２００９年０４月２４日
Ｂｙ　エーワン　Ｒｅｖ１．４３

統合環境Ｈｅｗ（Ｖｅｒ２．２）添付スタートアップ関数を使用した場合の新ワークスペースおよびプロジェクトを登録する方法（ＳＨ－２版）

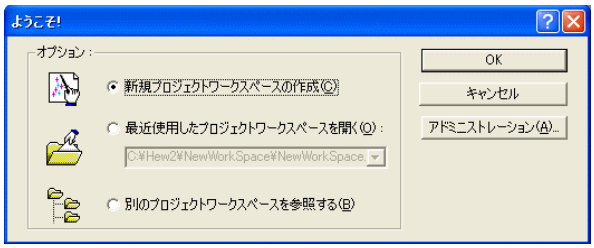
ルネサスＣ言語用統合環境「Ｈｅｗ　Ｖｅｒ２．２（ｒｅｌｅａｓｅ１５）」で H-debugger 用に新ワークスペース／プロジェクトを登録する手順方法を説明します。

説明を明確にするために、名前等を仮に決めて例に沿って説明を進めます。

ワークスペース名	NewWorkSpace2		
プロジェクト名	Project2		
登録モジュール名	SH705x.c	Ｃファイル	メインモジュール（アプリ用）
Ｈｅｗ添付ファイル	Resetprg.c	Ｃファイル	スタートアップモジュール
	Intprg.c		割込みハンドラモジュール
	Dbsct.c		定数転送用セクション管理宣言
	Vecttbl.c		ベクター定義モジュール
ＣＰＵタイプ	SH7051F		

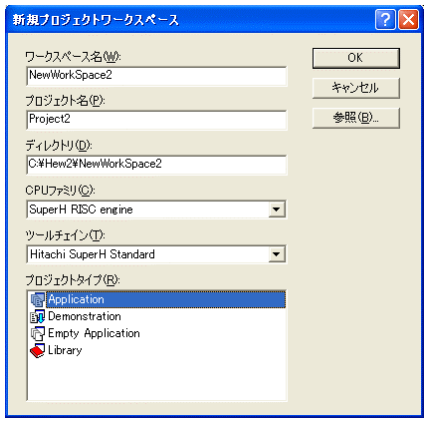
１．新ワークスペースの登録方法

“HEW”起動させます。



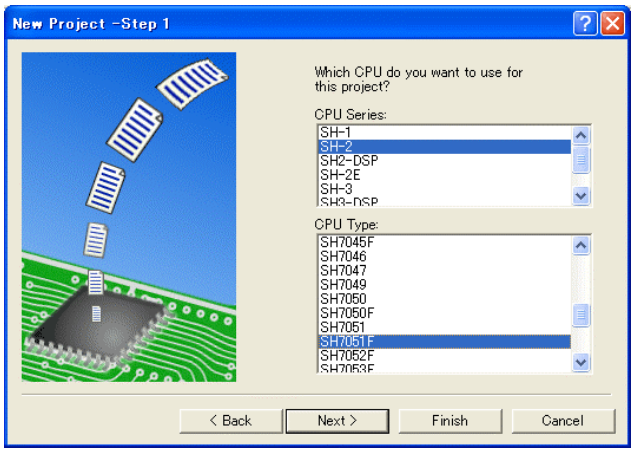
“新規プロジェクトワークスペース”をチェックしてのOKをクリックする。

もしくは、キャンセル後に、[ファイル]-[新規ワークスペース]をクリックします。



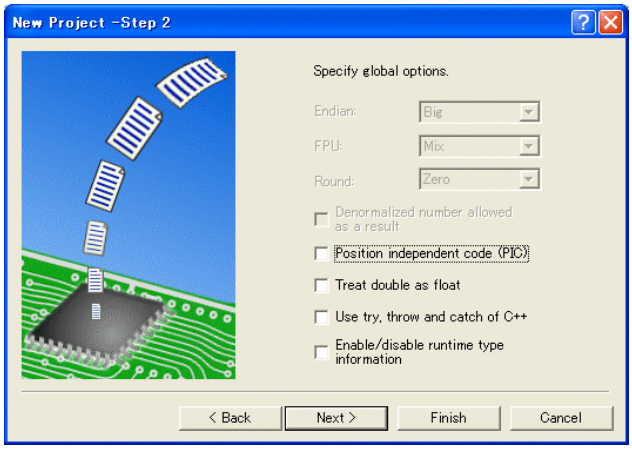
ﾜｰｸｽﾍﾟｰｽ名	“NewWorkSpace2”
ﾌﾟﾛｼﾞｪｸﾄ名	“Project2”
ﾃﾞｨﾚｸﾄﾘ	“C:¥HEW2¥NewWorkSpace2”
CPU 種別	“SuperH RISC engine”
ﾂｰﾙﾁｪｲﾝ	“Hitachi SuperH Standard”
ﾌﾟﾛｼﾞｪｸﾄ	Application

この項目を確認後、ＯＫをクリックして下さい。

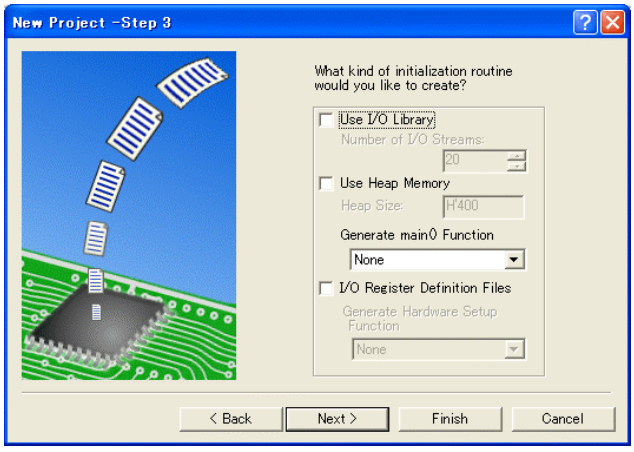


CPU ｼﾘｰｽﾞを“SH-2”に選択する。
CPU ﾀｲﾌﾟを“SH7051F”に選択する。

確認後、Ｎｅｘｔ＞をクリックします。



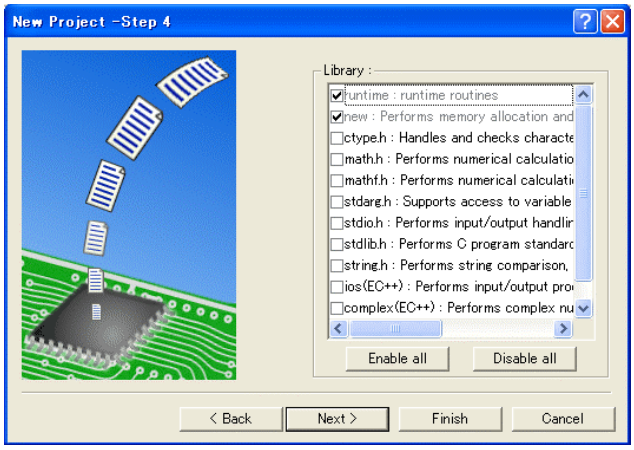
Ｎｅｘｔ＞をクリックして下さい。



ここでのＨｅｗが用意したＲＯＭ化支援関数は、使用しませんので全てのチェックを外して下さい。

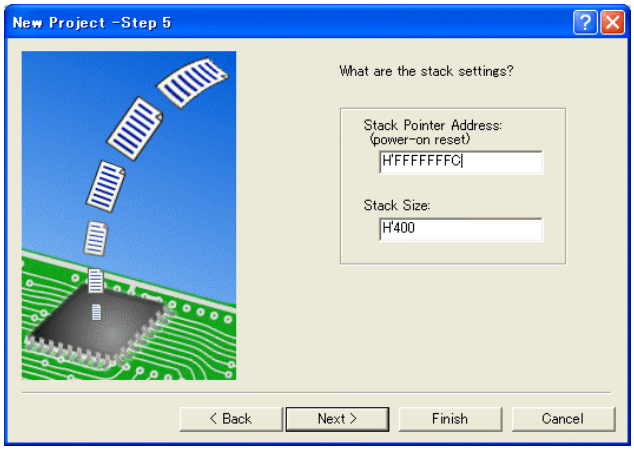
main() 関数生成を”None”に選択する。

確認後、Ｎｅｘｔ＞クリックして下さい。



Ｃ言語ライブラリの選択です。この例では、その他ライブラリを使用しません。

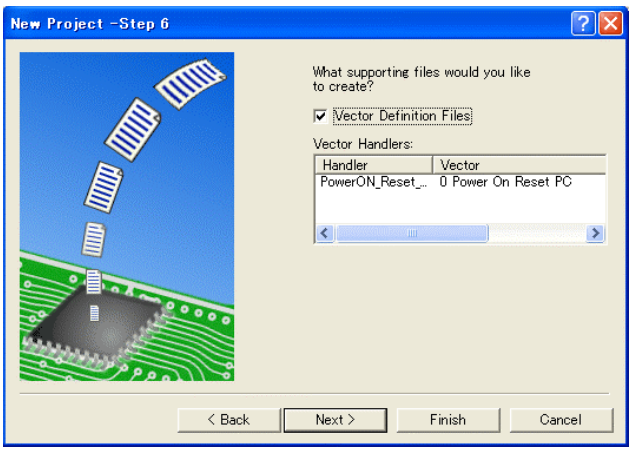
Ｎｅｘｔ＞クリックして下さい。



スタックボトムの設定です。
ＲＡＭ位置を示す為「H'FFFFFFFC」にする。

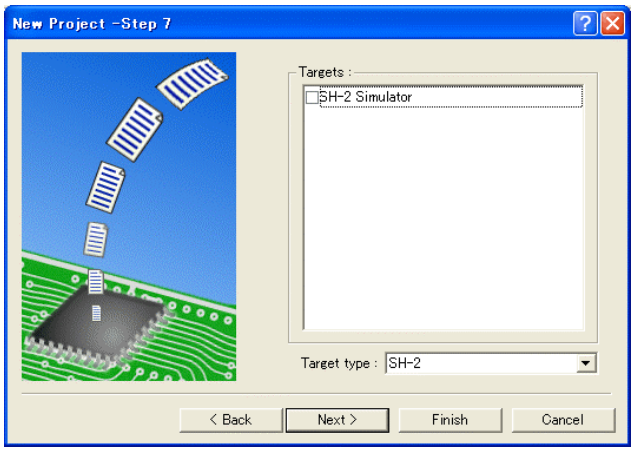
スタックサイズはデフォルト値にします。

Ｎｅｘｔ＞をクリックして下さい。



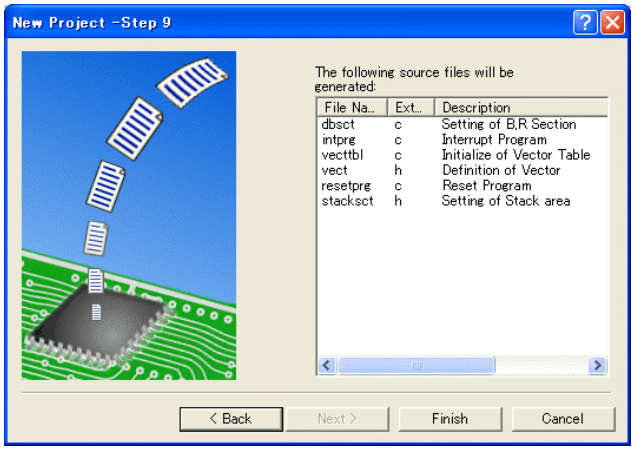
ここで明示されたＨｅｗ作成スタートアップ関数を使用しますので、デフォルトの状態で、

Ｎｅｘｔ＞をクリックして下さい。



シミュレータの設定ですが使用しませんのでチェック無しの状態で、

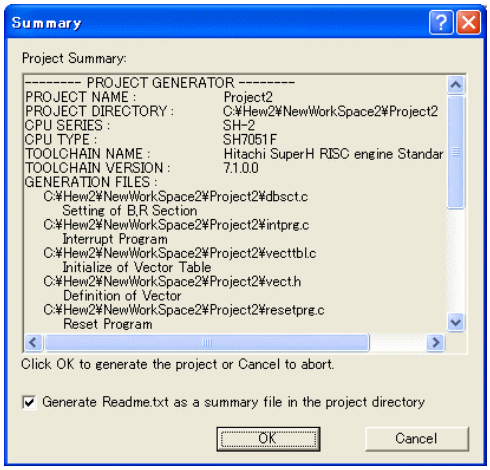
Ｎｅｘｔ＞クリックして下さい。



ここで最終になります。

使用するＣモジュールを表示します。

この状態でＦｉｎｉｓｈをクリックして下さい。



確認画面が表示されますので、ＯＫをクリックして下さい。

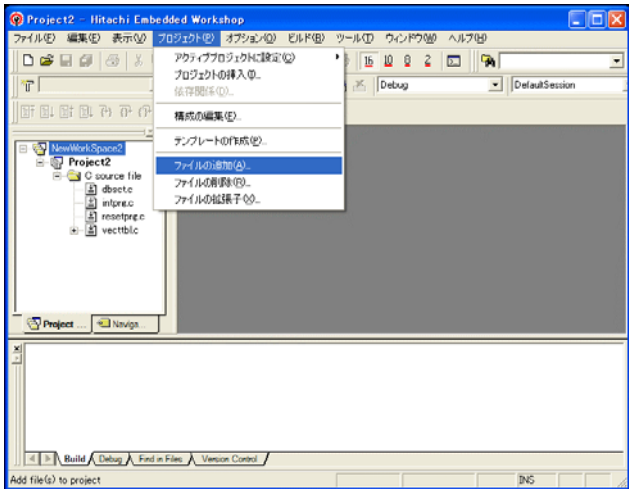
ここまでの操作が新規プロジェクトの登録方法です。

２．プロジェクトに希望モジュール（ソースファイル）を登録する方法

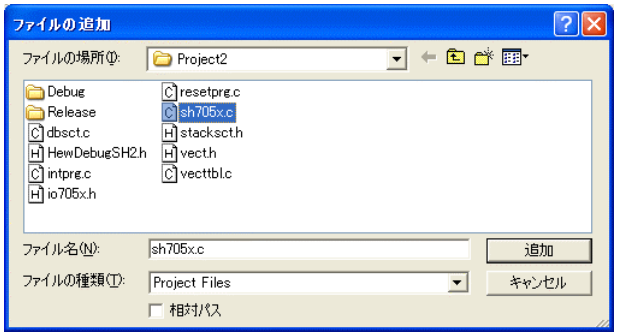
準備：　作成済みのソースファイルを”C:¥Hew2¥NewWorkSpace2¥Project2”にコピーして下さい。

SH705x.c
Io705x.h
（iodefine.h）
HewDebugsh2.h

”¥sample¥HewSH22_0¥SH7051¥”
(左記ソースは、製品CDの上記フォルダに入っています。)
又は、Ｈｅｗ提供のＩ／Ｏ定義ファイルを使用しても構いません。



[プロジェクト]-[ファイルの追加]をクリックします。



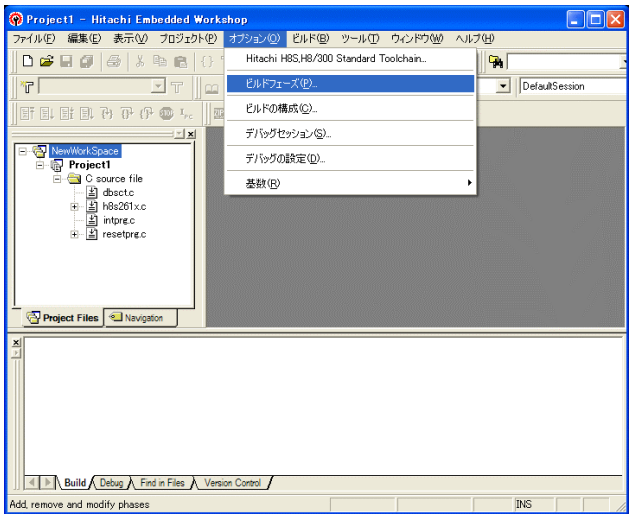
下記１ファイルを指定して下さい。

Sh705x.c

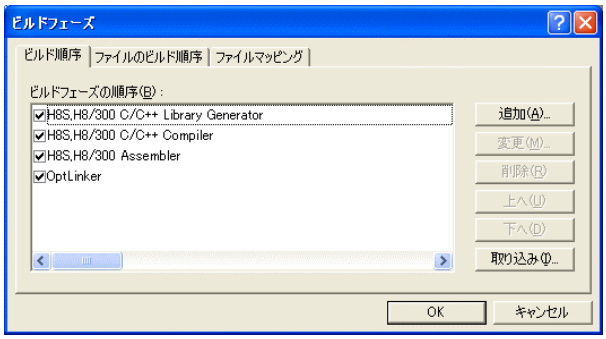
選択後、追加をクリックします。

この操作によりプロジェクトにモジュールが登録されました。

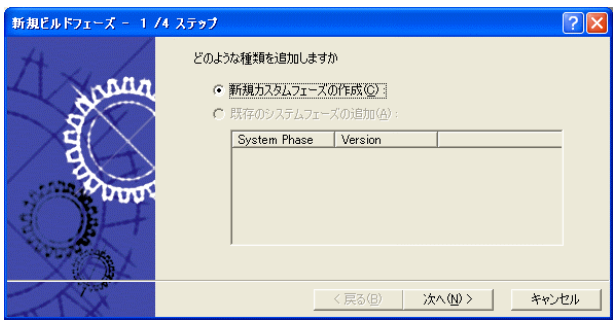
３．シンボルコンバータ「ＨＣｓｙｍｃｏｎｖ」を登録する。



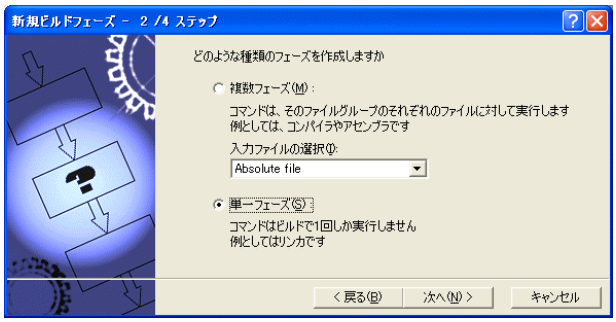
[オプション]-[ビルドフェーズ]をクリックします。



追加をクリックします。

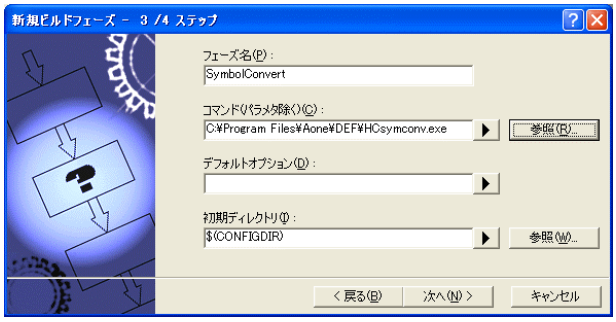


次へ>をクリックします。



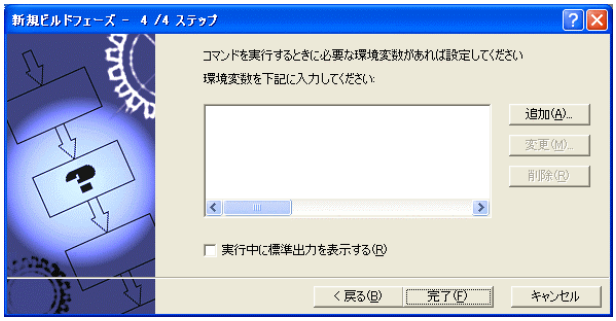
単一フェーズ側にチェックをします。

次へ>をクリックします。

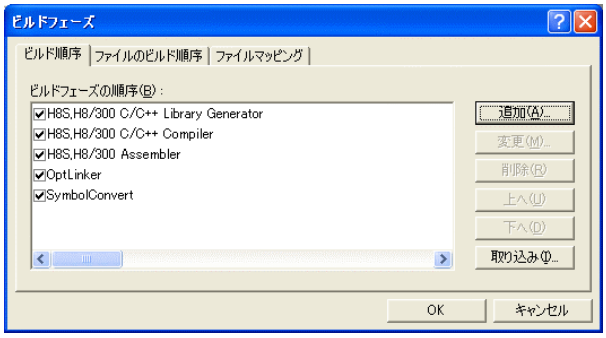


フェーズ：SymbolConvert
コマンド：C:¥ProgramFiles¥Aone¥DEF¥HCsymconv.exe を選択する。（ﾃﾞﾌｫﾙﾄ）
初期ディレクトリ：$(CONFIGDIR)

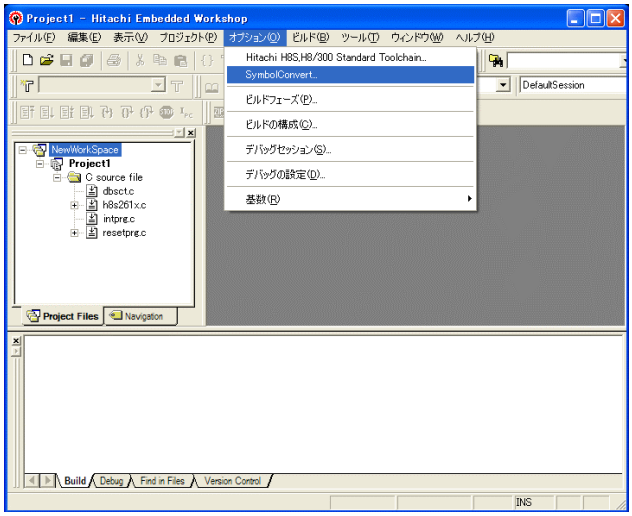
次へ>をクリックします。



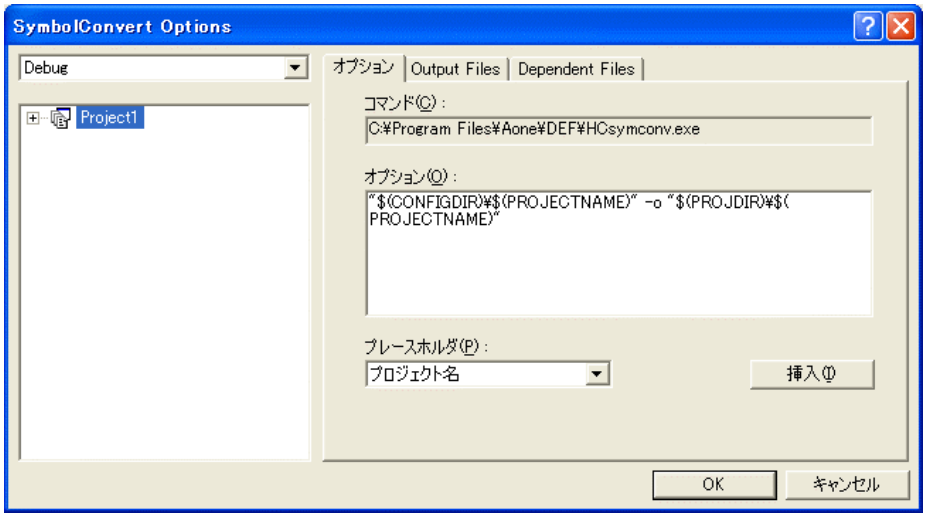
完了をクリックします。



OKをクリックします。



[オプション]-[ＳｙｍｂｏｌＣｏｎｖｅｒｔ]をクリックします。



オプションに下記内容を設定する。

"$(CONFIGDIR)¥$(PROJECTNAME)" -o "$(PROJDIR)¥$(PROJECTNAME)"

１（入力ファイル名）　　（出力先名）

注意事項

１）ディレクトリ名に‘　’スペースを使用している場合は、“ダブルクォートで囲んで下さい。
“$(CONFIGDIR)¥$(PROJECTNAME)”-o “$(PROJDIR)¥$(PROJECTNAME)”

２）$(PROJECTNAME)の先頭に「¥」記号を挿入して下さい。（手入力）

３）オプションＳＷ「－ｏ」の両端には、スペースを入れてください。（手入力）

４）この設定例は、後説明の「*.mot」ファイルの生成されるディレクトリと同じ場所にシンボルコンバータが生成する「*.sym/*.lin」を置く為の指定です。
＜コンフィグレーション DIR＞に生成させたい場合は、
“$(CONFIGDIR)¥$(PROJECTNAME)”
の指定のみで構いません。
この場合は「*.mot」の生成場所を同じく＜コンフィグレーション DIR＞にして下さい。

追加事項（ＨＣｓｙｍｃｏｎｖ．ｅｘｅ　スイッチ説明）

１）［－ｏ］（省略可）出力ファイル名を指定

２）［－ｒ］（省略可）モジュール毎のディレクトリ情報を作成しない。ELF 専用(Ｖｅｒ３．２ｘｘから)

３）［－ｓ］（省略可）ラインシンボル情報をソート（アドレス順）しない。(Ｖｅｒ３．２ｘｘから)

４）［－ｉ］（省略可）重複モジュール情報を削除する。(Ｖｅｒ３．３ｘｘから)

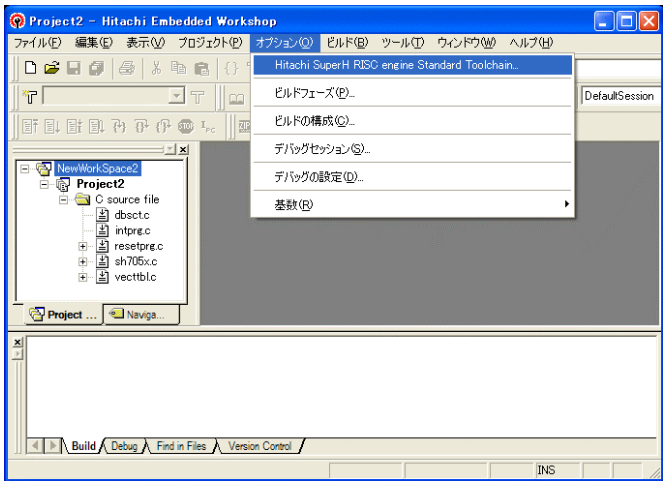
５）［－ｇ］（省略可）スタティック変数をグローバル化する。(Ｖｅｒ３．６ｘｘから)

６）［－m］（省略可）重複モジュール情報をＣソースにマージする。(Ｖｅｒ３．８０Ｂから)

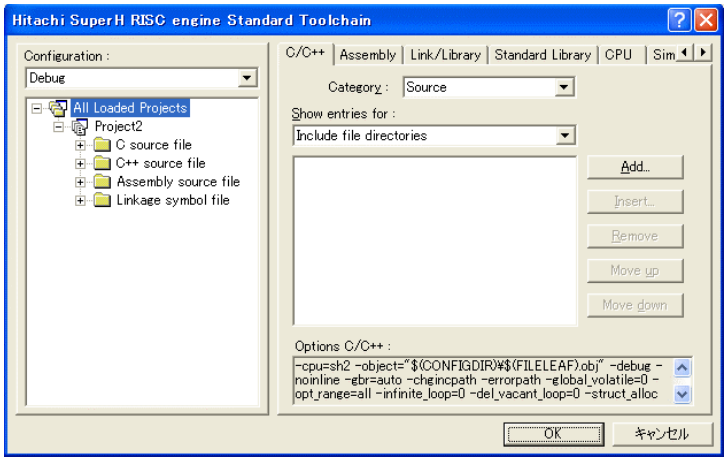
７）［－ｆ］（省略可）使用インクルードファイルをCViewに登録する。(Ｖｅｒ３．８０Ｂから)

4．ツール（ライブラリ）の設定

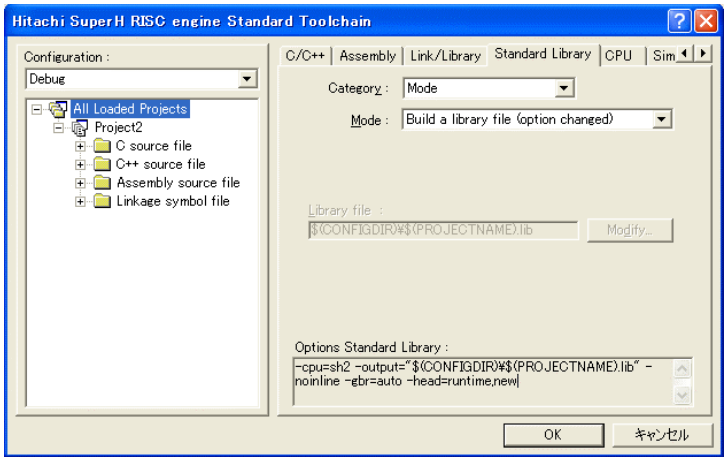
ＨＥＷは、プロジェクトごとにＣ言語用ライブラリを作成する仕様になっています。
ライブラリを作成および設定の確認をします。



[オプション]-[Hitach SuperH RISC engine Standard Toolchain]をクリックします。



[Standard Library]タグをクリックする。

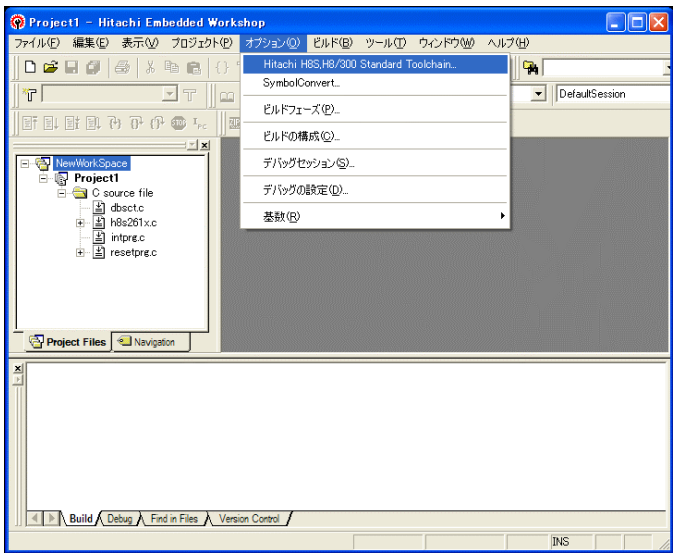


カテゴリの Mode が「Build a Library file(option changed)」指定になっている事を確認します。

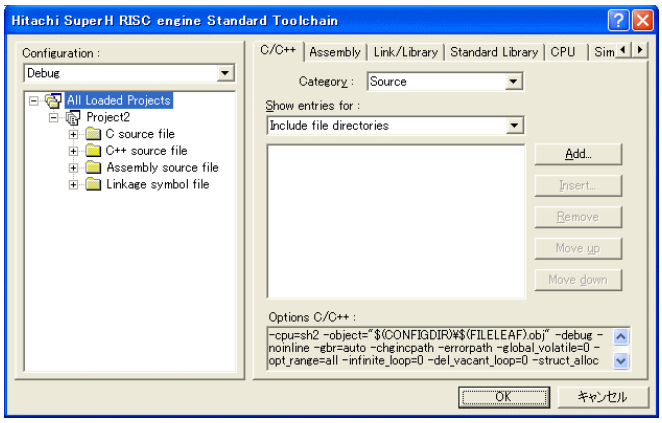
この指定によりオプション変更時のみライブラリを作成する事になります。

OKをクリックする。

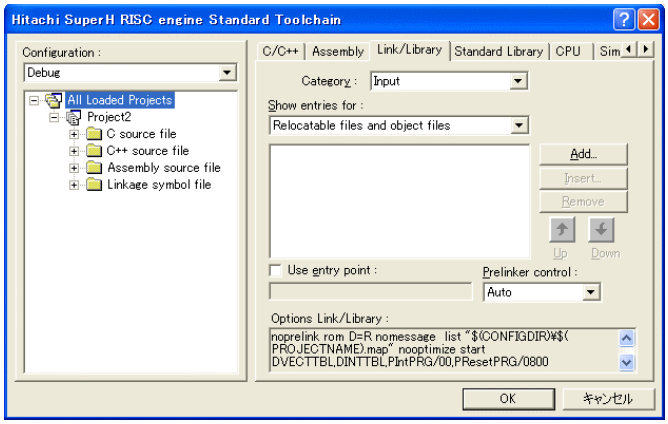
５．ツール（リンカ）の設定



[オプション]-[Hitach SuperH RISC engine Standard Toolchain]をクリックします。



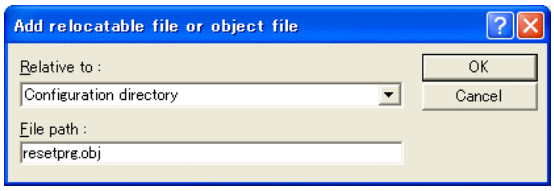
「Link/Library」タグをクリックする。



カテゴリの「input」を選択する。

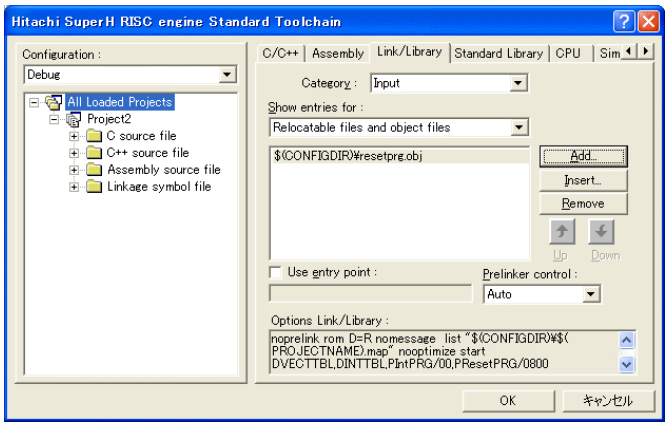
Show entries for:項目の「Relocatable files and object files」を選択する。

Ａｄｄをクリックする。



Relative to:「Configuration Directory」に選択します。
File Path:「resetprg.obj」と入力します。

OKをクリックします。

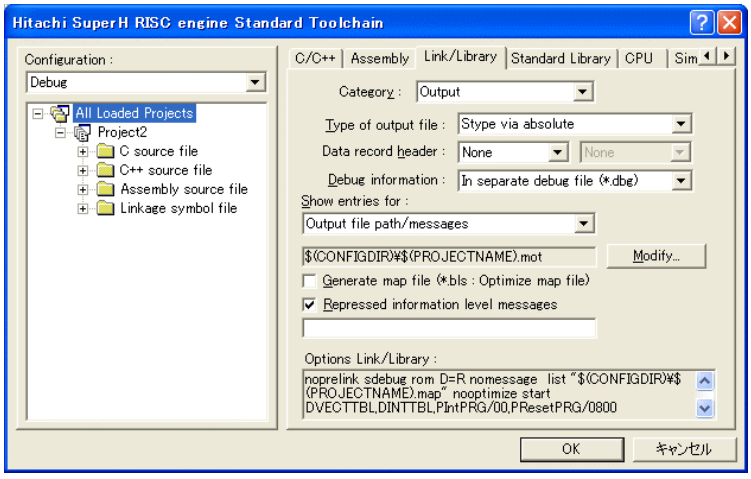


ファイルがセットされたのを確認します。

この指定は、“最初にこのリスト順にモジュールをリンクしなさい”との指示になります。

（DEF の C View にてアドレス順に他モジュールを表示させたい場合は、追加設定してください。）

（重要）　特に「resetprg.obj」の先頭に SP を設定するコードがありますので先頭を０ｘ１０００番地にする為の設定ですが、後説明のセクション指定で「PResetPRG」の先頭番地を０ｘ１０００番地にしていますので必要ありませんが、同セクション名の他モジュールを追加した場合でも、必ず最初にリンクさせるための手続きです。



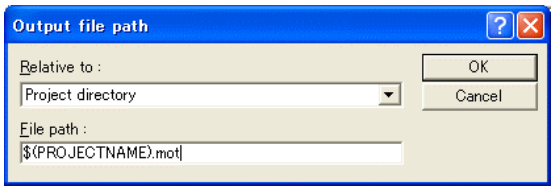
Category の
「Output」を選択する。

Type of output file:の
「Stype via absolute」を選択する。

Debug information:の
「In separate debug file (*.dbg)」を選択する。

「Output file path/….」の
Modifyをクリックします。

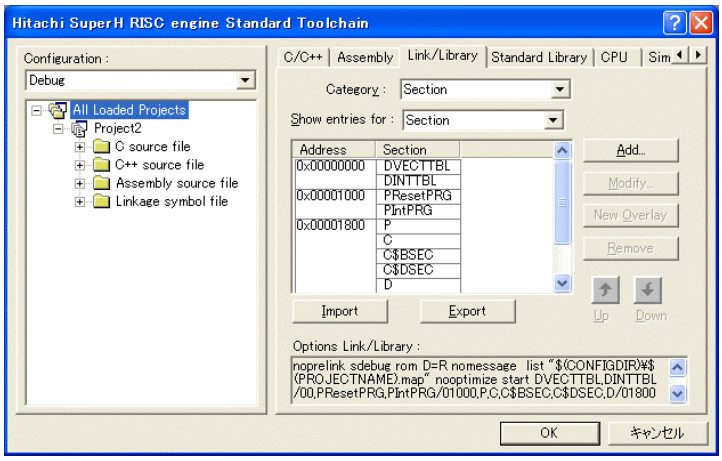
(重要)　シンボリックデバッグを可能にするために必要な設定です。



Relative to:を「Project directry」に設定します。

OKをクリックする。

(重要)　この指定は、ＨＥＸファイルをＣソースファイルのある同じディレクトリに置くための設定です。絶対条件として、「＊．ｍｏｔ／＊．ｓｙｍ／＊．ｌｉｎ」は、同じ場所に置く必要があります。
ＨＣｓｙｍｃｏｎｖで出力ファイルを「Configuration directory」にした場合は、上記の指定も「Configuration directory」にして下さい。今回の使用例は、「Project directry」になっています。



Category:の「Section」を選択する。

下図のようにセクション指定をする。

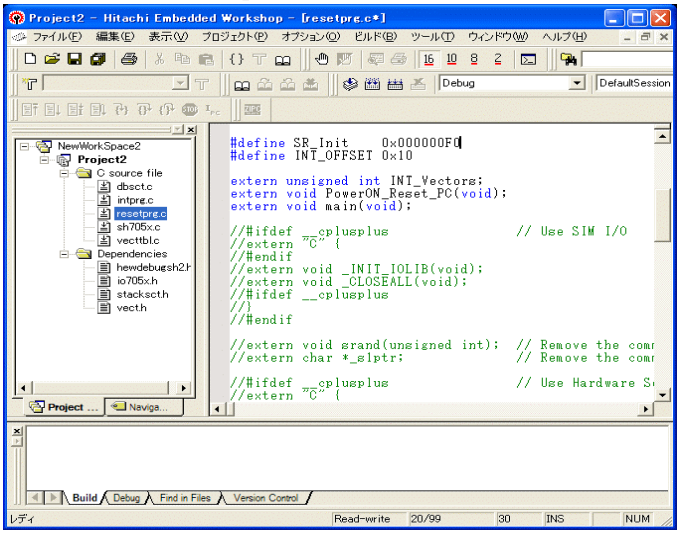
0x00000000	DVECTTBL
	DINTTBL
0x00001000	PResetPRG
	PIntPRG
0x00001800	P
	C
	C$DSEC
	C$BSEC
	D
0xFFFFE840	B
	R
0xFFFFFBFC	S

OKをクリックする。

(重要)　「Ｂセクション」０ｘＦＦＦＦＥ８４０（０ｘＦＦＦＦＥ８００）は、ソースブレーク使用時の例です。
ＤＥＦバージョン６．５０ｘから、ソースブレークを使用する場合は、モニタワーク方式をスタック方式に選択する必要があります。

６．スタートアップおよびベクターの変更と追加

１）　＜ｒｅｓｅｔｐｒｇ．ｃ＞の変更

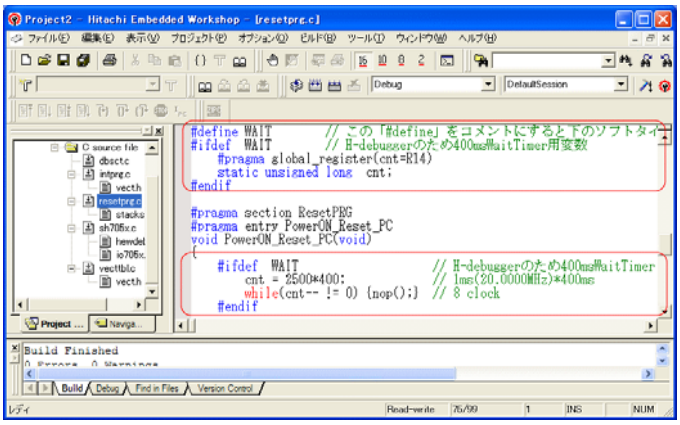


①SR_Init 値を「0x000000F0」で変更しません。

（備考）

「resetprg.c」関数内でブレークポイント及びトレース／ステップ実行をしたい場合は、DEF 操作のショートＰＢ「ＥＩ」をクリックして下さい。

この例での割り込み許可は、「_main()」関数内に記述してあります。

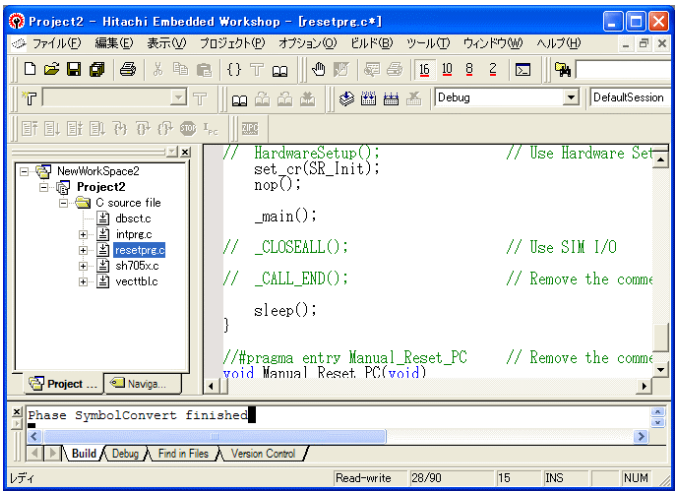


①左図のようなソースを追加しますとリセット時ソフトタイマ挿入になります。
このソースを追加することにより下記メリットがあります。

②ソフトタイマが不必要な場合は「#define WAIT」をコメントにして下さい。

③ソフトタイマによるメリット

・ＤＥＦの「Ｒｓｔ Ｍｏｎ」操作時ＳＰ値が設定値のままである。
・ＤＥＦの「Ｒｓｔ Ｍｏｎ」操作時「ｍａｉｎ ()」プログラムまで走行しないため初期設定等による内部レジスタが汚れない。



①「main()」を「_main()」に変更
特に意味はありません。
SH-2 のサンプルは全て「_main()」になっているので共通化する為です。

2）ブレークおよびトレース／ステップ実行について

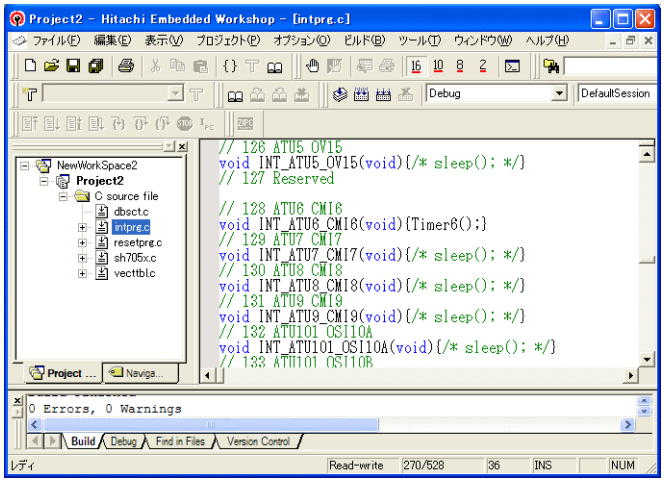
ＵＢＣブレーク割り込みのプライオリティは、１５になっていますのでＣＰＵのＳＲレジスタの割り込みマスクレベルを１４以下に設定する必要がありますので、下記方法のどちらかを都合に応じて実施して下さい。

なお、トレース／ステップはＵＢＣを利用したソフト判定での実施となります。

【方法①】 「set_imask(14)」を記述する。

【方法②】 ＤＥＦ操作でのショートＰＢ「ＤＩ」をクリックする。

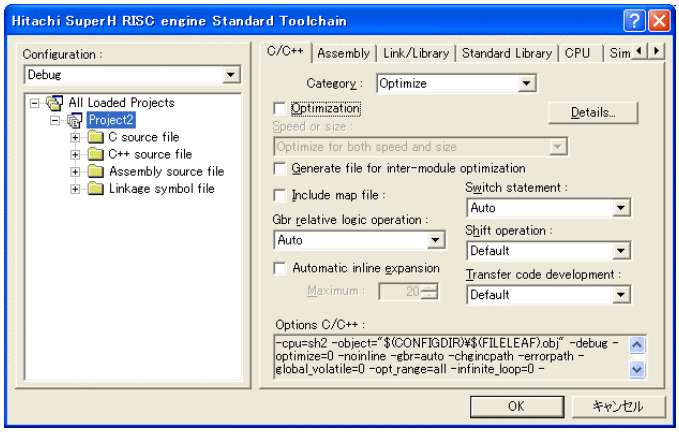
3） ＜ｉｎｔｐｒｇ．ｃ＞の変更



今回使用の「SH705x.C」は、ATU6のCMI6割り込みを使用した例ですのでベクターを設定します。

①「vector 128」に「Timer6();」関数を登録します。

4） コンパイラの「最適化」を外す



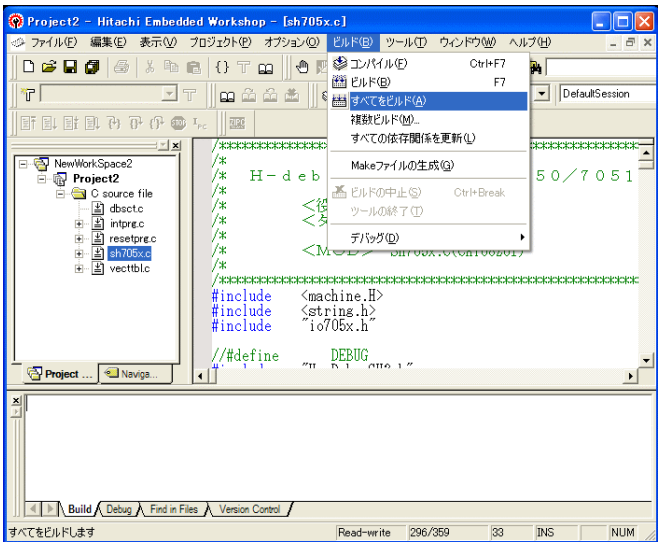
①Toolchain の「C/C++」を選択

②Category:「Optimaize」を選択

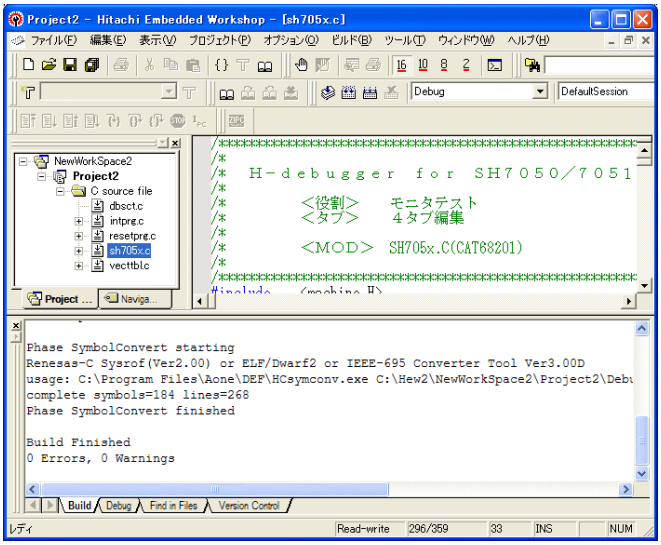
③「Optimization」のチェックを外す。

④ＯＫをクリックする。

7．ビルドの実行

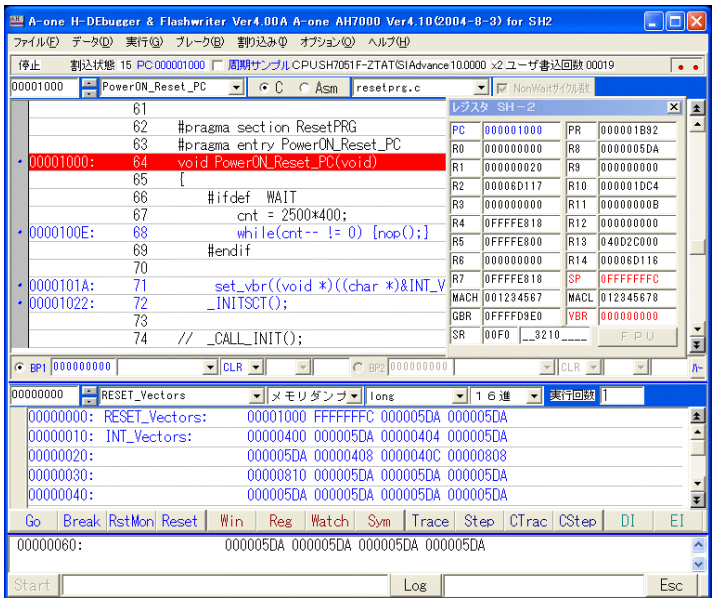


［ビルド］-[すべてをビルド]をクリックします。

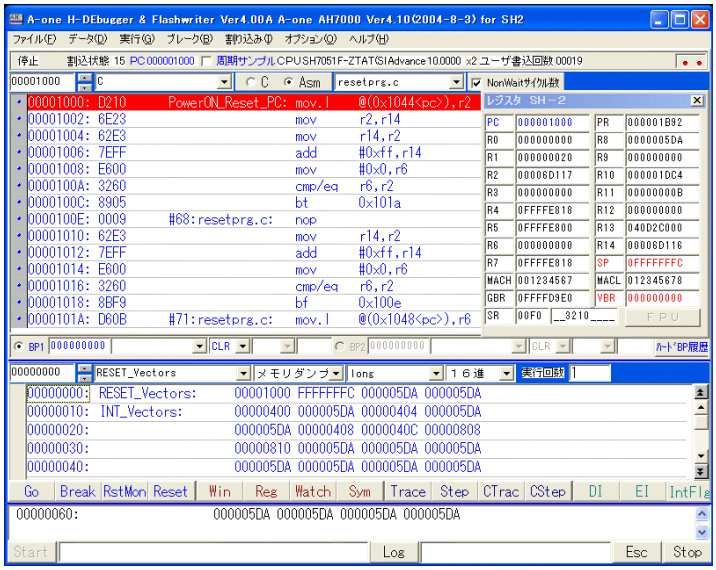


「0 Error　0 Warnings」になり作業終了です。

８．ＤＥＦでの確認



①０番地のリセットベクター値「0x1000」が確認できます。
②４番地のスタックポインタ値「0xFFFFFFFC」が確認できます。
③レジスタ画面を確認して下さい。



①ソフトタイマの逆アセンブラ表示です。(参考まで)

②SR 値および SP 値がリセット状態であることが確認できます。

これで「Ｈ－ｄｅｂｕｇｇｅｒ」用に設定ができたのが確認できました。

以上