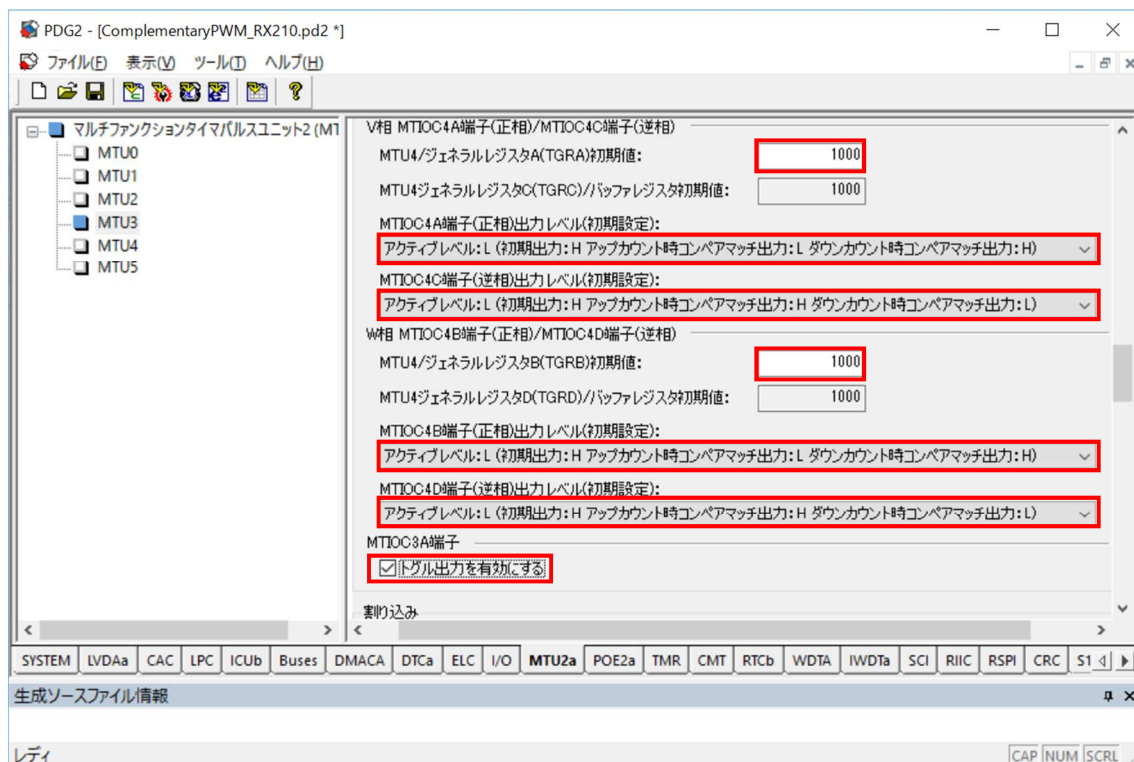
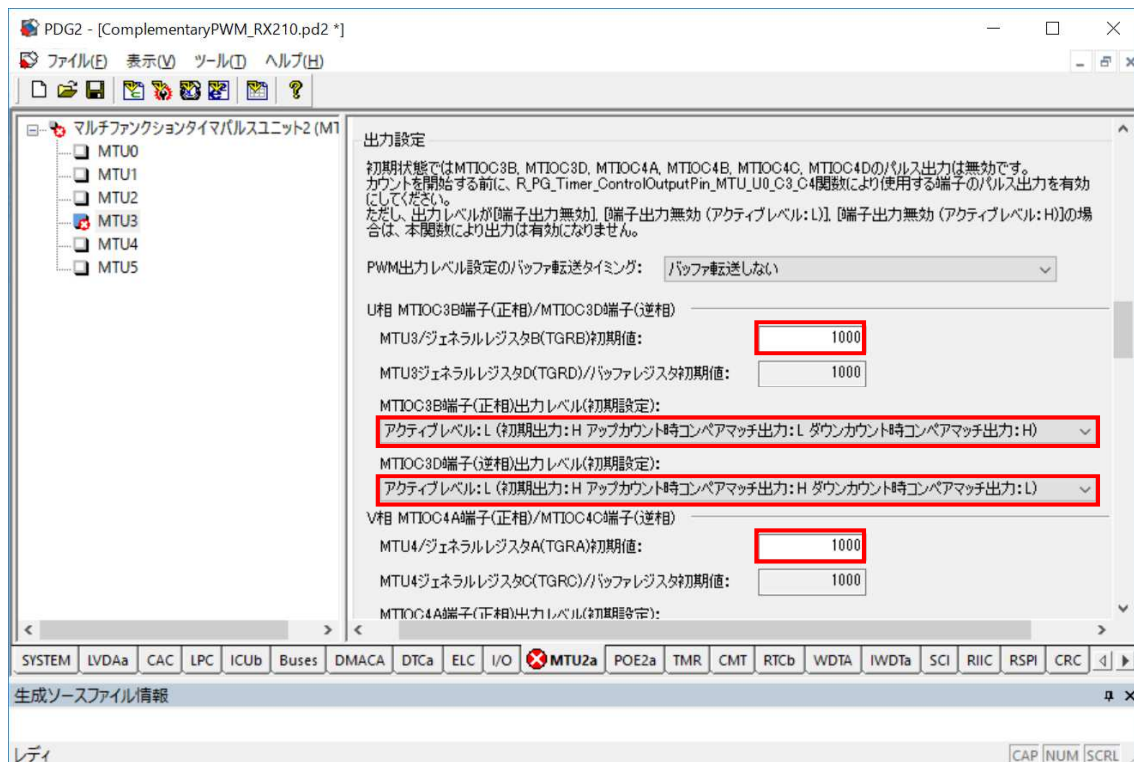
タイマ動作周期とデッドタイムを入力すると、それぞれ対応するレジスタの値が自動的に表示されます。



バッファレジスタの転送タイミングを「カウンタの谷で転送」に設定します。



出力端子の設定をします。各ジェネラルレジスタの初期値は PWM 波形 1 に設定します。相補 PWM 波形を出力する 6 つの端子は全て「アクティブレベル:L」を選択します。トグル出力を有効にします。

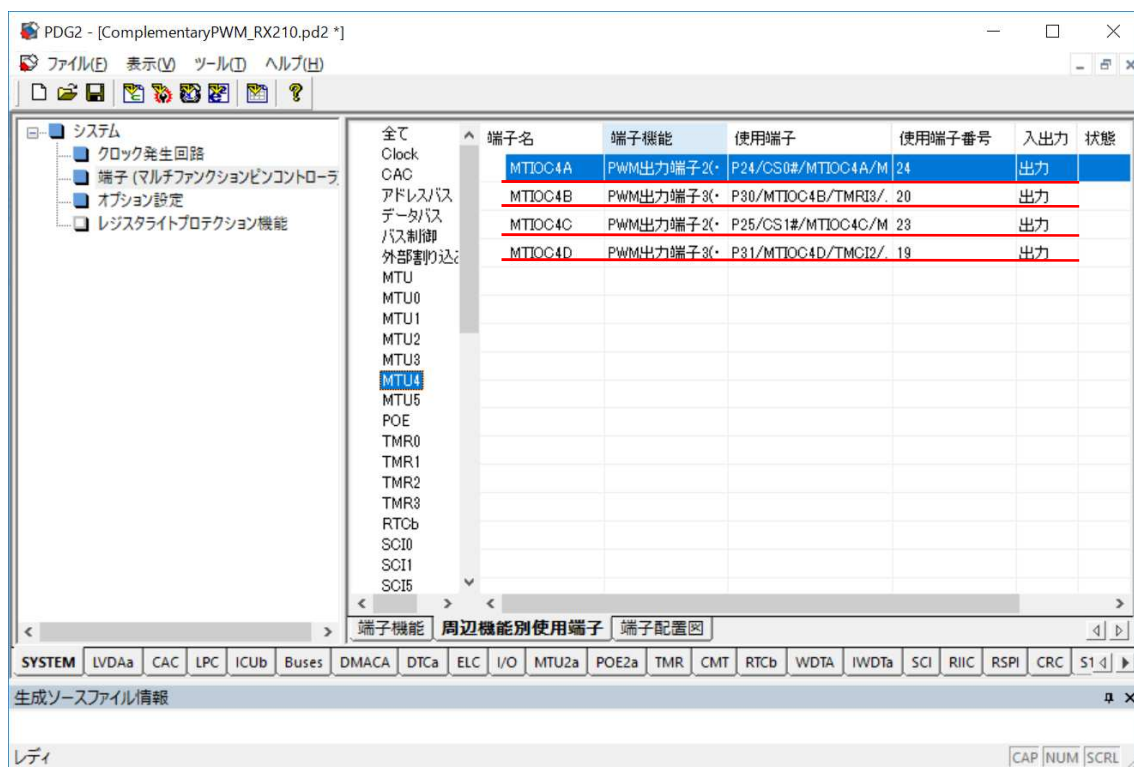
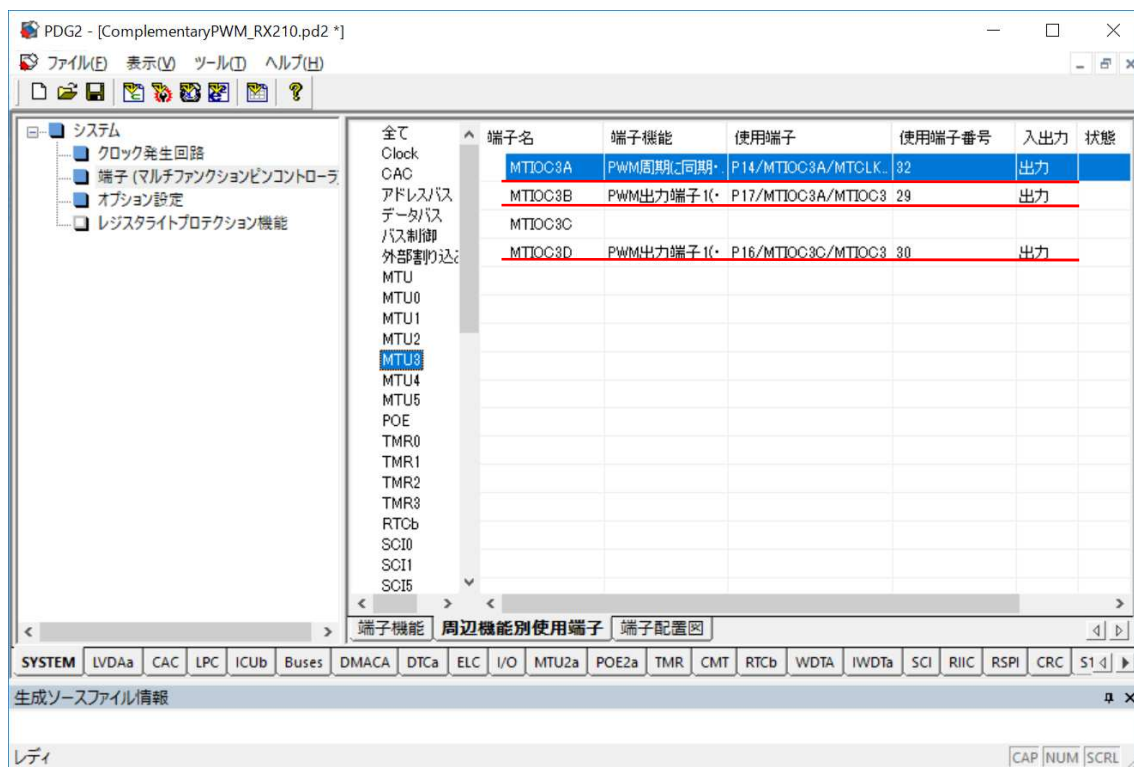


割り込み関数の設定を行います。「MTU3/TGRA コンペアマッチ(カウンタ値の山)割り込み(TGIAn)を使用する」にチェックを入れます。割り込み通知関数名は任意に設定してください。



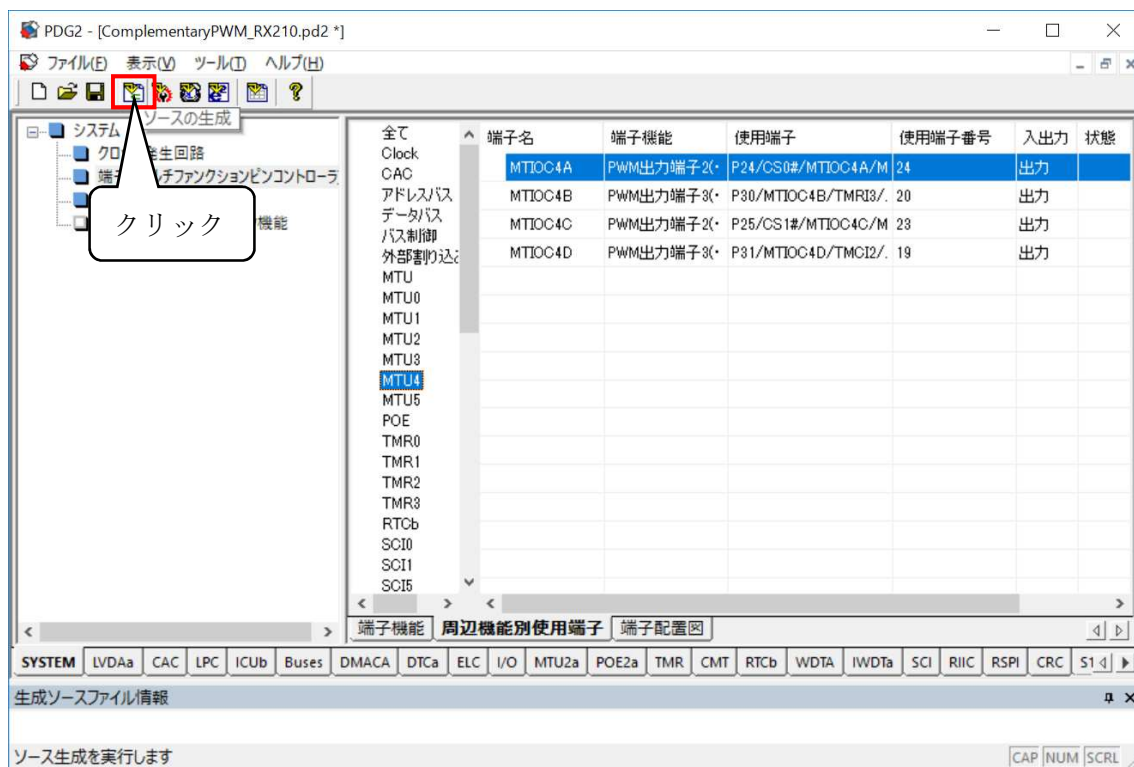
5.3 SYSTEM の端子設定

SYSTEM の端子設定を確認します。

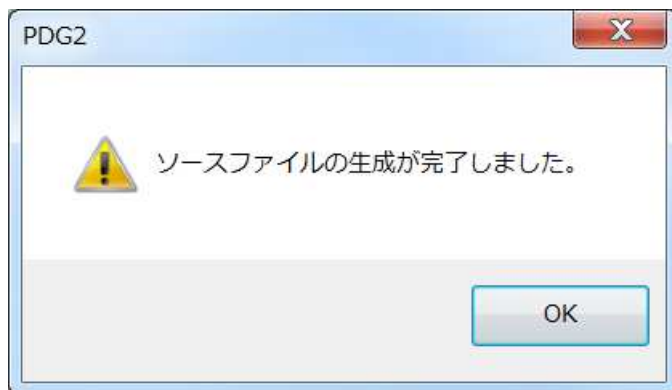


5.4 ソースの生成

以下の GUI をクリックすると、

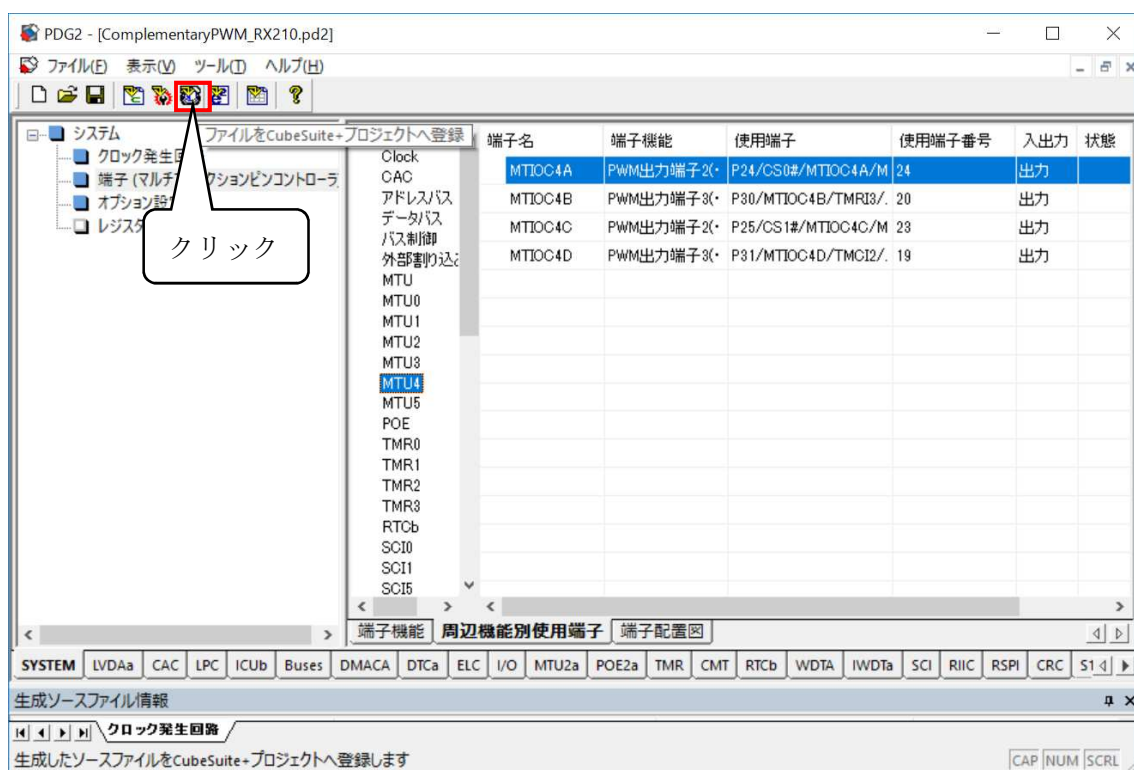


ソースファイルが生成されます。

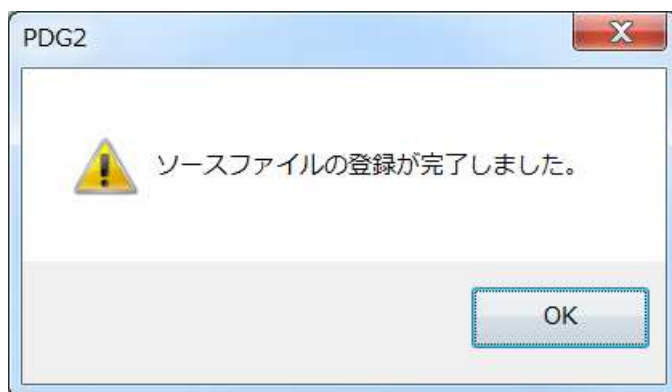


5.5 CS+への登録

対象のCS+プロジェクトを開き、PDG 上の以下の GUI をクリックします

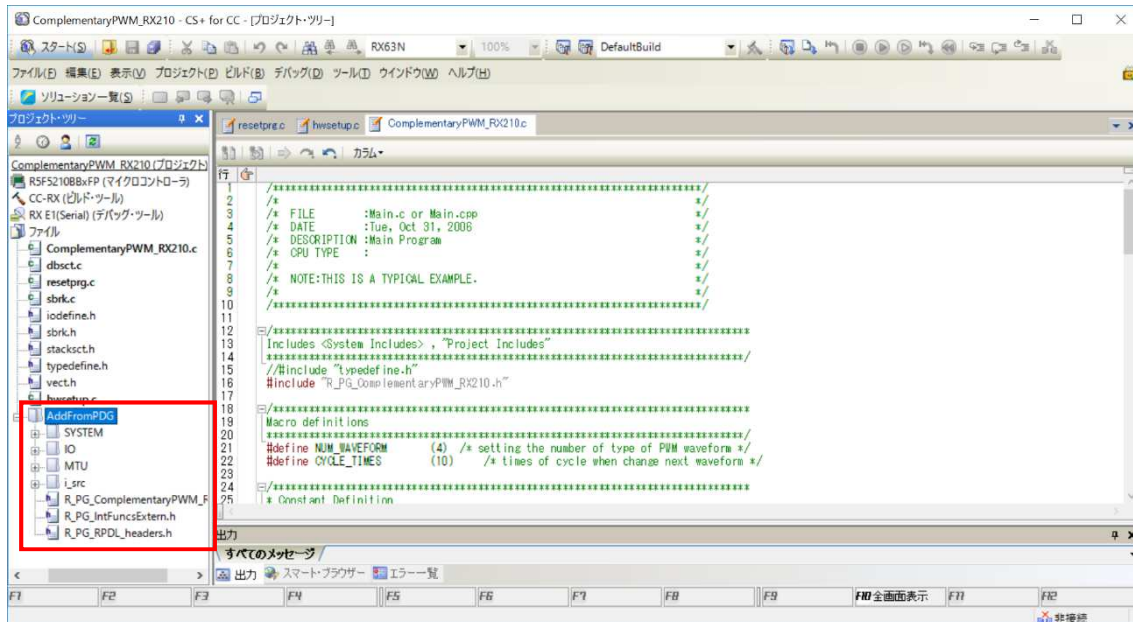


ソースファイルの登録が完了しました。



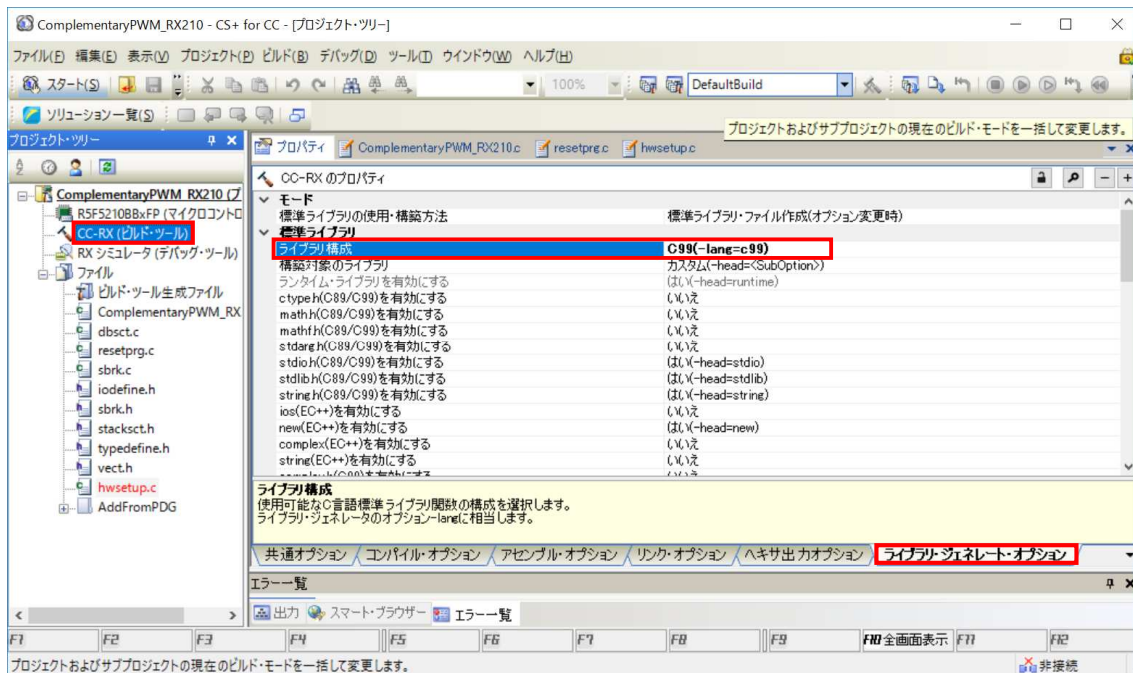
6. CS+のプロジェクトにPDGのソースファイルを登録する際の設定

CS+のプロジェクトにPDGで生成されたソースファイルを登録すると、プロジェクトのファイルにAddFromPDGフォルダが追加されます。

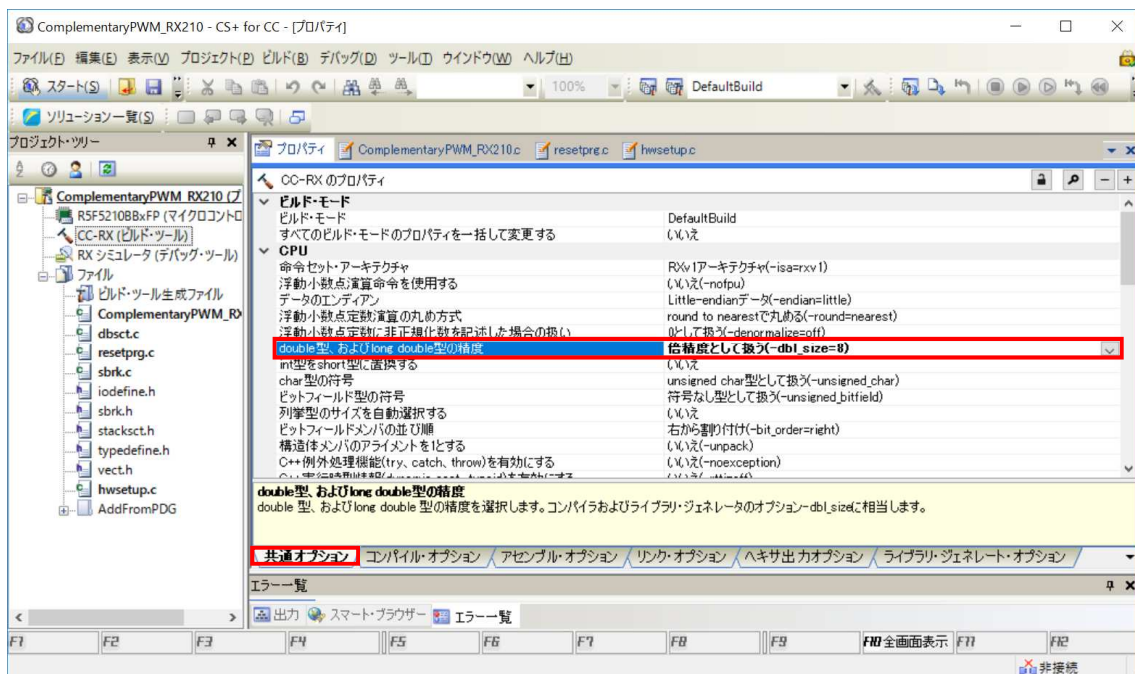


そのままビルドをすると、エラーおよび警告が発生します。解消する設定を以下に示します。

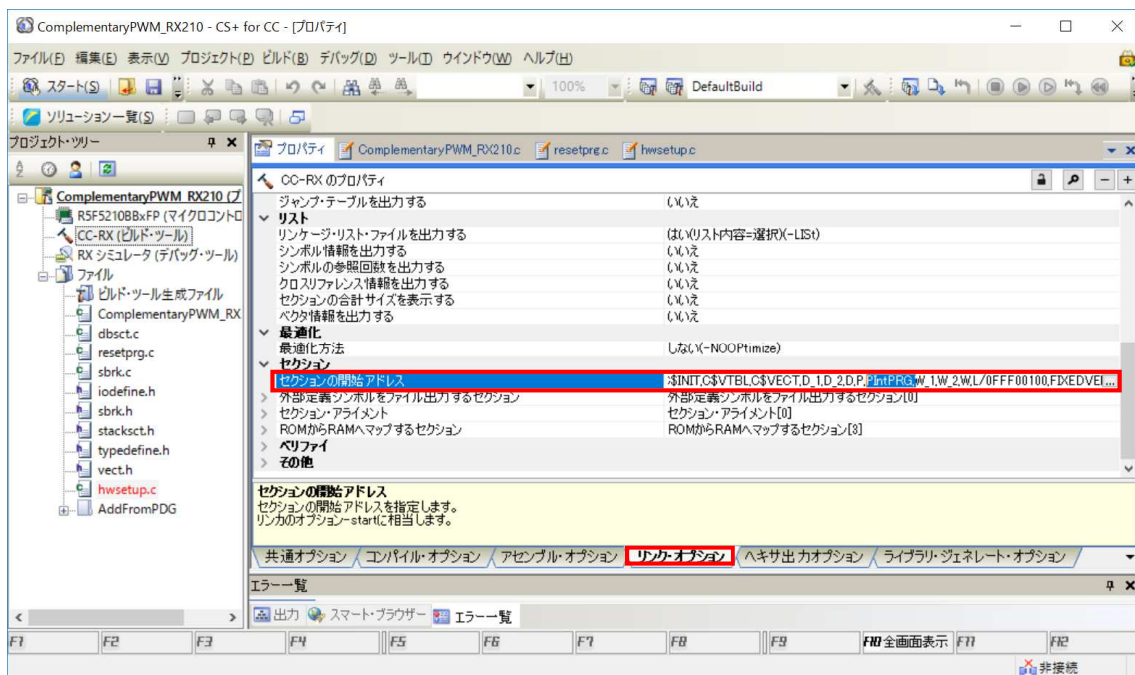
PDGで生成されるソースファイルはbool変数を使用しています。対応させるため、ビルド・ツールを右クリック→プロパティを表示し、ライブラリ・ジェネレート・オプションタブにある「ライブラリ構成」を”C99(-lang=c99)”に設定します。



PDG で生成されるソースファイルは double 型、および long double 型の精度を倍精度として扱っているため、ビルド・ツールを右クリック→プロパティを表示し、共通オプションタブにある「double 型、および long double 型の精度」を”倍精度として扱う(-dbl_size=8)”に設定します。



DG で生成されるソースファイルを登録すると PIntPRG セクションを使用しないため、CS+プロジェクトを生成した際にデフォルトで設定されている PIntPRG セクションを削除します。ビルド・ツールを右クリック→プロパティを表示し、リンクオプションタブにある「セクションの開始アドレス」から”PIntPRG”を削除します。

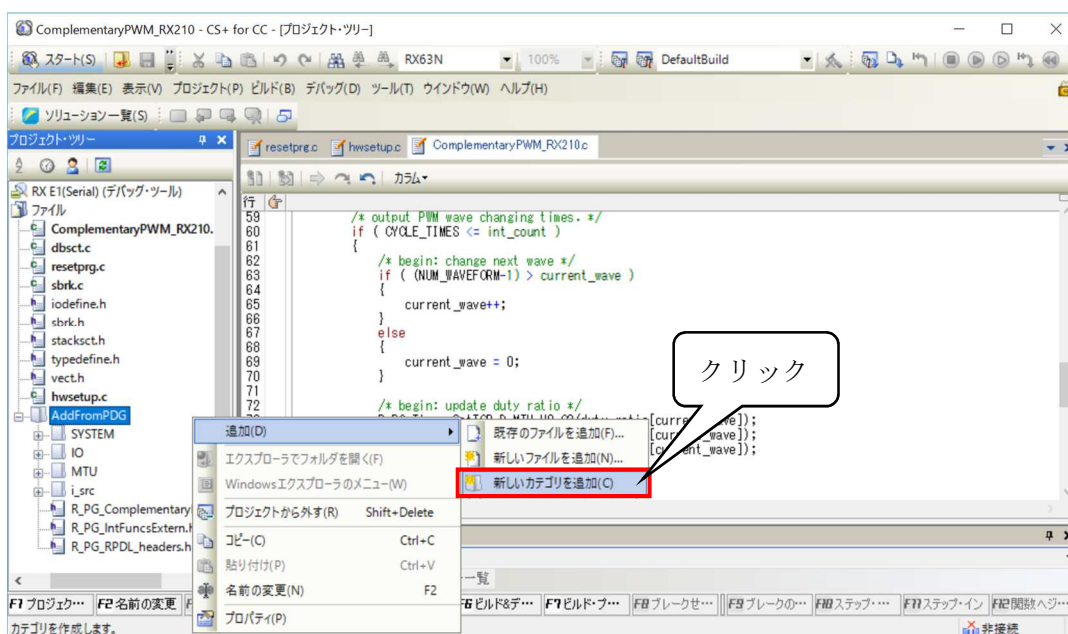


7. デバッグについて

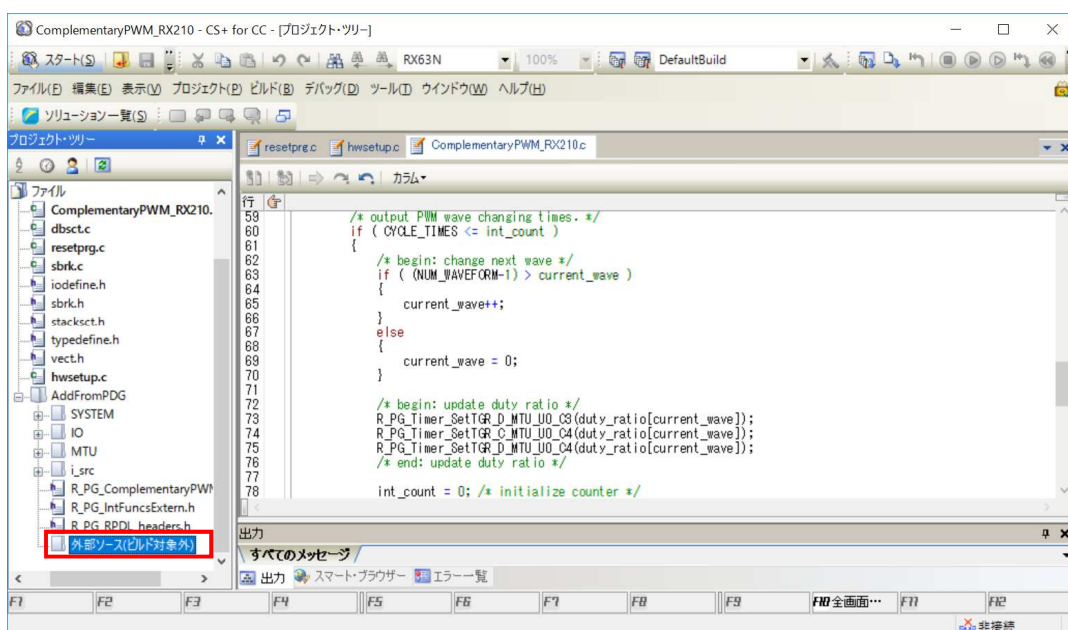
PDG2 が生成する関数が呼び出すライブラリの内部処理をデバッグする方法の一例を示します。

7.1 ライブラリの内部処理をデバッグする準備

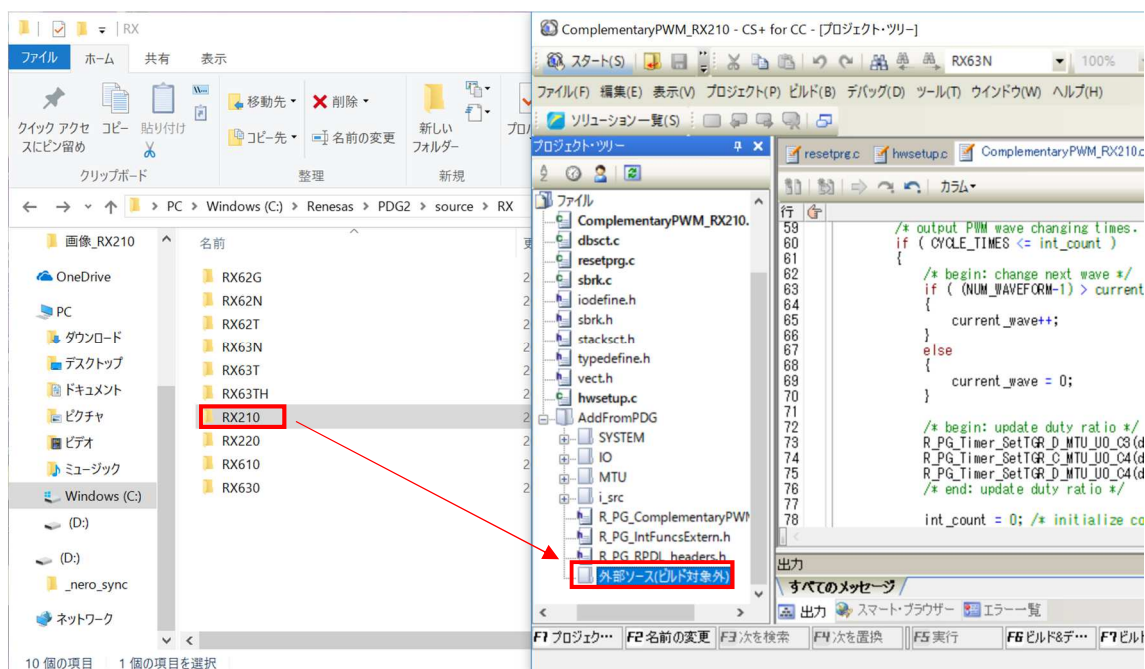
CS+の左部ツリーの「ファイル」以下の部分で左クリックし、新しいカテゴリを追加します。例ではAddFromPDGの下に新しいカテゴリを追加します。



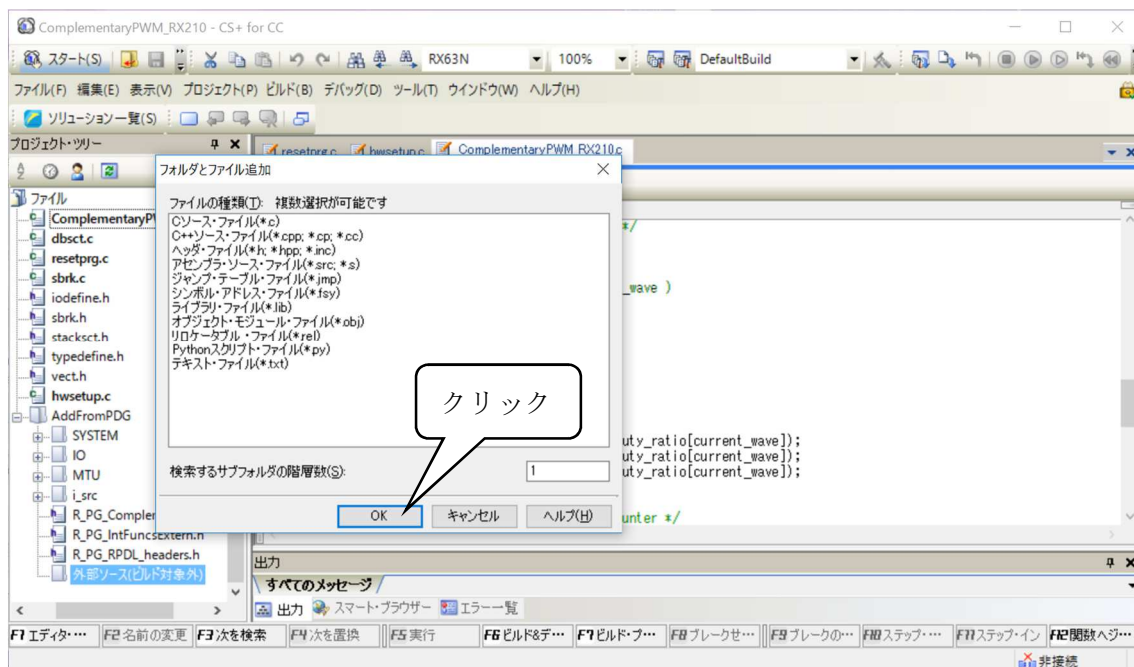
新しいカテゴリに適切な名前をつけます。



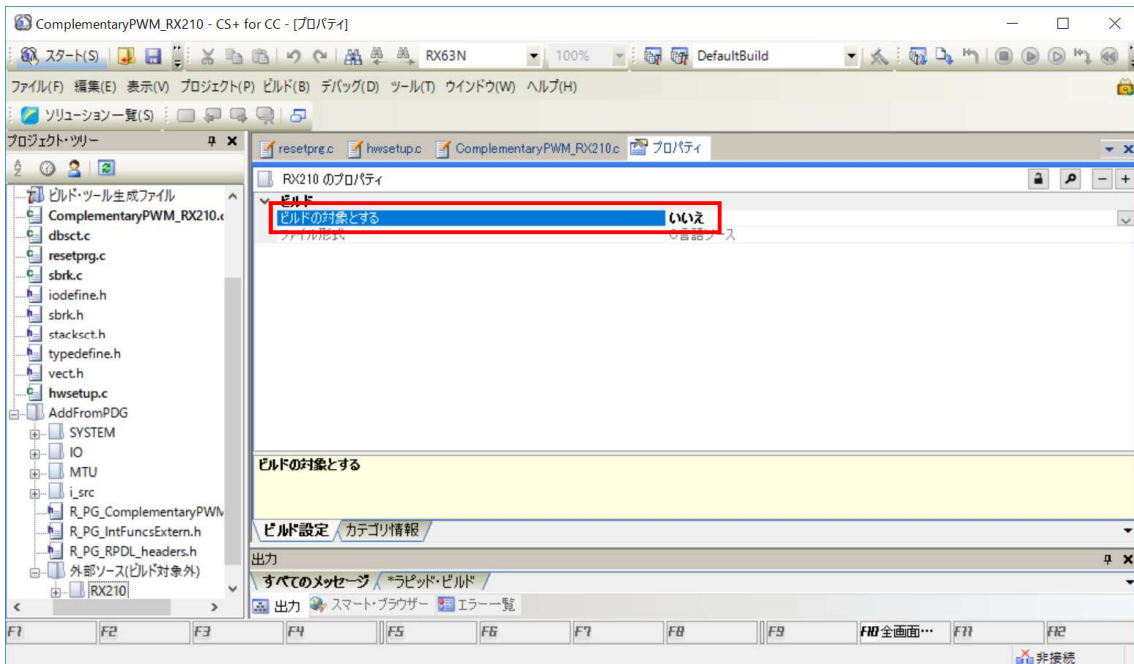
「C:\¥Renesas¥PDG2¥source¥RX」 下にあるフォルダの中から、現在作成中のマイコンに対応するフォルダを選択し、先ほど作成した新しいカテゴリにドラッグします。



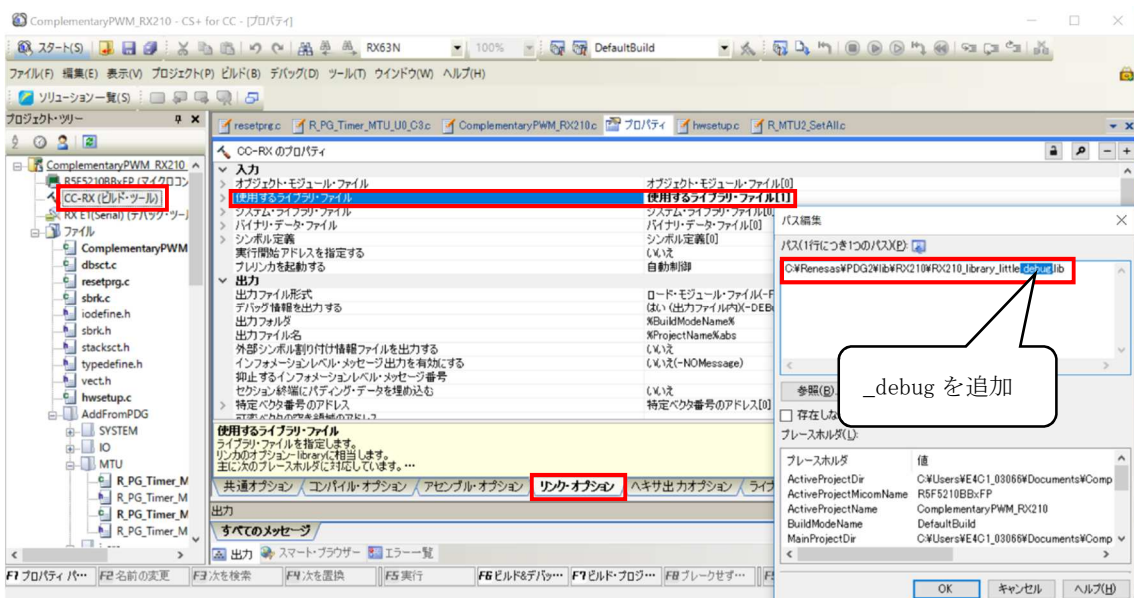
追加するファイルを問われるので「OK」を選択します。



ファイルが追加されるのを確認したあと、追加したフォルダのプロパティを開きます。
ビルド設定のタブから、「ビルドの対象とする」を「いいえ」に変更します。



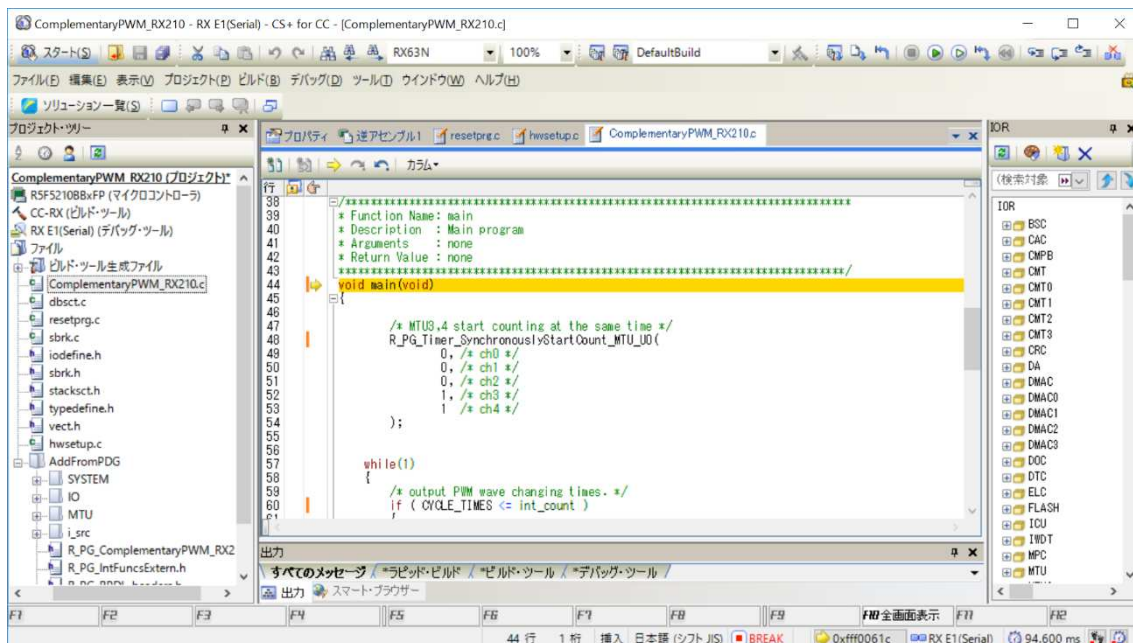
ビルド・ツールを右クリック->プロパティを表示し、リンク・オプションタブにある「使用するライブラリ・ファイル」を開き、現在指定されているファイルの拡張子の直前に「_debug」を追加します。



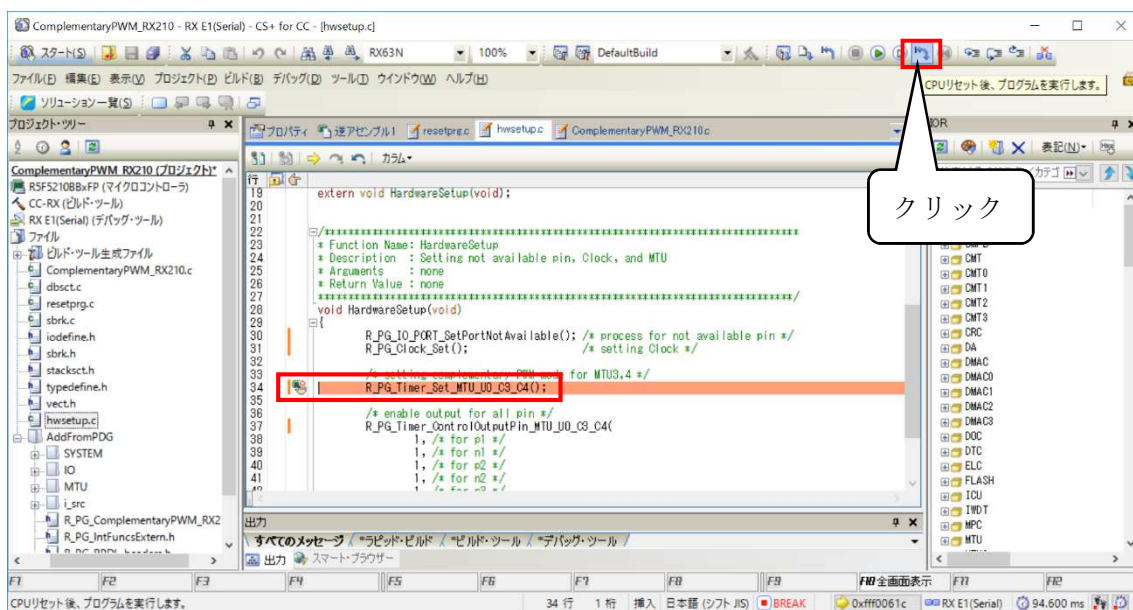
以上の設定を行うことで、PDG2 が生成した関数が呼び出すライブラリの内部処理を追うことができます。

7.2 デバッグ

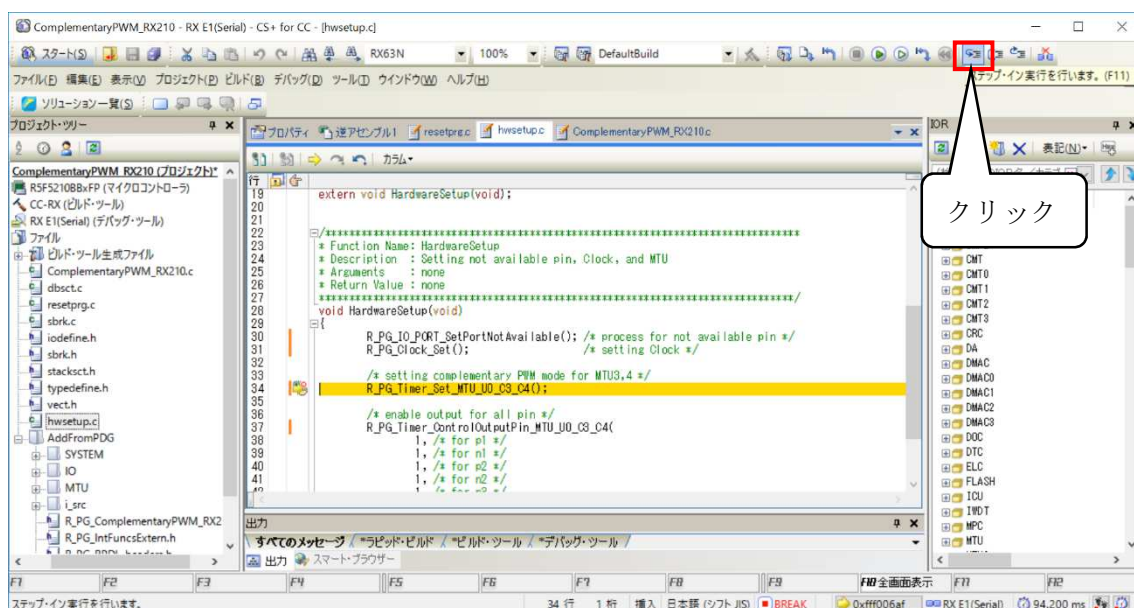
実際にデバッグしてみます。デバッグツールにプログラムをダウンロードすると、デフォルトの設定ではmain関数の開始でブレークされます。



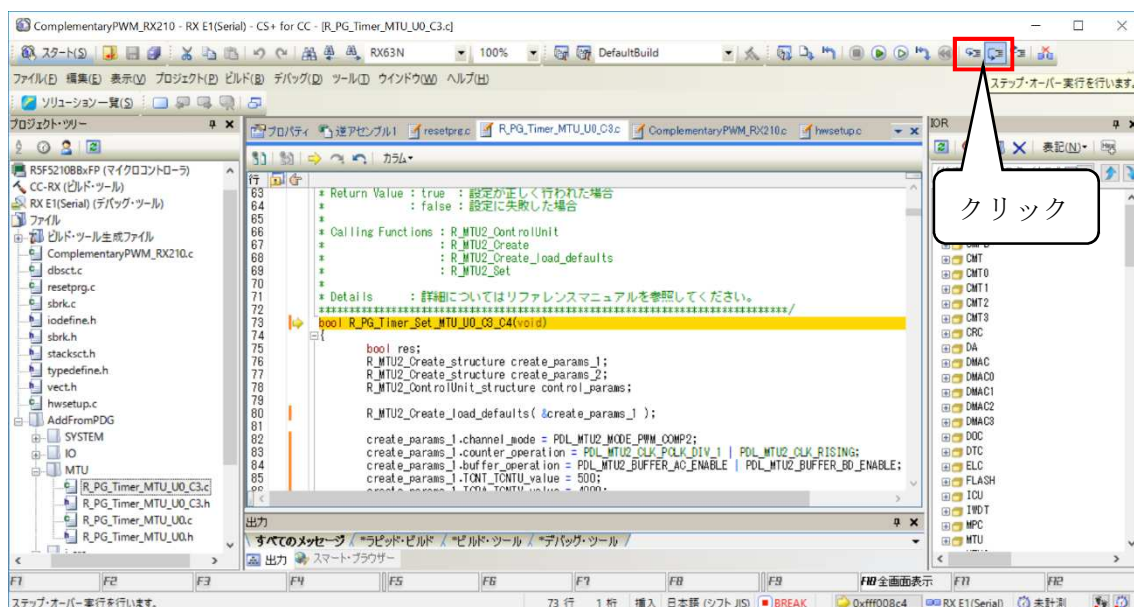
ここではhwsetup.cに記述した、PDGが生成した関数P_PG_Timer_Set_MTU_U0_C3_C4の動作を追ってみます。関数の呼び出し部分にブレークポイントを設定し、「CPUリセット後、プログラムを実行します。」をクリックします。



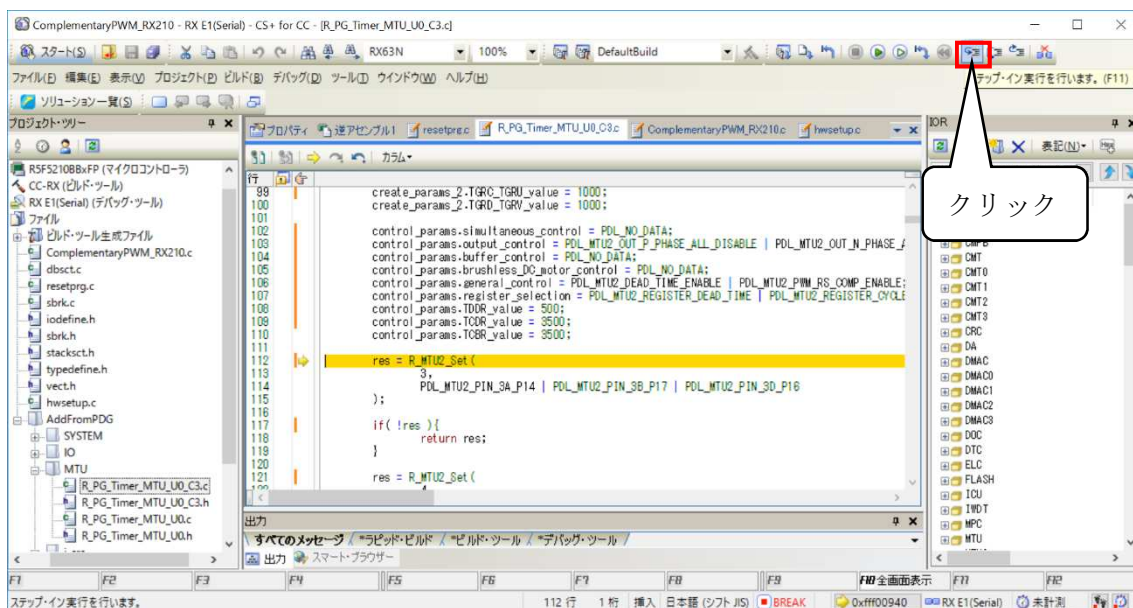
先ほど設定したブレークポイントの行を実行する直前で停止しました。「ステップ・イン実行を行います。」をクリックし、関数内の処理を追っていきます。



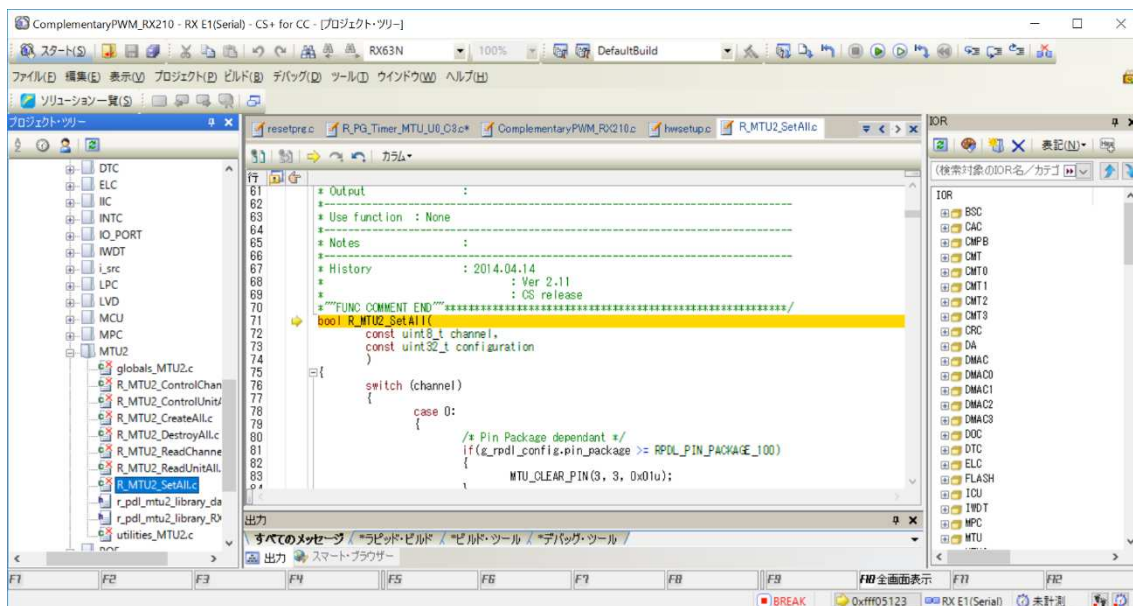
関数 P_PG_Timer_Set_MTU_U0_C3_C4 の内部に入りました。ステップ・イン、ステップ・オーバーを駆使して、関数の動作を追っていきます。



関数内部でさらに、関数 R_MTU2_Set を呼んでいます。ステップ・インして内部の処理を追っていきます。



関数 R_MTU2_SetAll の内部に移動しました。この関数は先ほど追加したライブラリのファイルです。呼び出した関数名の末尾に「All」が加えられた関数が追加されたライブラリの関数です。



8. 動作確認方法

作成したソフトウェアをボードにダウンロードし、出力端子をオシロスコープで観測することで確認しました。出力された波形は、4.1.1 および 4.1.2 に示した波形と同一のものが確認されました。

9. 参考ドキュメント

RX210 グループ ユーザーズマニュアルハードウェア編

RX210 グループ Peripheral Driver Generator リファレンスマニュアル

以上