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.1.03

ガイドブック

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2008.09

本資料ご利用に際しての留意事項

１．本資料は、お客様に用途に応じた適切な弊社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弊社または第三者の知的財産権その他の権利の実施、使用を許諾または保証するものではありません。
２．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など全ての情報の使用に起因する損害、第三者の知的財産権その他の権利に対する侵害に関し、弊社は責任を負いません。
３．本資料に記載の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、あるいはその他軍事用途の目的で使用しないでください。また、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、それらの定めるところにより必要な手続を行ってください。
４．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などの全ての情報は本資料発行時点のものであり、弊社は本資料に記載した製品または仕様等を予告なしに変更することがあります。弊社の半導体製品のご購入およびご使用に当たりましては、事前に弊社営業窓口で最新の情報をご確認頂きますとともに、弊社ホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意下さい。
５．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、弊社はその責任を負いません。
６．本資料に記載の製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を流用する場合は、流用する情報を単独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下さい。弊社は、適用可否に対する責任は負いません。
７．本資料に記載された製品は、各種安全装置や運輸・交通用、医療用、燃焼制御用、航空宇宙用、原子力、海底中継用の機器・システムなど、その故障や誤動作が直接人命を脅かしあるいは人体に危害を及ぼすおそれのあるような機器・システムや特に高度な品質・信頼性が要求される機器・システムでの使用を意図して設計、製造されたものではありません（弊社が自動車用と指定する製品を自動車に使用する場合を除きます）。これらの用途に利用されることをご検討の際には、必ず事前に弊社営業窓口へご照会下さい。なお、上記用途に使用されたことにより発生した損害等について弊社はその責任を負いかねますのでご了承願います。
８．第７項にかかわらず、本資料に記載された製品は、下記の用途には使用しないで下さい。これらの用途に使用されたことにより発生した損害等につきましては、弊社は一切の責任を負いません。
　１）生命維持装置。
　２）人体に埋め込み使用するもの。
　３）治療行為（患部切り出し、薬剤投与等）を行なうもの。
　４）その他、直接人命に影響を与えるもの。
９．本資料に記載された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、実装条件およびその他諸条件につきましては、弊社保証範囲内でご使用ください。弊社保証値を越えて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、弊社はその責任を負いません。
１０．弊社は製品の品質および信頼性の向上に努めておりますが、特に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。弊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、誤動作防止設計などの安全設計（含むハードウエアおよびソフトウエア）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エアは、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致します。
１１．本資料に記載の製品は、これを搭載した製品から剥がれた場合、幼児が口に入れて誤飲する等の事故の危険性があります。お客様の製品への実装後に容易に本製品が剥がれることがなきよう、お客様の責任において十分な安全設計をお願いします。お客様の製品から剥がれた場合の事故につきましては、弊社はその責任を負いません。
１２．本資料の全部または一部を弊社の文書による事前の承諾なしに転載または複製することを固くお断り致します。
１３．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弊社営業窓口までご照会下さい。

はじめに

本ガイドブックでは、Peripheral Driver Generator(以降、PDGと略語で記述します)を使用していただく場合の具体的な操作例を示します。PDGの操作方法やHigh-performance Embedded Workshop(以降、HEWと略語で記述します)の操作方法などは、PDGおよびHEWのユーザーズマニュアルを参照してください。

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
コンタクトセンタcsc@renesas.com
ホームページhttp://japan.renesas.com/tools

目次

1.　概要

本章では、PDG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。
PDG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。
PDG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 I/O モジュールの設定

PDG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。

(2) ビルド/デバッグなどの環境作成

HEW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。
PDG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、SH/7125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

PDGで生成した関数を呼び出します。
PDG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PDG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) PDG で生成したソースファイルの登録

PDG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HEW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HEW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、

- 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
- リンクオプション-LなどによるAPIライブラリの指定

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。
使用するHEWのバージョンによってはソースファイルの登録時に自動で行われます。

(6) デバッグ・評価

ビルドしたアプリケーションをデバッグ・評価することで、アプリケーションを完成させます。

2.　PDGを使ってみる

本章では、PDGのサンプルプロジェクトとHEW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**PDG**と記載されている場合は、PDGの操作説明、**HEW**と記載されている場合は、HEW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、PDG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。）

2.1　HEWでワークスペースを作成［HEW］

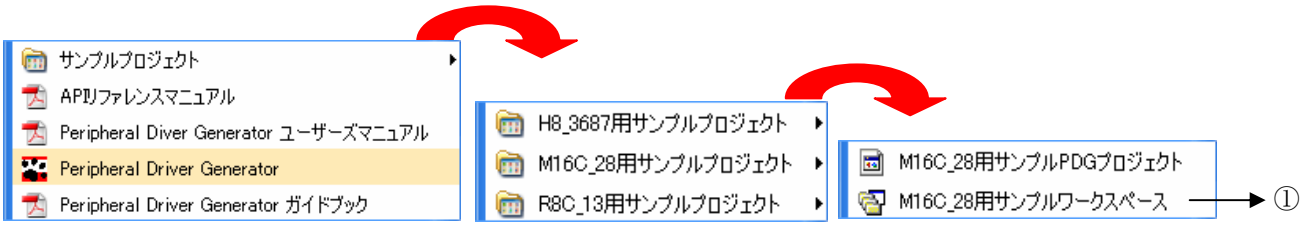
PDG パッケージで用意している HEW 用空ワークスペースを使用します。

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
・SH/7125 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。

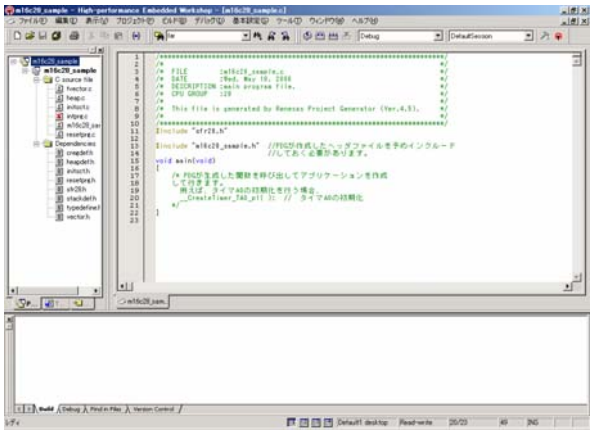
ここでは、M16C/28 用のサンプルワークスペースを使用します。



①　M16C/28 用の HEW サンプルワークスペースです。

①を選択すると、HEW が立ち上がります。HEW は立ち上げたまま次に進んでください。

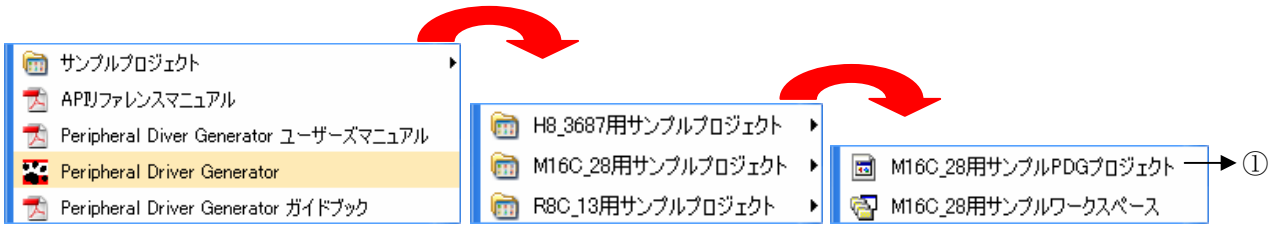
サンプルワークスペースは HEW V.4.00 で作成してあります。新しいバージョンの HEW を使用している場合、アップデートの確認メッセージが表示されますが、OK ボタンを押して開いてください。



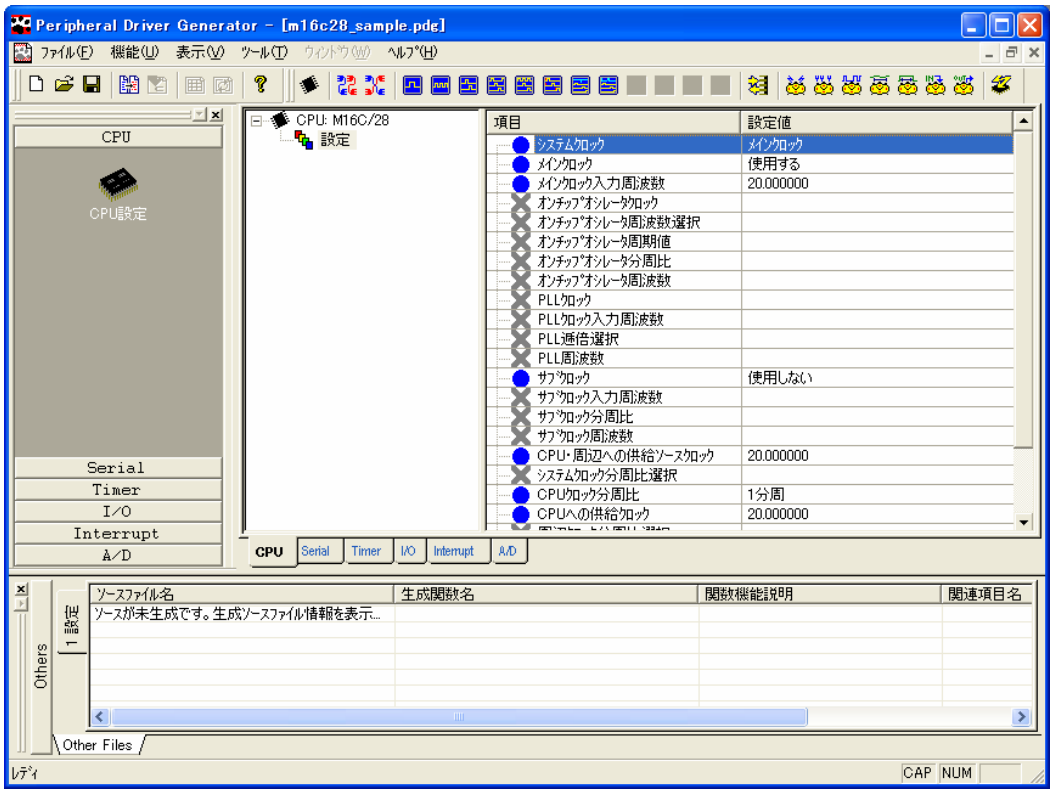
注意：H8/3687 用、M16C/28 用、および R8C/13 用サンプルワークスペースは HEW V.4.00.03 で、SH7125 用サンプルワークスペースは HEW V.4.04.01 で作成されています。それ以下のバージョンの HEW では開くことができません。また、それ以上のバージョンで開く場合、バージョンアップに伴うメッセージが表示されます。[OK]をクリックしてワークスペースを開いてください。

2.2　PDGでプロジェクト作成［PDG］

PDG パッケージで用意している PDG 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
・SH/7125 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。

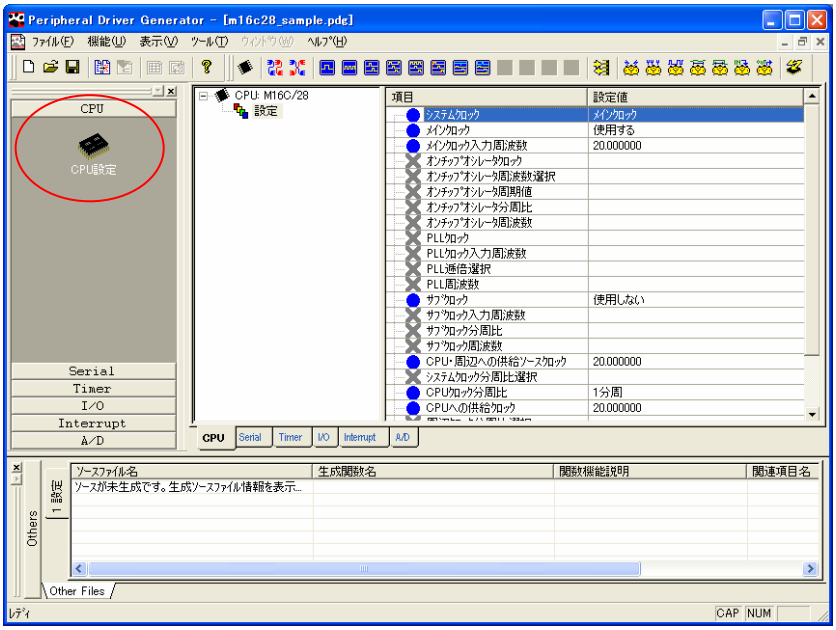


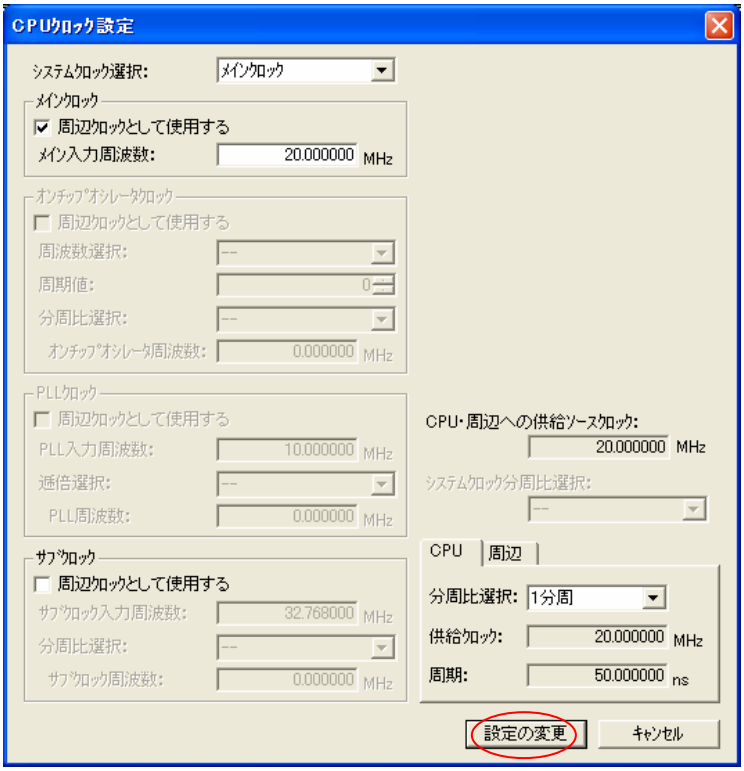
①を選択すると、PDG が立ち上がります。



2.3　周辺I/Oモジュールの設定　［PDG］

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

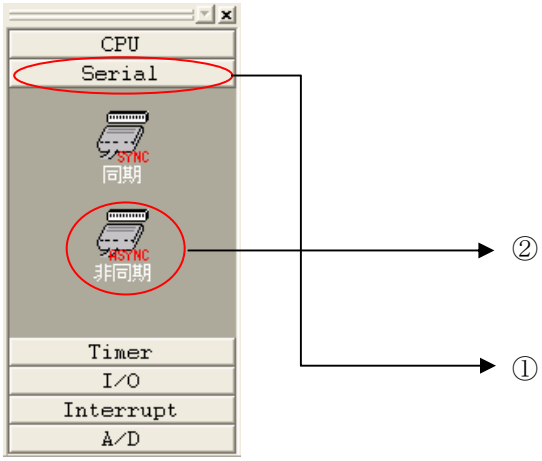




“CPU クロック設定”ダイアログが立ち上がりますので、作成するプログラムに応じて各設定を行い、“設定の変更”ボタンを押して設定を終了します。

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、

① まず、周辺設定ウィンドウから”Serial”タブを選択します。
② 次に、“非同期“を選択します。



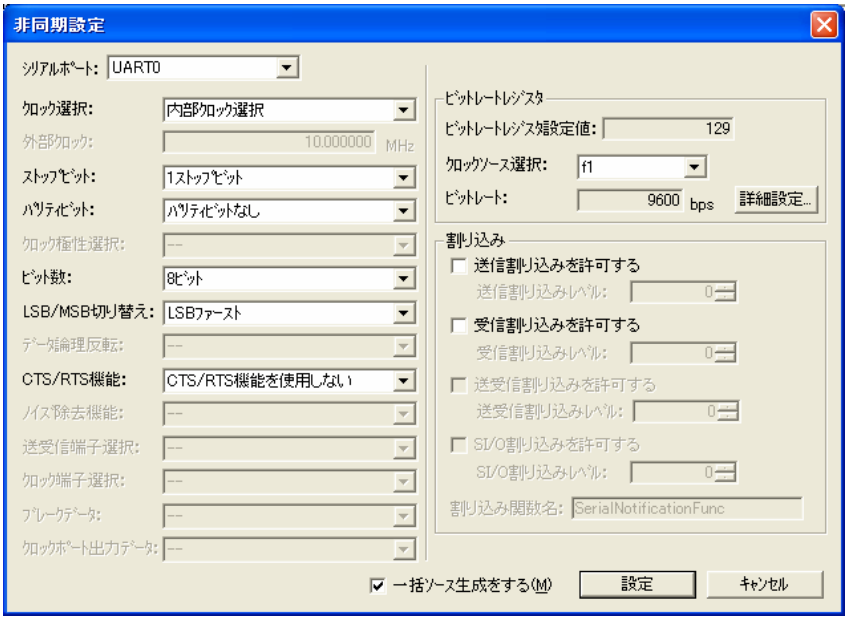
非同期設定ダイアログが開きますので、各項目（ビット数、パリティビット有無、転送ボーレートなど）を設定し、最後にシリアルポート（UART0,UART1など）を決定して入力は完了です。

□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。

※SH/Tiny では”UART 機能に必要な端子が設定されていません”というメッセージが表示され、この状態ではソース生成を行うことができません。I/O ポート設定でシリアル通信に使用する端子の設定をする必要があります。設定方法を次に示します。



SH/Tiny ではシリアルなど端子を使用する機能を設定した場合、I/O ポート設定で端子機能の選択を行います。

例えばシリアルのチャネル0を設定した場合、以下の手順で対応する端子の機能をシリアルに設定します。

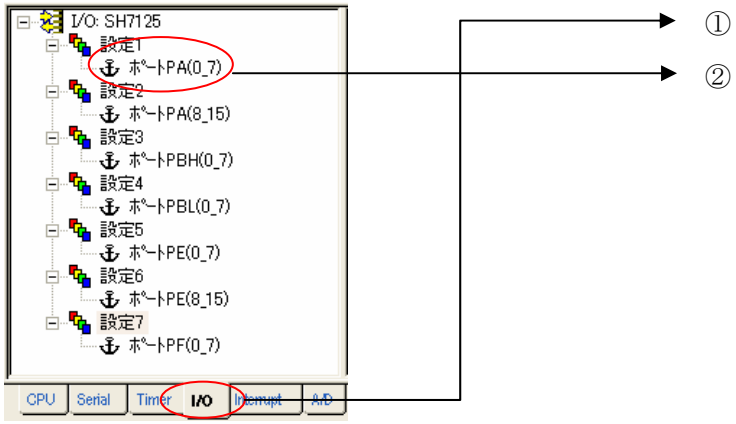
① 周辺設定ウィンドウから”I/O”タブを選択します。

② シリアル通信のチャネル 0 に対応する I/O ポートは次の通りです。

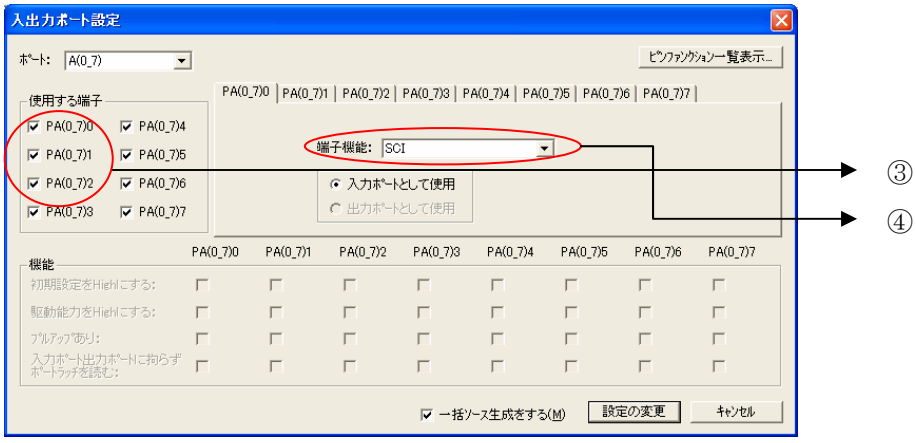
RXD0：PA0、PA10 または PE1
TXD0：PA1、PA11 または PE2
SCK0：PA2、PA12 または PE3

シリアル通信では RXD と TXD のいずれかと、SCK を設定してください。
ここでは RXD0:PA0、TXD0:PA1、SCK0:PA2 と指定します。

周辺設定ウィンドウの左側画面で、ポート PA(0_7)をダブルクリックします。[入出力ポート設定]ダイアログが表示されます。



③ [使用する端子]の [PA(0_7)0]、[PA(0_7)1]、[PA(0_7)2] をチェックします。それぞれ PA0、PA1、PA2 に対応します。チェックするとそれぞれの端子に対応した設定タブが表示されます。

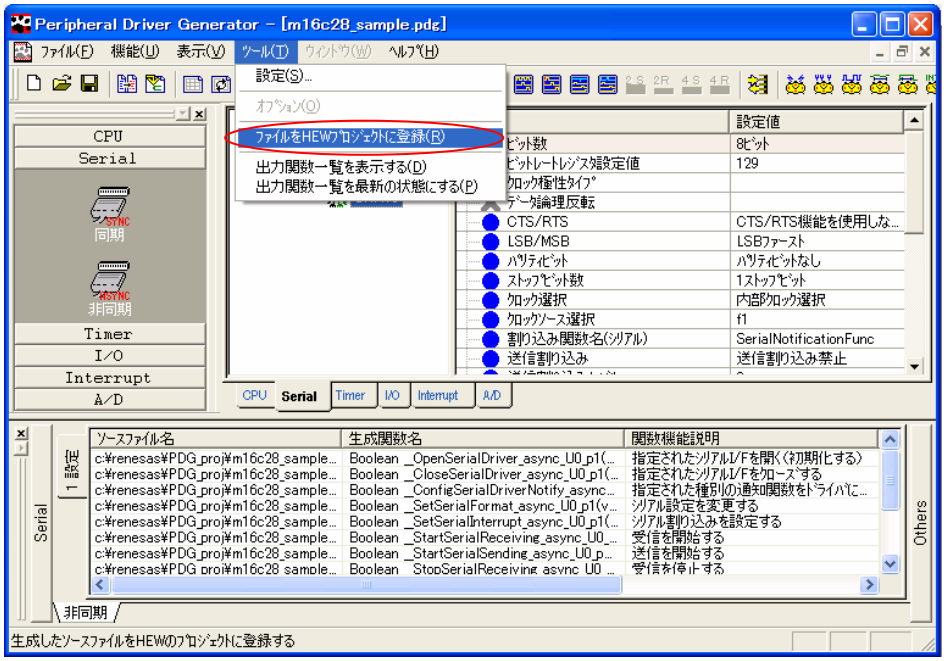


④ 各端子の設定タブで、端子機能を[SCI]に設定します。

⑤ [設定の変更]ボタンをクリックすると、ソースの生成を行います。

2.4　出力Cソースファイルを登録［PDG］

“２.３　周辺I/Oモジュールの設定“で生成したCソースをHEW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、PDG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。

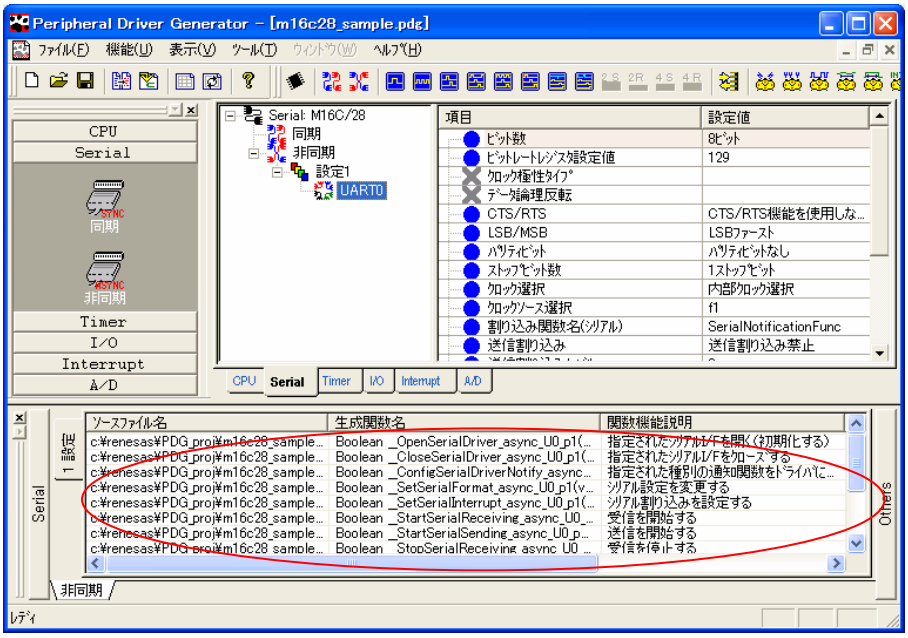


注）

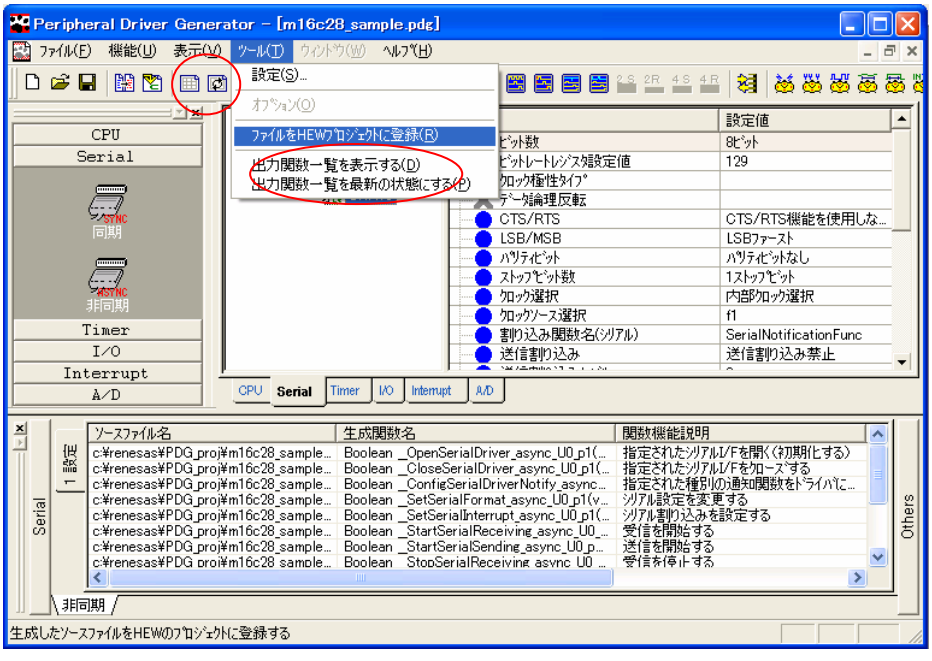
- HEWのワークスペースへPDGが生成したソースファイルを登録すると、PDGからファイル登録を解除することができません。登録を解除する場合は、HEW側から解除してください。
- HEWのワークスペース作成時に選択したCPU グループとPDGのプロジェクト作成時に選択したCPUグループが同じであるか確認の上、ソースファイルを登録してください。

2.5　作成した関数の確認［PDG］

“２.３　周辺I/Oモジュールの設定“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。

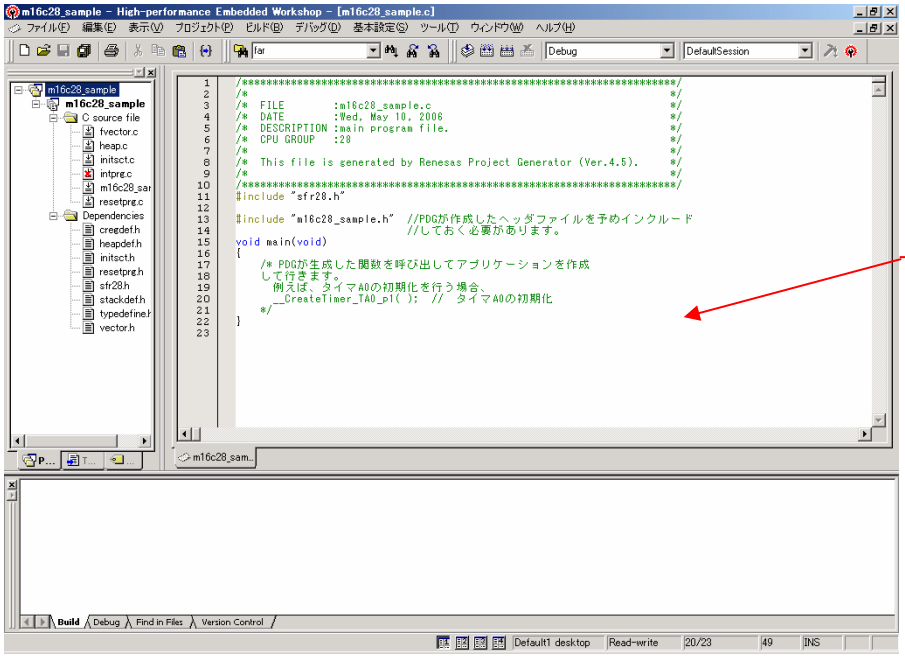


ツールメニューの“出力関数一覧を表示する”を選択もしくはボタンを押すことで、csv形式のファイルで出力関数一覧を見ることができます。拡張子.csvが関連付けられたアプリケーションが起動します。



2.6　アプリケーションの作成［HEW］

アプリケーションの作成は、HEWを使用して行います。



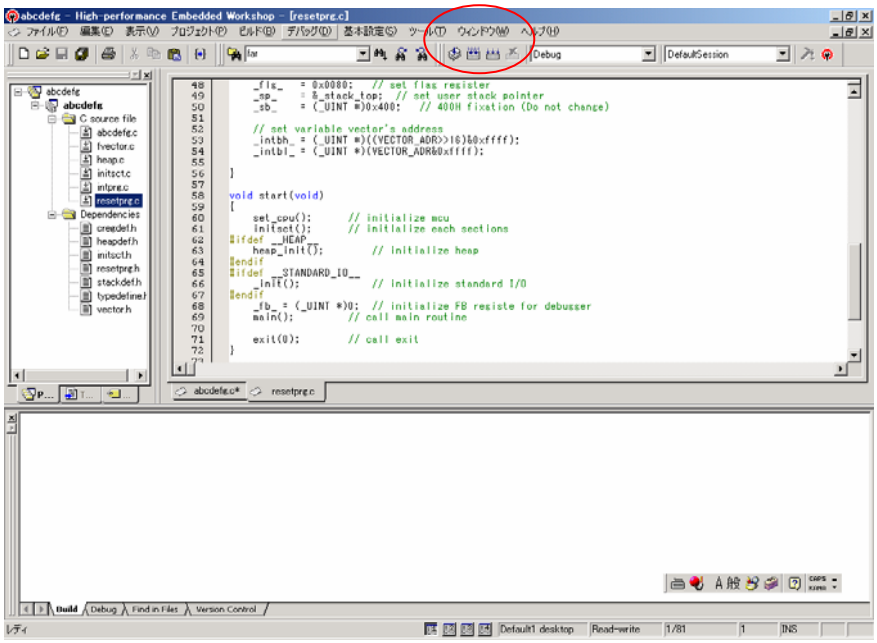
ここにプログラムを記述していきます。

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
```

2.7　コンパイル／リンク［HEW］

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HEWのビルドボタンを押すことで行います。



※

(1) HEW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（PDG をインストールしたディレクトリ下のlib¥M16C_Tiny, lib¥H8_Tiny, lib¥R8C_Tiny, lib¥SH_Tiny 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、「3.2 ライブラリの指定」を参照してください。

(2) M16Cシリーズ用Cコンパイラパッケージ M3T-NC30WA V.5.40 Release 00を使用している場合にビルドを行うと、依存関係の更新でエラーとなることがあります。V.5.42へバージョンを変更するか、ライブラリの指定方法を変更してください。

これで全ての作業は完了です。

2.8　実行［HEW］

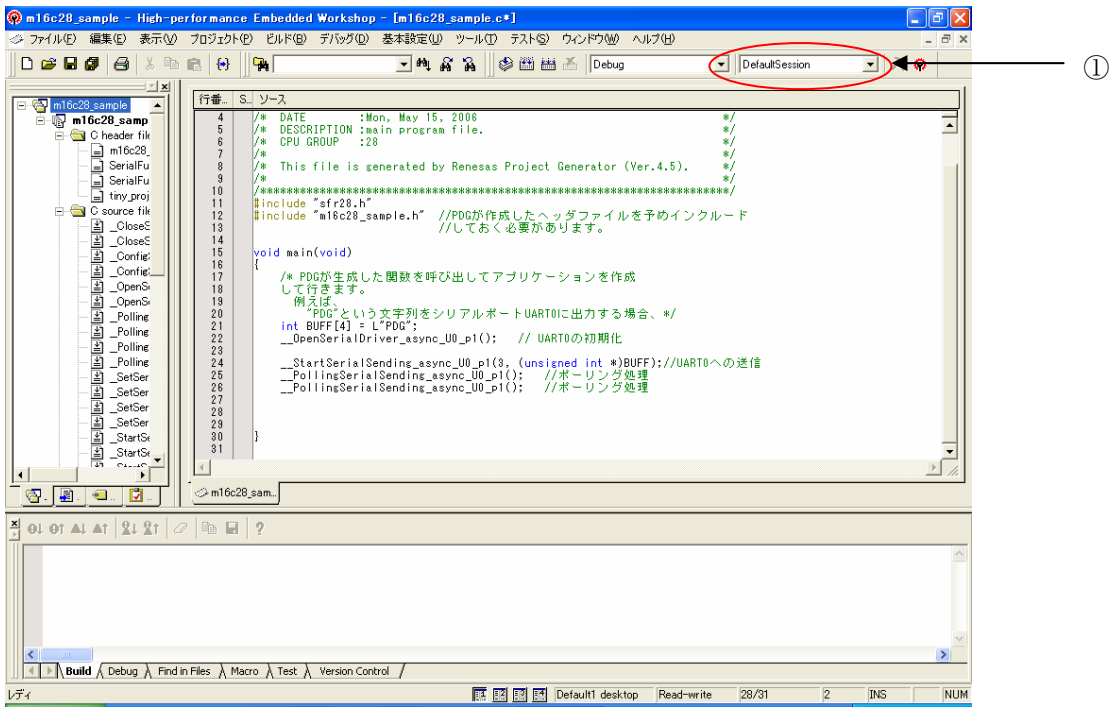
ここまでの作業を行って、シミュレータを使用して動作確認する場合について説明します。
実行するには、前準備として、ここまで説明してきた以下の処理が必要です。

(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) PDGを使用して生成ソースファイルをHEWへ登録する。
(3) ビルドする。

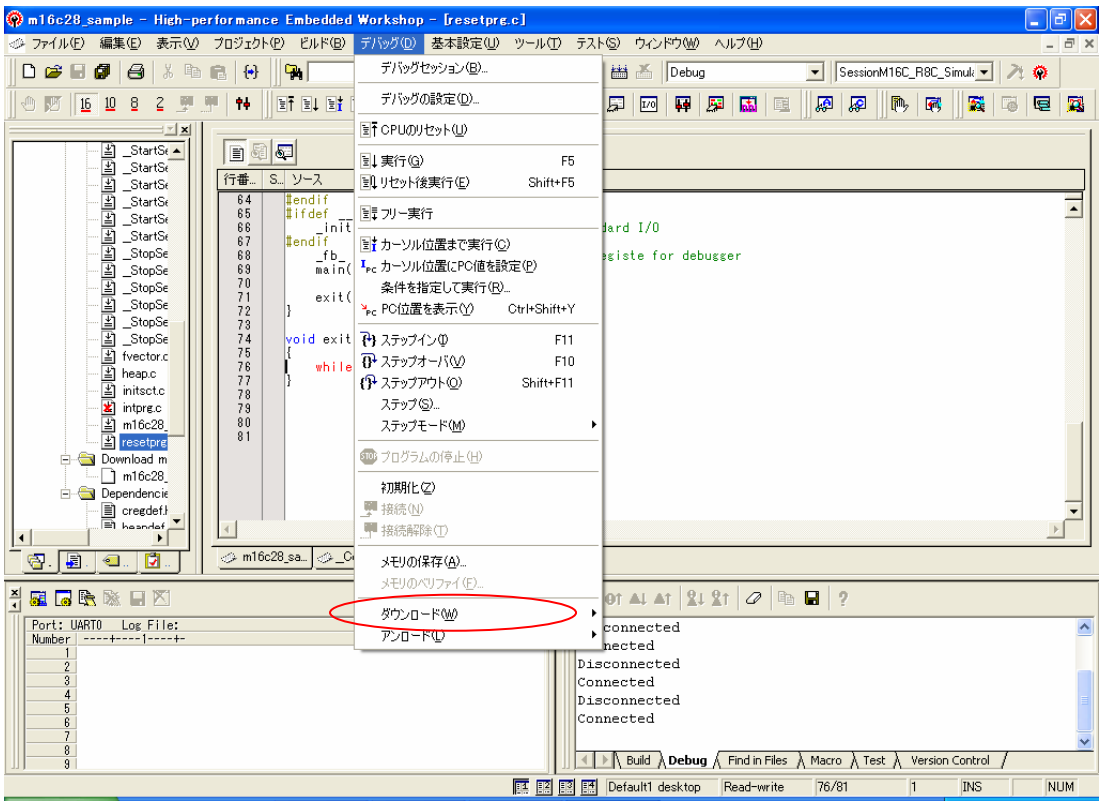
- シミュレータ実行



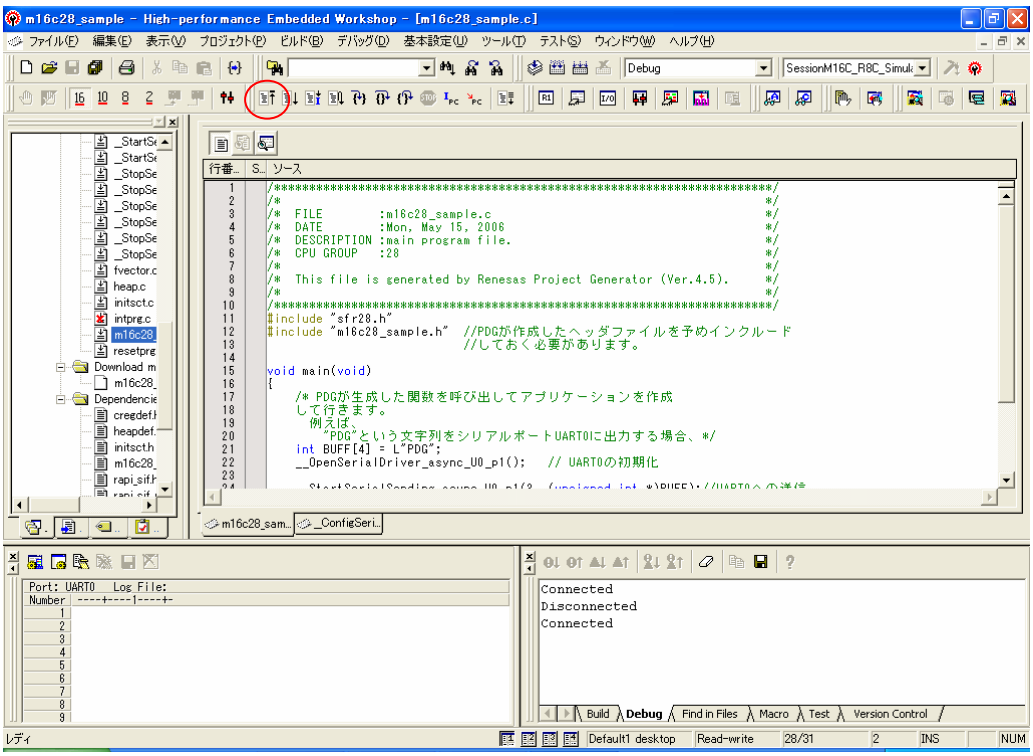
① セッションを SessionM16C_R8C_Simulator を選択する
② “はい”を選択する。

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）



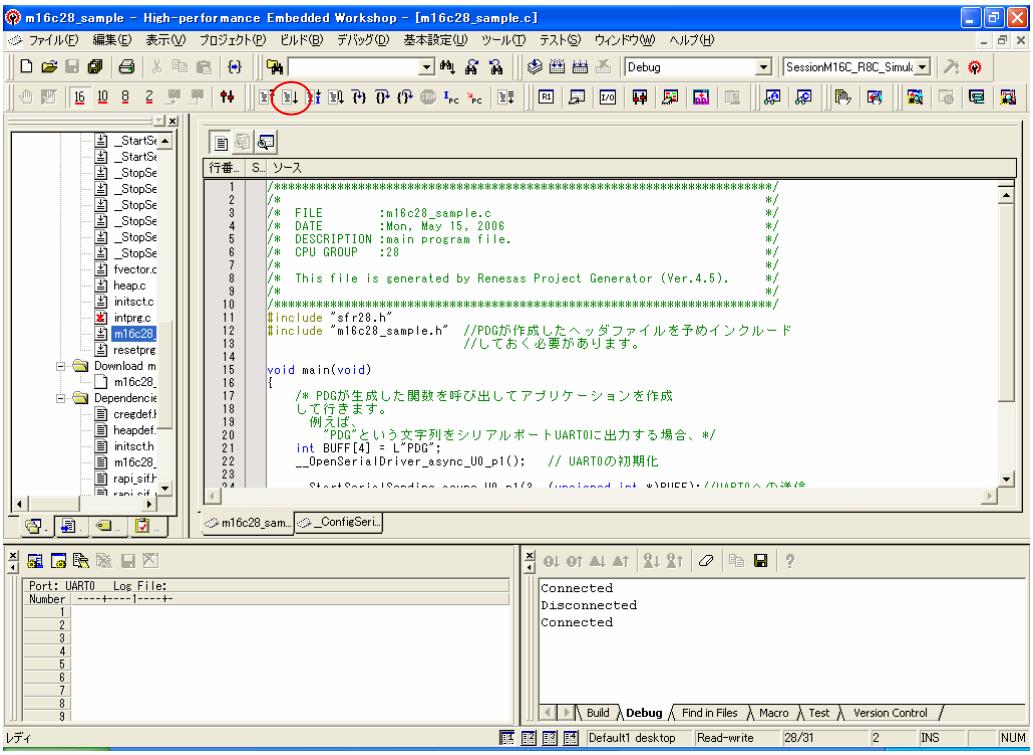
⑤　リセットボタンを押す



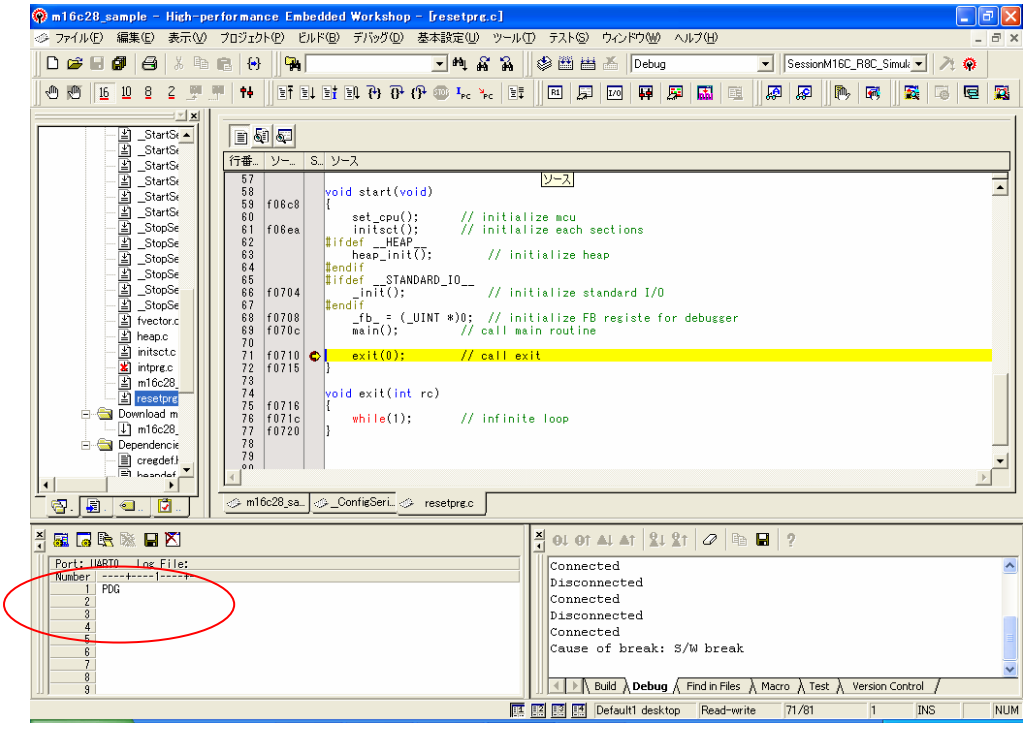
⑥　resetprg.c の exit(0)呼び出し箇所に Breakpoint を設定する。

```
_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、サンプルを使った操作は終わりです。

3.　ビルドの設定

HEW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）※1

・APIライブラリをリンクするための設定 (-Lの指定)　※2

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。

※1　High-performance Embedded Workshop V.4.05以降のバージョンを使用する場合、既にHEWのプロジェクトに追加されているソースファイルとPDGから追加するソースファイルについてはPDGからのソース登録時に-Iオプションの指定を自動的に行います。手動での-Iオプションの指定は、V.4.04以前のバージョンを使用する場合と、V.4.05以降のバージョンを使用し、PDGからのソース登録後、HEWのプロジェクトにソースファイルを追加した場合に必要です。

※2　High-performance Embedded Workshop V.4.02以前のバージョンを使用する場合に必要です。

3.1　ヘッダファイルの指定

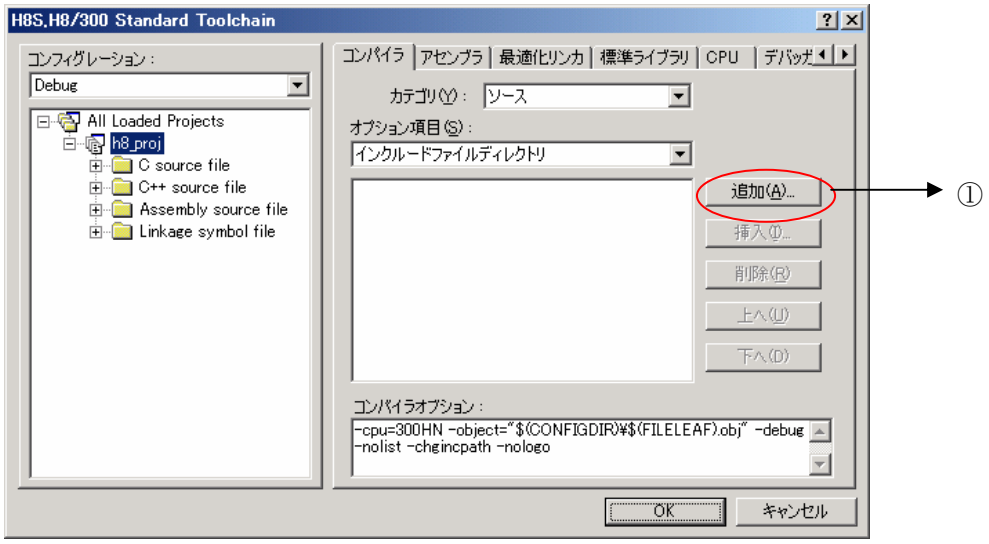
PDGで生成した関数を呼び出す場合、関数のプロトタイプ宣言を記述したヘッダファイルをインクルードする必要があります。

ヘッダファイル名は、プロジェクト名.hになります。

HEWで作成したディレクトリとPDG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。

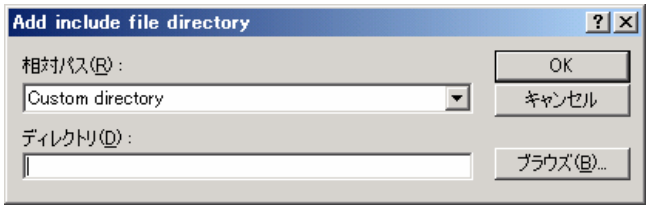
```
<sample.c>
#include "m16c28_sample.h"

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…」、SH/Tinyの場合は「SuperH RISC engine Standard Toolchain…」を選択します。



(2) “コンパイラ“タブ(M16C/Tiny, R8C/Tiny の場合は、”C”タブ)を選択します。
H8/300H Tiny, SH/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します。
H8/300H Tiny, SH/Tinyの場合、
・相対パス(R) : Custom directory
・サブディレクトリ(S):　ディレクトリ名
M16C/Tiny、R8C/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
をそれぞれ選択、入力します。

(4) 全てのダイログを閉じて完了です。

3.2　ライブラリの指定

3.2.1　サンプルプロジェクト用ライブラリー覧

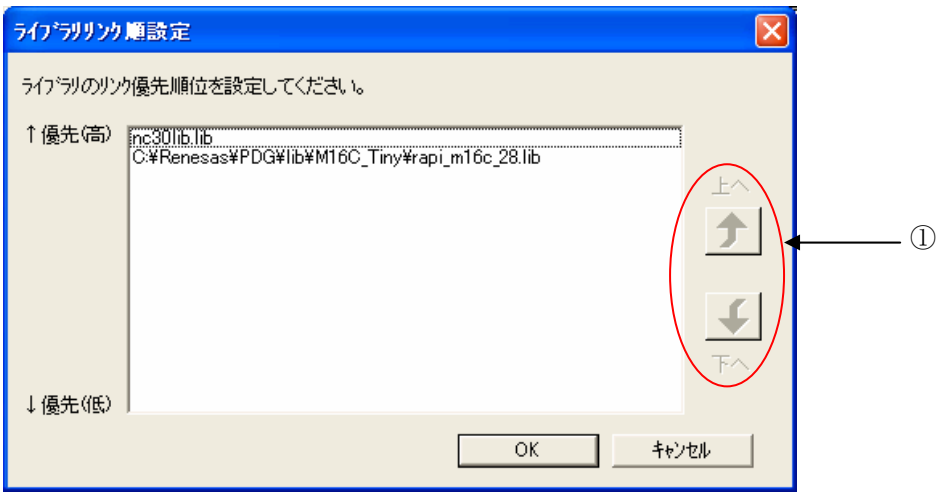
サンプルプロジェクトで作成した各周辺 I/O モジュールの設定ソースをビルドする場合、次に示す各ライブラリをリンクする必要があります。

サポートしているマイコンの全ライブラリファイル名は PDG のユーザーズマニュアルを参照してください。

CPU	格納ディレクトリ	ライブラリファイル名
H8/3687	lib¥h8_3687	rapi_h8_3687.lib
R8C/13	lib¥r8c_13	rapi_r8c_13.lib
M16C/28	lib¥m16c_28	rapi_m16c_28.lib
SH7125	lib¥sh7125	Rrapi_sh7125.lib

3.2.2　HEW V.4.02 以降のバージョンを使用している場合

HEW V.4.02 以降のバージョンを使用している場合は、PDG 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。

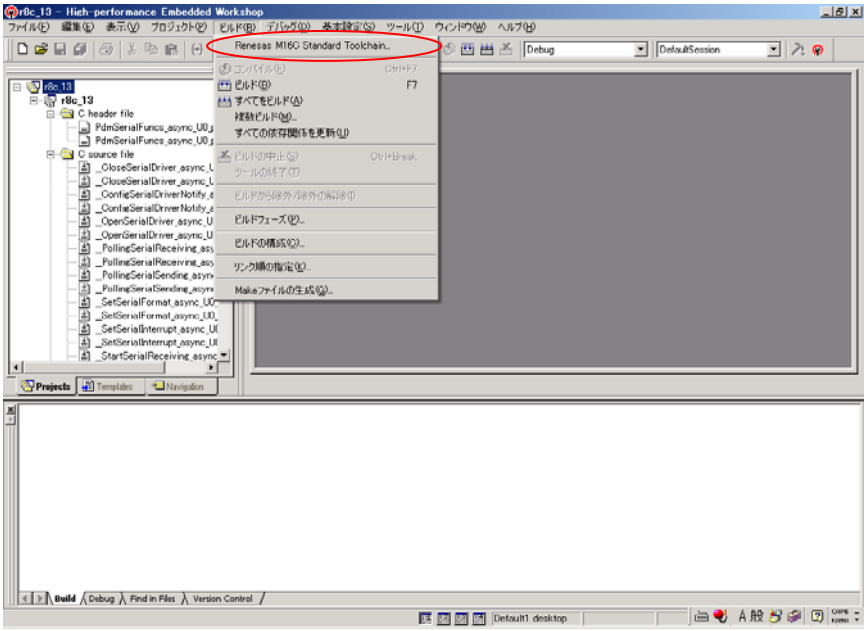
優先度の変更は、対象となるライブラリを選択した状態で、①で示した矢印ボタンを押してください。

上に行くほど優先度は高くなります。

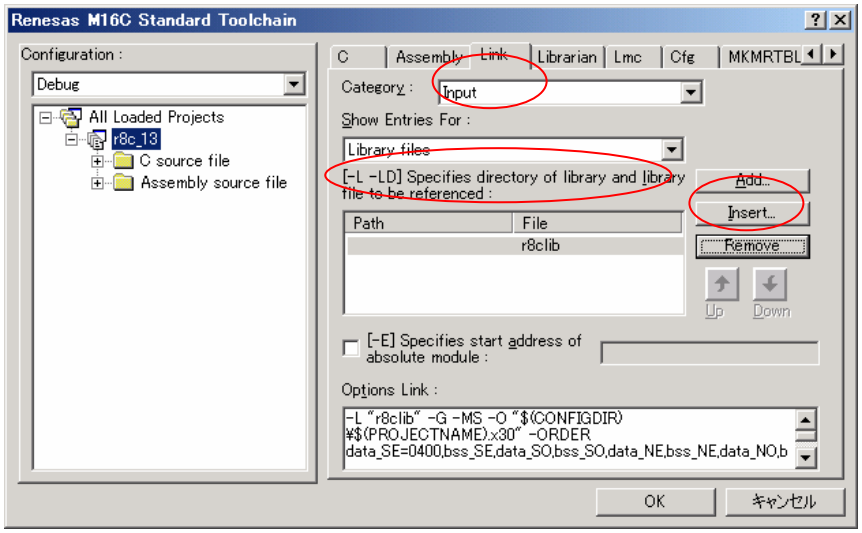
3.2.3　　HEW V.4.02 より前のバージョンを使用している場合

HEW V.4.02 より前のバージョン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…」、SH/Tiny の場合は「SuperH RISC engine Standard Toolchain…」を選択します。

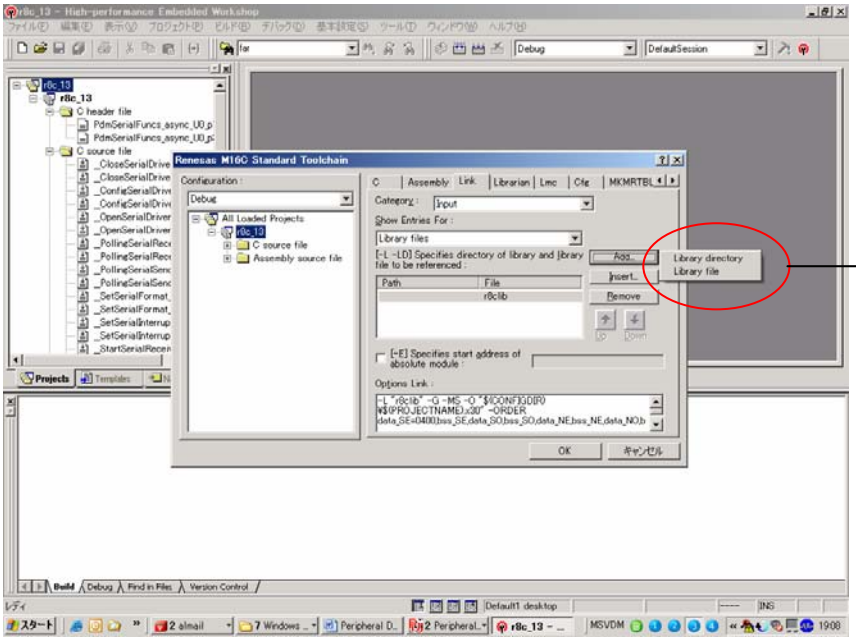


(2) Renesas M16C Standard Toolchain ダイアログの“リンカ”タブ、または H8S,H8/300 Standard Toolchain (SH/Tiny の場合は SuperH RISC engine Standard Toolchain) ダイアロの“最適化リンカ“タブを選択します。

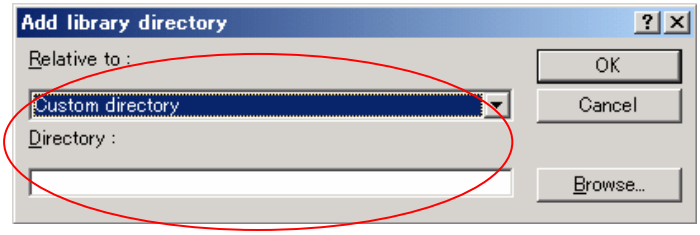


Show Entries For: Library files (H8/300H Tinyの場合は、ライブラリファイル)を選択し、Add（追加）ボタンを押します。

(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。



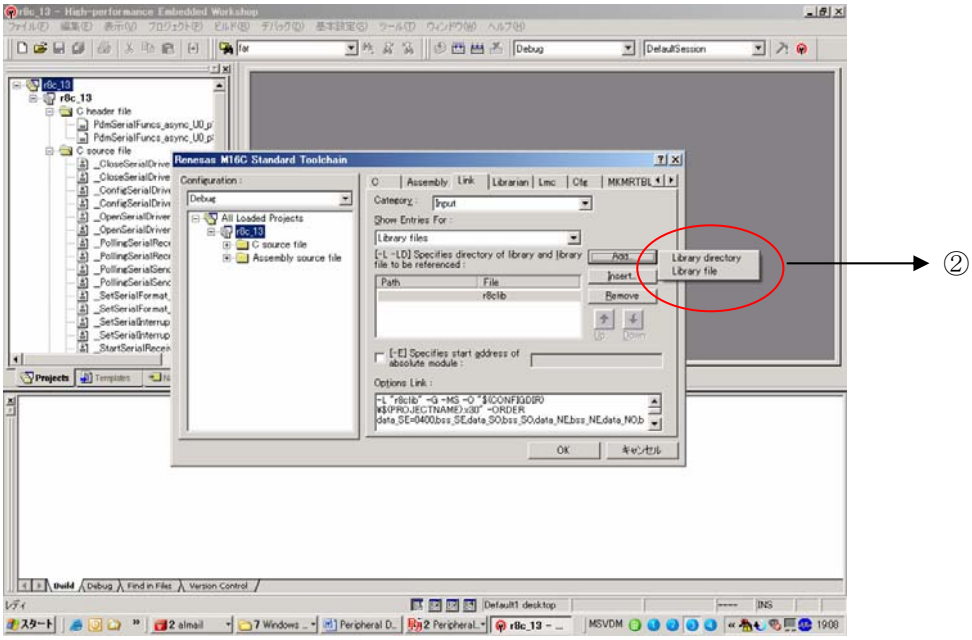
①Library directoryを選択する。（R8C/Tiny, M16C/Tiny, M16C/60の場合）



Relative to : Custom directory　を指定します。
Directory : APIライブラリの格納ディレクトリを指定します。
　例）C:¥Renesas¥PDG¥lib¥M16C_28

H8/300H TinyまたはSH/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。（R8C/Tiny, M16C/Tiny, M16C/60の場合）



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です。

3.3　割り込みベクタテーブルの除外

PDG は、割り込みベクタも同時に作成します。

そのため、HEW 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。

なお、HEW V.4.02 以降のバージョンを使用している場合は、PDG から自動的に除外します。以下に割り込みベクタ使用方法を示します。

3.3.1　H8/300H Tiny,M16C/Tiny,R8C/Tinyの場合

HEW のスタートアッププログラムには割り込みベクタの関数を宣言するファイル intprg.c が含まれます。一方、PDG で割り込み許可を設定した場合、生成されるソースには割り込みベクタが含まれます。これらが重複することを避けるため、PDG から HEW へソースを登録する際、intprg.c はビルド対象から除外されます。(HEW V.4.02 より前のバージョンを使用している場合は手動で除外してください。)　独自に割り込みベクタ関数を追加する場合は intprg.c を使用せず、別ファイルに関数を宣言し追加してください。

例えば M16C/28 で PDG の設定からシリアルの UART0 の送信割り込みを許可した場合、intprg.c と PDG が生成する InterruptTxU0_m16c_28.c でベクタ番号 17 の関数が重複するため、intprg.c はビルドから除外します。

ここで例えば DMA0 割り込み(ベクタ番号 11)の処理を追加する場合は、その関数のファイルを追加してください。

InterruptTxU0_m16c_28.c (PDGが生成するファイル)

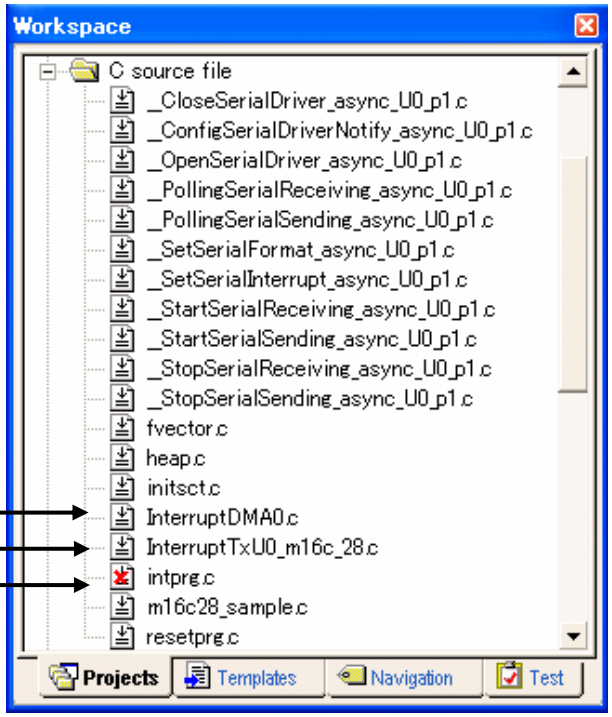
```
#pragma INTERRUPT _InterruptTxU0(vect=17)
void _InterruptTxU0(void)
{
    _SifdComTxU0();
}
```

intprg.c (HEWのスタートアップに含まれるファイル)

```
#pragma interrupt  _uart0_trance(vect=17)
void _uart0_trance(void){}
```

例:InterruptDMA0.c (必要に応じて作成、追加する)

```
#pragma interrupt  _Interrupt_dma0(vect=11)
void _Interrupt_dma0(void)
{
    //DMA0割り込み処理を記述する
}
```



HEW ワークスペースウィンドウ

3.3.2　SH/Tinyの場合

HEW のスタートアッププログラムには割り込みベクタの関数を宣言するファイル intprg.c が含まれます。一方、PDG は割り込みベクタ関数のファイル intprg_sh7125_pdg.c および intprg_sh7125.c を生成します。これらの関数が重複することを避けるため、PDG から HEW へソースを登録する際、intprg.c はビルド対象から除外されます。(HEW V.4.02 より前のバージョンを使用している場合は手動で除外してください。)

intprg_sh7125_pdg.c は PDG から設定可能な割り込み種別に対応した関数が集められています。このファイルは PDG による生成および編集専用で、コード生成の度に PDG により上書きされます。

intprg_sh7125.c は PDG から設定しない割り込み種別の関数が集められており、PDG により上書きされないため編集することができます。

intprg_sh7125.c
（PDGが生成するファイル　編集可能）

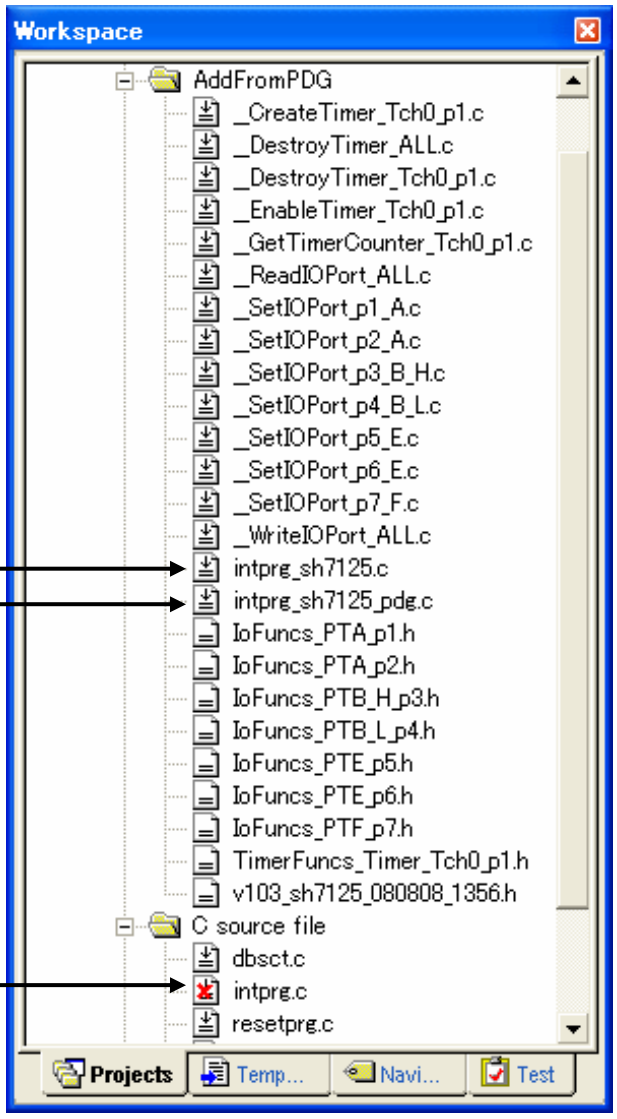
```
// 4 Illegal code
void INT_Illegal_code(void)
{
      //割り込みの処理を追加できます
}

// 6 Illegal slot
void INT_Illegal_slot(void){}

// 9 CPU Address error
void INT_CPU_Address(void){}
```

intprg_sh7125_pdg.c
（PDGが生成するファイル　PDGによる編集専用）

```
// 11 NMI
void INT_NMI(void){}

// 64 Interrupt IRQ0
void INT_IRQ0(void){}

// 65 Interrupt IRQ1
void INT_IRQ1(void){}
```

intprg.c（HEWのスタートアップに含まれるファイル）

```
// 4 Illegal code
void INT_Illegal_code(void){}

// 6 Illegal slot
void INT_Illegal_slot(void){}

// 9 CPU Address error
void INT_CPU_Address(void){}

// 11 NMI
void INT_NMI(void){}
```



HEW ワークスペースウィンドウ

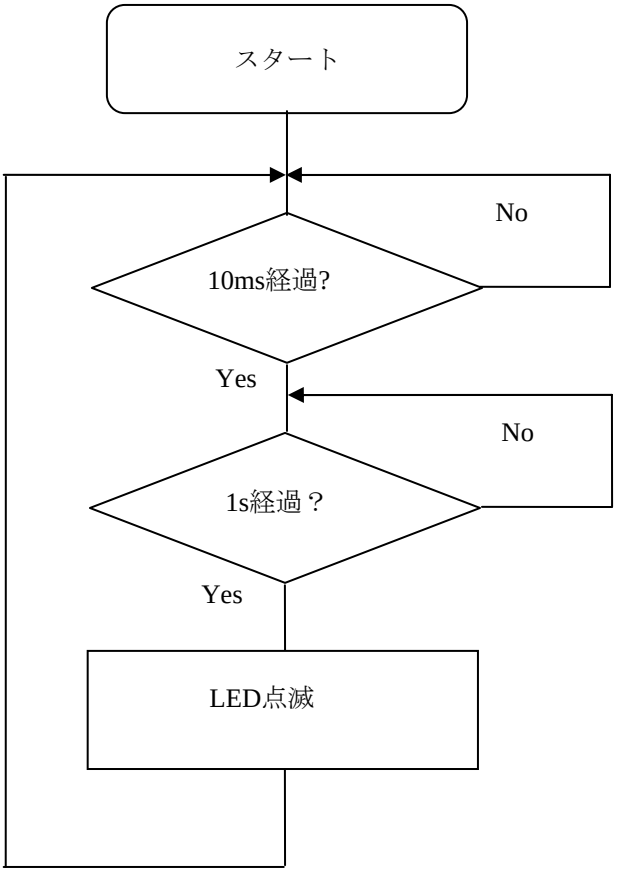
注意：SH7125でHEW V.4.04以前のバージョンを使用している場合、intprg_sh7125.cおよびintprg_sh7125_pdg.cに対し、HEWのスタートアッププログラムのファイルvect.hへのパスを設定する必要があります。HEW V.4.05以降のバージョンを使用している場合、インクルードパスは自動的に登録されます。

4.　アプリケーションの作成例

本章では、PDGを使用してアプリケーションを作成する手順を説明します。

4.1　作成するアプリケーションのフローチャート

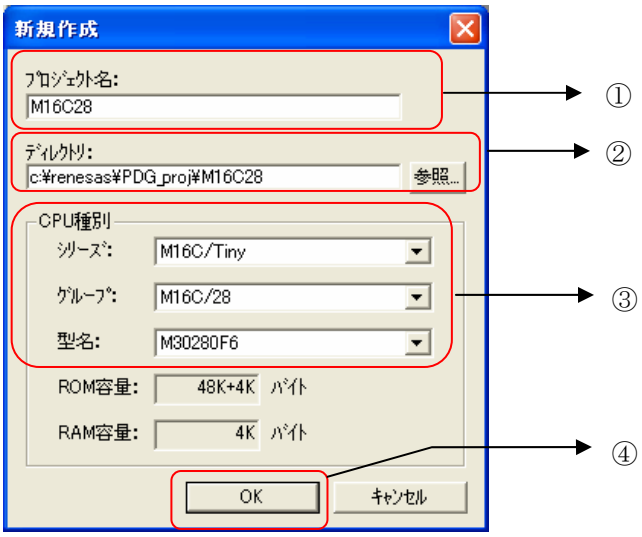
以下のフローチャートに基づいたアプリケーションを作成します。
使用マイコン：M16C/28



4.2　PDGを使用して周辺を設定する

4.2.1　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。

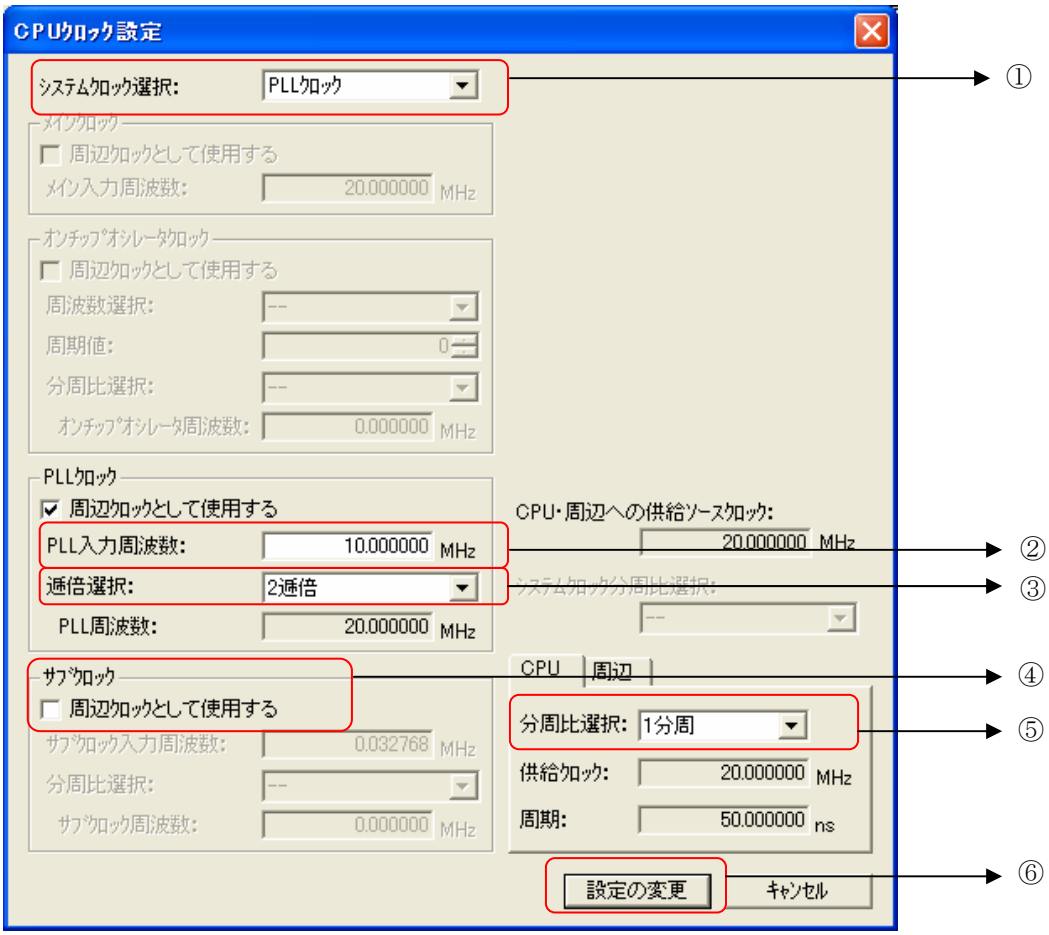
② プロジェクトを格納するディレクトリを指定します。
デフォルトでは、c:¥renesas¥PDG_projの下にプロジェクト名のディレクトリを作成します。

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。

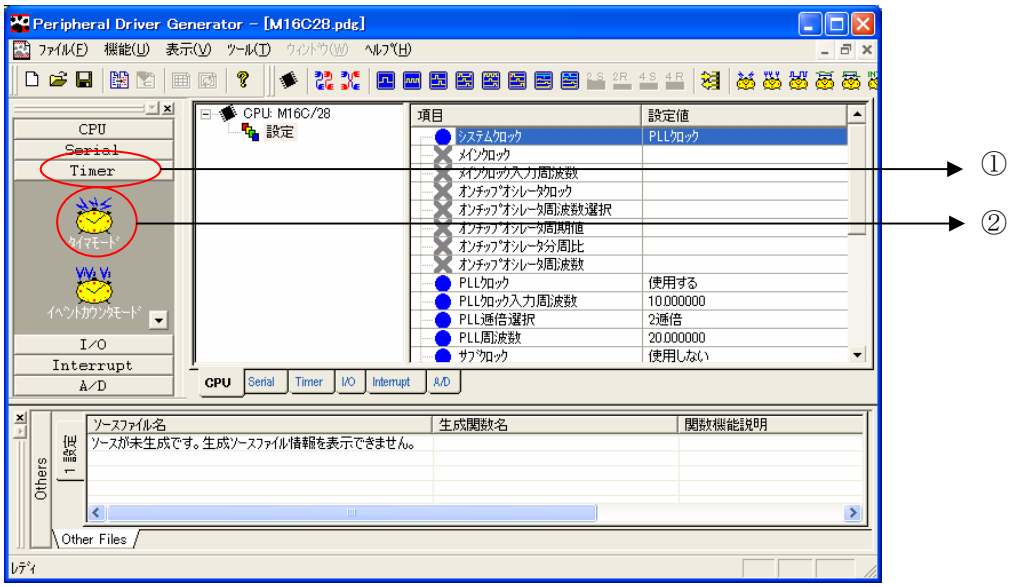
4.2.2　　クロックの設定

4.2.1 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。

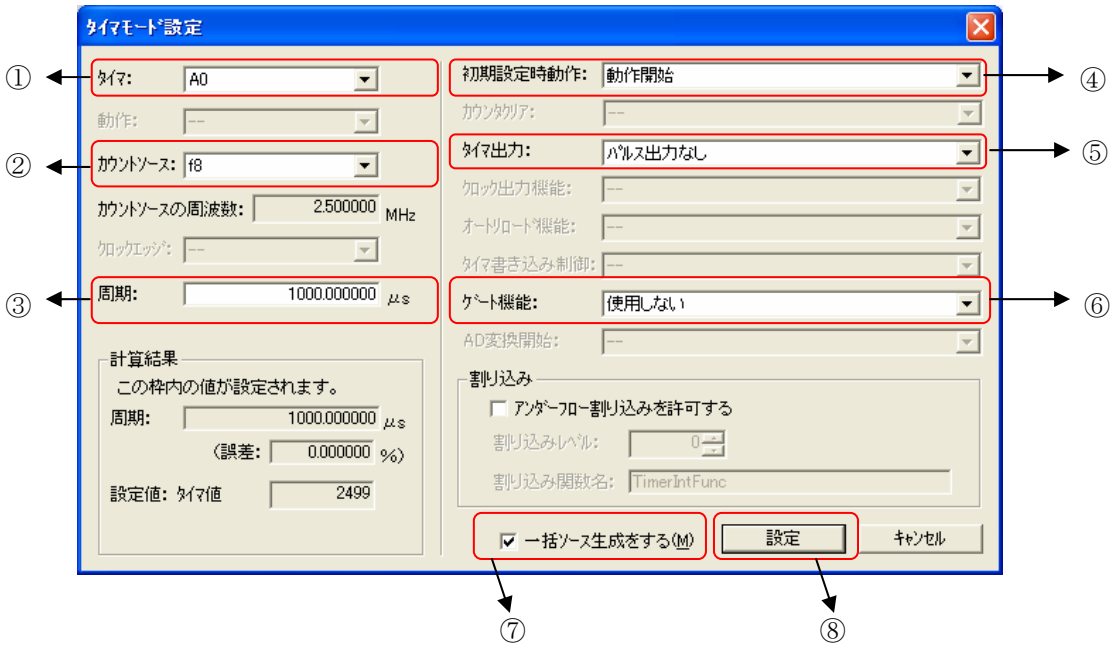


① システムクロック選択で **PLL クロック**を選択する。
② 周波数は、**10MHz** に設定する。
③ 逓倍選択は、**２逓倍**を選択する。
④ サブクロックは、**周辺クロックとして使用しない**。
⑤ CPU クロックの分周比選択は、**1 分周**を選択する。
⑥ 設定の変更を選択して、設定を終了する。

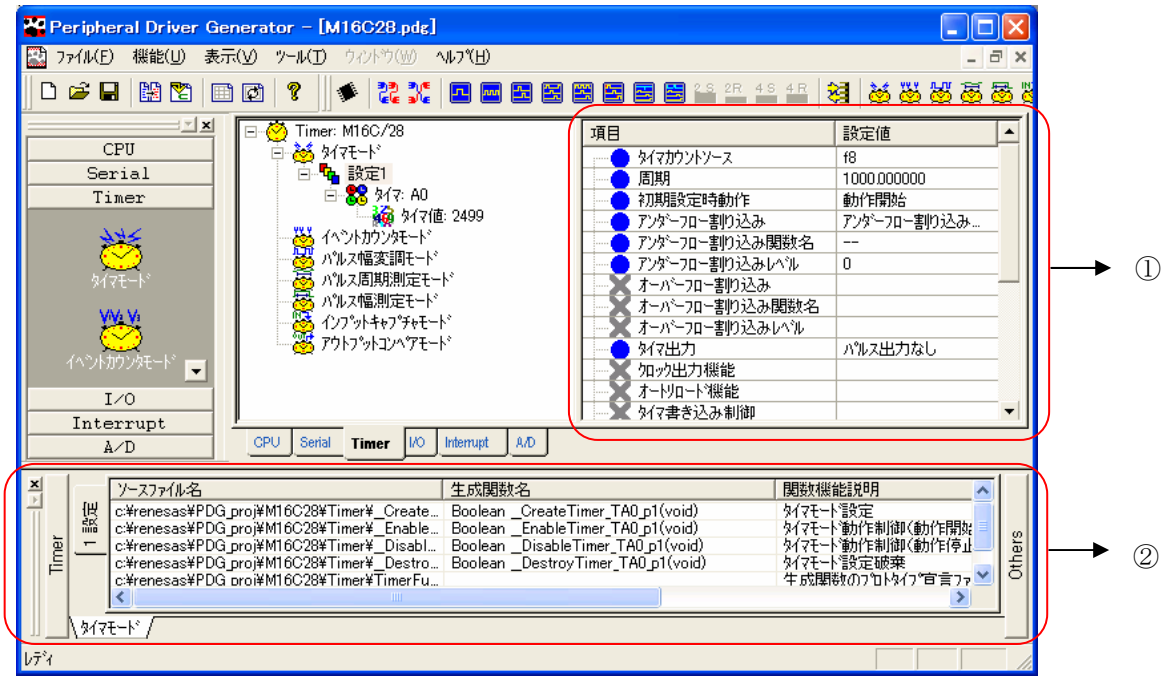
4.2.3　タイマモードの設定



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 周期は、**1000μs** 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

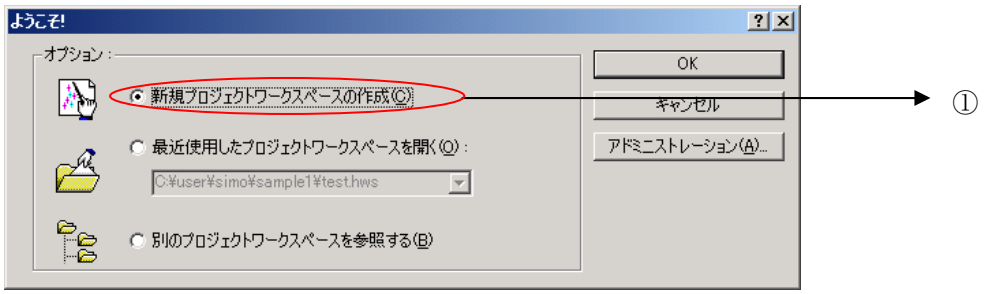
② に生成したソースファイル、関数名、機能

の一覧を表示します。

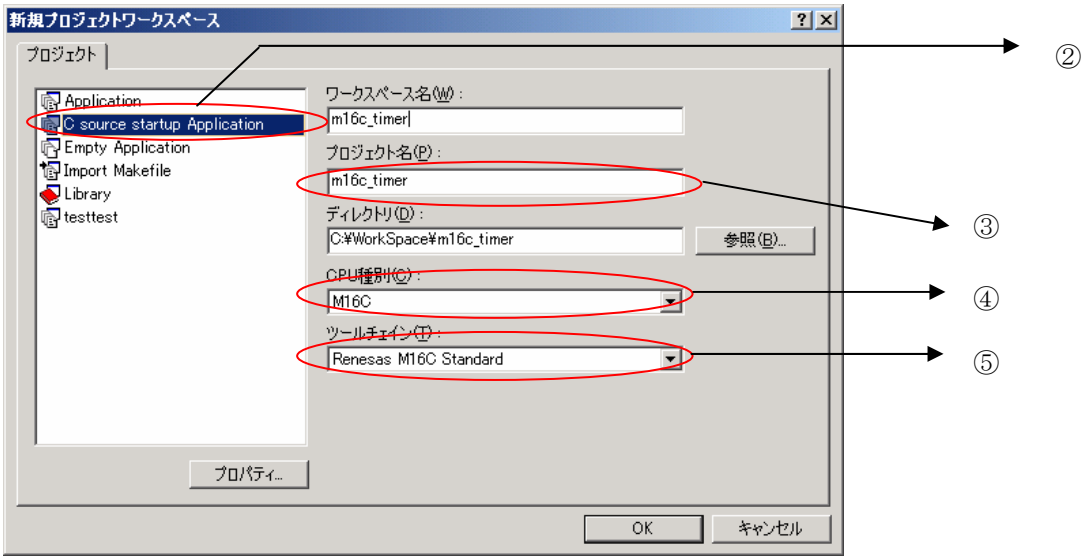
4.3　プログラム作成

プログラムの作成は、HEWを使用して行います。

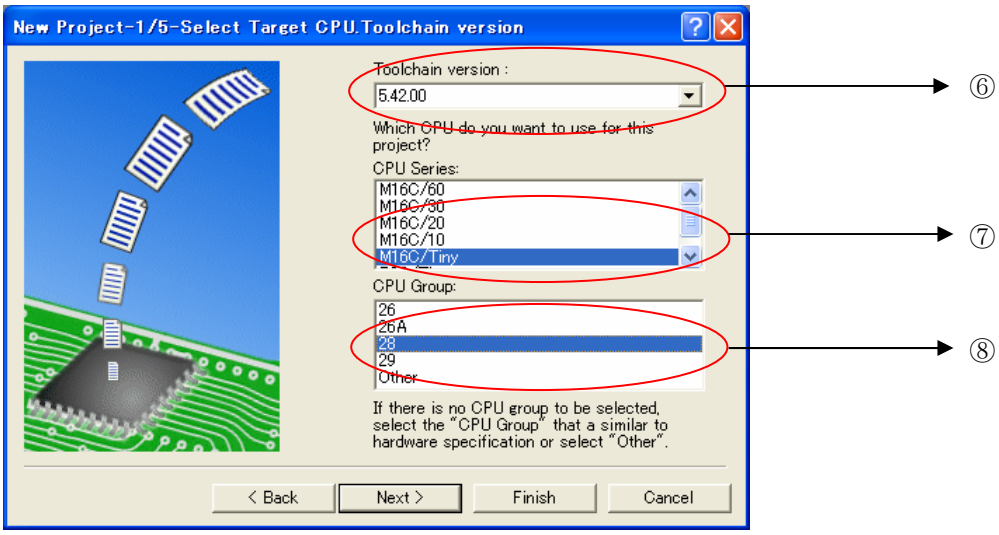
4.3.1　ワークスペース作成



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 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。
④ CPU 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。

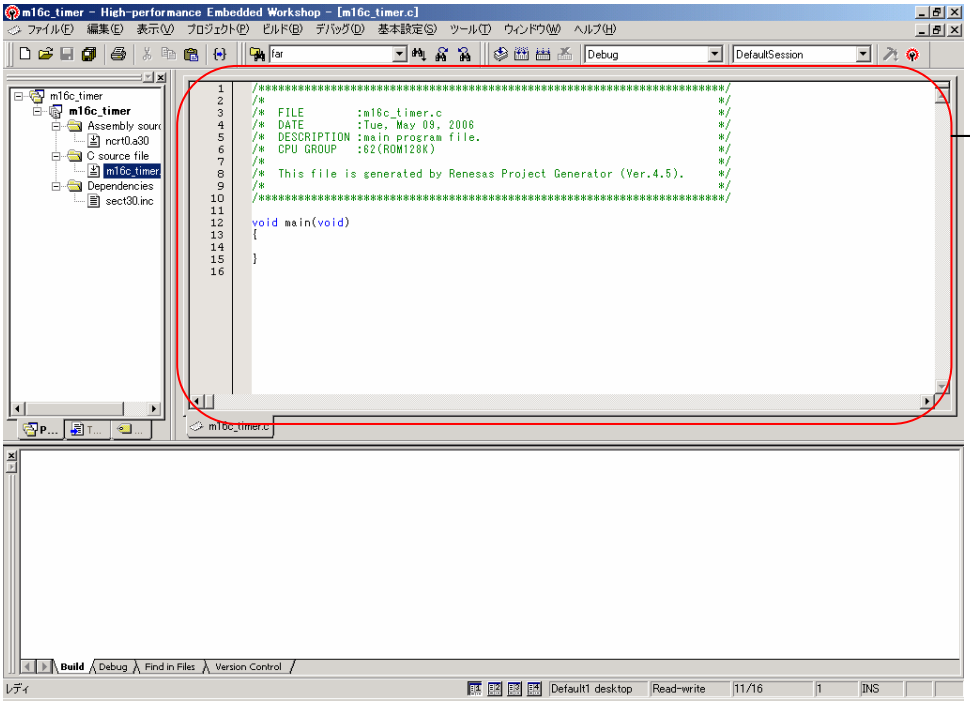


⑥　コンパイラのバージョンを **5.40.00**、もしくはそれ以上にする
⑦　CPU Series:は **M16C/Tiny** を選択する
⑧　CPU Group は、**28** を選択する。
Next>ボタンをクリックして次に進む。

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。

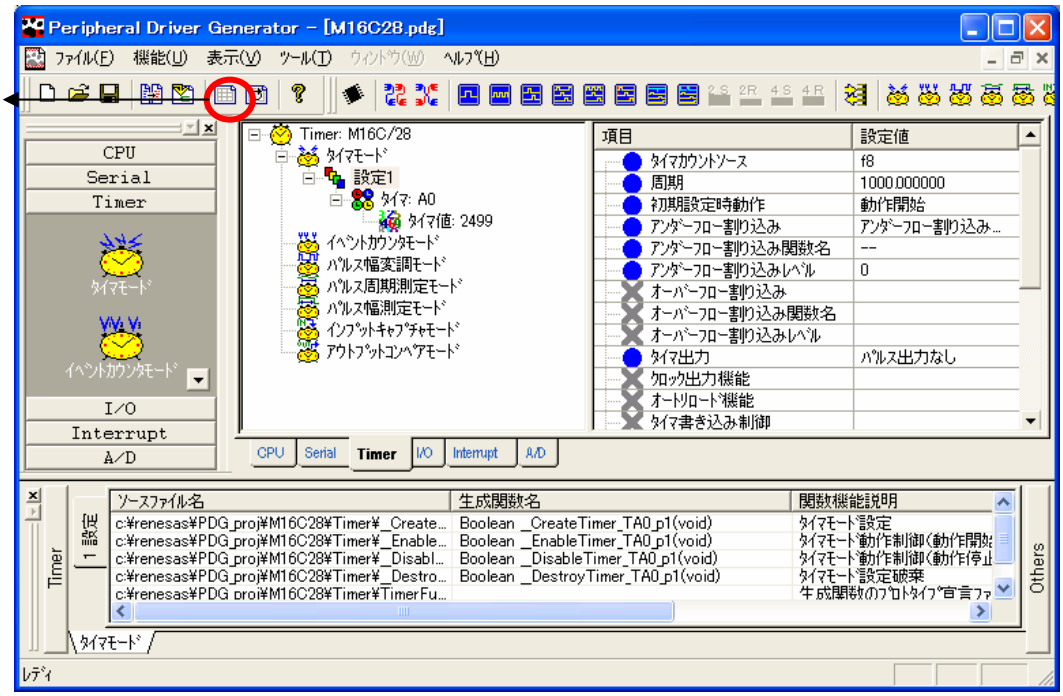
4.3.2　プログラムの作成

HEW 上の①でプログラムをコーディングしていきます。



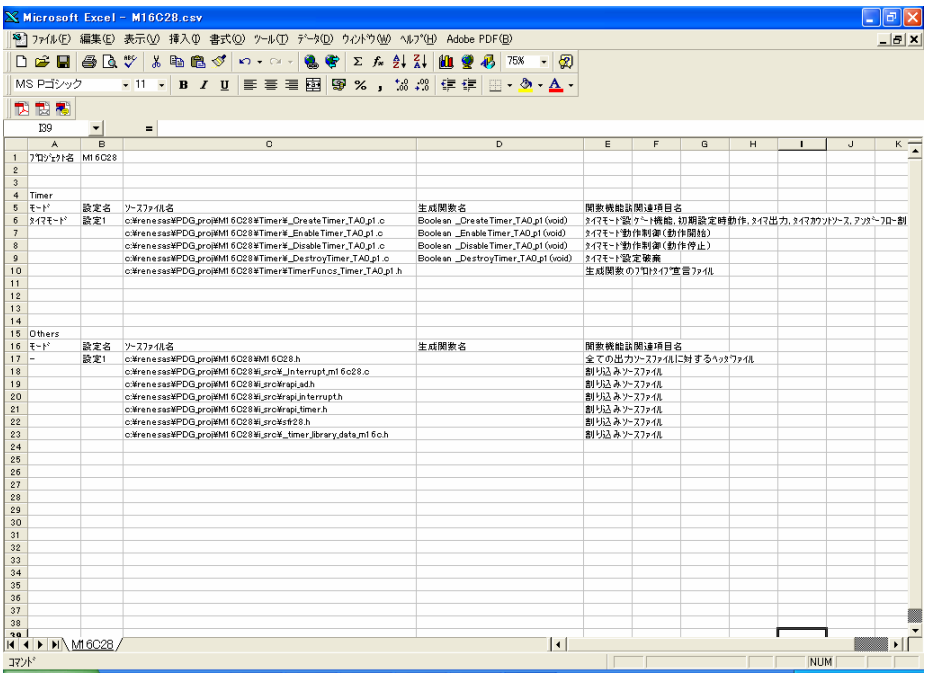
コーディングでは、PDG で作成した関数を使用します。

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。



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

4.3.3　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PDG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ります
#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( ) == RAPI_TRUE )              /* timer setting */
        {
                if( __EnableTimer_TA0_p1(    ) == RAPI_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 */
```

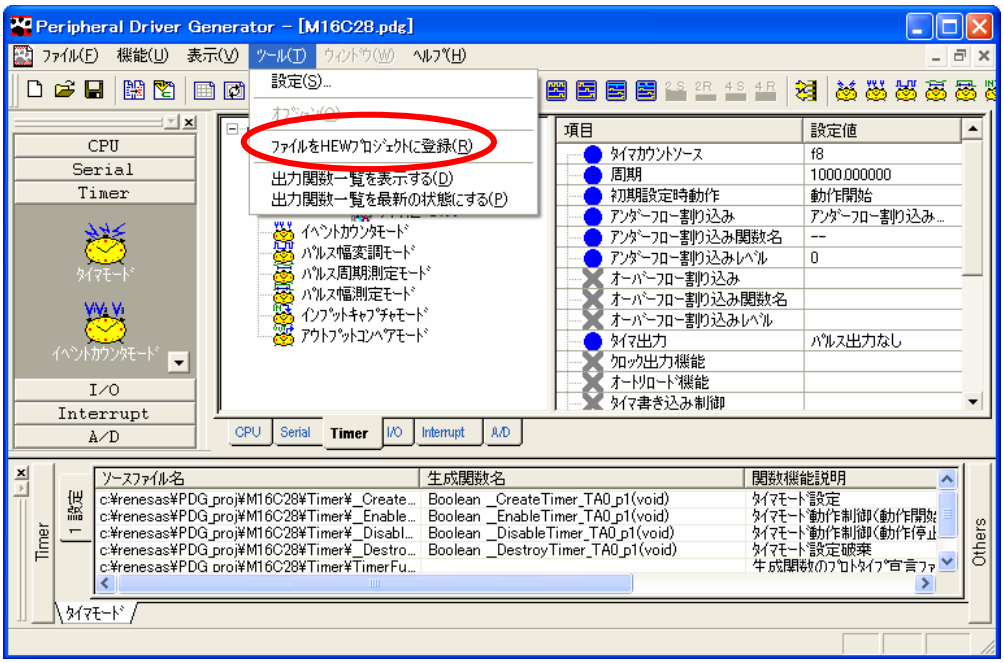
4.4　ビルド作業

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。

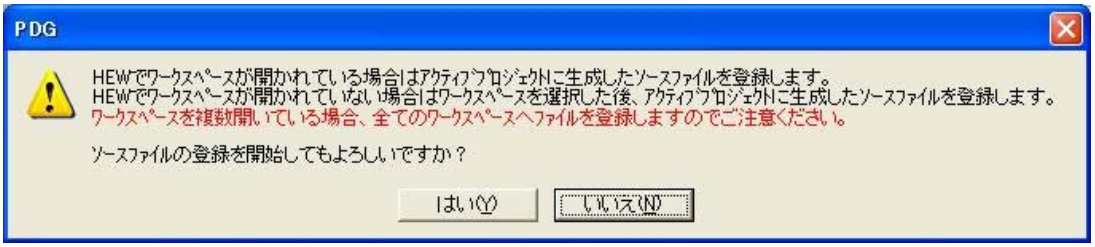
- Peripheral Driver Generator が生成したファイルを High-performance Embedded Workshop へ登録
- コンパイルに必要なオプションの設定（ヘッダファイルのインクルード先指定）
- リンクに必要なオプションの設定（Renesas Embedded Library の指定）

4.4.1　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。

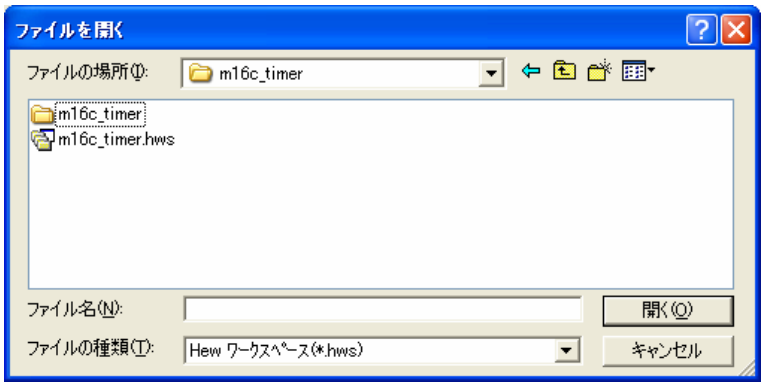


[ツール]→“ファイルを HEW プロジェクトに登録“を選択する。



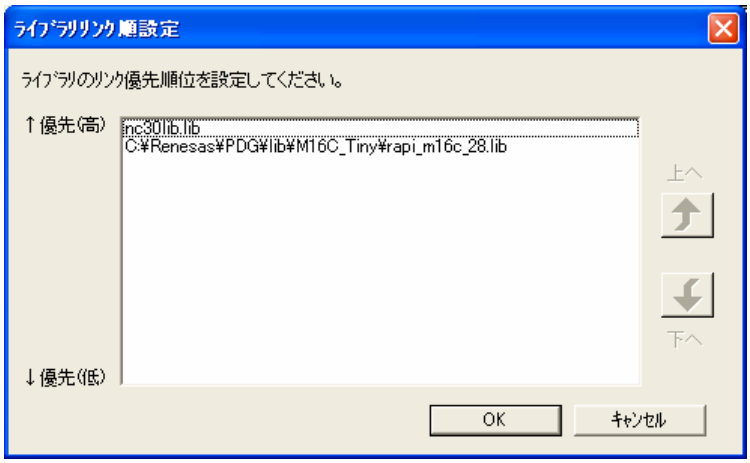
登録確認ダイアログが開くので、“はい”を選択する。
現在 HEW が開いているプロジェクトへ登録します。

異なるプロジェクトへ登録する場合は、一旦 HEW 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。



HEW V.4.02 以降のバージョンを使用している場合は、「ライブラリリンク順設定ダイアログ」が開くので、リンクの優先順位が高いものを上、低いものを下へ移動します。「OK」を選択すると、HEW プロジェクトへのファイル登録を開始します（※）

※ファイル登録の確認メッセージにもあるように、複数の HEW ワークスペースが開いている状態では、すべてのアクティブプロジェクトにファイルを登録します。ファイルを登録したくないワークスペースは、ファイル登録実行の前に閉じてください。



ファイルの登録完了メッセージが表示されれば、登録は完了です。

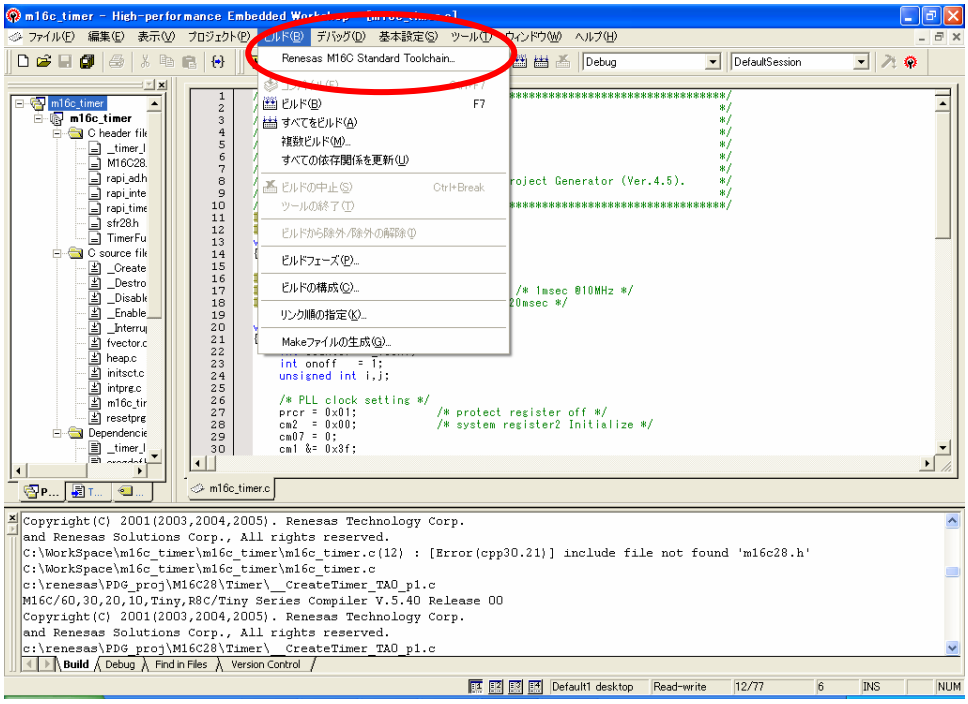
4.4.2　コンパイルオプションの設定

PDG 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。

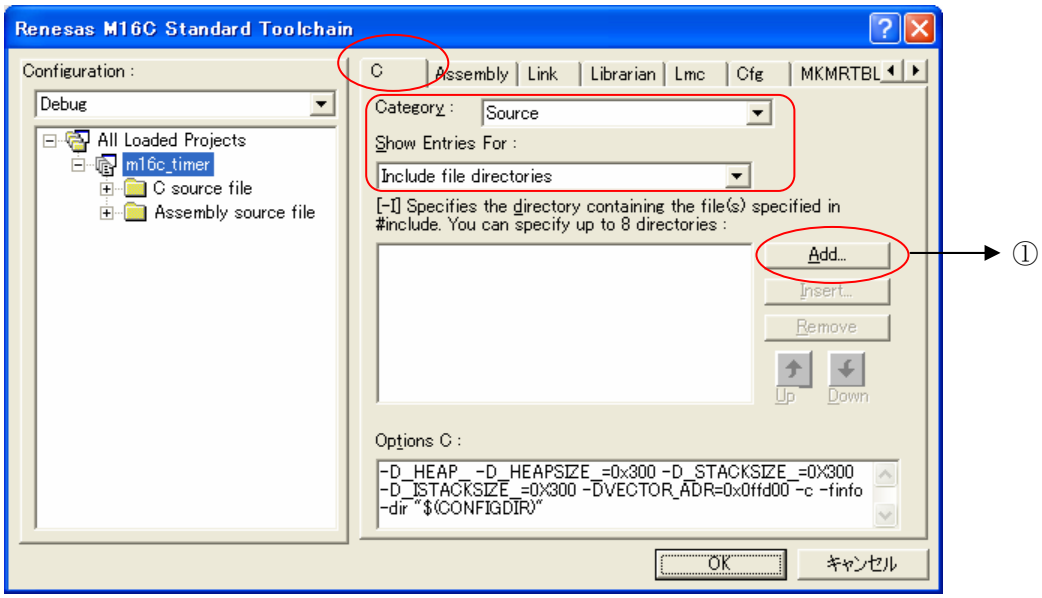
```
#include "sfr28.h"

#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ります
#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。インクルードファイルのディレクトリの指定方法を以下に示します。

High-performance Embedded Workshop V.4.05 以降のバージョンを使用する場合、既に HEW のプロジェクトに追加されているソースファイルと PDG から追加するソースファイルについては、PDG からのソース登録時にインクルードファイルのディレクトリ指定が自動的に行われるため、以下の操作は不要です。



HEW のビルドメニューから”Renesas M16C Standard Toolchain”を選択します。



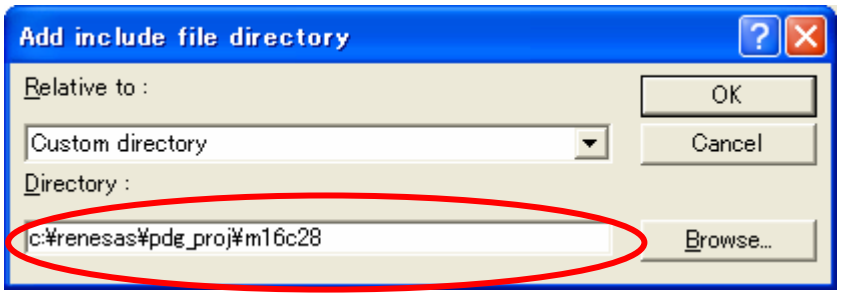
オプションダイアログからCタブを選択して、

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。



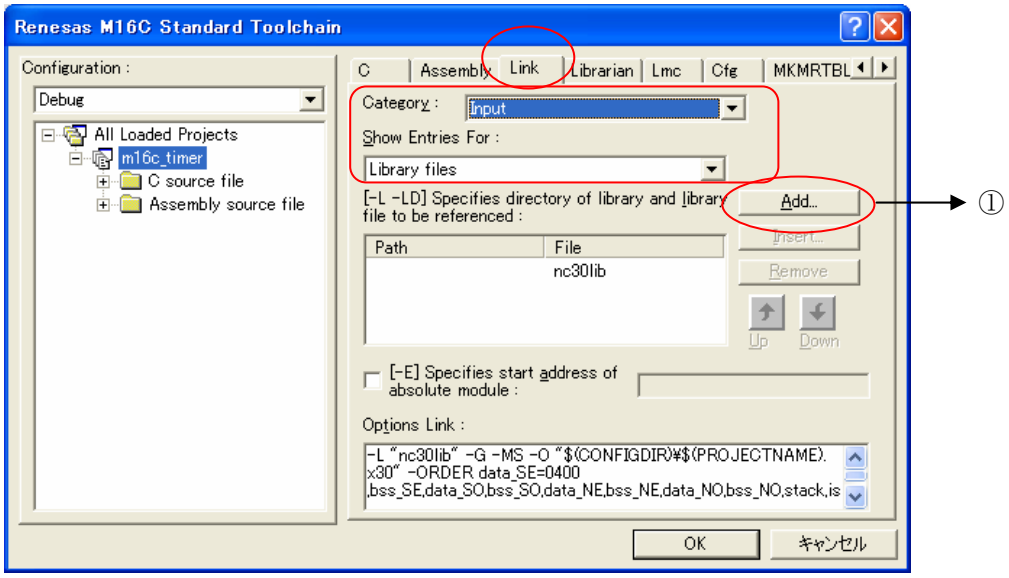
PDG が出力したヘッダファイルが存在するディレクトリを指定します。
このディレクトリは、PDG で作成したプロジェクトのディレクトリです。

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。

4.4.3　　リンクオプションの設定

HEW V.4.02 以降のバージョンを使用している場合は、ライブラリの指定操作は不要です。

(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、

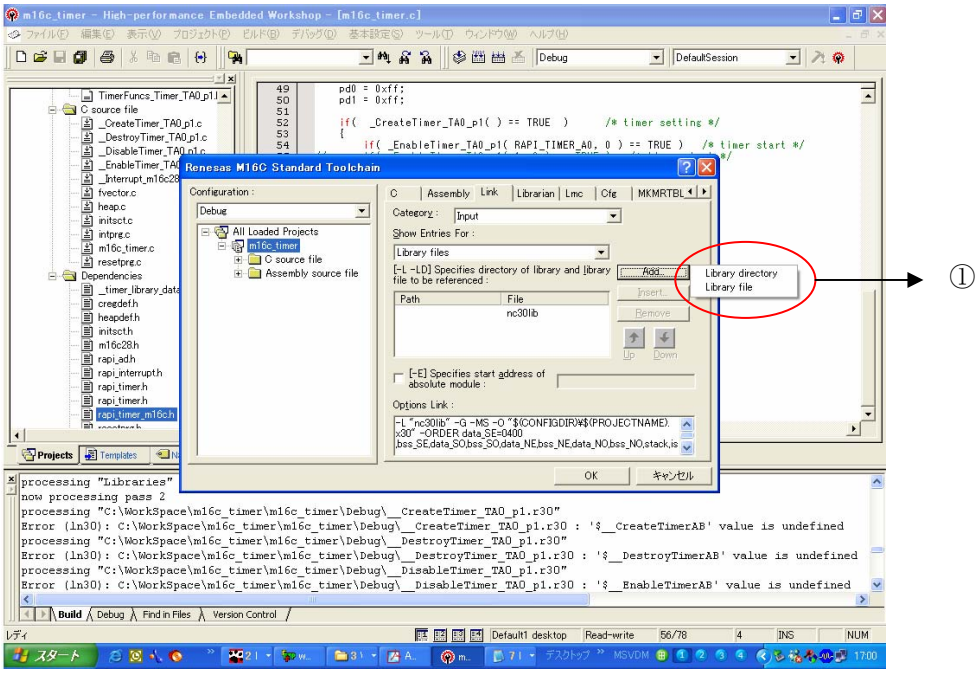
Category: Input

Show Entries For: Library files

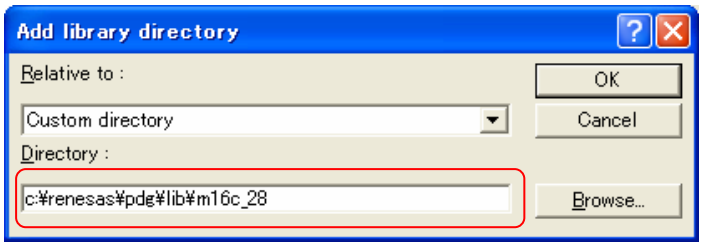
をそれぞれ選択します。

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。



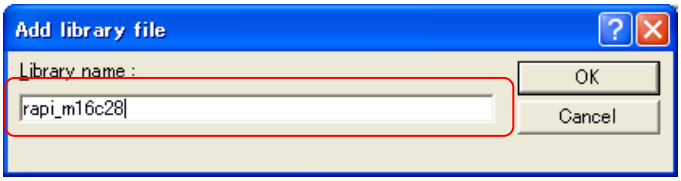
①でまず、Library directory を選択します。



Relative to : Custom directory　を指定します。
Directory : API ライブラリの格納ディレクトリを指定します。
C:¥Renesas¥PDG¥lib¥m16c_tiny

ライブラリのディレクトリ名入力後、OK を押します。

(3) 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。



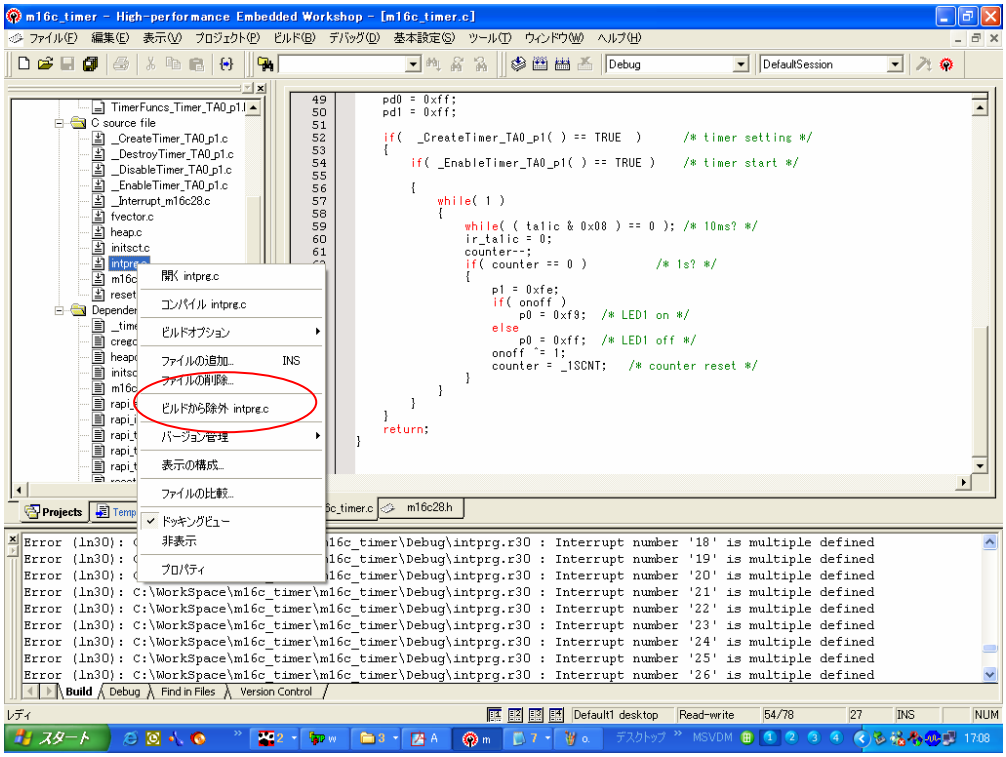
ライブラリ名に、rapi_m16c_28 を入力します。(拡張子 .lib は付けないでください。)

4.4.4　　割り込みベクタエントリファイルの除外

HEW 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)。

PDG 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。

なお、HEW V.4.02 以降のバージョンを使用している場合は、PDG から自動的に除外します。



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。

以上で全ての設定が完了です。

ビルドして、デバッガを使用して実行してみてください。

Peripheral Driver Generator V.1.03
ガイドブック

発行年月日　2008年9月19日　　Rev.1.00

発行　　株式会社 ルネサス テクノロジ　営業統括部
〒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.1.03
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

RJJ10J2349-0100