お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

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

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、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に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。お客様が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。
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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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その総株主の議決権の過半数を直接または間接に保有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。

User's Manual

Peripheral Driver Generator V.1.01.000

ガイドブック

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2007.01

安全設計に関するお願い

- 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

- 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズは責任を負いません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズは、予告なしに、本資料に記載した製品又は仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前に株式会社ルネサス テクノロジ、株式会社ルネサス ソリューションズ、株式会社ルネサス販売又は特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ（http://www.renesas.com）などを通じて公開される情報に常にご注意ください。
- 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズはその責任を負いません。
- 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズは、適用可否に対する責任は負いません。
- 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、株式会社ルネサス テクノロジ、株式会社ルネサス ソリューションズ、株式会社ルネサス販売又は特約店へご照会ください。
- 本資料の転載、複製については、文書による株式会社ルネサス テクノロジおよび株式会社ルネサス ソリューションズの事前の承諾が必要です。
- 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたら株式会社ルネサス テクノロジ、株式会社ルネサス ソリューションズ、株式会社ルネサス販売又は特約店までご照会ください。

製品の内容及び本書についてのお問い合わせ先

株式会社ルネサス テクノロジ

ホームページ http://japan.renesas.com/tools

本ガイドブックでは、Peripheral Driver Generator を使用頂く際の High-performance Embedded Workshop への組み込み方法などを説明します。

目次

1. パッケージとインストール

1.1. インストール

インストーラ起動後、インストーラの手順に従ってインストールしてください。

インストールが完了すると、スタートメニュー ->[Renesas]に[Peripheral Driver Generator]が表示されます。
[Peripheral Driver Generator]には、

- サンプルプロジェクト
- Peripheral Driver Generator
- API リファレンスマニュアル
- Peripheral Driver Generator ユーザーズマニュアル
- Peripheral Driver Generator ガイドブック(本 PDF ファイル)

が登録されます。

※ インストールは、Administrator 権限で行ってください。

1.2. パッケージ

インストールするファイル及びプログラムは、以下の通りです。

ディレクトリ		ファイル名	説明
chipDll		H8_3687.dll,H8_36049.dll,H8_36077.dll,H8_36109.dll	デバイス毎のデータファイル
		M16C_28.dll,M16C_28B.dll,M16C_29.dll	
		R8C_13.dll,R8C_24.dll,R8C_25.dll	
lib	h8_3687	rapi_h83687.lib	APIライブラリファイル
	m16c_28	rapi_m16c28.lib	
	r8c_13	rapi_r8c13.lib	
i_src	h8_3687	割り込み用Cソースファイル	
	m16c_28	割り込み用Cソースファイル	
	r8c_13	割り込み用Cソースファイル	
manual		RJJ10J1644_pdg_u.pdf	PDGユーザーズマニュアル
		RJJ10J1651_pdg_g.pdf	PDGガイドブック
		RJJ10J1643_rapi_r.pdf	APIリファレンスマニュアル
source		APIライブラリソース及びライブラリ再構築用プロジェクト	
SrcGenerator	ChipSrcDll	H8_3687_Src.dll,H8_36049_Src.dll,H8_36077_Src.dll	デバイス毎のデータファイル
		H8_36109_Src.dll	
		M16C_28_Src.dll,M16C_29_Src.dll	
		R8C_13_Src.dll,R8C_24_Src.dll	
		SrcGenerator.dll	ソース出力用管理データファイル
startup_files	h8_3687	H8/3687用スタートアッププログラム	
	m16c_28	M16C/28用スタートアッププログラム	
	r8c_13	R8C/13用スタートアッププログラム	
sample	h8_3687	H8/3687用High-performance Embedded Workshop用空ワークスペースサンプル及びPeripheral Driver Generator用空サンプルプロジェクト	
	m16c_28	M16C/28用High-performance Embedded Workshop用空ワークスペースサンプル及びPeripheral Driver Generator用空サンプルプロジェクト	
	r8c_13	R8C/13用High-performance Embedded Workshop用空ワークスペースサンプル及びPeripheral Driver Generator用空サンプルプロジェクト	
その他		PDG.exe	PDG本体
		PDG.hrf	HEW登録用テキストファイル
		projConv.exe	コンバート変換用エンジン
		license.txt	使用権許諾契約書

注意）

Peripheral Driver Generator をインストール際には、予め High-performance Embedded Workshop がインストール済みである必要があります。

Peripheral Driver Generator からのソースファイルの登録などは High-performance Embedded Workshop の HewTargetServer を使用します。

そのため Peripheral Driver Generator をインストールすると自動的に HewTargetServer を有効にします。

Peripheral Driver Generator を先にインストールした場合は、High-performance Embedded Workshop インスト

ール後、HewTargetServer を有効にしてください。

HewTargetServer を有効にする方法は、「9.High-performance Embedded Workshop への登録」を参照ください

2. Peripheral Driver Generator を使用した開発の流れ概要

本章では、Peripheral Driver Generator を使用してアプリケーションを開発する際の概要を説明します。
Peripheral Driver Generator は、選択された周辺及び設定内容を盛り込んだ関数群を C ソースファイルで生成します。
Peripheral Driver Generator が生成した関数を呼び出して行く事で、アプリケーションを開発します。
基本的には、以下の流れで開発を行います。

(1) 周辺の設定

Peripheral Driver Generator を使用してプロジェクトを作成します。
CPU グループ、使用する周辺を選択してファイルを生成します。
「4.1　プロジェクトの作成」　を参照

(2) ビルド/デバッグなどの環境作成

High-performance Embedded Workshop を用いてアプリケーションのワークスペースを作成します。
新規ワークスペース作成などで開発するアプリケーション用のワークスペースを作成します。
Periphera Driver Generator をインストールしたディレクトリ下の sample ディレクトリに M16C/28 用、H8/3687 用、R8C/13 用それぞれのサンプルワークスペースを用意しています。

(3) アプリケーションの作成

Peripheral Driver Generator で生成した関数を呼び出します。
Peripheral Drive Generator で生成した関数をアプリケーションから**適所で呼び出します。**
例えば、タイマ A0 を用いたタイマモードの初期設定を行う場合、
_CreateTimer_TA0_p1( );
を呼び出します。
また同時に Peripheral Driver Generator が生成したヘッダファイルをインクルードしておく必要があります。
生成ヘッダファイル名：プロジェクト名.h

(4) Peripheral Driver Generator で生成したソースファイルの登録

Peripheral Driver Generator が生成した関数を呼び出すだけでは、ビルド(リンク)時にエラーとなります。
関数実態が格納されている C ソースファイルを High-performace Embedded Workshop で作業しているワークスペースへ登録する必要があります。

(5) ビルド

ビルドに必要なオプションを **High-performance Embedded Workshop** のビルドオプションへ登録します。
設定するオプションは、

- リンクオプション-L などによる API ライブラリの指定
- コンパイルオプション-I によるインクルードファイルのパス設定

添付のサンプルワークスペースでは、上記オプションは設定済みです。

サンプルワークスペースを使用しない場合に必要となります。

(6) 完了

４章以降でそれぞれ詳細を説明します。

3. 今すぐ使ってみる

本章では、Peripheral Driver Generator のサンプルプロジェクトと High-performance Embedded Workspace のサンプルワークスペースを使って、オブジェクト作成までの手順を説明します。

なお、各項横の[　]内が Peripheral Driver Generator と記載されている場合は、Peripheral Driver Generator の操作説明、High-performance Embedded Workshop と記載されている場合は、High-performance Embedded Workshop の操作説明を意味しています。

※本サンプルワークプロジェクトの雛形は、Peripheral Driver Generator インストールディレクトリ下の sample.bak にバックアップ用があります。（元の状態に戻す場合は、sample.bak 下ディレクトリを sample ディレクトリへコピーしてください。）

3.1. High-performance Embedded Workshop でワークスペース作成 [High-performance Embedded Workshop]

Peripheral Driver Generator パッケージで用意している High-performance Workshop 用空ワークスペースを使用します。

スタートメニュー　→　[Renesas] → [Peripheral Driver Generator] → [サンプルプロジェクト]　から

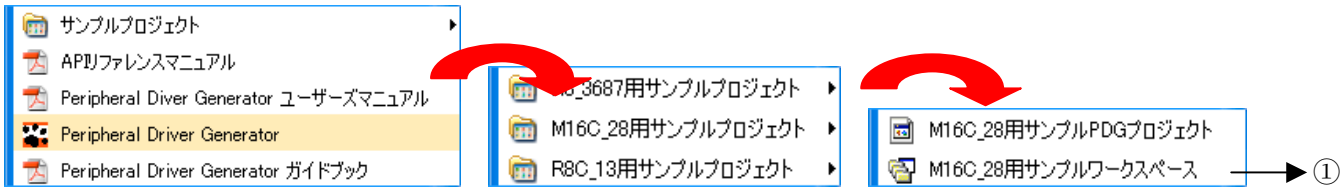
H8/3687 用サンプルワークスペース

M16C/28 用サンプルワークスペース

R8C/13 用サンプルワークスペース

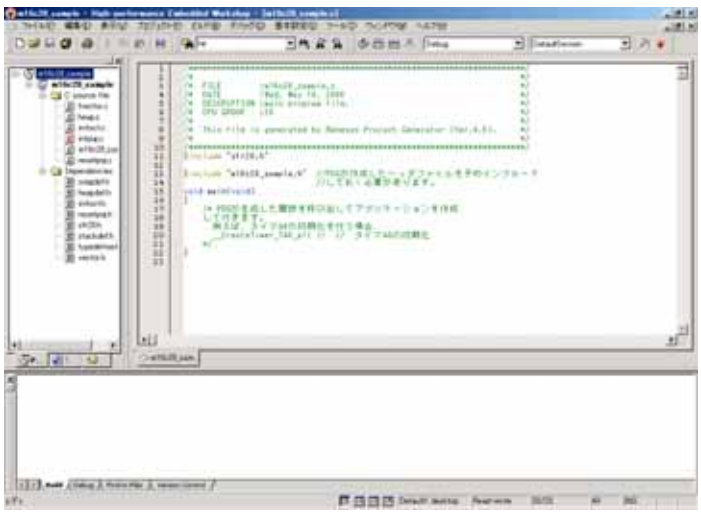
のいずれかをオープンします。

ここでは、M16C/28 用のサンプルワークスペースを使用します。



①M16C/28 用の High-performance Embedded Workshop サンプルワークスペースです。

①　を選択すると、High-performance Embedded Workshop が立ち上がります。



High-performance Embedded Workshop は立ち上げたまま次に進んでください。

3.2. Peripheral Driver Generator でプロジェクト作成 [Peripheral Driver Generator]

Peripheral Driver Generator パッケージで用意している Peripheral Driver Generator 用空プロジェクトを使用します。

スタートメニュー → [Renesas] → [Peripheral Driver Generator] → [サンプルプロジェクト] から

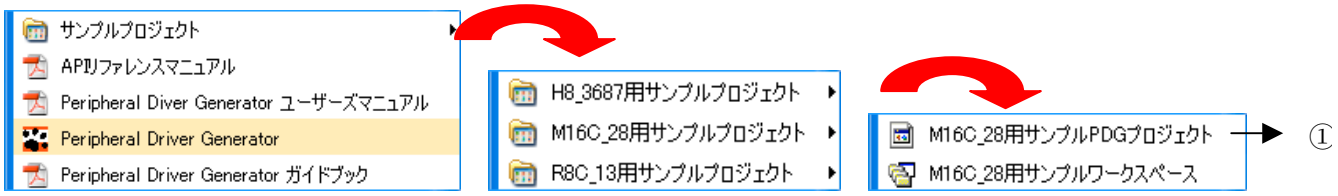
H8/3687 用サンプル PDG プロジェクト

M16C/28 用サンプル PDG プロジェクト

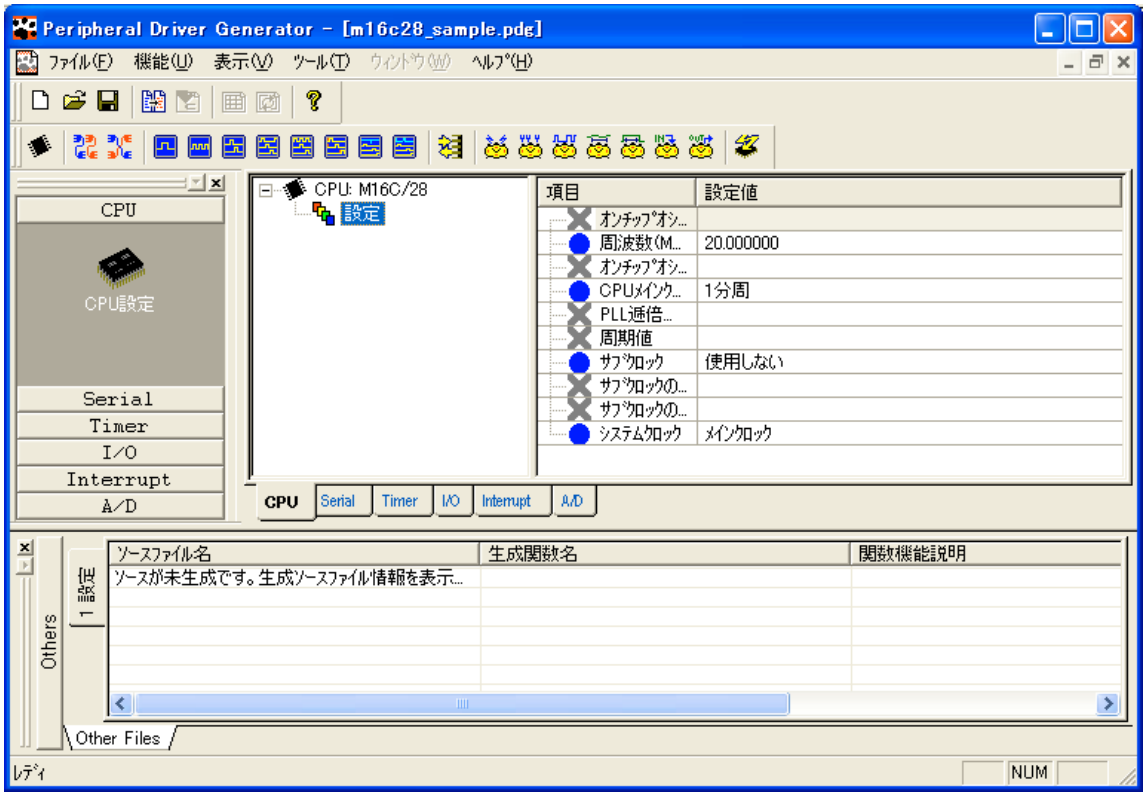
R8C/13 用サンプル PDG プロジェクト

のいずれかをオープンします。

ここでは、M16C/28 用サンプル PDG プロジェクトを使用します。



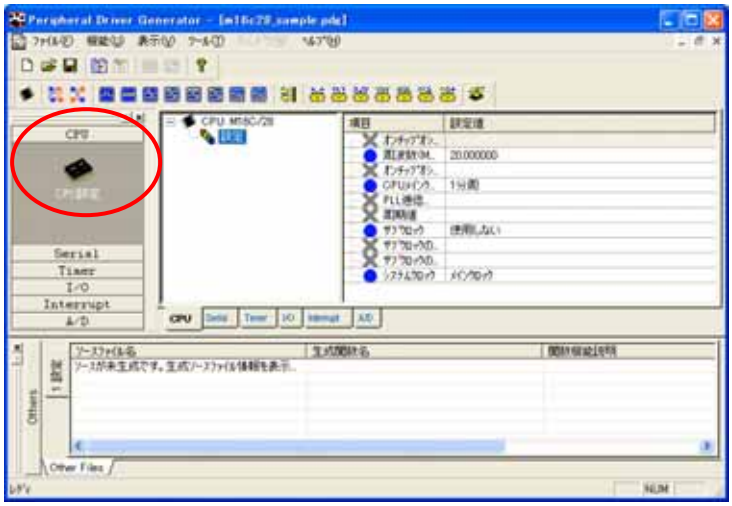
① を選択すると、Peripheral Driver Generator が立ち上がります。

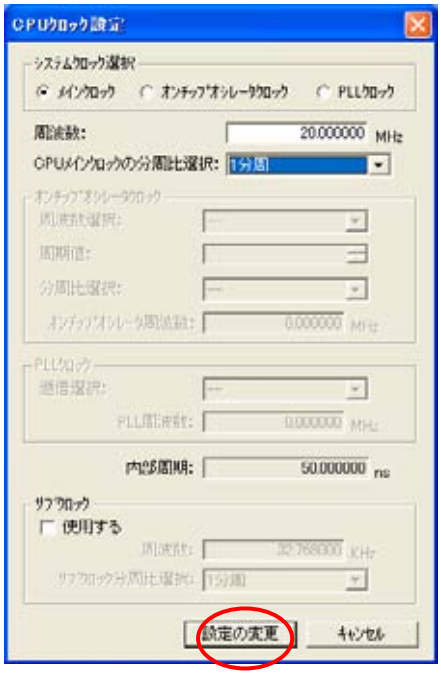


3.3. 周辺の設定 [Peripheral Driver Generator]

まず、**最初に行うのは CPU クロックの設定**です。

周辺 IO 選択ウィンドウから **CPU 設定**を選択します。





“ CPU クロック設定 ” ダイアログが立ち上がりますので、作成するプログラムに応じて各設定を行い、“ 設定の変更 ” ボタンを押して設定を終了します。

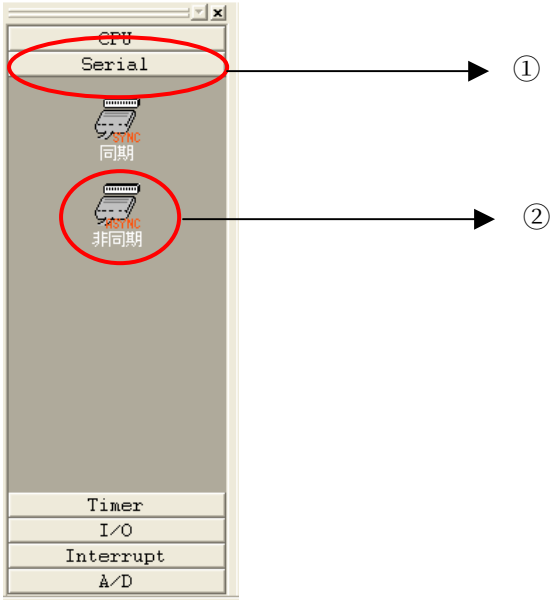
次に行うのは、周辺の設定です。

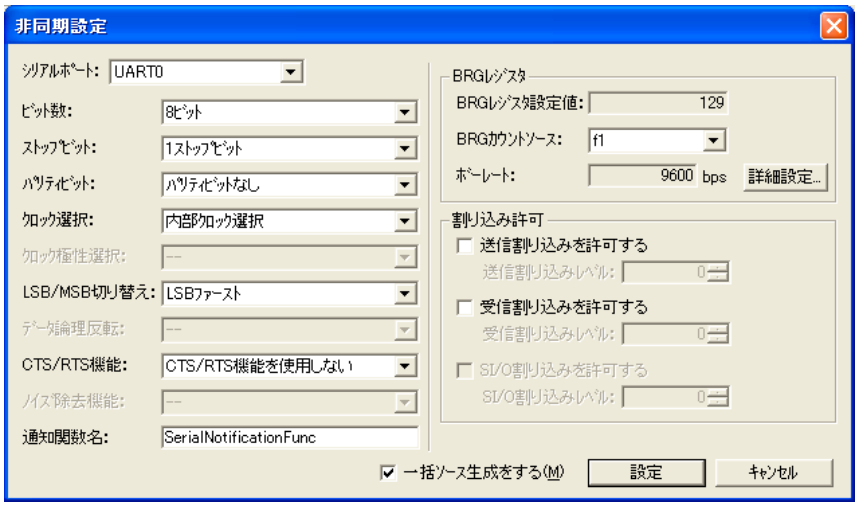
作成するプログラムに応じて、まず**周辺を選択し、その中から機能**を選びます。

例えば、非同期のシリアル通信プログラムを作成する場合は、

② まず、周辺設定ウィンドウから”Serial”タブを選択します。

③ 次に、“ 非同期 “ を選択します。





非同期設定ダイアログが開きますので、各項目（ビット数、パリティビット有無、転送ボーレートなど）を設定し、最後にシリアルポート（UART0,UART1 など）を決定して入力は完了です。

□一括ソース生成をする(M)をチェックしている状態で設定ボタンを押します。

これで設定した内容で C ソースの生成を行います。

※　シリアルポートが設定されていない状態では、C ソースファイルを生成することはできません。

3.4. 出力 C ソースファイルを登録　[Peripheral Driver Generator]

“ 3.4 周辺の設定 “ で生成した C ソースを High-performance Embedded Workshop へ登録する必要があります。この作業は、コンパイル/リンクするために必要な作業です。

登録は、Peipheral Driver Generator のツールメニュー “ ファイルを HEW プロジェクトへ登録 ” を選択することで行います。

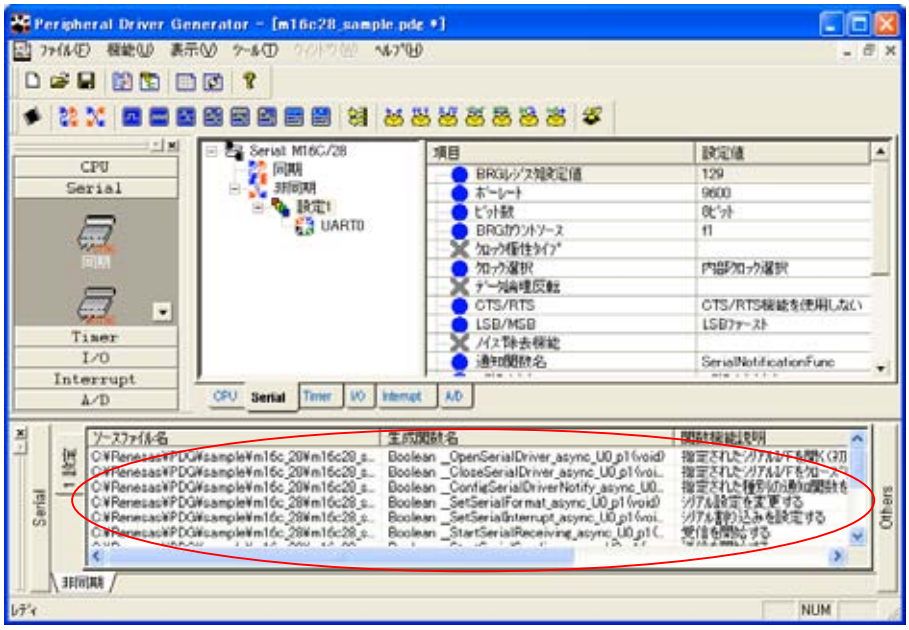
注意）

High-performance Embedded Workshop のワークスペースへ Peripheral Driver Generator が生成したソースファイルを登録すると、Peripheral Driver Generator からファイル登録を解除することができません。
登録を解除する場合は、High-performance Embedded Workshop 側から解除してください。
High-performance Embedded Workshop でのファイル登録の削除は、「5.3 登 録したソースファイルの削除」を参照ください。

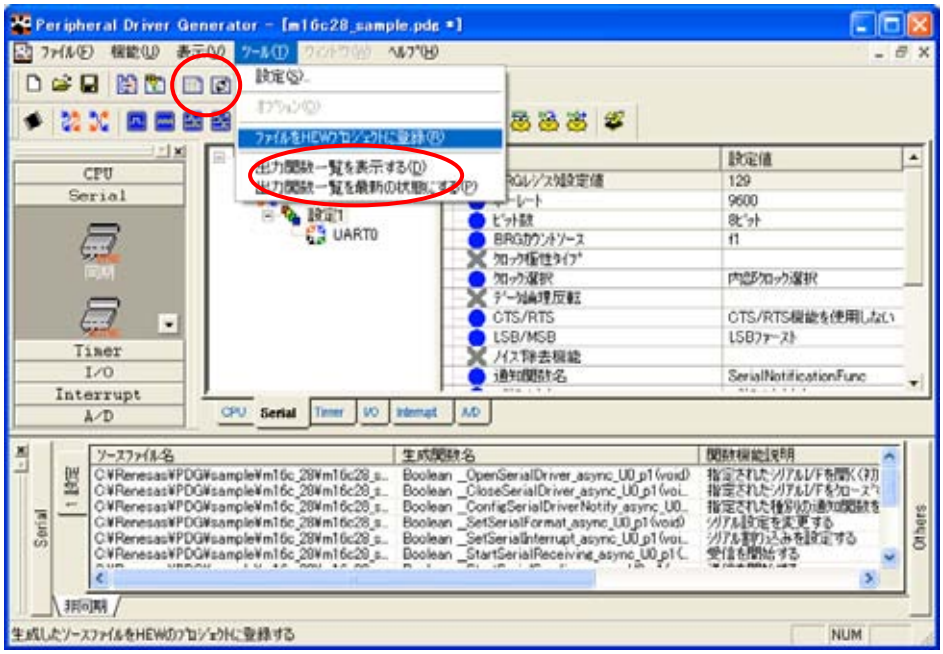
High-performance Embedded Workshop のワークスペース作成時に選択した CPU グループと Peripheral Driver のプロジェクト作成時に選択した CPU グループが同じであるか確認の上、ソースファイルを登録してくだ
さい。

3.5. 作成した関数の確認 [Peripheral Driver Generator]

“ 3.4 周辺の設定 “ で作成した関数を呼びだすことでアプリケーションを作成していきます。
使用可能な関数は、ソース出力ウィンドウで確認することができます。



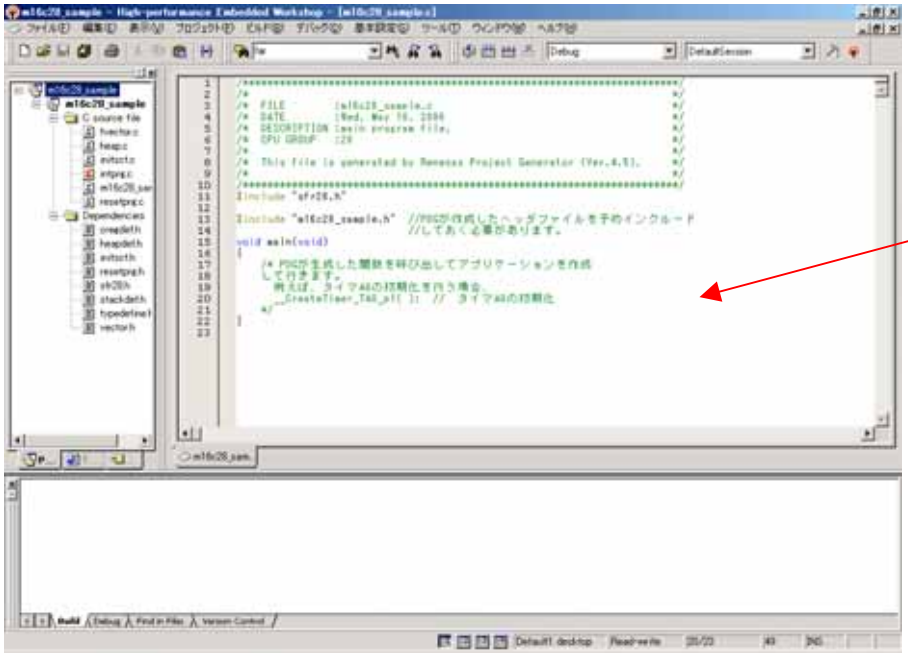
ツールメニューの“出力関数一覧を表示する”を選択もしくはボタンを押すことで、csv 形式のファイルで出力関数一覧を見ることができます。拡張子.csv が関連付けられたアプリケーションが起動します。



出力関数一覧表示メニュー及びボタン

3.6. アプリケーションの作成 [High-performance Embedded Workshop]

アプリケーションの作成は、High-performance Embedded workshop を使用して行います。



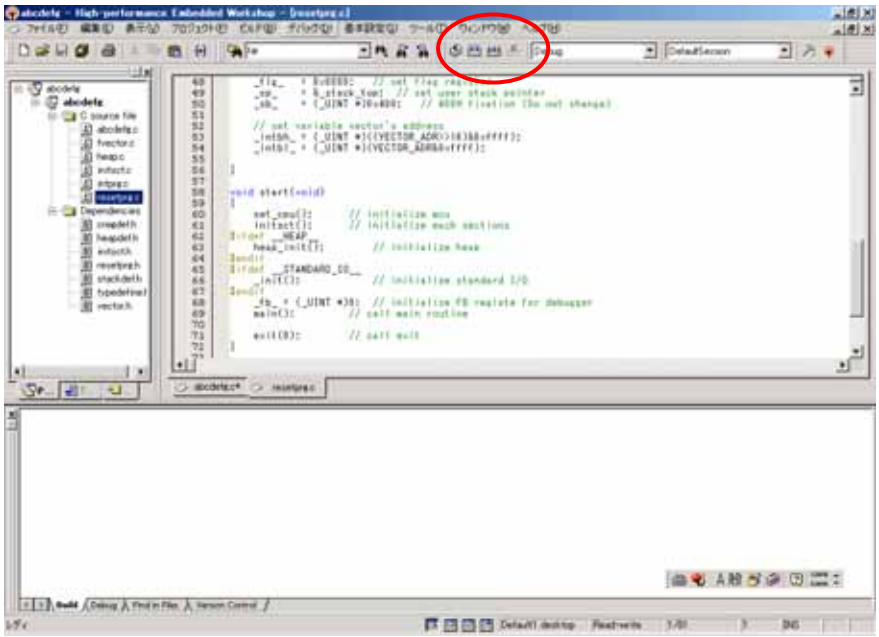
ここにプログラムを
記述していきます。

例えば、シリアルポート UART0 にデータを送信する場合、
__StartSerialSending_async_U0_p1( )関数をコールします。

```
if( __StartSerialSending_async_U0_p1(15,(unsigned int *)"hello PDG World") == 0)
        printf("False¥n"); //送信失敗
```

3.7. コンパイル/リンク [High-performance Embedded Workshop]

アプリケーションの作成が終了するとビルド作業を行います。ビルドは High-performance Embedded Workshop のビルドボタンを押すことで行います。



※High-performance Embedded Workshop が C ドライブ以外にインストールされている場合は、ビルドを行うとリンク時にエラーが発生します。
この場合は、リンクオプションで API が存在するディレクトリ（Peripheral Driver Generator をインストールしたディレクトリ下の lib¥m16c_28,lib¥h8_3687,lib¥r8c_13 のいずれか）を指定してください。
API ライブラリの指定方法については、「6.3 ライブラリの指定」を参照してください。

これで全ての作業は完了です。

3.8. 実行 [High-performance Embedded Workshop]

ここまでの作業を行って、シミュレータを使用して動作確認する場合について説明します。

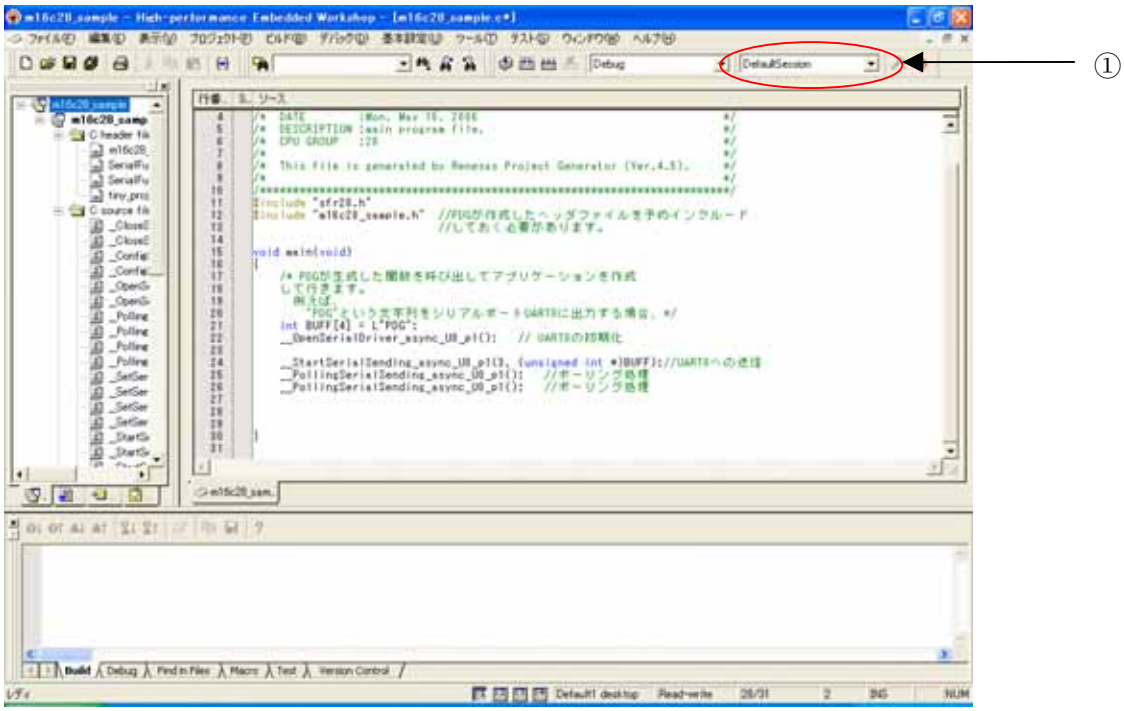
実行するには、前準備として、ここまで説明してきた以下の処理が必要です。
1. サンプルワークスペースの main 関数中に記載している以下のプログラムについてコメントを外して有効にする。

```
int BUFF[4] = L"PDG";
__OpenSerialDriver_async_U0_p1();          // UART0 の初期化

__StartSerialSending_async_U0_p1(3, (unsigned int *)BUFF);//UART0 への送信
__PollingSerialSending_async_U0_p1();      //ポーリング処理
__PollingSerialSending_async_U0_p1();      //ポーリング処理
```

2. Peipheral Driver Generator を使用して生成ソースファイルを High-performance Embedded Workshop へ登録する。
3.ビルドする。

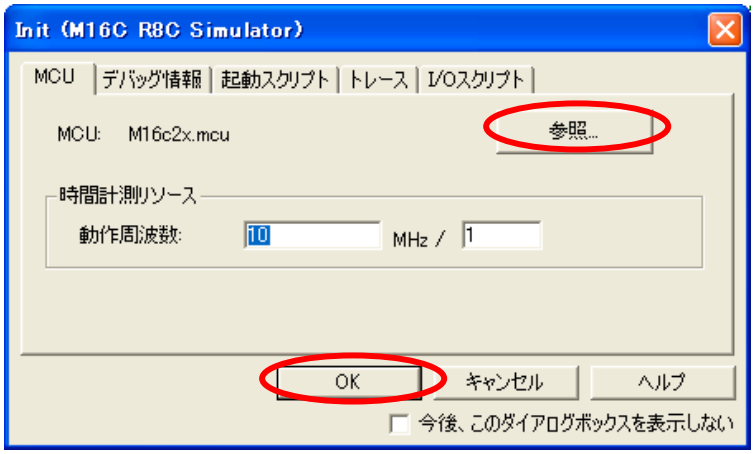
- シミュレータ実行



① セッションを SessionM16C_R8C_Simulator を選択する

② “はい”を選択する。

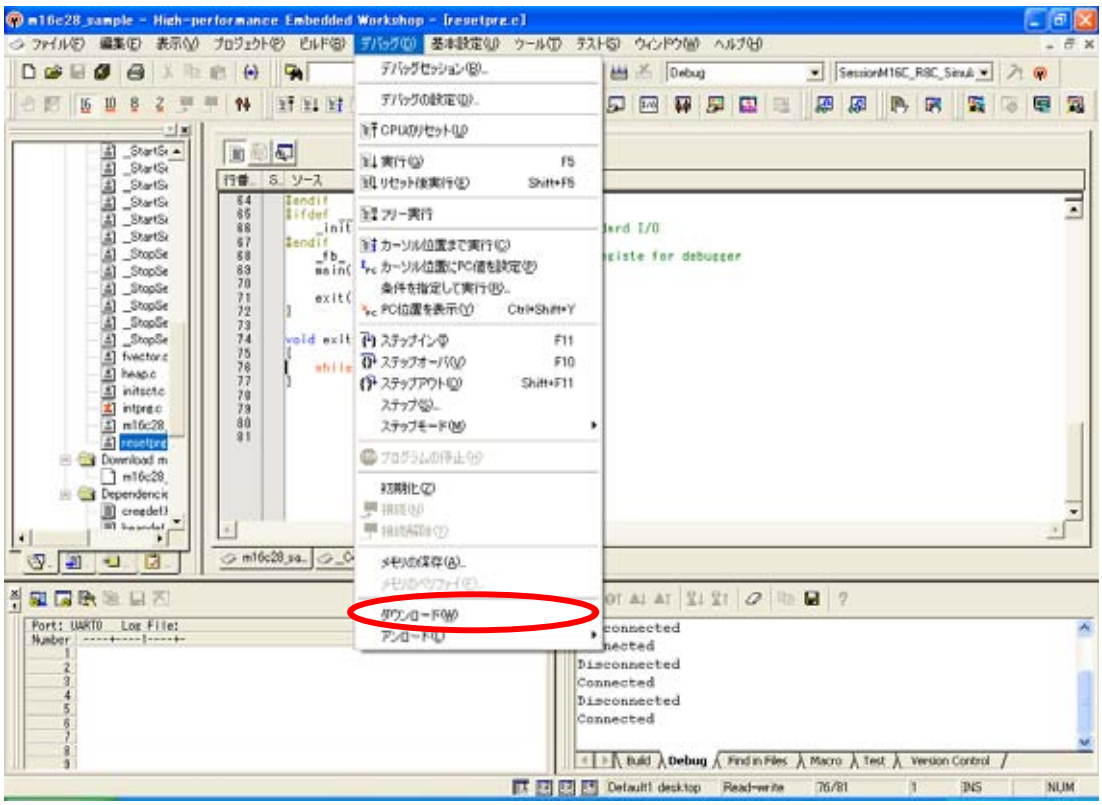
③ “OK”を選択する。



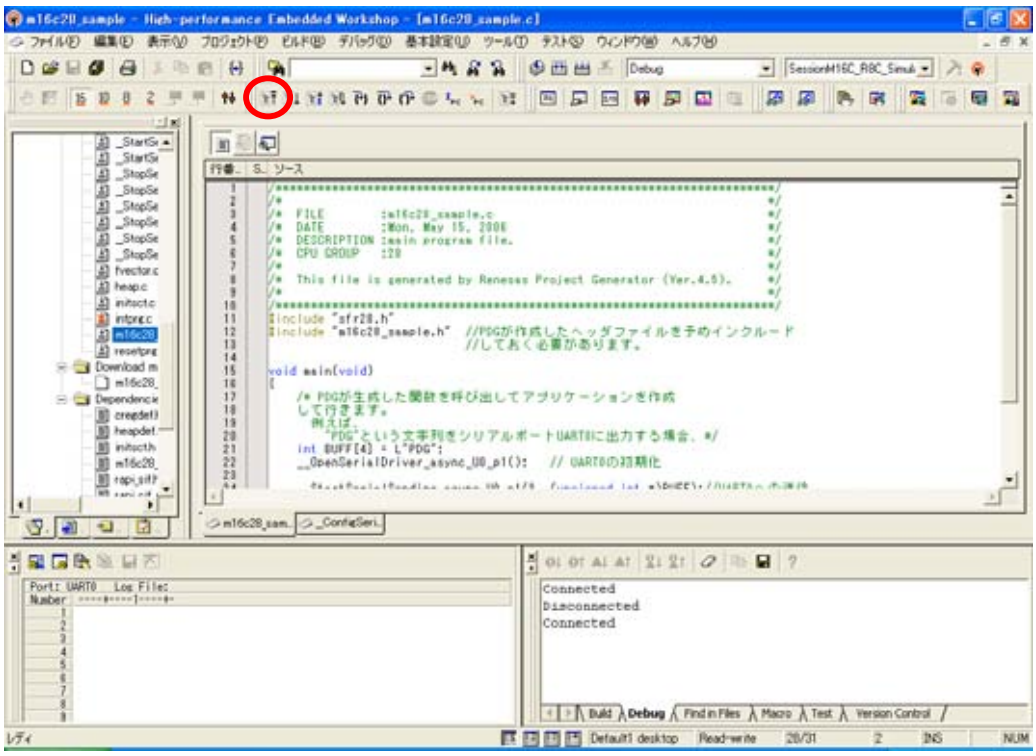
④ダウンロードを選択して、

C:¥renesas¥PDG¥sample¥m16c_28¥m16c28_sample¥debug¥m16c28_sample.x30

をダウロードする。(①～③の手順によりビルド終了後自動ダウンロードする場合もあります)



⑤リセットボタンを押す

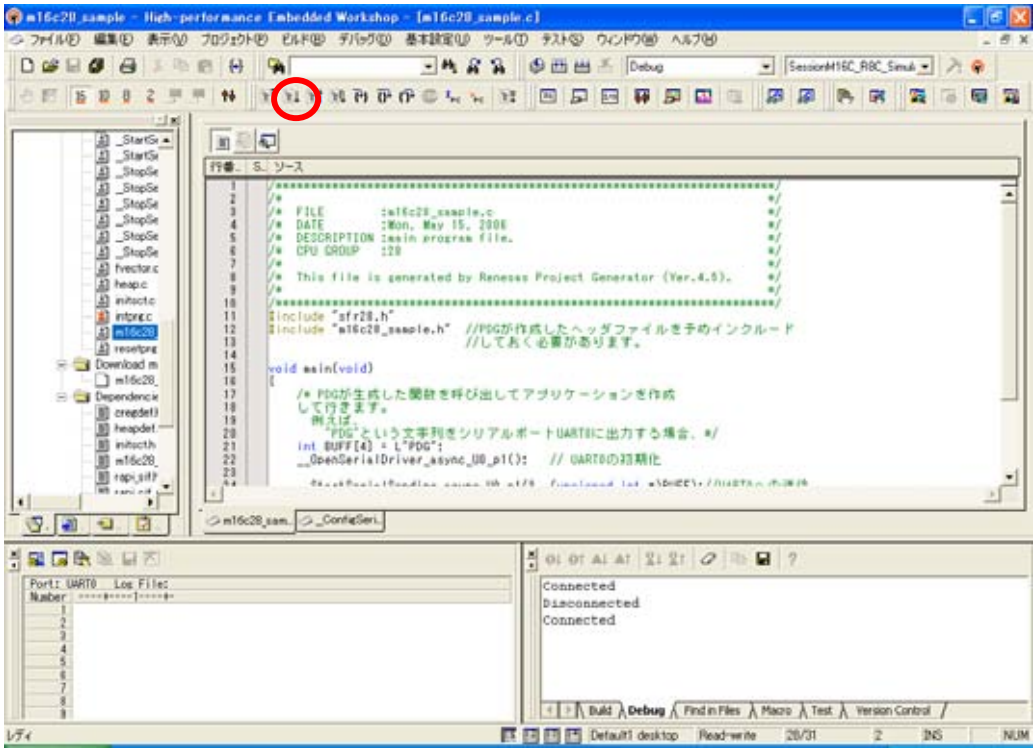


⑥ resetprg.c の exit(0)呼び出し箇所に Breakpoint を設定する。

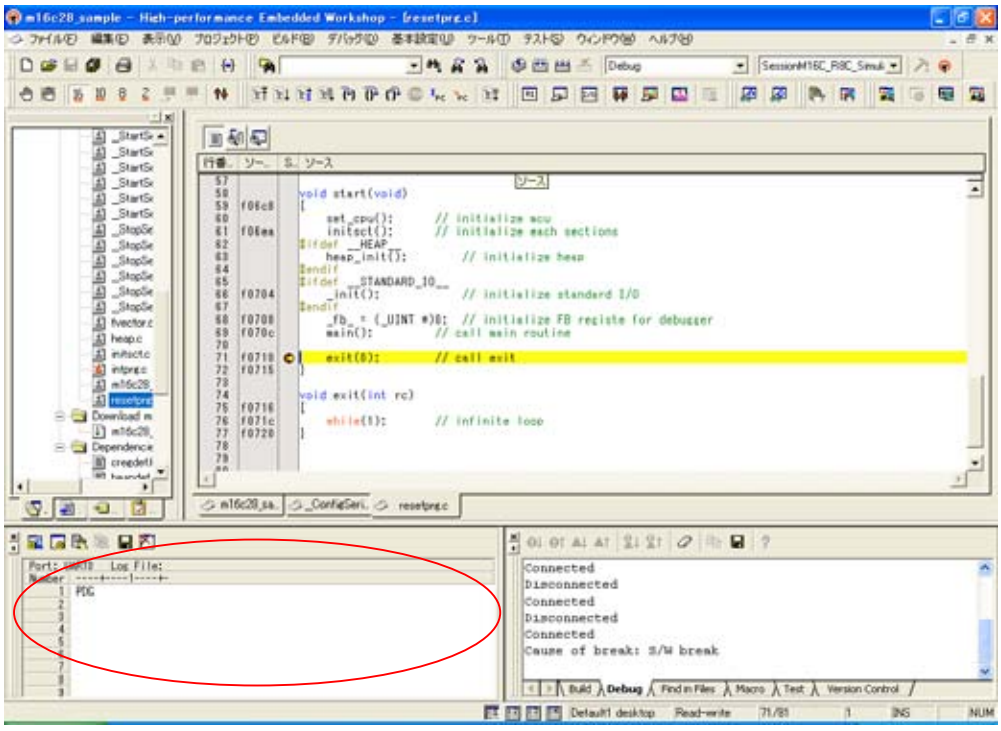
```
_fb_ = (_UINT *)0;          // initialize FB registe for debugger
main();                     // call main routine


exit(0);                // call exit
```

⑦実行ボタンを押して実行する。



⑧”PDG“という文字が出力されている事を確認する。



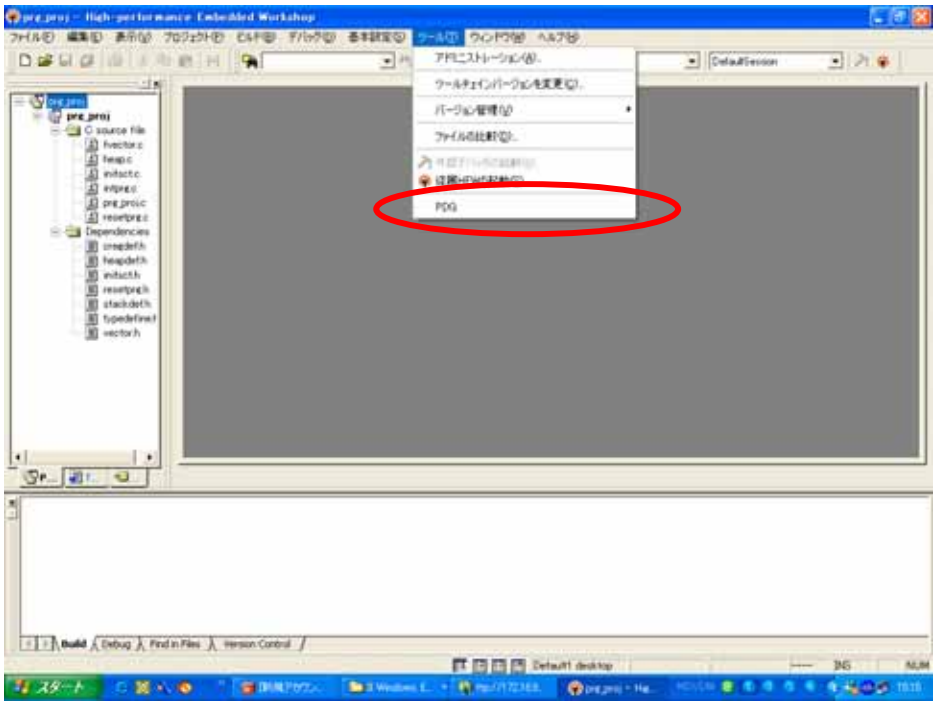
これで、実行は完了です。
以降の章で、より詳しく説明していきます。

4. Peripheral Driver Generator の起動方法

Peripheral Driver Generator を起動する方法を説明します。

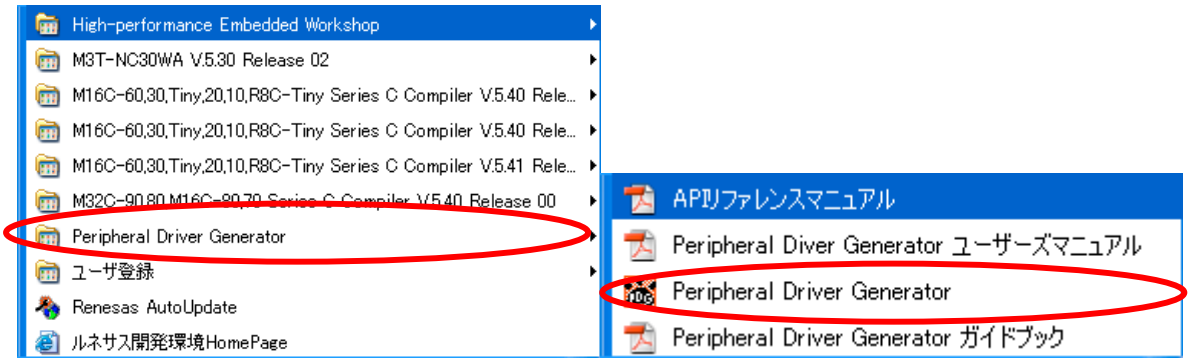
4.1. High-performance Embedded Workshop から起動する場合

(1)新規ワークスペースを作成する。
(2)ツールメニューから“ PDG”を選択する。
ワークスペースを開いていない状態では、“ PDG”は表示されません。



4.2. スタートメニューから起動する場合

スタートメニュー → Renesas → Peripheral Driver Generator → Peripheral Driver Generator を選択する。

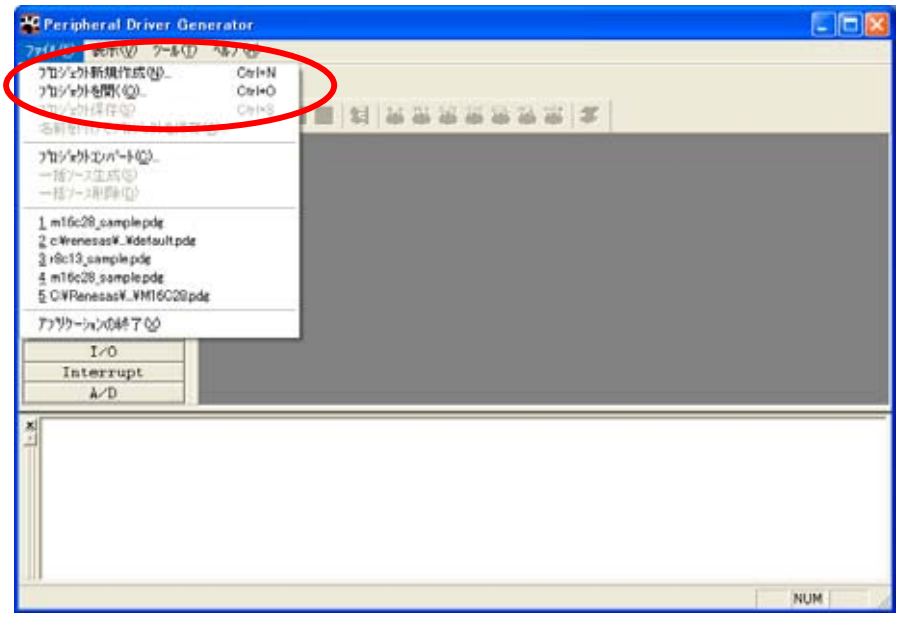


5. Peripheral Driver Generator から High-performance Embedded Workshop へのソース登録

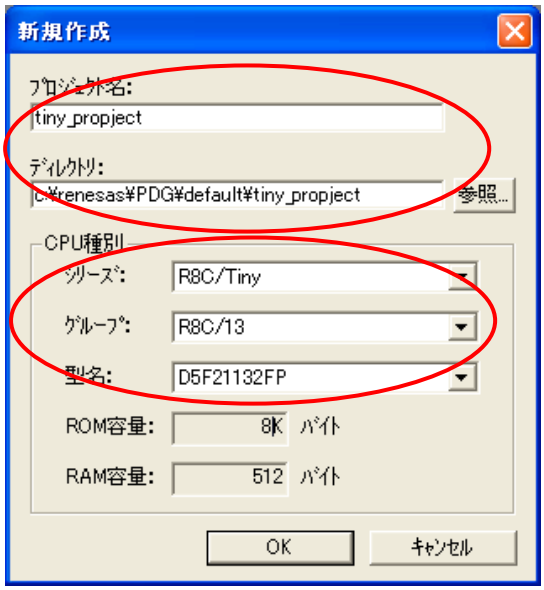
本章では、Peripheral Driver Generator で作成したソースファイルを High-performance Embedded Workshop のワークスペースへ登録する方法を説明します。

5.1. プロジェクトの作成

(1)Peripheral Driver Generator のファイルメニューから“プロジェクト新規作成(N)…”もしくは、“プロジェクトを開く(O)…“を選択してプロジェクトをオープンします。



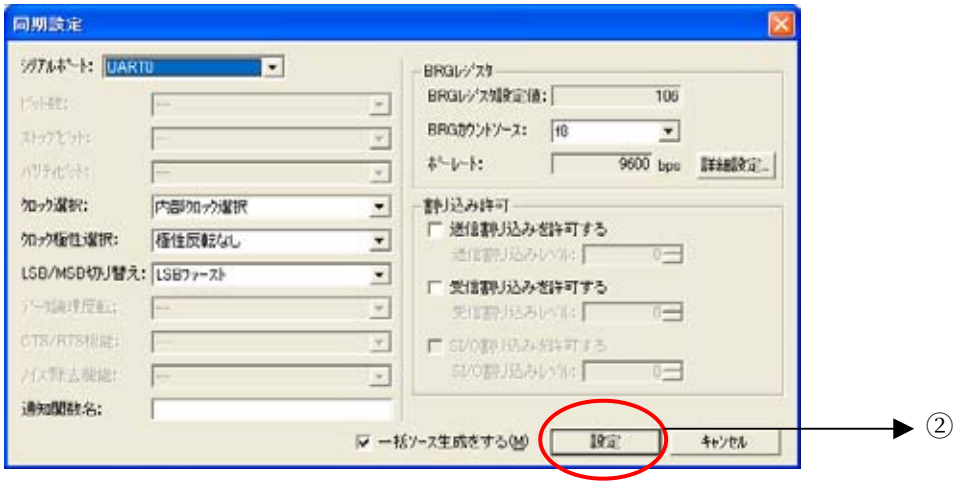
(2)“プロジェクト新規作成(N)…”を選択して新規にプロジェクトを作成する場合は、プロジェクト名、プロジェクトファイル格納ディレクトリ及び CPU 種別を選択してください。



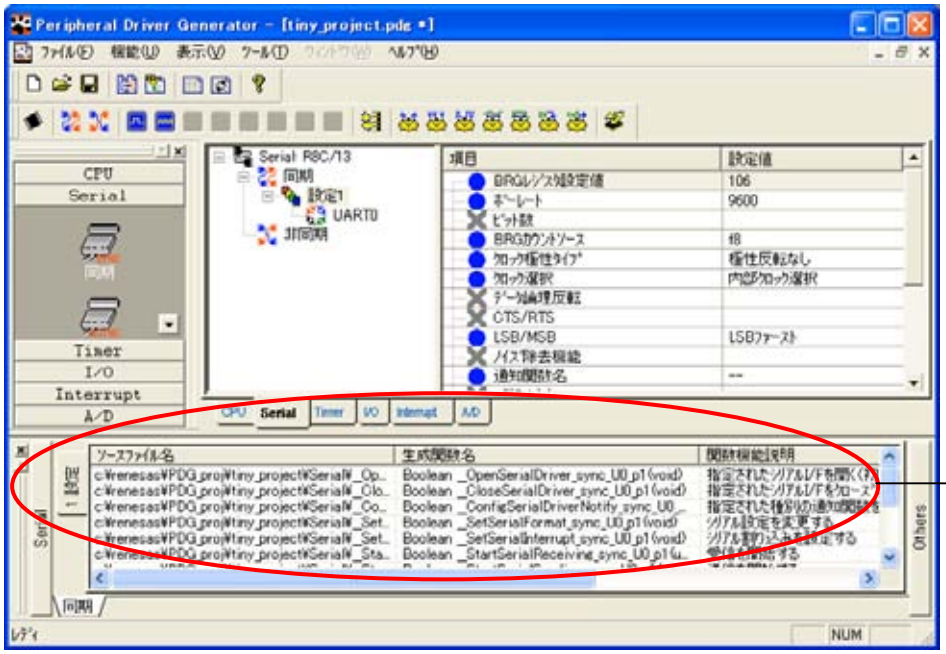
(3)CPU クロックの設定、各周辺 IO の設定を行いソースの生成を行います。

CPU クロックの設定画面

①周辺 IO を選択します。



②“ 設定 ”ボタンを押すとソースを生成します。

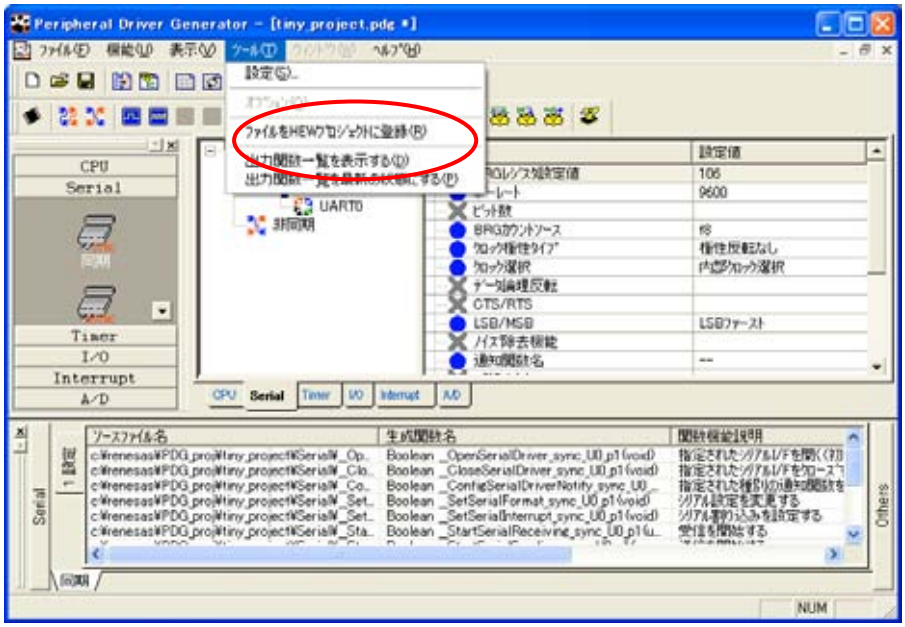


③生成したソース及び関数一覧です。

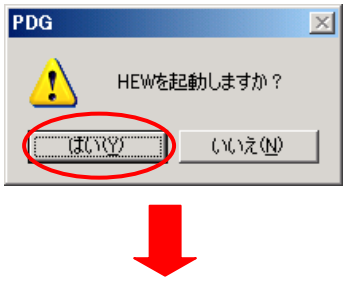
ここに記載された関数を処理に応じて呼び出す事でアプリケーションを開発していきます。

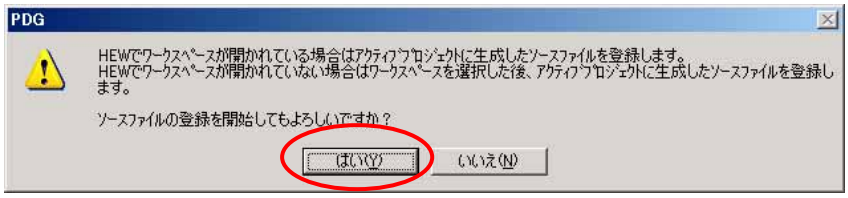
5.2. ソースファイルの登録

(1) ツールメニューから“ファイルを HEW プロジェクトへ登録”を選択する。

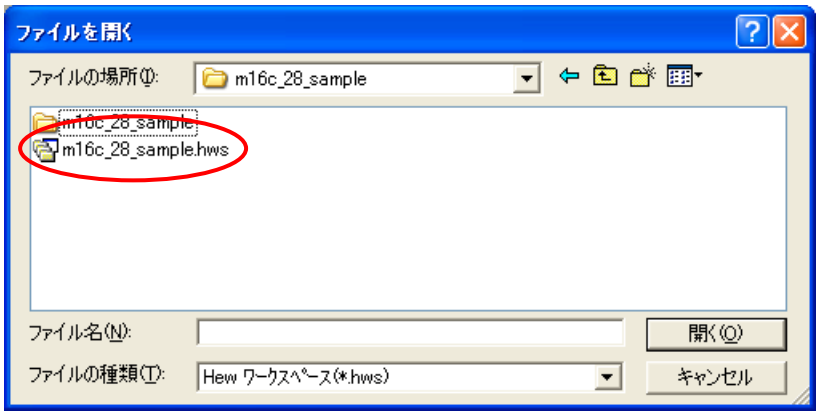


(2)High-performance Embedded Workshop を起動していない場合、High-performance Embedded Workshop を起動後、ファイルの登録を開始します。





(3)High-performance Embedded Workshop が起動しており、ワークスペースがオープンしていない場合は、
ワークスペース選択ダイアログがオープンしますので、ワークスペースを選択します。
ワークスペースがオープンしている場合は、そのワークスペースへファイルを登録します。

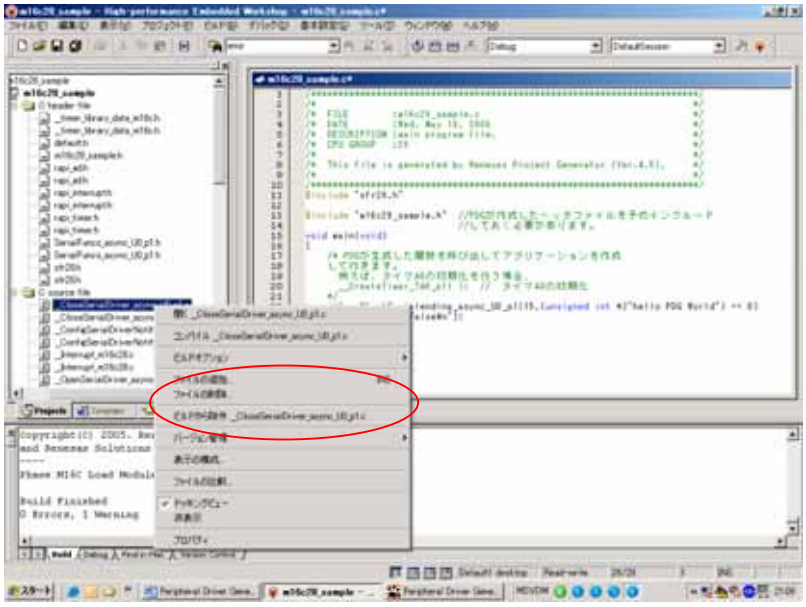


5.3. 登録したソースファイルの解除

High-performance Embedded Workshop へ一度登録すると、Peripheral Driver Generator から解除することはできません。

High-performace Embedded Workshop の機能を使って解除します。

登録を解除するファイル名にマウスポインタを合わせて右クリックします。



開いたメニューから“ファイルの削除”、もしくは“ビルドから除外”を選択します。

“ファイルの削除“を選択した場合、ファイルのツリー表示から削除されます。

“ビルドから除外”を選択した場合は、そのファイル名にXマークが表示されます。

6. ビルドの設定

High-performance Embedded Workshop でコンパイル、ビルドを行うには、２で示した生成ソースを登録する以外に

・参照ヘッダファイルのディレクトリ指定（-I オプションの指定）

・API ライブラリをリンクするための設定 (-l の指定)　※

※High-performance Embedded Workshop V.4.02 以前のバージョンを使用する場合

が必要になります。

本章では、各項目の設定方法を説明します。

なお、Peripheral Drive Generator パッケージに添付している各サンプルワークスペースでは、設定済みです。

6.1. ヘッダファイルの指定

Peripheral Driver Generator で生成した関数を呼び出す場合、関数のプロトタイプ宣言を記述したヘッダファイルをインクルードする必要があります。

ヘッダファイル名は、プロジェクト名.h になります。

High-performance Embedded Workshop で作成したディレクトリと Peripheral Driver Generator で作成したプロジェクトディレクトリが異なる場合には、コンパイラオプションの-I でヘッダファイルの存在するディレクトリ、すなわちプロジェクトを作成したディレクトリを指定する必要があります。

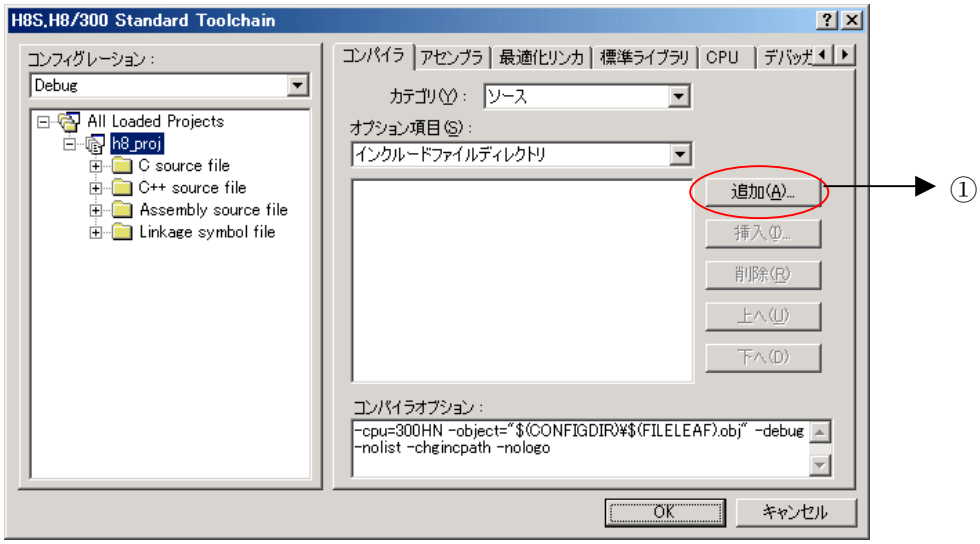
```
<sample.c>
#include "m16c_28_sample.h"

void main(void)
{
    __OpenSerialDriver_sync_U0_p1();
    __ConfigSerialDriverNotify_sync_U0_p1();
    __SetSerialFormat_sync_U0_p1();
        :
        :
        :
}
```

(1) ビルドメニューから、

M16C/Tiny 及び R8C/Tiny の場合は、「Renesas M16C Standard Toolchain...」

H8/300H Tiny の場合は、「H8S,H8/300 Standard Toolchain...」を選択する。



(2) “コンパイラ“ タブ(M16C/Tiny,R8C/Tiny の場合は、”C”タブ)を選択する。

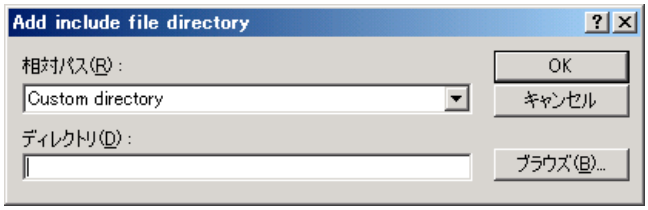
H8/300H Tiny 選択時：オプション項目(S):インクルードファイルディレクトリ

M16C/Tiny,R8C/Tiny 選択時：Show Entries For: Inlcude file directories

を選択する。

①「追加(A)...」(Add...)をクリックする。

(3) インクルードファイルが格納されているディレクトリの指定



Add include file directory ダイアログに相対パスのカテゴリとディレクトリ名を入力する。

H8/300H Tiny の場合、

相対パス(R) : Custom directory

ディレクトリ(D): ディレクトリ名

M１６C/Tiny、R8C/Tiny の場合、

Relative to: Custom directory

Directory: ディレクトリ名

をそれぞれ選択、入力する。

(4)全てのダイログを閉じて完了

6.2. ライブラリの指定

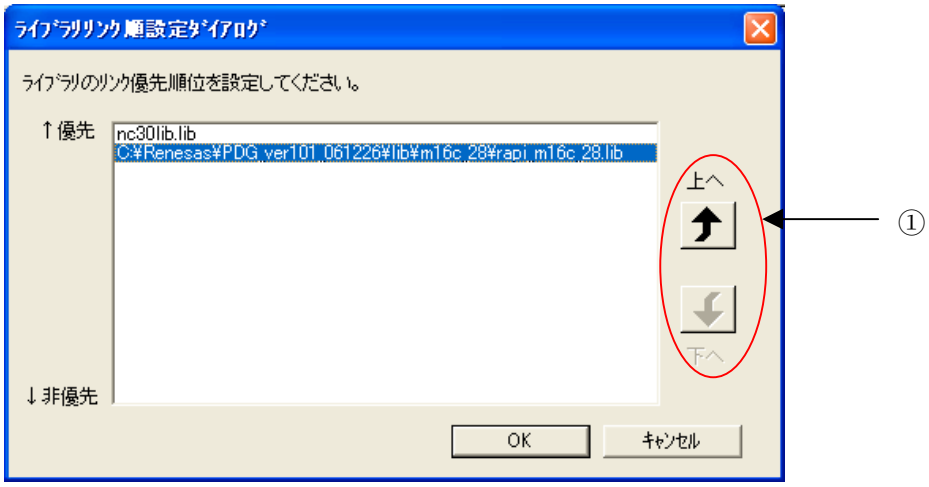
6.2.1. ライブラリー覧

各周辺の設定ソースをビルドする場合、下に示す各ライブラリをリンクする必要があります。

CPU	格納ディレクトリ	ライブラリ名
H8/3687	lib¥h8_3687	rapi_h83687.lib
R8C/13	lib¥r8c_13	rapi_r8c13.lib
M16C/28	lib¥m16c_28	rapi_m16c28.lib

6.2.2. ライブラリ指定方法

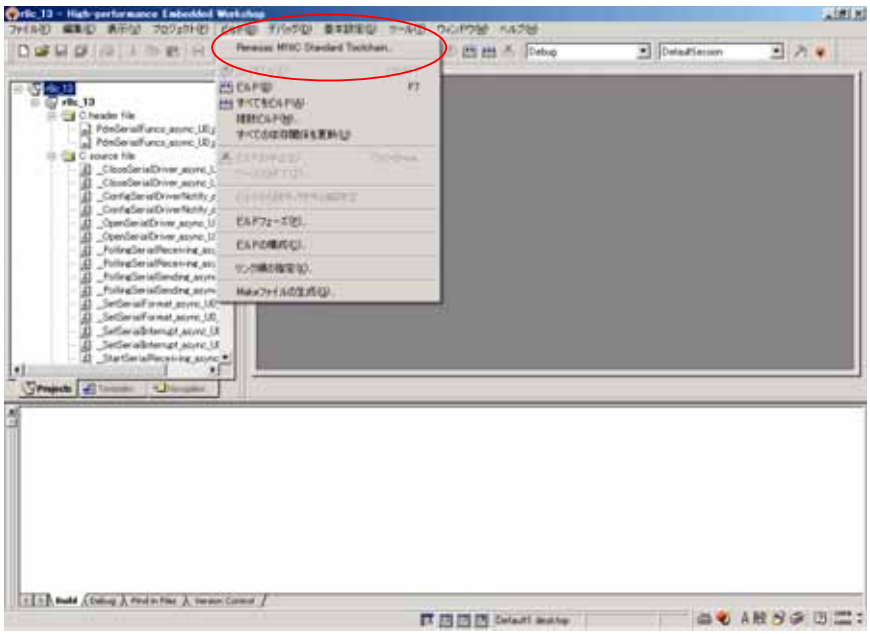
High-performance Embedded Workshop V.4.02 以降を使用している場合は、Peripheral Driver Generator からソースを登録する際に、以下のダイアログが表示されます。



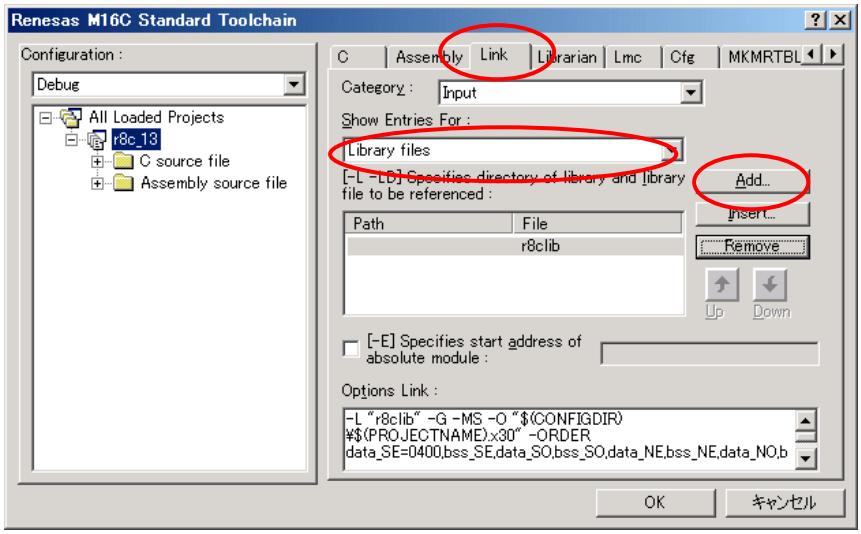
このダイアログを使用して各ライブラリの優先度を決めてください。
ライブラリの優先度は、異なるライブラリ中に同一のシンボル名が存在した場合にどちらのライブラリから選択するかを決定します。
優先度の変更は、対象となるライブラリを選択した状態で、①で示した矢印ボタンを押してください。
上に行くほど優先度は高くなります。

High-performance Embedded Workshop V.4.02 以前を使用している場合は、以下の方法でライブラリを設定してください。

(1) ビルドメニューから、
M16C/Tiny 及び R8C/Tiny の場合は、「Renesas M16C Standard Toolchain...」
H8/300H Tiny の場合は、「H8S,H8/300 Standard Toolchain...」を選択する。

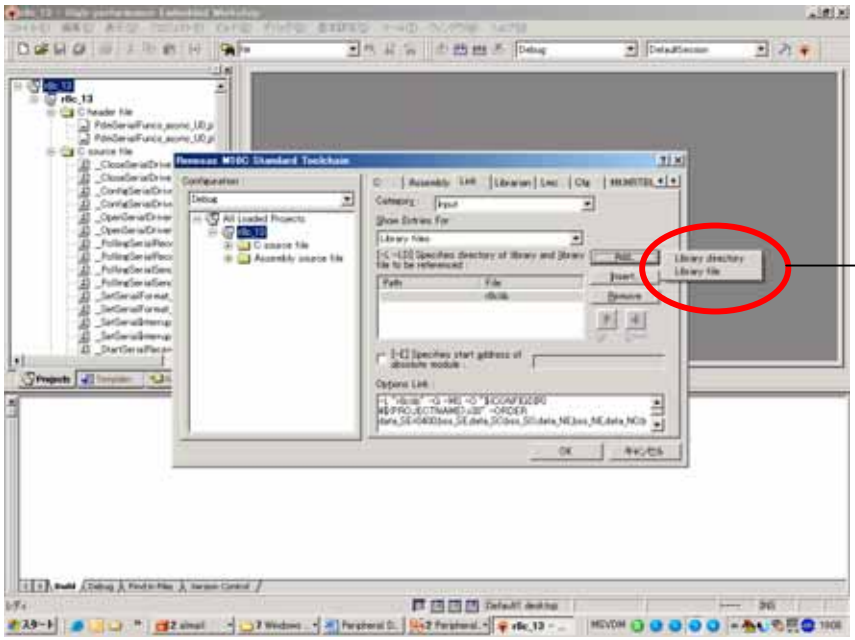


(2) Renesas M16C Standard Toolchain ダイアログで“ Link ”タブ
もしくは、
H8S,H8/300 Standard Toolchain ダイアログで“ 最適化リンカ“ タブ
を選択する。

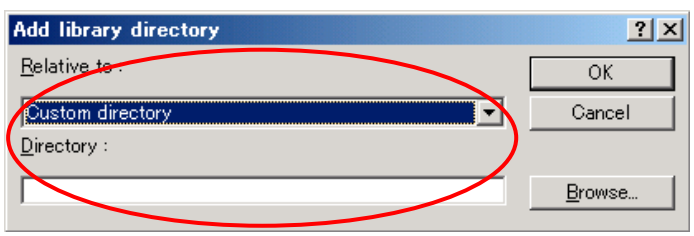


Show Entries For: Library files (H8/300H Tiny の場合は、ライブラリファイル)を選択
Add（追加）ボタンを押す

(3) API ライブラリが格納されているディレクトリを指定します。



①Library directory を選択する。（M16C/Tiny,R8C/Tiny のみ）



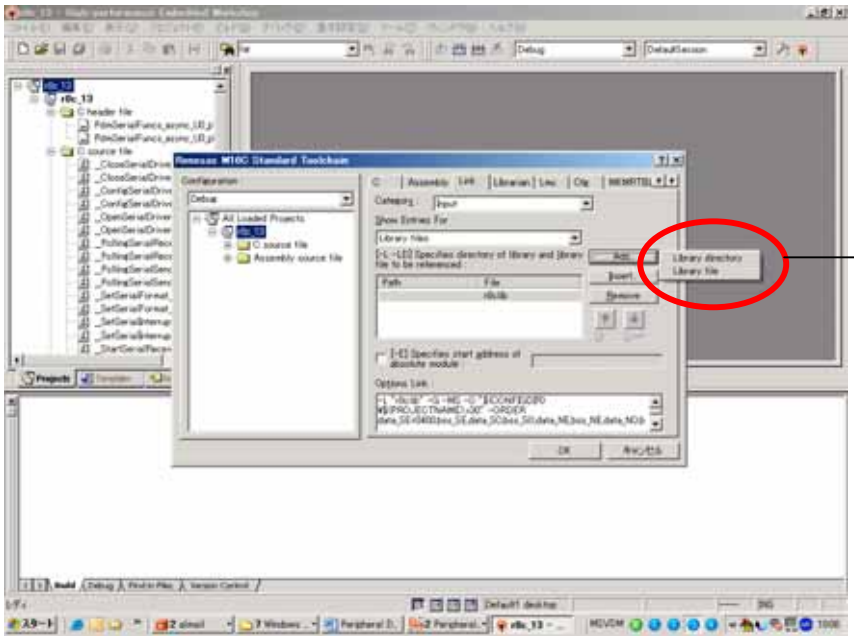
Relative to : Custom directory　を指定する。

Directory : API ライブラリの格納ディレクトリを指定する。

例）C:¥Renesas¥PDG¥lib¥M16C_28

H8/300H Tiny の場合は、ここでディレクトリ名とライブラリファイル名を指定します。

(4) Library 名を指定する。(M16C/Tiny,R8C/Tiny のみ)



③ Library file　を選択する。

④ 使用する周辺の API ライブラリファイル名を記述します。

例） rapi_m16c28　(拡張子.lib は付けないでください。)

(5)全てのダイアログを閉じて完了

6.2.3. 割り込みベクタテーブルの除外

Peripheral Driver Generator は、割り込みベクタも同時に作成します。

そのため、High-performance Embedded Workshop でワークスペースを作成した際に、スタートアップファイルも同時生成すると割り込みベクタが重複するため、スタートアップ側の割り込みベクタをコンパイル対象から除外する必要があります。

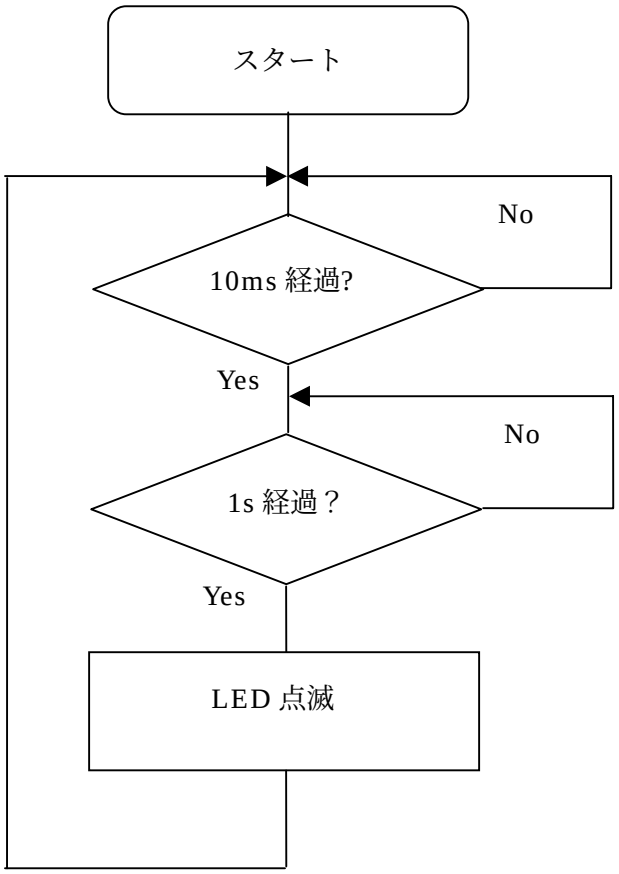
7. Peripheral Driver Generator を使用して実際にプログラムを作成する

7.1. プログラム

7.1.1. フローチャート

以下のフローチャートに基づいたプログラムを作成します。

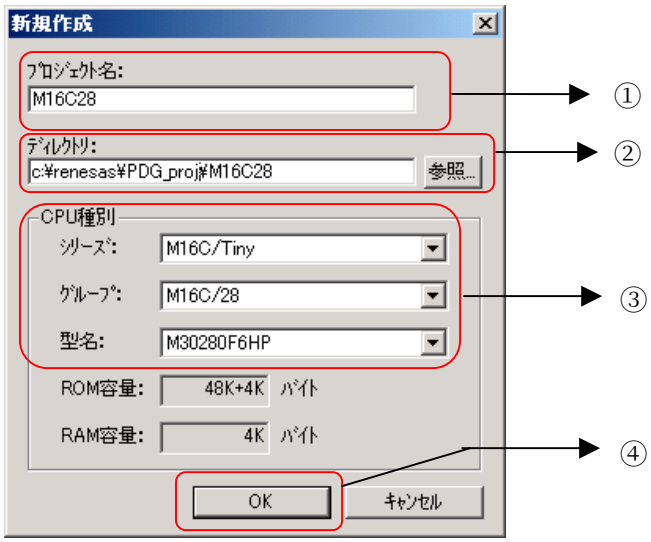
使用マイコン：M16C/28



7.2. Peripheral Driver Generator を使用して周辺を設定する。

7.2.1. プロジェクト作成

[ファイル]→[プロジェクト新規作成]を選択して新規作成ウィンドウをオープンします。



① 任意のプロジェクト名を入力します。
ここでは、“ M16C28 “ とします。
注意）Peripheral Driver Generator は、プロジェクト名.h でヘッダファイルを作成します。
既存の同名ヘッダファイルが存在する場合は、別名称のプロジェクト名にしてください。

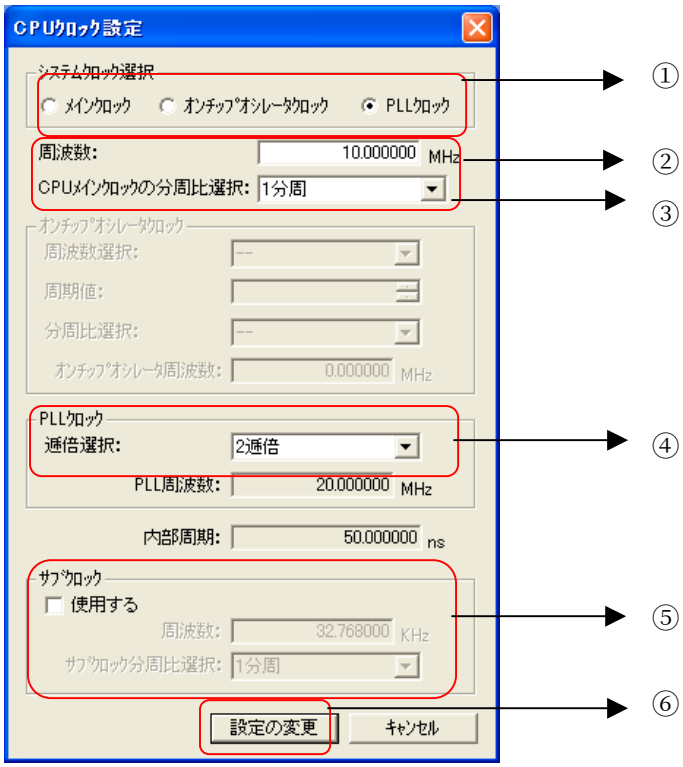
② プロジェクトを格納するディレクトリを指定します。
デフォルトでは、c:¥renesas¥PDG_proj の下にプロジェクト名のディレクトリを作成します。

③ CPU を選択します。
ここでは、
シリーズ：M16C/Tiny
グループ：M16C/28
型名：M30280F6
を、それぞれ選択します。

④ OK ボタンを押して、プロジェクト作成を終了します。

7.2.2. クロックの設定

7.2.1 のプロジェクト作成が終了すると、**CPU クロック設定ウィンドウ**が起動します。



① システムクロック選択で **PLL クロック**を選択する。

② 周波数は、**10MHz** に設定する。

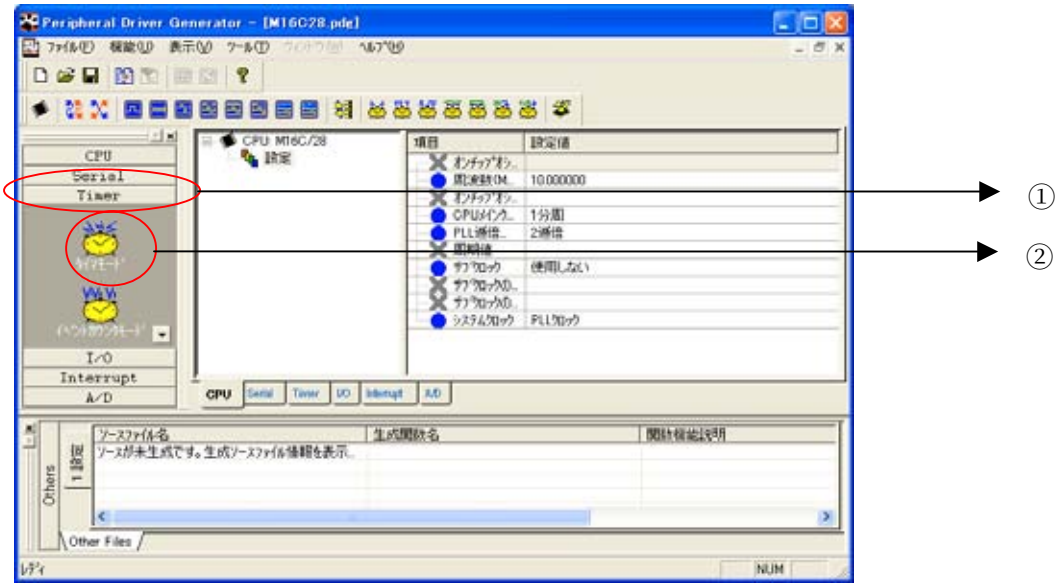
③ CPU メインクロックの分周比選択は、**1 分周**を選択する。

④ 逓倍選択は、**２逓倍**を選択する。

⑤ サブクロックは、**使用しない**。

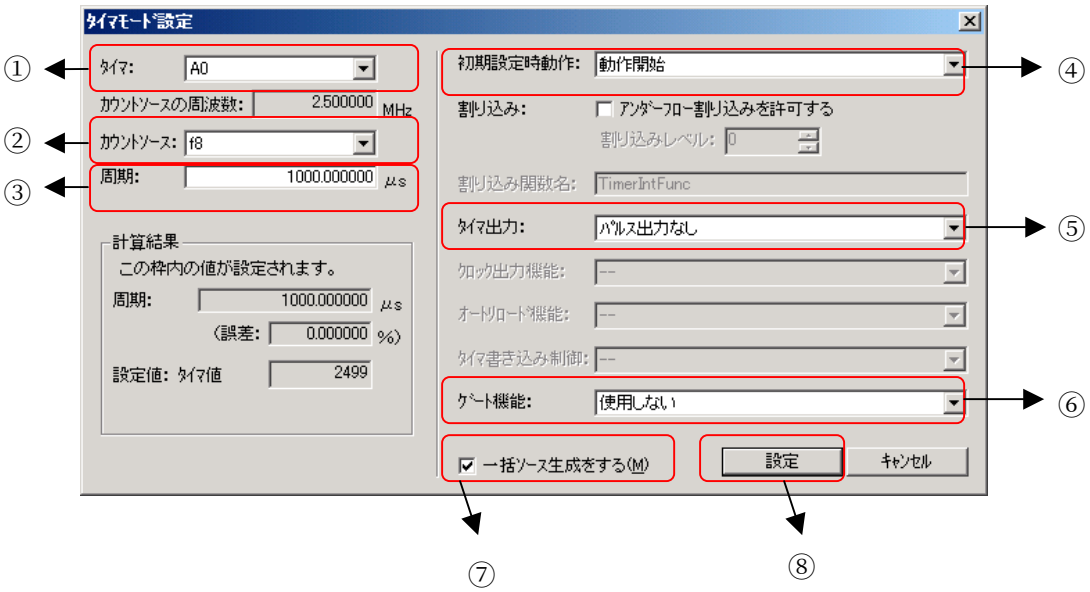
⑥ 設定の変更を選択して、設定を終了する。

7.2.3. タイマモードの設定



①**Timer** タブを選択します。

②**タイマモード**を選択してクリックします。



①使用するタイマは **A0** を選択する

②内部カウントソースでは、**f8** を選択する。

③カウンタ値は、**10ms** を設定する。

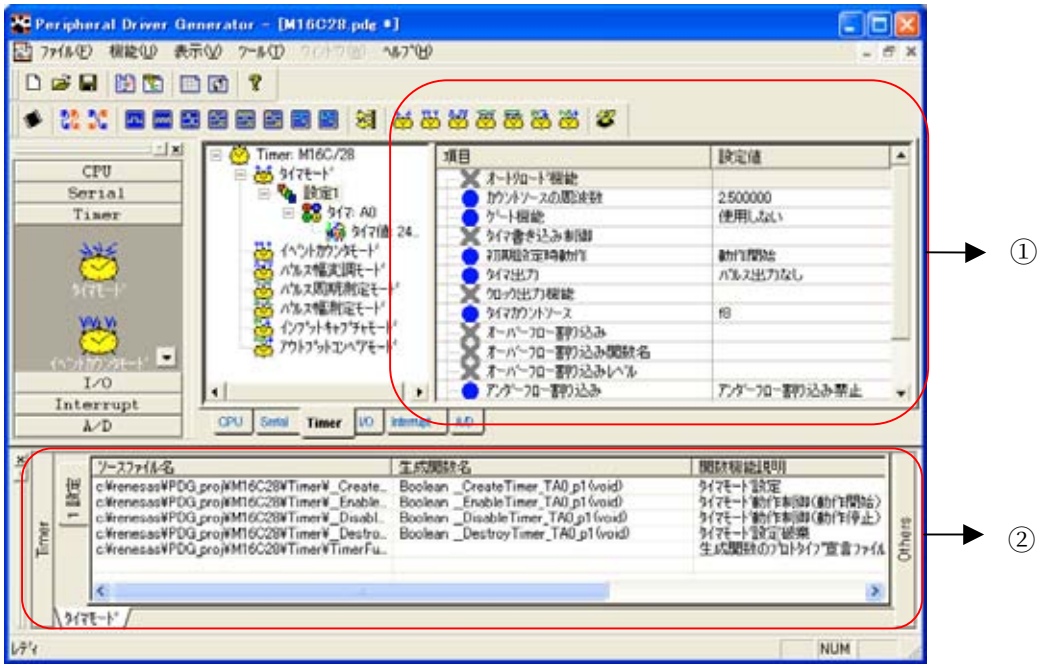
④初期化後の動作では、**動作開始**を選択する。

⑤タイマ出力は、**パルス出力なし**を選択する。

⑥ゲート機能は、**使用しない**を選択する。

⑦ 設定した内容でソース生成をするために、**一括ソース生成をする**にチェックする。

⑧ **設定**をクリックして設定を完了する。



タイマモードの設定が終了すると、

① に設定した内容

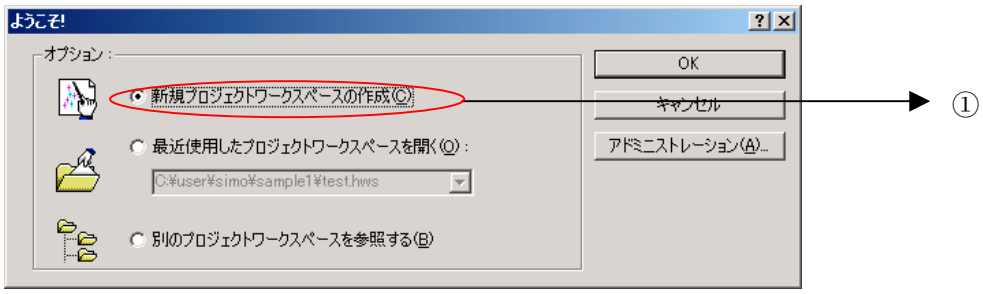
② に生成したソースファイル、関数名、機能

の一覧を表示します。

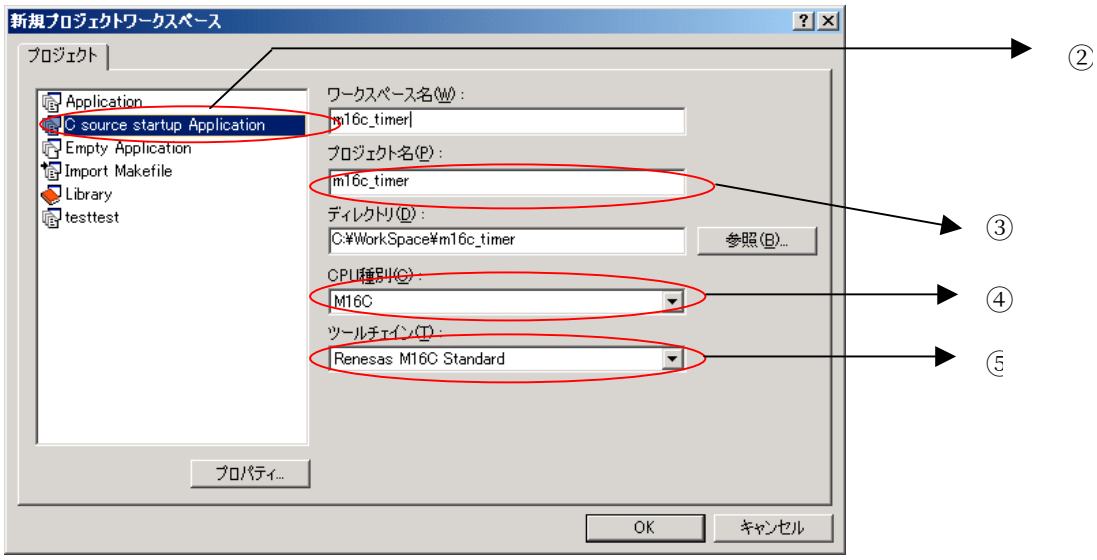
7.3. プログラム作成

プログラムの作成は、High-performance Embedded Workshop を使用して行います。

7.3.1. ワークスペース作成



① ようこそ！ダイアログで “ **新規プロジェクトワークスペースの作成(C)**”を選択して OK ボタンをクリックする。



② “**C source startup Aplication**”を選択します。

（アセンブラスタートアップを使用する場合は、Aplication を選択します。）

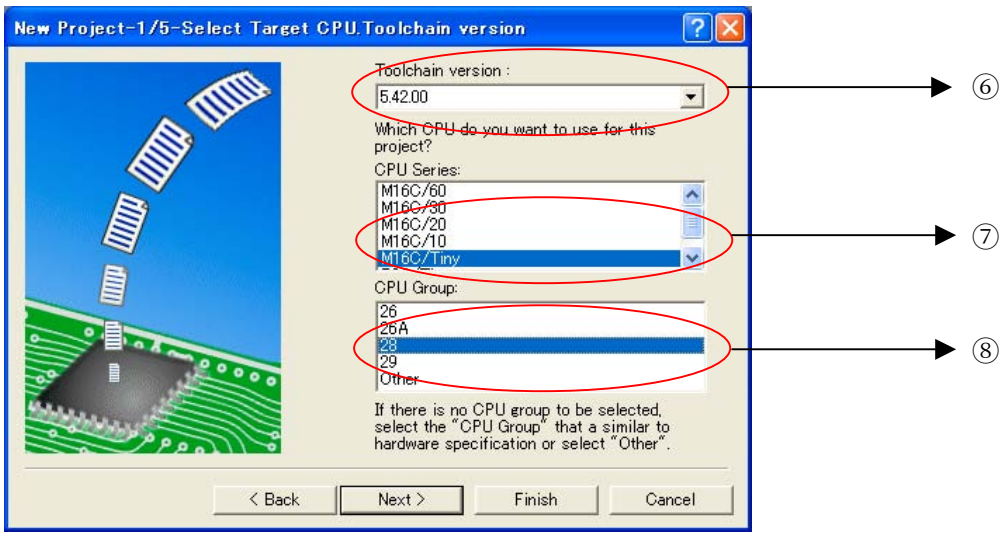
③ワークスペース名を入力します。

ここでは、**m16c_timer** とします。

④CPU 種別では、**M16C** を選択します。

⑤ツールチェインは、”**Renesas M16C Standard**”を選択します。

OK ボタンをクリックします。



⑥ コンパイラのバージョンを 5.40.00、もしくは 5.42.00 にする

⑦ CPU Series:は M16C/Tiny を選択する

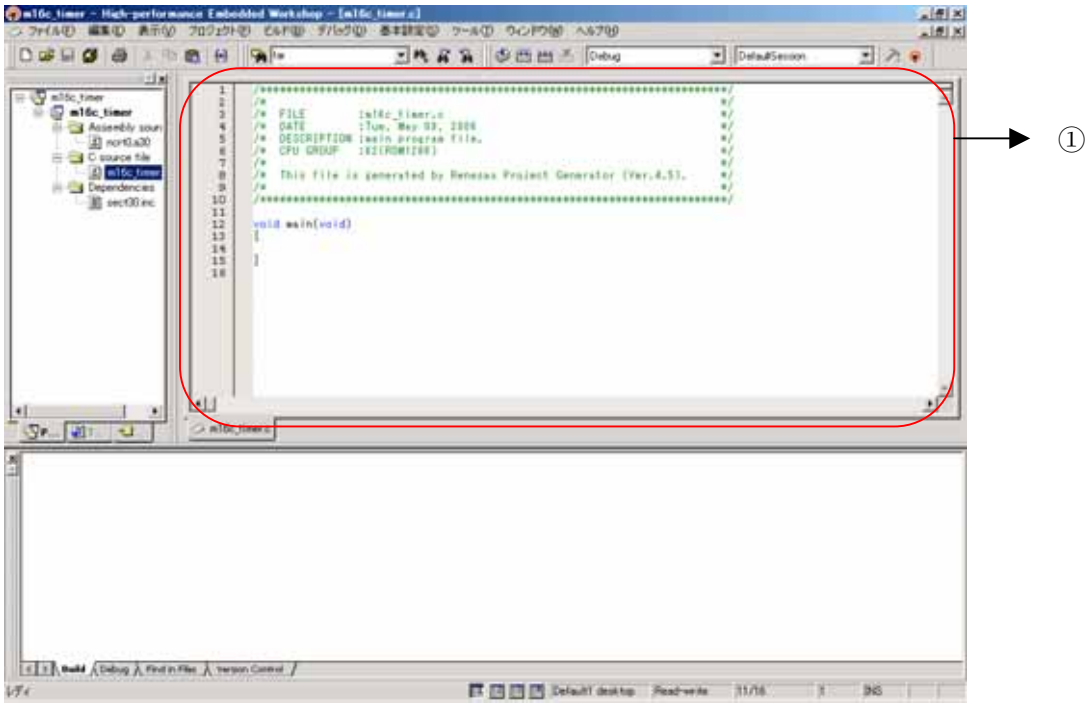
⑧ CPU Group は、28 を選択する。

Next>ボタンをクリックして次に進む。

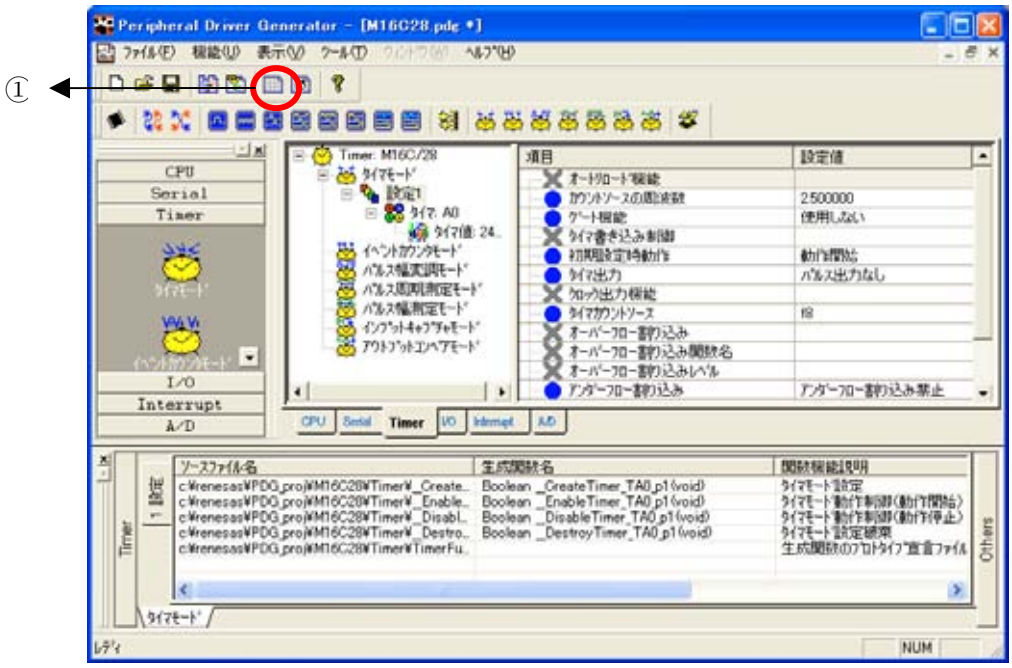
新規ワークスペース作成ウィザードを最後まで行い、新規ワークスペースの作成を完了する。

7.3.2. **プログラムの作成**

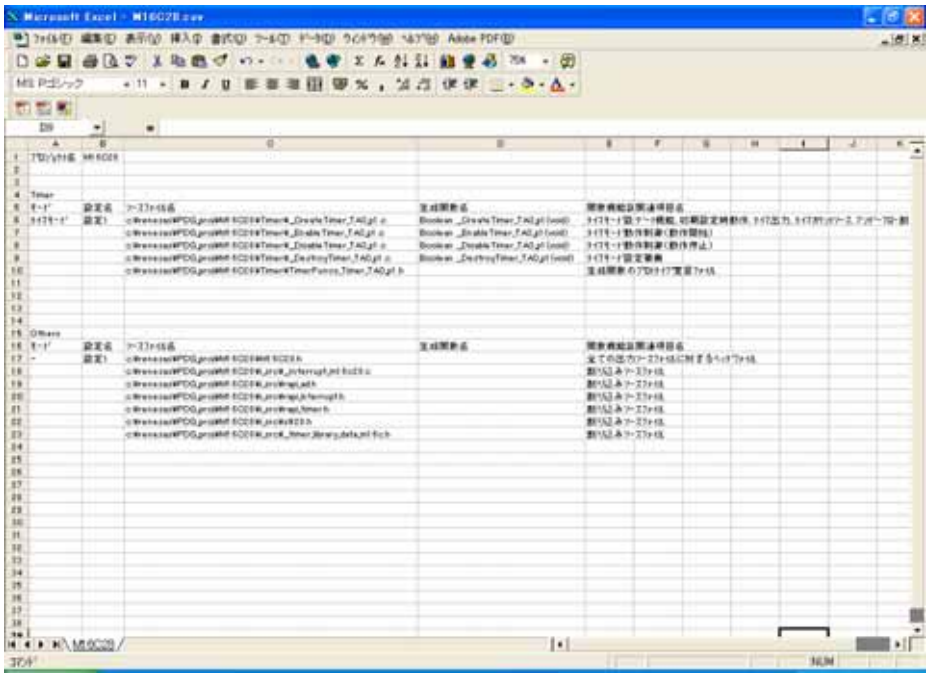
High-performance Embedded Workshop 上の①でプログラムをコーディングしていきます。



コーディングでは、Peripheral Driver Generator で作成した関数を使用します。
生成した関数の一覧を見るには、GUI 上の表示で確認する他に、エクセルデータとして確認することもできます。



① ボタンを押すことにより、エクセルが起動し関数一覧を見ることができます。
ただし、拡張子 csv ファイルを出力しますので、csv ファイルがエクセルに関連付けられている必要があります。（関連付けられていない場合は、テキストファイルとして表示します。）



表示したエクセルデータ

7.3.3. サンプルプログラム

赤文字で記述されたヘッダファイル及び関数が Peripheral Drive Generator が生成した関数です。

```
#include "sfr28.h"

#include “m16c28.h”              //Peripheral Driver Generator で生成した
                                 //関数を使用する場合は、本ヘッダをインクルードする必要があ
る
#define  _1SCNT(1000/10)
#define PLL_WAIT_1MS  10000                    /* 1msec @10MHz */
#define PLL_WAIT_CNT  20                       /* 20msec */

void main(void)
{
        int       counter  = _1SCNT;
        int       onoff     = 1;
        unsigned int i,j;

        /* PLL clock setting */
        prcr = 0x01;                    /* protect register off */
        cm2    = 0x00;                  /* system register2 Initialize */
        cm07 = 0;
        cm1 &= 0x3f;
        cm06 = 0;
        plc0 = 0x11;                    /* 2 multiplying */
        pm20 = 0;                       /* 2 wait */
        plc07 = 1;                      /* PLL operation */

        for (i = 0; i < PLL_WAIT_CNT; i++) {            /* about 20ms wait */
                for (j = 0; j < PLL_WAIT_1MS; j++) {    /* Main clock 10MHz */
                }
        }
        cm11 = 1;
        prcr = 0x00;                                    /* protect register on    */

        prcr = 0x04;                                    /* protect register off */
        pacr = 0x03;                                    /* 80pin type */
        prcr = 0x00;                                    /* protect register on */

                        ↓
                        ①
```

```
                                ①
                                ↓
        p0 = 0xff;
        p1 = 0xff;
        pd0 = 0xff;
        pd1 = 0xff;

        if(   __CreateTimer_TA0_p1( ) == TRUE    )          /* timer setting */
        {
                if( __EnableTimer_TA0_p1(   ) == TRUE )  /* timer start */
                {
                        while( 1 )
                        {
                                while( ( ta1ic & 0x08 ) == 0 );       /* 10ms? */
                                ir_ta1ic = 0;
                                counter--;
                                if( counter == 0 )                    /* 1s? */
                                {
                                        p1 = 0xfe;
                                        if( onoff )
                                                p0 = 0xf9;          /* LED1 on */
                                        else
                                                p0 = 0xff;          /* LED1 off */
                                        onoff ^= 1;
                                        counter = _1SCNT;           /* counter reset */
                                }
                        }
                }
        }
        return;
}
/* end */
```

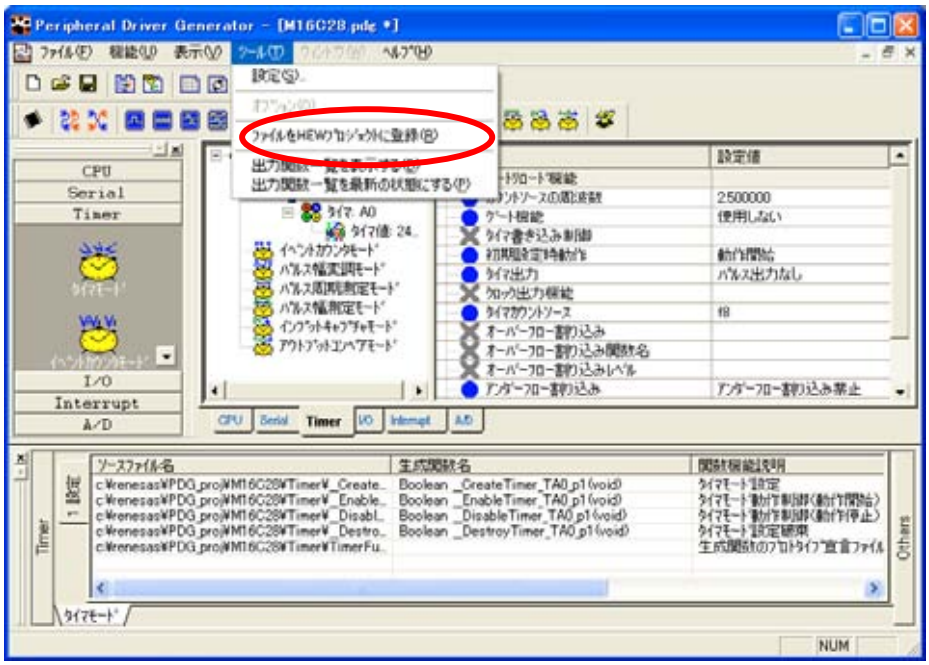
7.4. ビルド作業

プログラムのコーディングが完了して、コンパイル/リンクを行うには、以下の作業が必要です。

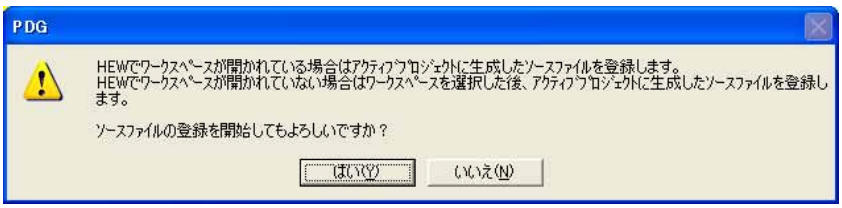
- Peripheral Driver Generator が生成したファイルを High-performance Embedded Workshop へ登録
- コンパイルに必要なオプションの設定（ヘッダファイルのインクルード先指定）
- リンクに必要なオプションの設定（Renesas Embedded Library の指定）

7.4.1. 生成ファイルの登録

Peripheral Driver Generator が生成したファイル群を High-performance Embedded Workshop のワークスペースへ登録します。



[ツール]→“ ファイルを HEW プロジェクトに登録 “ を選択する。

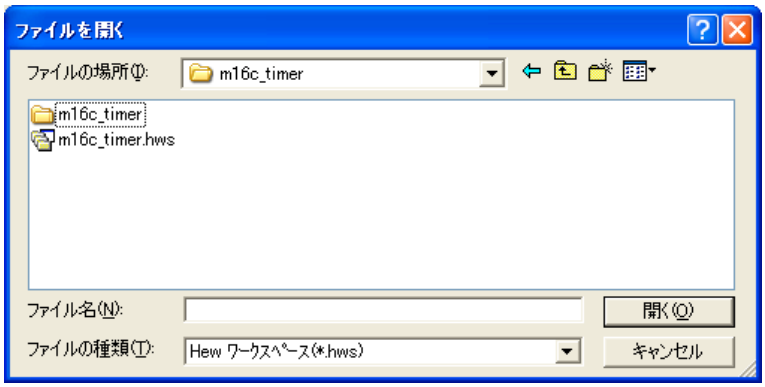


登録確認ダイアログが開くので、“ はい ” を選択する。

現在 High-performance Embedded Workshop が開いているプロジェクトへ登録します。

異なるプロジェクトへ登録する場合は、一旦 High-performance Embedded Workshop で開いているワークスペースを閉じてください。

ワークスペースを閉じた状態で、ファイル登録を行うと、登録先ワークスペースの選択ダイアログが開きますので、登録先ワークスペースを選択してください。



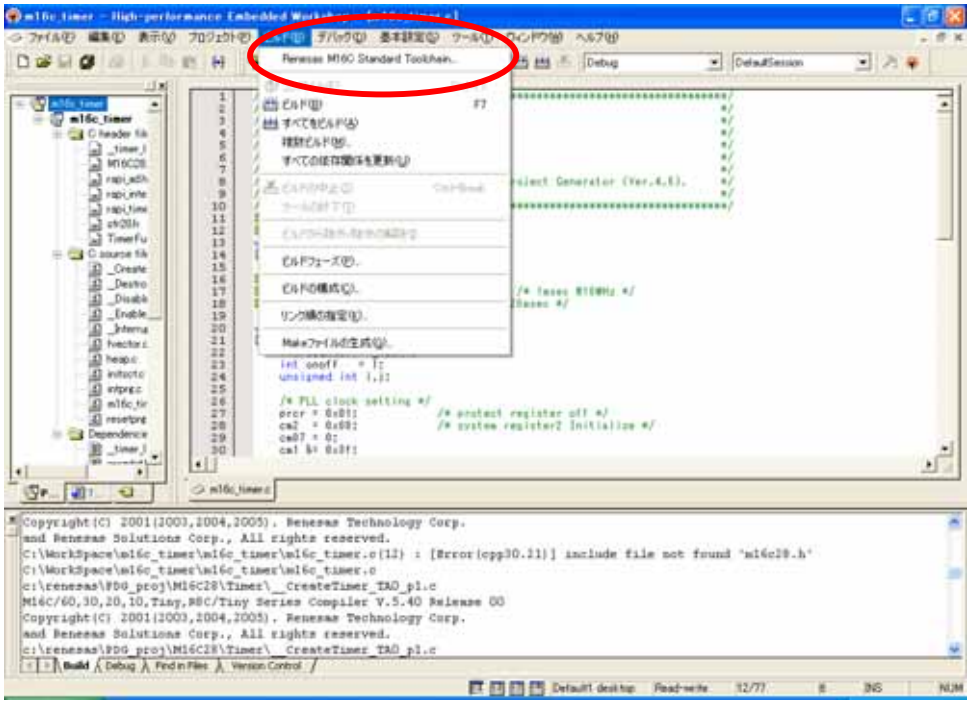
7.4.2. コンパイルオプションの設定

Peripheral Driver Generator で生成した関数を呼び出すためには、ヘッダファイルをインクルードする必要があります。

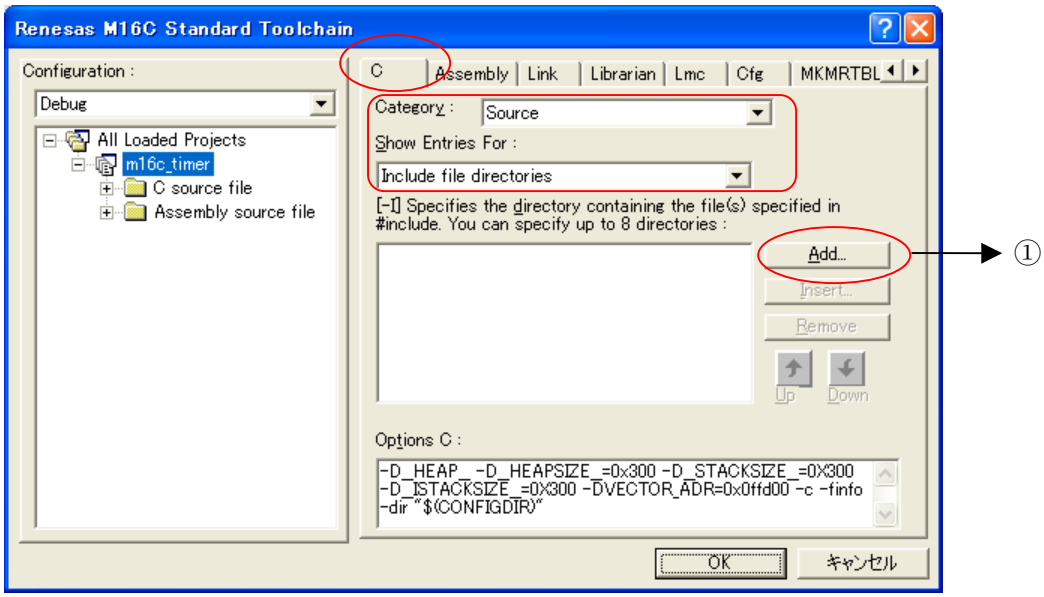
```
#include "sfr28.h"

#include “m16c28.h”                    //Peripheral Driver Generator で生成した
                                       //関数を使用する場合は、本ヘッダをインクルードする必要がある
#define  _1SCNT(1000/10)
```

このヘッダファイルをインクルードしたソースファイルをコンパイルするためには、ヘッダファイルが格納されているディレクトリを指定する必要があります。



High-performance Embedded Workshop のビルドメニューから”Renesas M16C Standard Toolchain”を選択します。



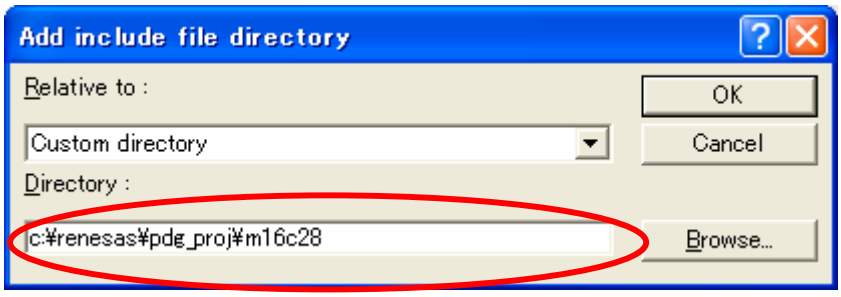
オプションダイアログから C タブを選択して、

Category: Source

Show Entries For: Include file directories

をそれぞれ選択します。

① で示した Add ボタンを押します。



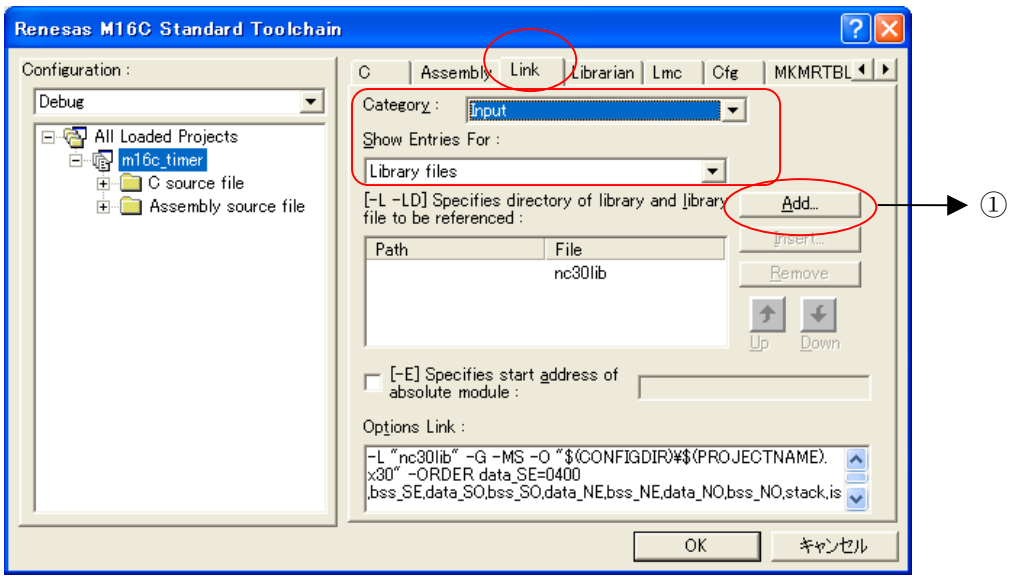
Peripheral Driver Generator が出力したヘッダファイルが存在するディレクトリを指定します。

このディレクトリは、Peripheral Driver Generator で作成したプロジェクトのディレクトリです。

OK ボタンを押してコンパイルオプションの設定は完了です。

7.4.3. リンクオプションの設定

(1)ビルドメニューから、「Renesas M16C Standard Toolchain...」を選択します。



オプションダイアログから Link タブを選択して、

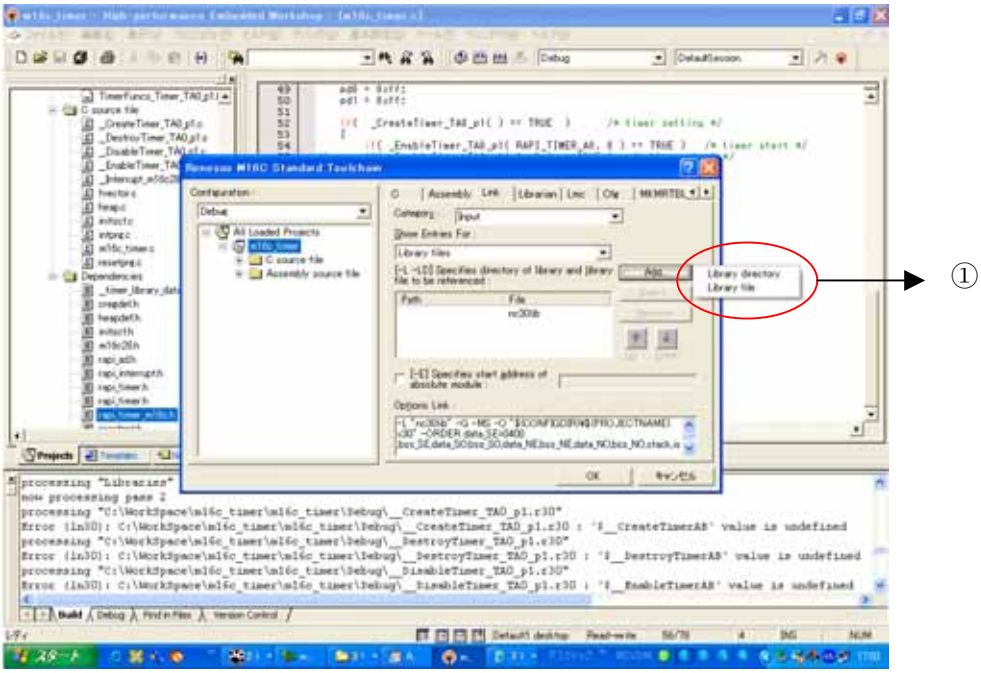
Category: Input

Show Entries For: Library files

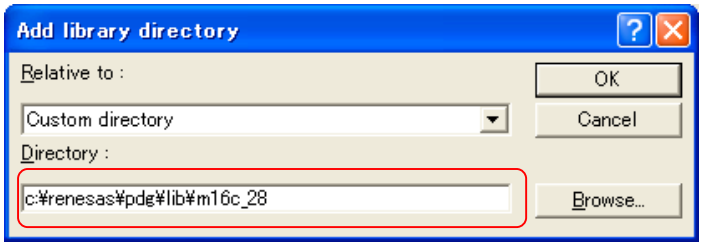
をそれぞれ選択します。

① で示した Add ボタンを押します。

(2) API ライブラリが格納されているディレクトリを指定します。



①でまず、Library directory を選択します。



Relative to : Custom directory　を指定する。

Directory : API ライブラリの格納ディレクトリを指定する。

C:¥Renesas¥PDG¥lib¥M16C_28

ライブラリは、Peripheral Driver Generator をインストールしたディレクトリ下にあります。

なお、M16C/Tiny,R8C/Tiny,H8 300H/Tiny それぞれのライブラリ格納ディレクトリは以下の通りです。

M16C/28 : lib¥m16c_28

H8 /3687 : lib¥h8_3687

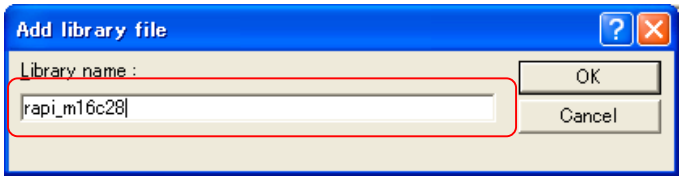
R8C/13　:　lib¥r8c_13

ライブラリのディレクトリ名入力後、OK を押します。

(2) API ライブラリ名を指定します。

再度 Add ボタンを押して、今度は

①で、Library file を選択します。



ライブラリ名を入力します。(拡張子 .lib は付けないでください。)

なお、M16C/Tiny,R8C/Tiny,H8 300H/Tiny それぞれのライブラリ名は以下の通りです。

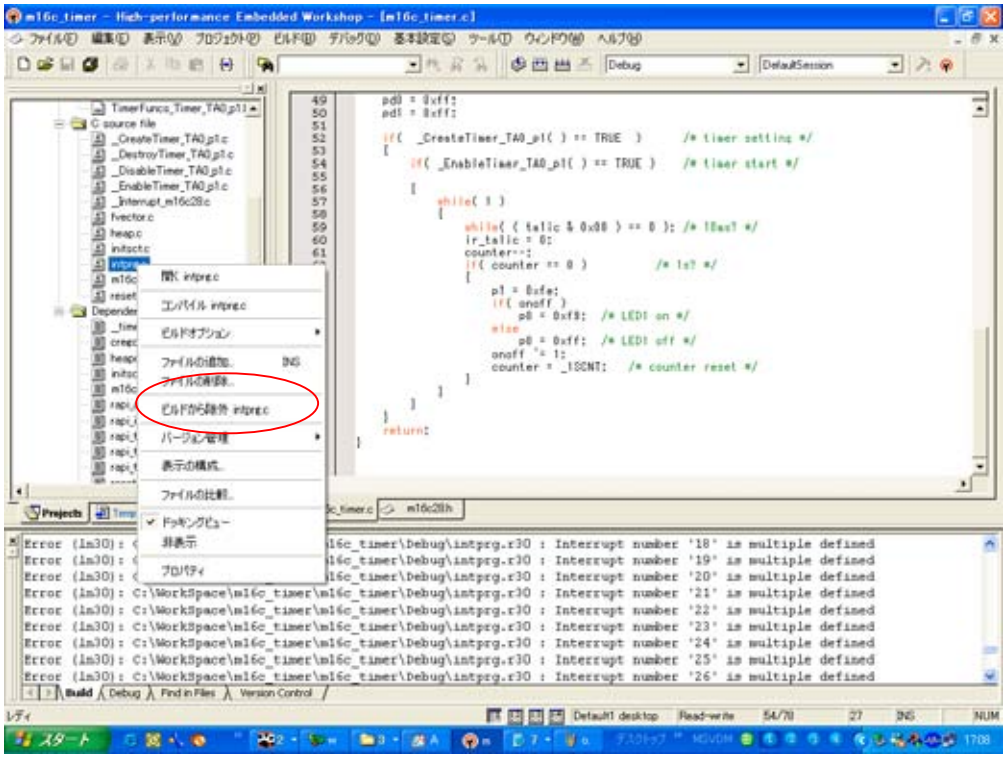
M16C/28 : rapi_m16c28.lib

H8 /3687 : rapi_h83687.lib

R8C/13　: rapi_r8c13.lib

7.4.4. 割り込みベクタエントリファイルの除外

High-performance Embedded Workshop でワークスペースを作成すると、スタートアップファイルの１つ intprg.c が登録されます。（H8 300H/Tiny の場合は、ワークスペース作成ウィザードで intprg.c の作成有無の確認あり）
Peripheral Driver Generator も割り込みベクタ関数を作成するため、ベクタが intprg.c と重複します。
そのため、intprg.c をビルド対象から除外する必要があります。



intprg.c を選択した状態で右クリックします。
メニューが開きますので、その中から“ ビルドから除外 intprg.c”を選択します。

アセンブラスタートアップ ncrt0.a30 ，sect30.inc を使用している場合は、「6.2.3 割り込みベクタテーブルの除外」を参照してください。

以上で全ての設定が完了です。

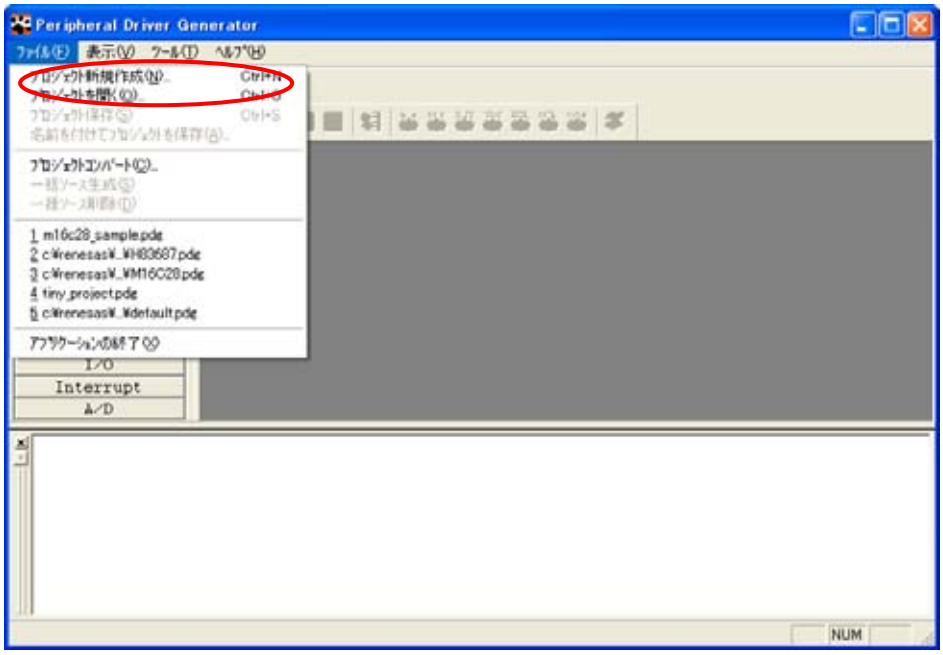
8. コンバート

ここでは、Peripheral Driver Generator の機能の１つであるコンバートについて説明します。

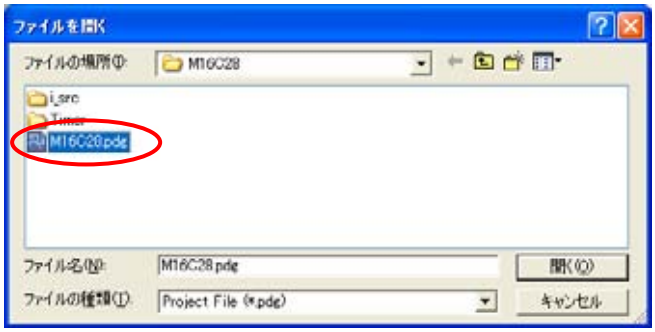
７章で作成した環境を H8/3687 用に変換してみます。

8.1. 既存プロジェクトを開く

７章で作成したプロジェクトをオープンします。



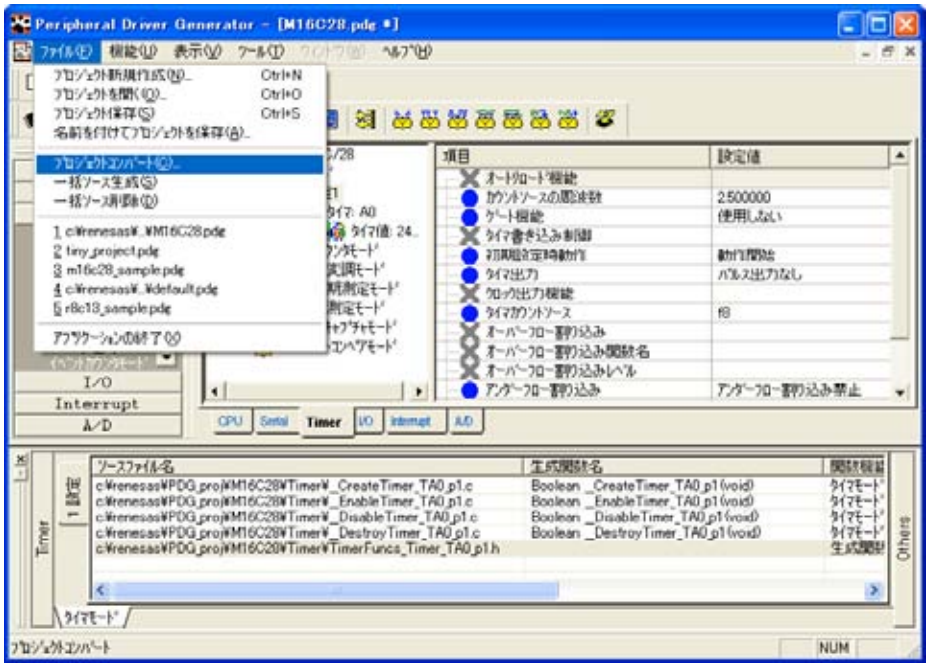
ファイルメニューの“ プロジェクトを開く(O)...”を選択します。



７章で作成した M16C/28 用プロジェクト M16C28.pdg を選択します。

8.2. コンバート

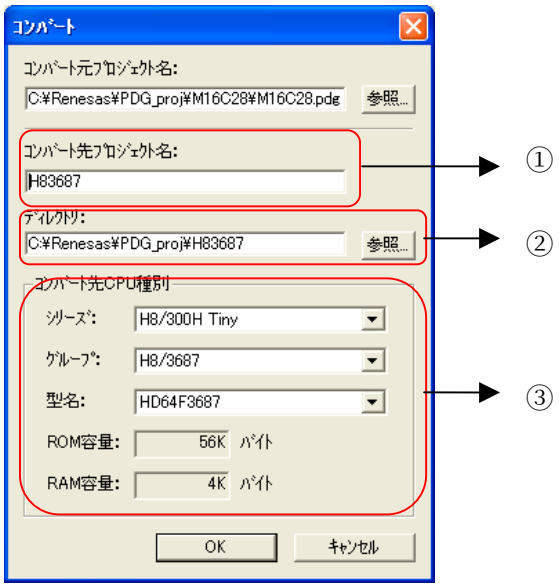
8.2.1. コンバート開始



ファイルメニューの“ プロジェクトコンバート(C)...”を選択するか、ツールバーの  をクリックします。

8.2.2. コンバート先プロジェクトの作成

8.2.1 の作業を行うと、コンバートウィンドウが開きます。



① コンバート先のプロジェクト名を入力します。

② ①で作成したプロジェクトの格納ディレクトリを指定します。

③ コンバートする CPU を選択します。

シリーズ名：H8/300H Tiny

グループ名：H8/3687

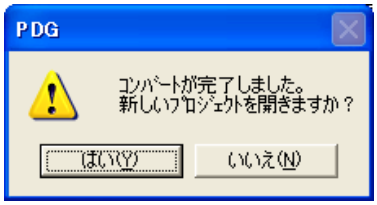
型名　　：HD64F3687

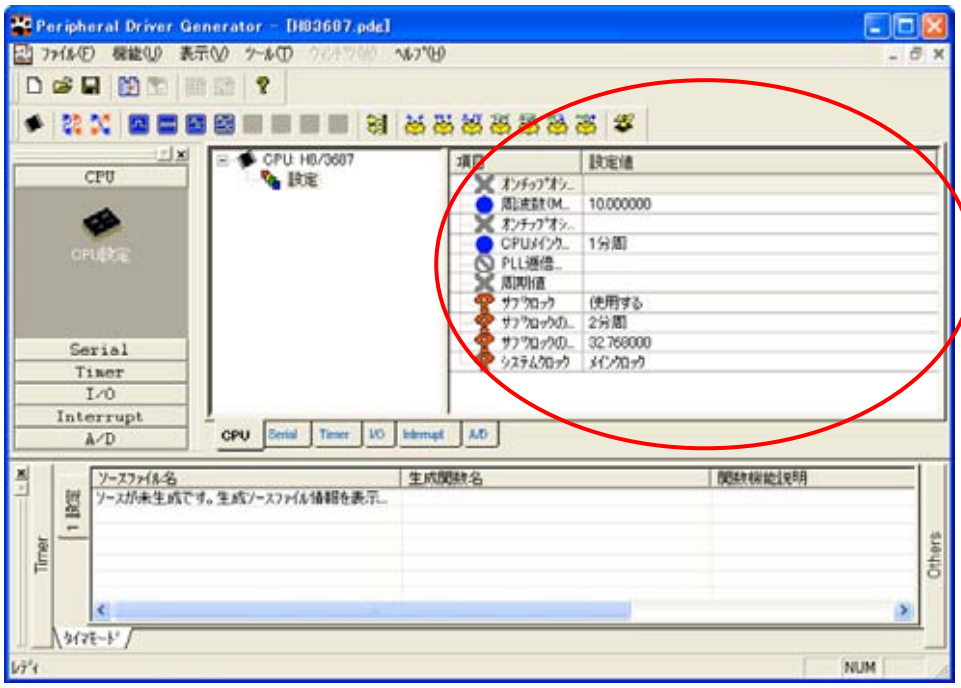
をそれぞれ選択します。

OK ボタンを押すと、コンバート処理を開始します。

8.2.3. コンバート後の編集

コンバート処理が終了すると、完了したことを知らせるダイアログが開きますので、“ はい ” を選択してコンバート後のプロジェクトをオープンさせます。





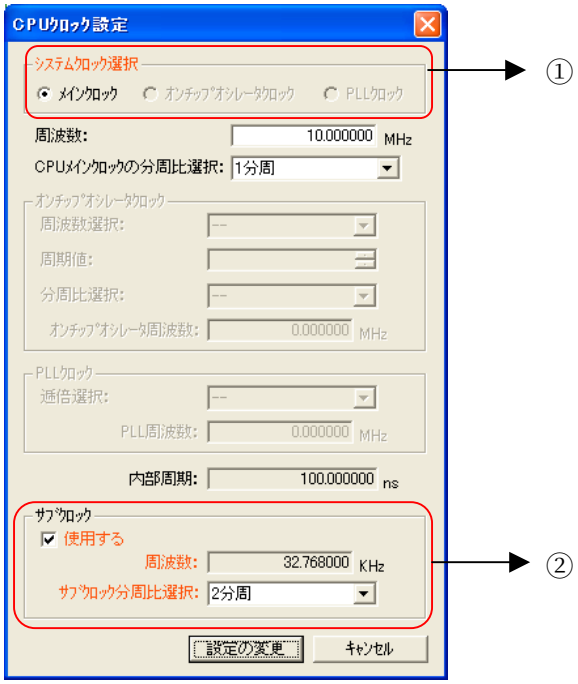
コンバート前の CPU 及び設定内容により、コンバート後に設定が無効になる項目、または再設定が必要になる項目が発生する場合があります。

このような項目は、上記 ◯ で示した表示で確認するこができます。

- コンバート前の設定がそのままコンバート後も設定されたことを意味します。
- ✖ コンバート前/後共に設定不可の項目を意味します。
- ⃠ コンバート前に設定した項目が、コンバートの際に選択した CPU ではサポートしていない項目であったため無効になったことを意味します。
- ? コンバート後の CPU の仕様が異なる等により再度設定が必要もしくは確認が必要な項目を意味します。

? で示された項目に対して再設定が必要であるため、? をクリックします。

CPU クロックに対する再設定項目なので、“ CPU クロック設定 ” ダイアログが開きます。

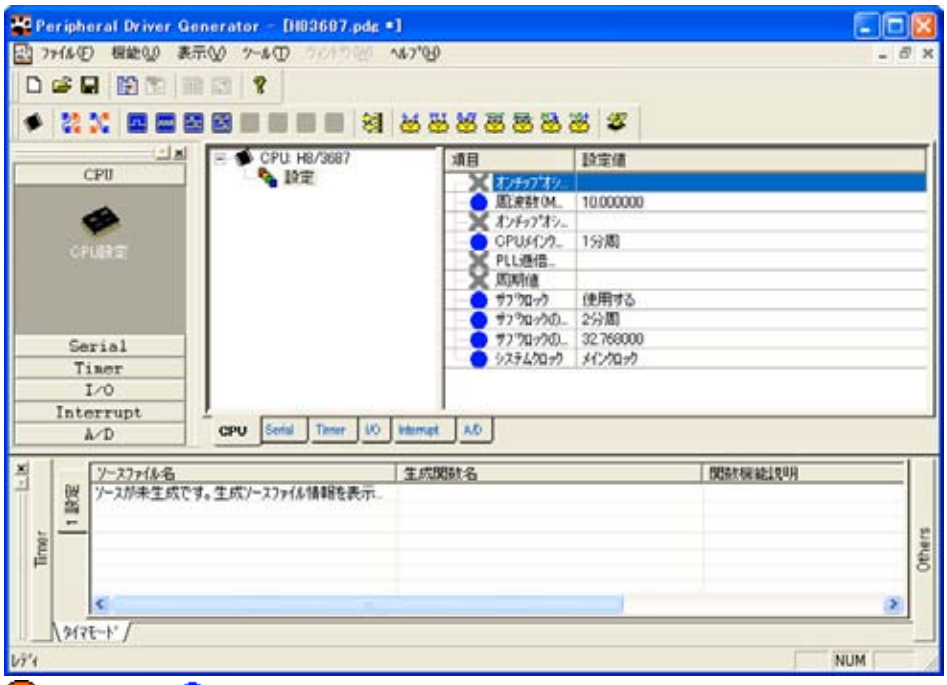


赤文字で示した項目が再設定もしくは確認が必要な項目です。

① 第 7 章では、M16C/28 のシステムクロックは PLL クロックを選択しました。
しかし、H8/3687 では PLL クロックが存在しません。そのため強制的にメインクロックを選択していることを意味しています。

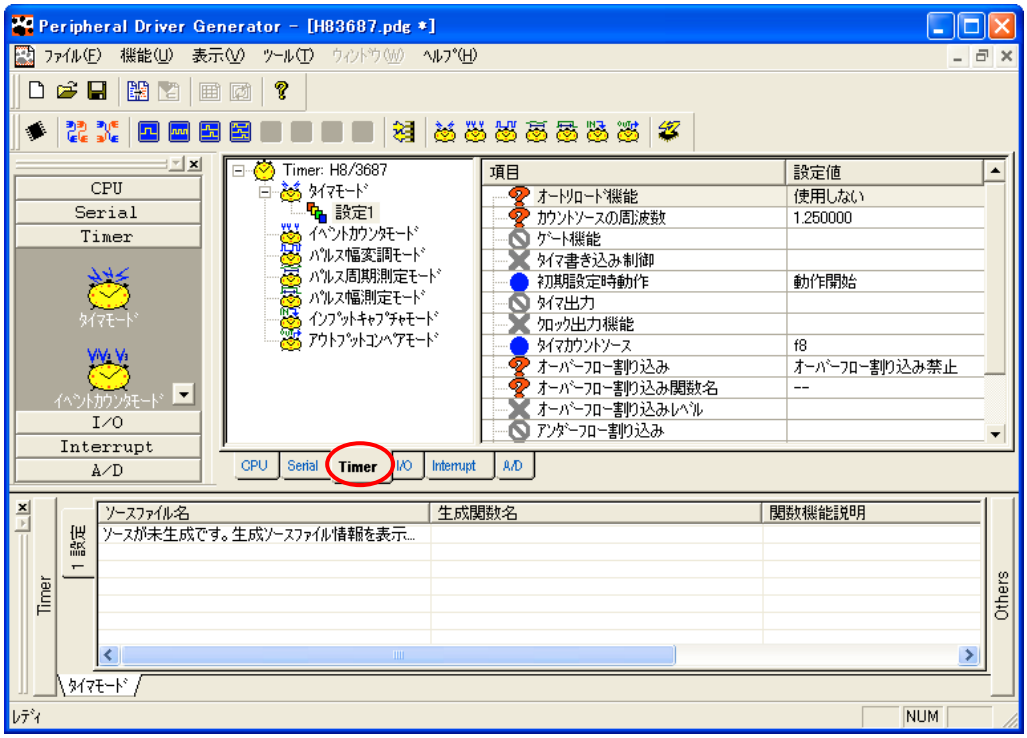
② H8/3687 では、サブクロックという概念がありません。また分周も 2、４，８からしか選択できないため 2 分周を Peripheral Driver Generator は選択しています。

この設定でよければ、“ 設定の変更 ” ボタンを押します。



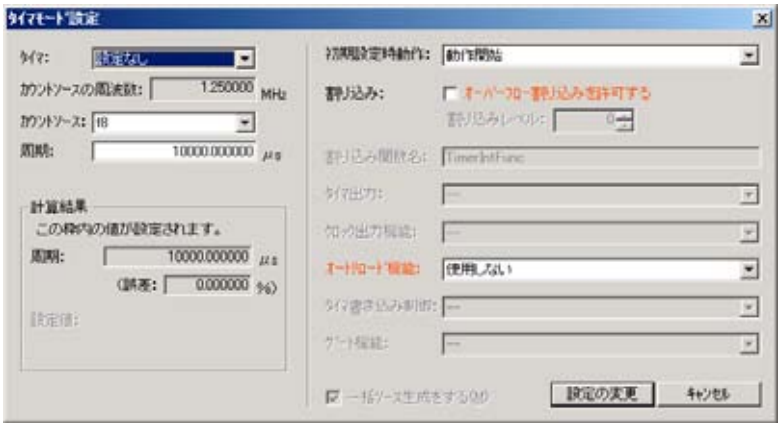
マークが、●マークに変わります。

第 7 章では、タイマモードの設定を行っているので、同様にタイマモードの確認も行う必要があります。



タイマタブを選択して、タイマの設定内容表示画面へ切り替えます。

CPU の設定内容変更時と同様に  をクリックします。



タイマの設定が“ 設定なし ”になっているので、H8/3687 で選択可能な B1 もしくは V のいずれかを選択します。

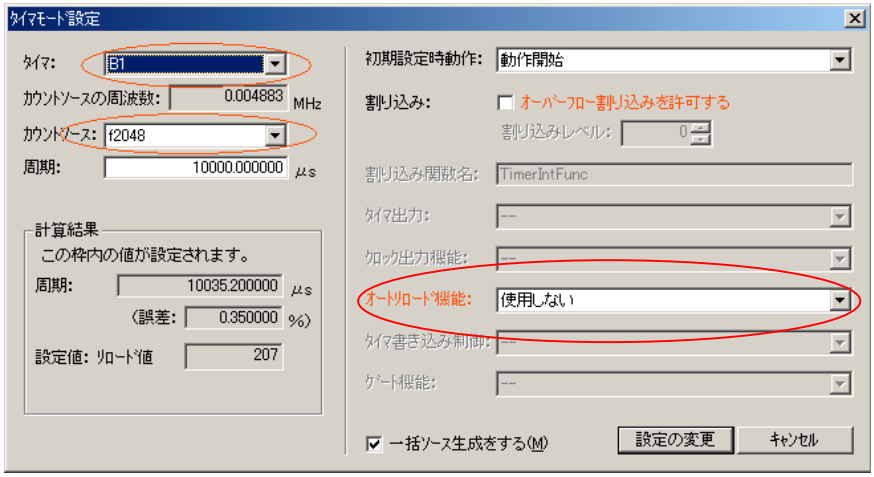
ここでは、B1 を選択します。

加えてカウントソースを選択します。

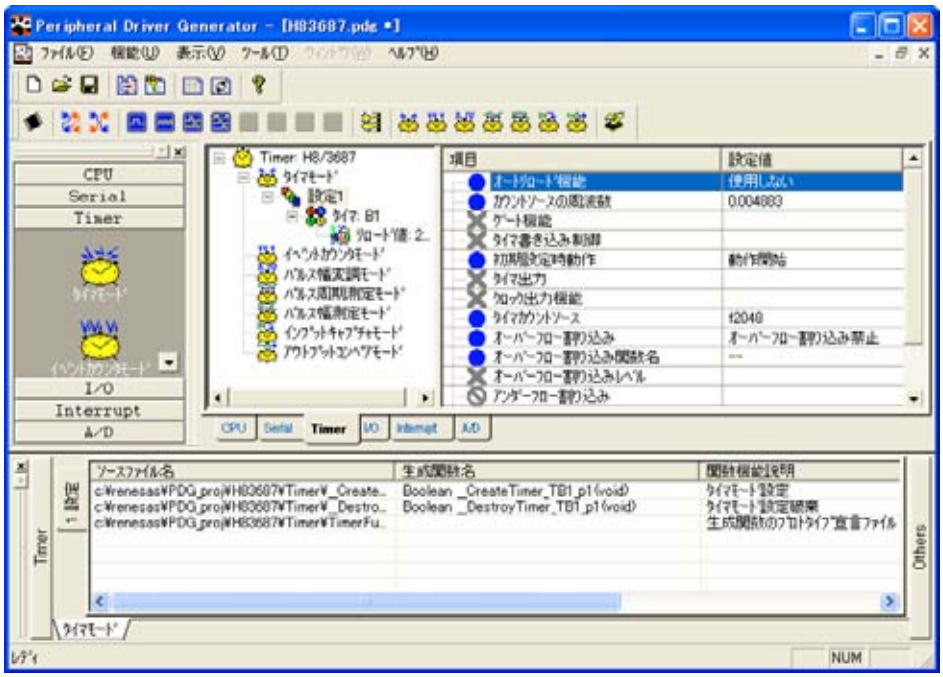
ここでは、f2048 を選択します。

更に、M16C/28 には無いオートリロード機能が H8/3687 にはあります。

この機能に対して、使用する/使用しないのいずれかを選択します。



この設定で“設定の変更”ボタンをクリックします。



マークが、●マークに変わります。

これでコンバート作業は全て終了です。

9. High-performance Embedded Workshop への登録

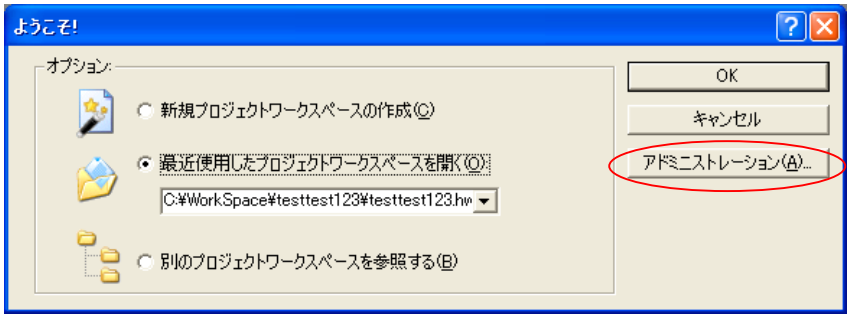
High-performance Embedded Workshop から Peripheral Driver Generator を起動、及び Peripheral Driver Generator で生成したソースファイルを High-performance Embedded Workshop のワークスペースへ登録するために必要な処理を説明します。

9.1. hrf ファイルの登録

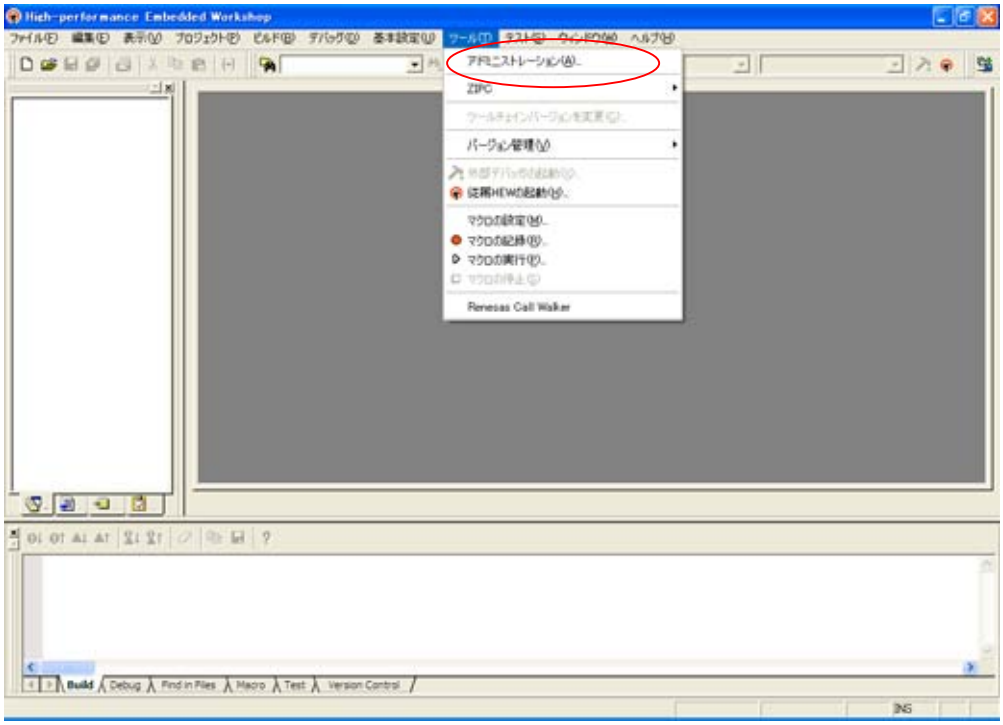
(1)High-performance Embedded Workshop を立ち上げます。

既に立ち上がっている場合は、全てのワークスペースを終了してください。

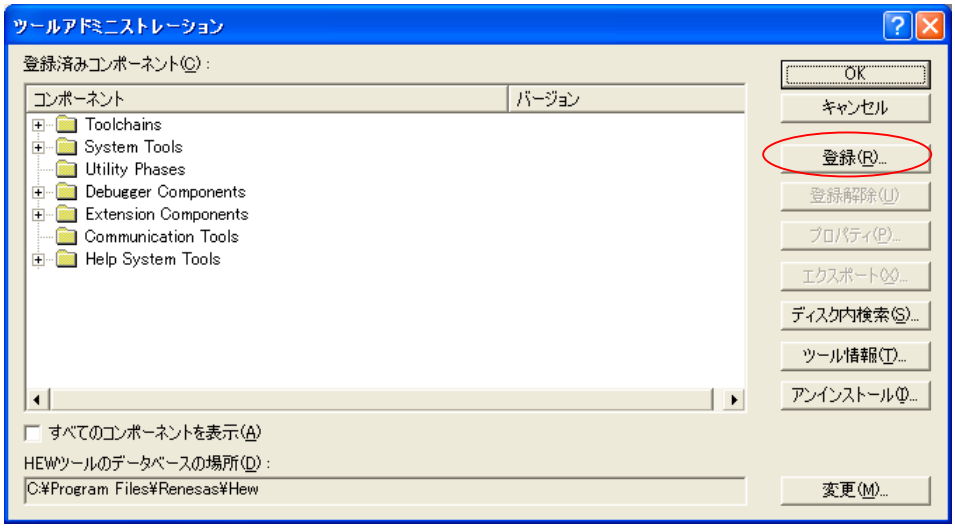
(2)ダイアログの「アドミニストレーション(A)...」を選択する



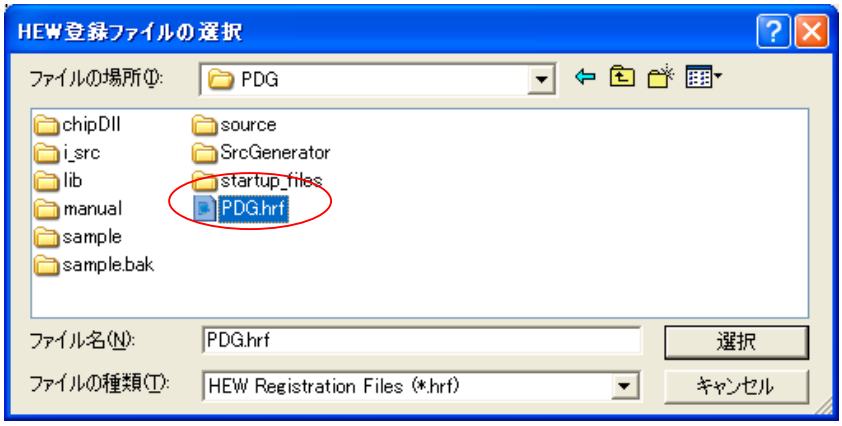
(3)既に High-performanceEmbeddedorkshop を起動している場合は、ツールメニューの「アドミニストレーション(A)...」を選択する。



(4)登録ボタンを押す

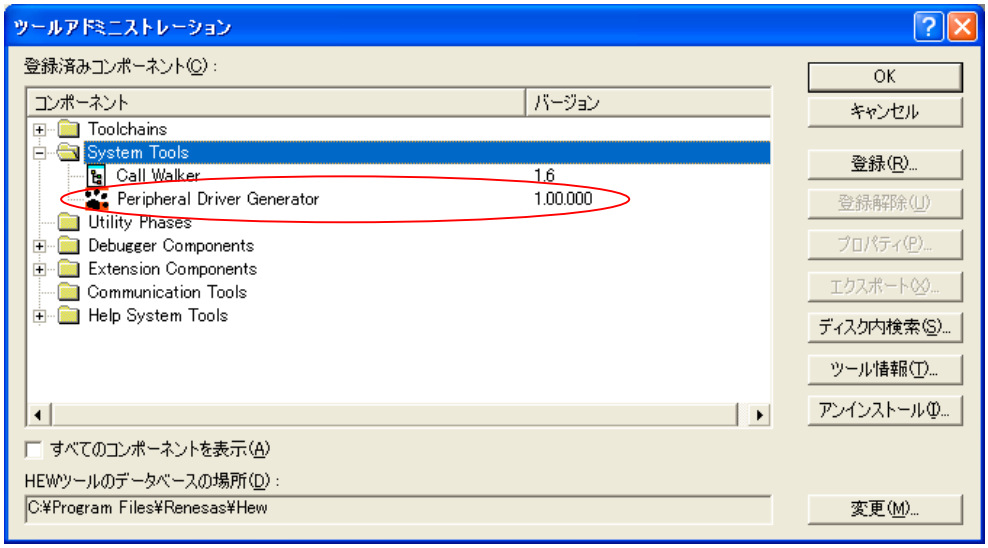


(5)Peripheral Driver Generator をインストールしたディレクトリにある “PDG.hrf”ファイルを選択する。



※Peripheral Driver Generator のデフォルトインストールディレクトリは、”c:¥renesas¥PDG”です

(6)アドミニストレーションダイアログで、Peripheral Driver Generator が登録されていることを確認する。



System Tools に

Peripheral Driver Generator　　1.01.000

が表示されていることを確認する。

(7) OK ボタンを押して登録を終了する。

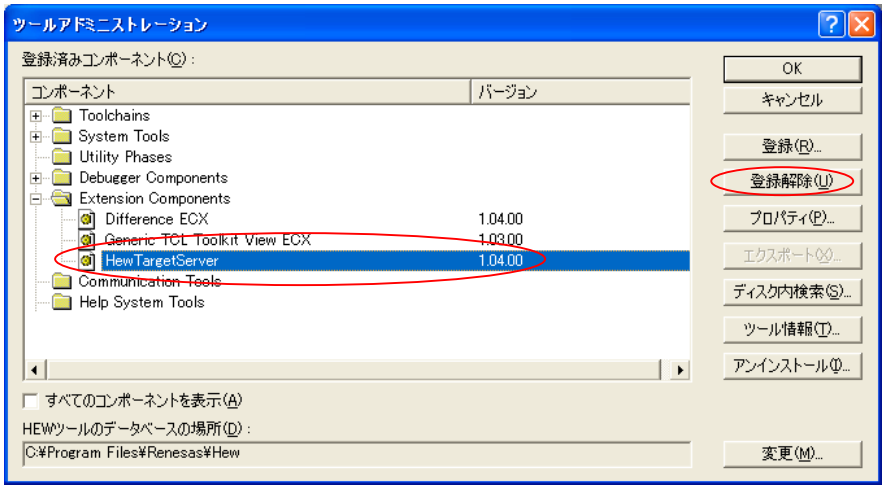
9.2. HewTargetServer の登録

(1) HEW インストールフォルダにある REGISTERSERVER.bat を実行する。
デフォルトインストールディレクトリの場合は、
c:¥Program Files¥Renesas¥Hew¥REGISTERSERVER.bat
を実行する。

(2)ツールメニューの「アドミニストレーション(A)...」を選択する。

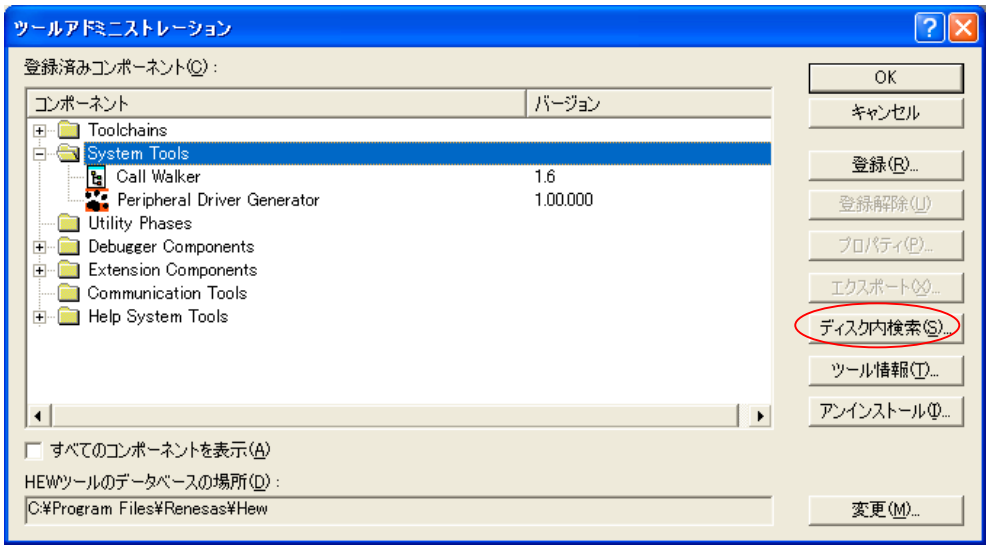
(3)Extension Components に HewTargetServer のバージョンが 1.05.000 かどうかを確認します。
1.05.00 より古いバージョンが表示されている場合は、一旦 HewTargetServer を選択した状態で

登録解除ボタンを押して登録を解除してください。

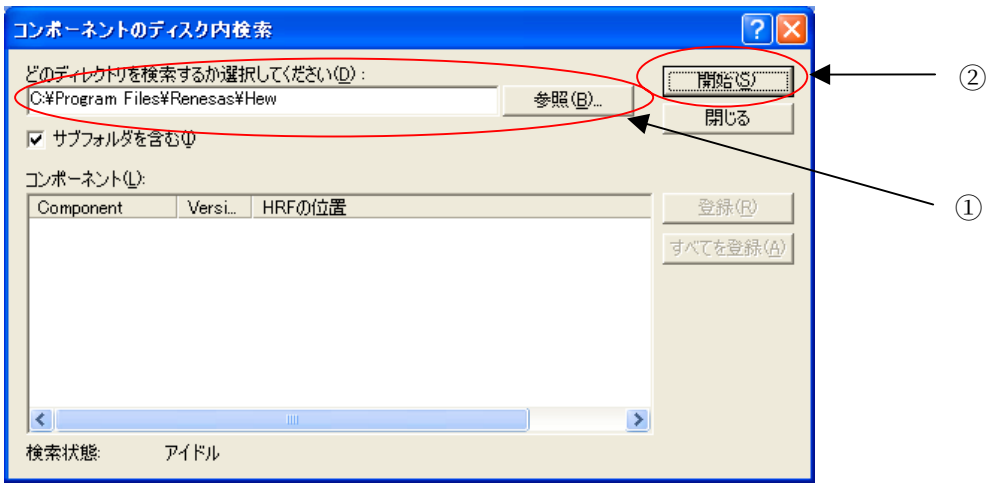


HewTargetServer 自体が表示されていない場合は、(3)は必要ありません。

(4)ツールアドミニストレーションダイアログの「ディスク内検索(S)...」ボタンを押す。

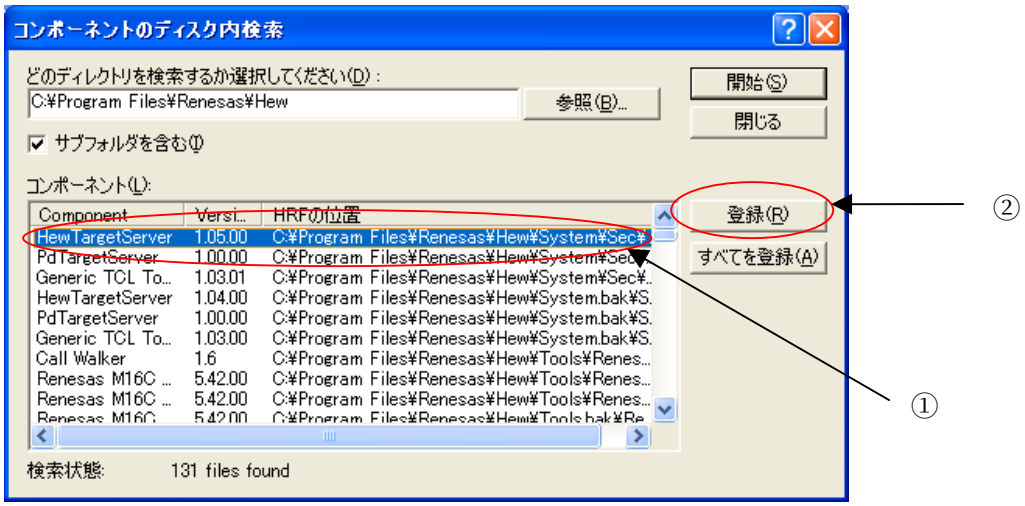


(5)「コンポーネントのディスク内検索」ダイアログで HewTargetServer を検索する。



①High-performance Embedded Workshop をインストールしたディレクトリを指定する。
② “開始”ボタンを押す

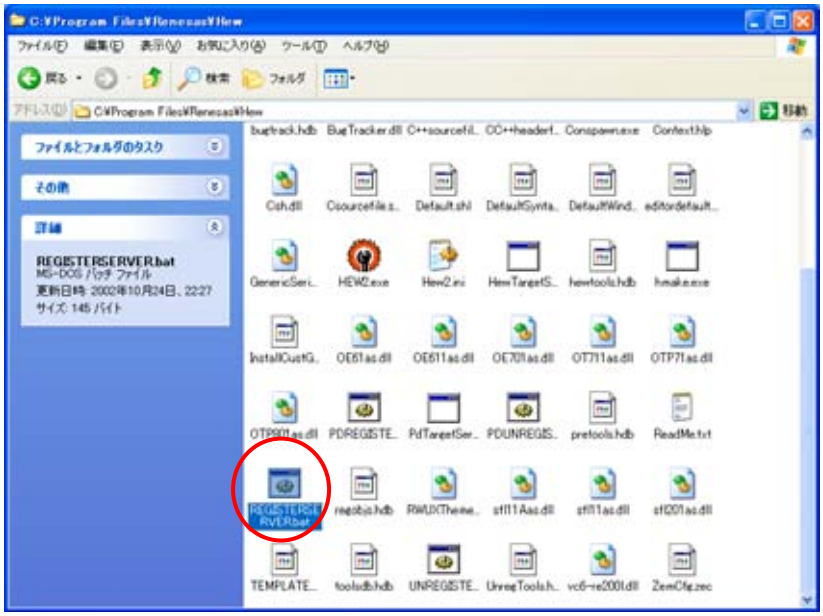
(6) HewTargetServer を登録する。



①HewTargetServer を選択する。
② 登録ボタンを押す
(6) 全てのダイアログを閉じて完了

9.3. REGISTERSERVER.bat の実行

High-performance Embedded Workshop をインストールしたディレクトリ直下にある REGISTERSERVER.bat を実行してください。



これで全ての作業は完了です。

10. API ライブラリの MISRA C ルール適合に関して

10.1. API ライブラリ

Peripheral Driver Generator の API ライブラリの C ソースコードは、MISRA C ルールに対して
H8 300H/Tiny 用 API ライブラリ：24
M16C/Tiny 用 API ライブラリ：12
R8C/Tiny 用 API ライブラリ：22
のルール違反[1]が認められますが、これらの違反は動作に支障がありません。

10.1.1. ルール違反の要因

Peripheral Driver Generator の API ライブラリの C ソースコードにおいて、ルール違反となった主な要因は次の通りです。

- typedef を用いて宣言を行っていない。
- ビットフィールドメンバの型は、signed int/unsigned int のいずれかでなければならない
- char の代わりに unsigned char/signed char と宣言しなければならない
- 関数の出口(exit)は一箇所でなければならない
- ポインタへ CAST してはならない
- 引数を持たない関数は,void で宣言しなくてはならない

10.1.2. ルール違反となった検査番号

ルール違反になった検査番号は次のとおりです。

[H8 300H/Tiny]

1	13	14	18	22	36	37	43	45	46
49	54	59	60	71	76	77	82	99	101
104	110	111	113						

[M16C/Tiny]

1	13	14	18	22	37	43	60	82	99
110	111								

[R8C/Tiny]

1	13	14	18	21	22	36	37	38	43
45	46	49	59	77	82	99	101	104	110
111	113								

10.1.3. 評価環境

コンパイラ	M3T-NC30WA V.5.40 Release 00
コンパイルオプション	-OR -ONA
MISRA C チェッカ	SQMlint V.1.03 Release 00

[1] MISRA C ルールチェッカ SQMLint による検査結果値です。

コンパイラ	H8S,H8/300 Standard Toolchain (V.6.1.2.0)
コンパイルオプション	サイズ優先(-optimize=1)
MISRA C チェッカ	SQMlint V.1.03 Release 00

周辺I/O設定ソフトウエア
ガイドブック
Peripheral Driver Generator

発行年月日　2006**年**01**月**16**日**　Rev.1.00

発行　　**株式会社 ルネサス テクノロジ　営業企画統括部**
〒100-0004　**東京都千代田区大手町**2-6-2

編集　　**株式会社 ルネサス ソリューションズ　ツール開発部**

Peripheral Driver Generator V.1.01.000
ガイドブック

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

RJJ10J1905-0100