HEXファイル指定でダウンロードする場合

２００８年０４月１８日
Ｂｙ　エーワン　Ｒｅｖ１．２３

Ｈｅｗ（Ｖｅｒ４．０４）＋ＫＰＩＴ（ｖ０７０３）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

（ＳＨ７０５１　ＢＯＯＴ版）

ＫＰＩＴ版の統合環境「Ｈｅｗ　Ｖｅｒ４．０４」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	KPIT7051		
プロジェクト名	Project		
登録モジュール名	SH7051.c	C	メインモジュール（アプリ用）
	KpitDebugSH2.h	ヘッダ	ソフトパーツ用定義ファイル （ソフトパーツを使用しない場合は不要です。）
KPIT 添付ファイル	start.asm	ＡＳＭ	スタートアップモジュール
	hwinit.c	C	ハード初期化用モジュール
	vects.c	C	リセット／割込みベクターテーブル
	inthandler.c	C	割込みハンドラー用
	iodefine.h	ヘッダ	I/O 定義ビットフィールド記述用
	inthandler.h	ヘッダ	割込みハンドラー用
ＣＰＵタイプ	SH7051F		

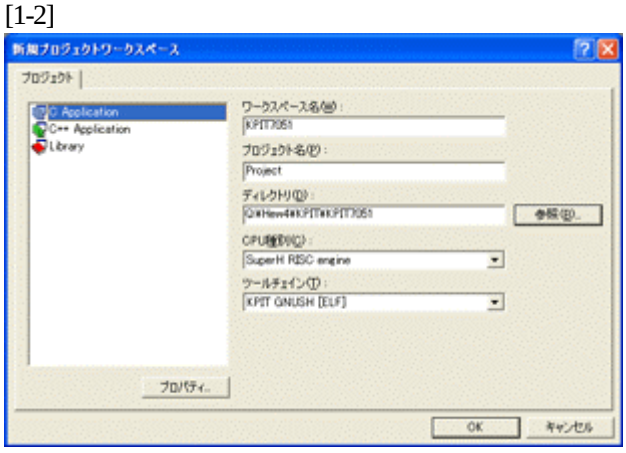
１．新ワークスペースの登録方法　　“HEW”起動させます。

[1-1]

“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。

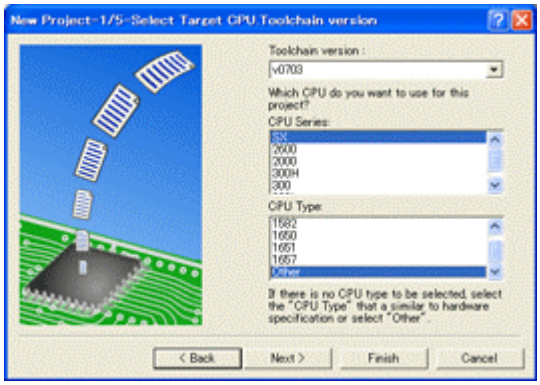
[1-2]



ﾜｰｸｽﾍﾟｰｽ名	KPIT7051
ﾌﾟﾛｼﾞｪｸﾄ名	Project
ﾃﾞｨﾚｸﾄﾘ	C:¥Hew4¥KPIT¥KPIT7051
CPU 種別	SuperH RISC engine
ﾂｰﾙﾁｪｲﾝ	KPIT GNUSH[ELF]
ﾌﾟﾛｼﾞｪｸﾄ	Application

この項目を設定確認後ＯＫをクリックして下さい。

[1-3]



ＣＰＵスペックを選択します。

①SH2
②SH7051F

Ｎｅｘｔ＞をクリックします。

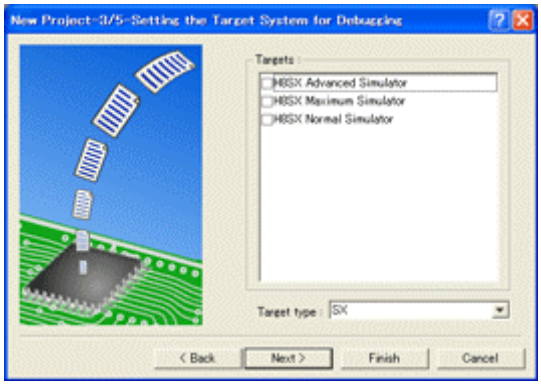
[1-4]

ＣＰＵオプションを選択します。

①「Use Optimized Libraries」
チェックを外します。

Ｎｅｘｔ＞をクリックします。

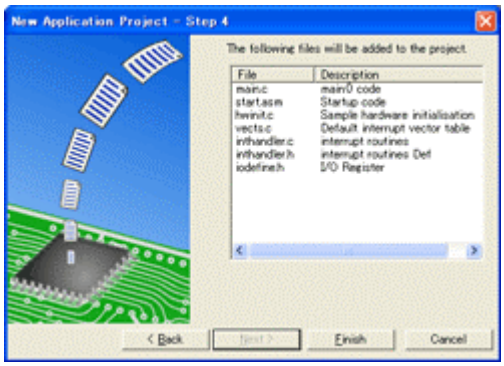
[1-5]



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＞をクリックします。

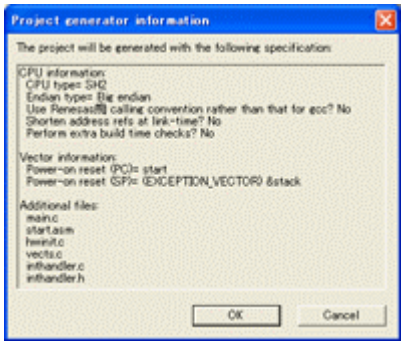
[1-6]



作成されるファイル一覧表示です。

Ｆｉｎｉｓｈ＞をクリックします。

[1-7]



最終確認画面です。

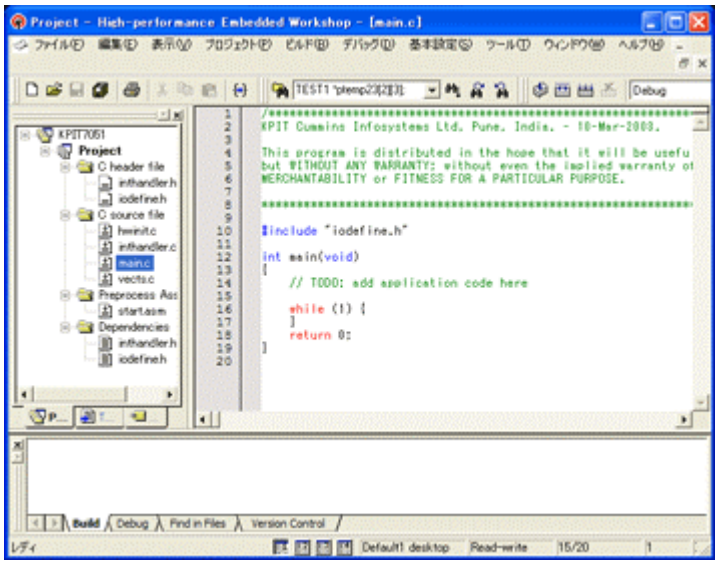
OKをクリックします。

ここまでの操作が新規プロジェクトの登録方法です。

２．プロジェクトから不要モジュール（ソースファイル）を削除します。

目的：　ＫＰＩＴにより準備されたモジュールを使用しない場合に削除しておきます。

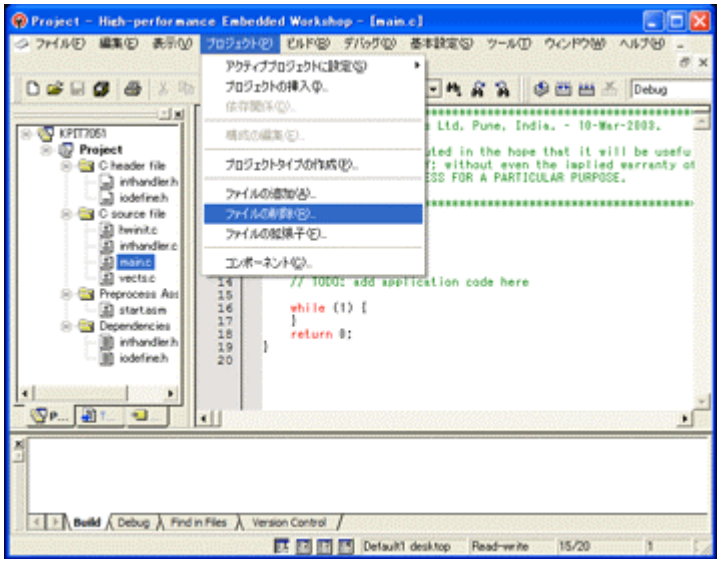
[2-1]



今回の使用例では下記１ファイルを削除します。

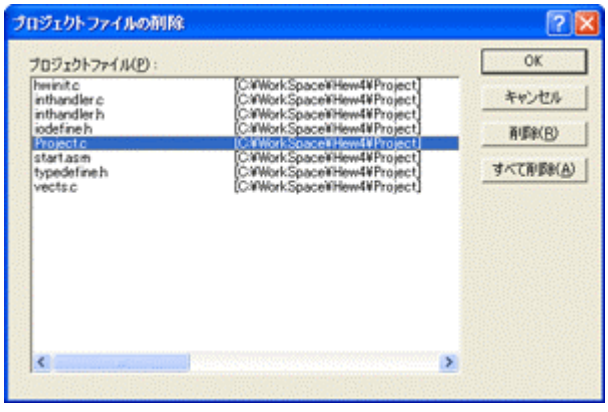
①Project.c

[2-2]



［プロジェクト］－
［ファイルの削除］をクリックします。

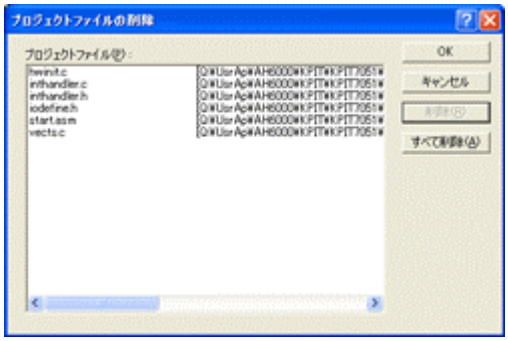
[2-3]



① project.c
の１ファイルを選択する。

削除をクリックします。

[2-4]



確認画面です。

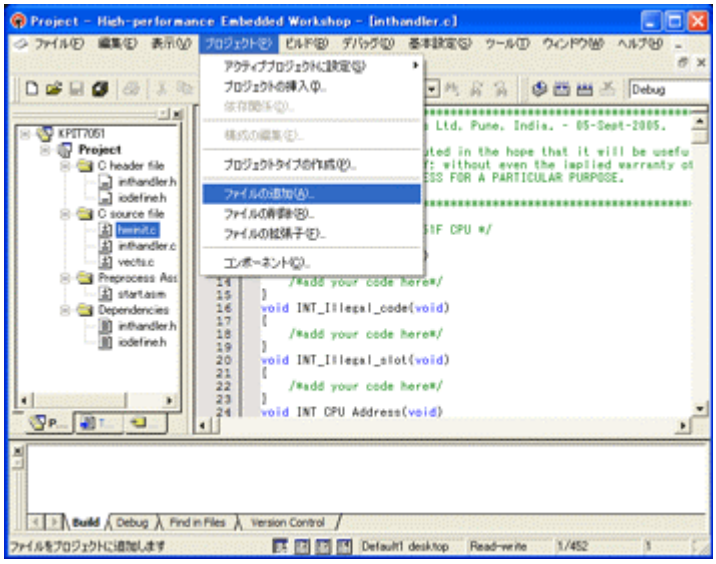
OK をクリックします。

３．プロジェクトに希望モジュール（ソースファイル）を登録します。

準備：　作成済みの２ファイルを”C:¥Hew4¥KPIT¥KPIT7051¥Project”にコピーします。

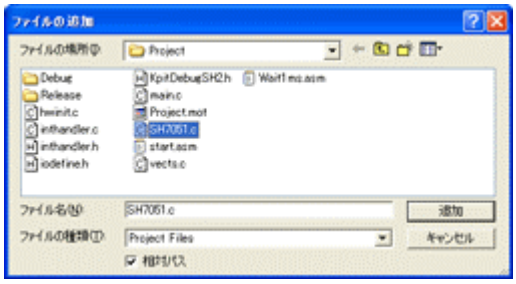
SH7051.c	HP よりダウンロードします。（ＧＮＵ／ｇｃｃ）
KpitDebugSH2.h	KPIT7051_v0703_1.LZH

[3-1]



[プロジェクト]-
[ファイルの追加]をクリックします。

[3-2]



登録ファイルを選択します。

① SH7051.c

追加をクリックします。

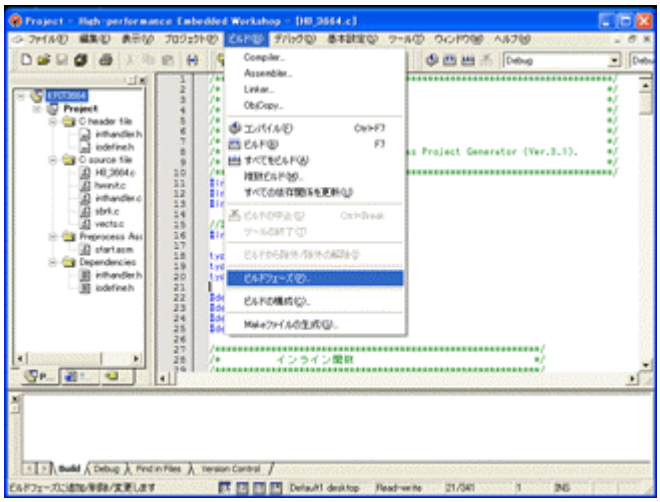
この操作によりプロジェクトにモジュールを登録します。

４．シンボルコンバータ「ＧＣｓｙｍｃｏｎｖ」を登録します。

目的：H-debugger でシンボリックデバッグする為にアブソリュートファイル【Project.x】からシンボル情報抽出します。

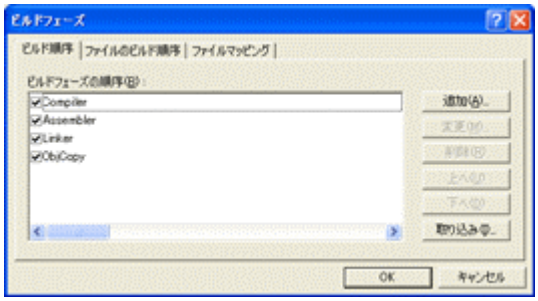
準備：KPIT 用シンボルコンバータ【GCsymconv.exe】をホームページよりダウンロードし解凍後、DEF インストールＤＩＲ「C:¥Program Files¥Aone¥DEF」下にコピーして下さい。

[4-1]



［ビルド］−
［ビルドフェーズ］をクリックします。

[4-2]

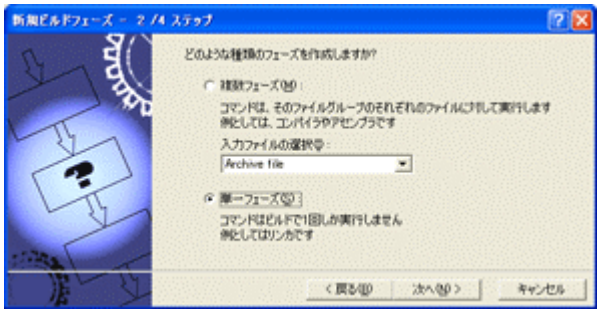


追加をクリックします。

[4-3]

次へ>をクリックします。

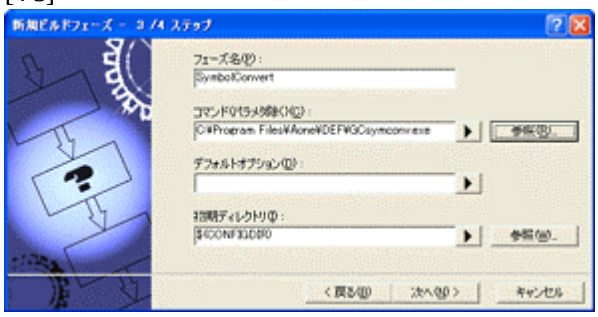
[4-4]



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。

[4-5]



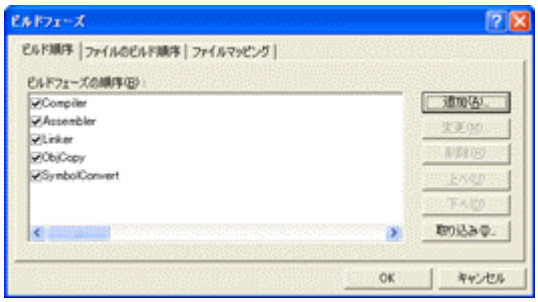
①フェーズ：SymbolConvert
②コマンド：
C:¥ProgramFiles¥
Aone¥DEF¥GCsymconv.exe を選択する。
③初期ディレクトリ：$(CONFIGDIR)

次へ>をクリックします。

[4-6]

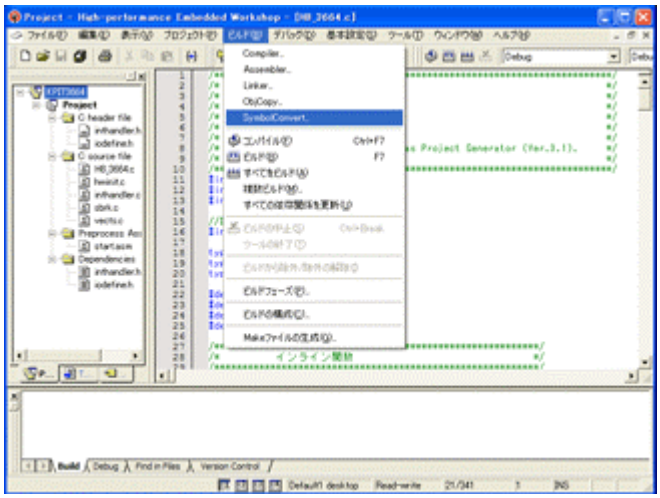
完了をクリックします。

[4-7]



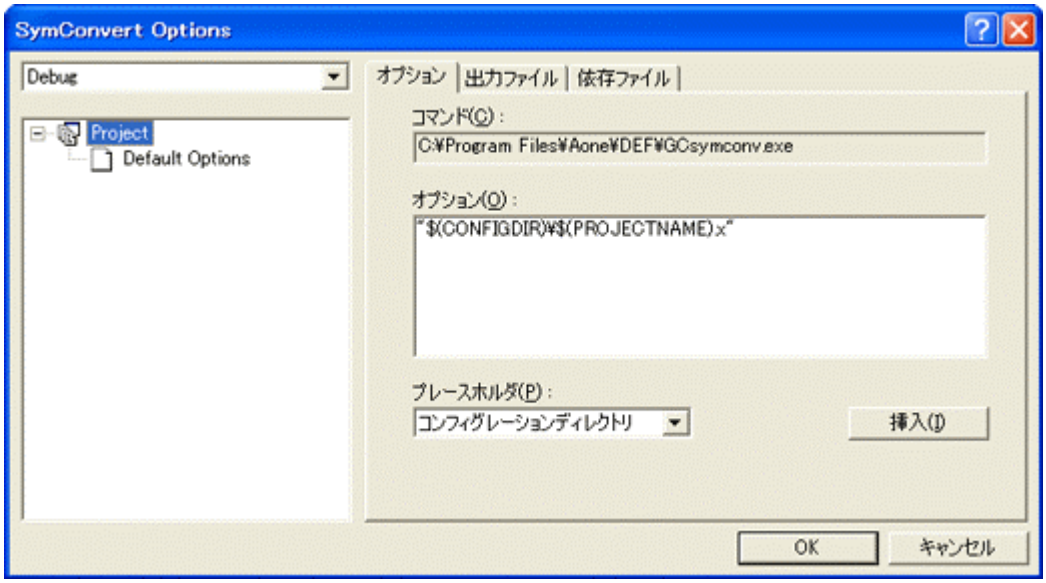
OKをクリックします。

[4-8]



[ビルド]-
[ＳｙｍｂｏｌＣｏｎｖｅｒｔ]をクリックします。

[4-9]



①オプションに下記内容を設定する。
"$(CONFIGDIR)¥$(PROJECTNAME).x"
（入力ファイル名）

②ＯＫをクリックします。

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).x”
②$(PROJECTNAME)の先頭に「¥」記号を入力して下さい。（手入力）
③オプションＳＷを使用する場合は両端にスペースを入れてください。（手入力）

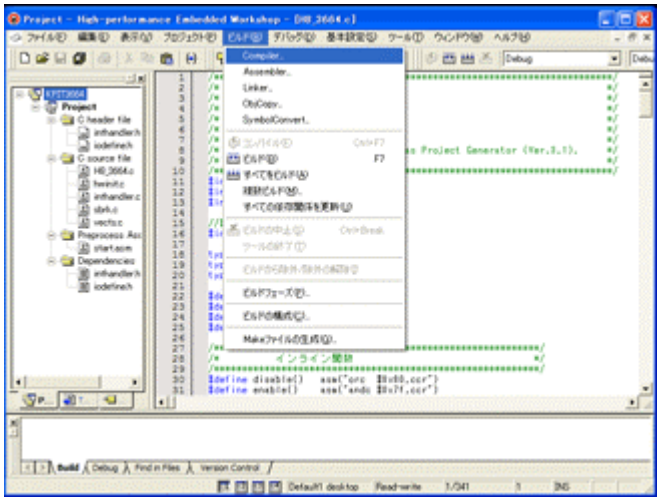
追加事項（スイッチ説明）

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。
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。
４）［－ｉ］（省略可）重複モジュール情報を削除する。
５）［－ｇ］（省略可）スタティック変数をグローバル化する。(Ｖｅｒ１．２０ｘ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。(Ｖｅｒ１．４０Ｂ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。(Ｖｅｒ１．４０Ｂから)

５．コンパイラーオプションの確認と設定をします。

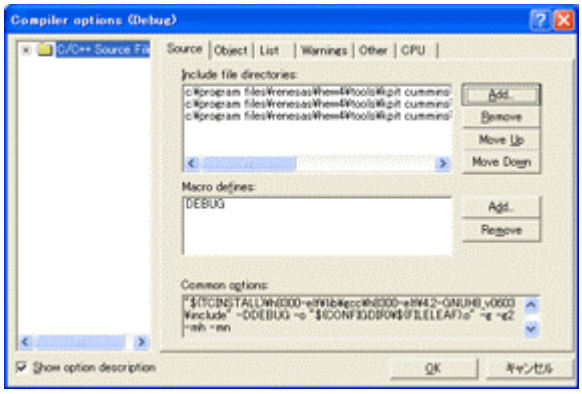
目的：　H-debugger でシンボリックデバッグを可能にする為、コンパイラーオプションの確認と設定をします。

[5-1]



［ビルド］-
［Ｃｏｍｐｉｌｅｒ］をクリックします。

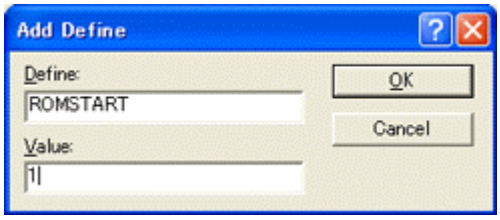
[5-2]



［Ｓｏｕｒｃｅ］タグ
①「Macro defines」の

Ａｄｄをクリックします。

[5-2-1]

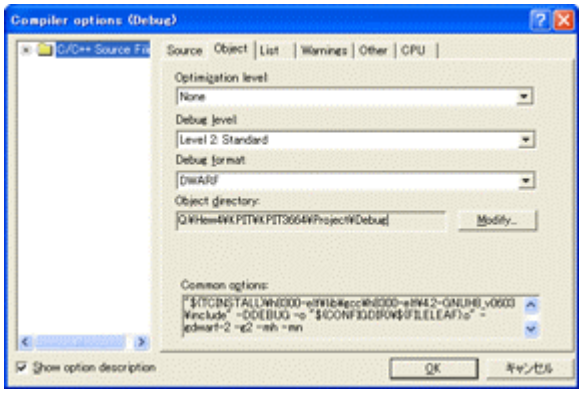


②Define:「ROMSTART」を登録します。
③Value:「１」を登録します。

④ＯＫをクリックします。

初期値を ROM->RAM にコピーする為の重要な設定です。

[5-3]



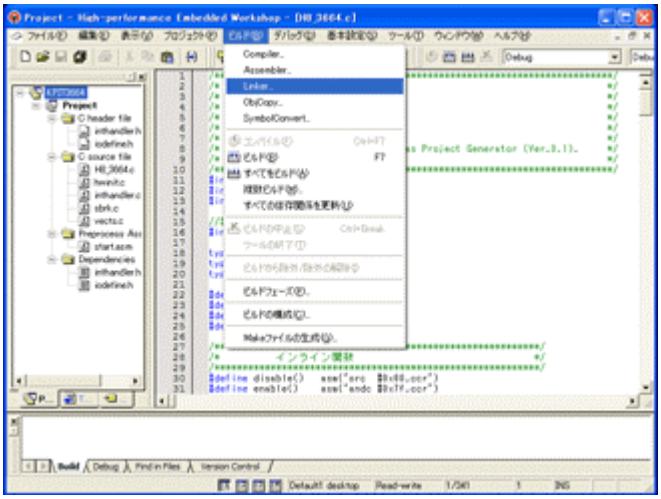
［Ｏｂｊｅｃｔ］タグ
①Optimization : None(Default)
②Debug level :　Level2:Standard(Default)
③Debug format:**DWRAF** に指定する。
④Object directory : (Default)状態

ＯＫをクリックします。

６．リンカーオプションの確認と設定をします。

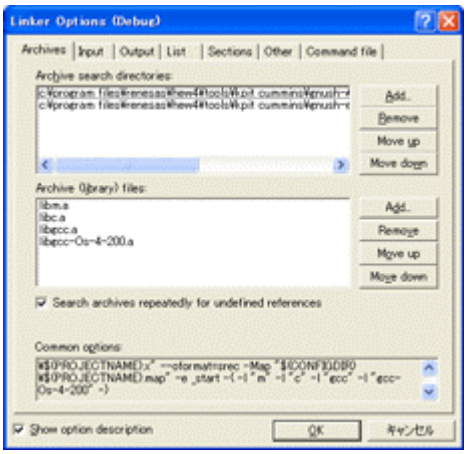
目的：　H-debugger でシンボリックデバッグを可能にする為、リンカーオプションの確認と設定をします。

[6-1]



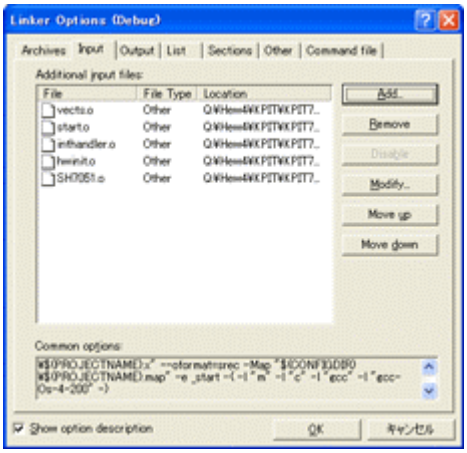
［ビルド］-
［Ｌｉｎｋｅｒ］をクリックします。

[6-2]



［Ａｒｃｈｉｖｅｓ］タグ
デフォルト状態です。(変更の必要なし)

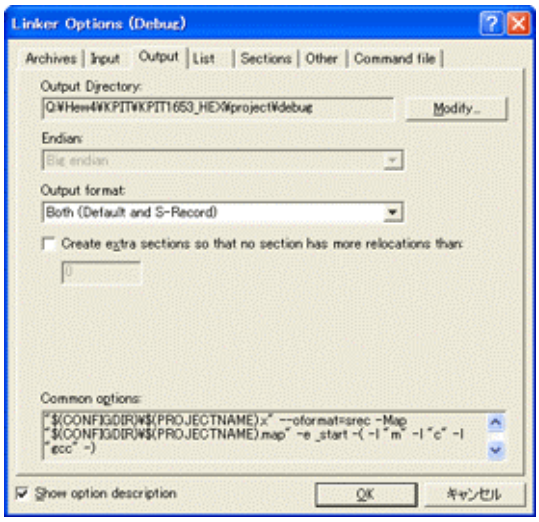
[6-3]



［Ｉｎｐｕｔ］タグ
　基本的には何も設定しなくて良いですが、各モジュールのリンク順番を指定したい場合に全モジュールをここで指定します。

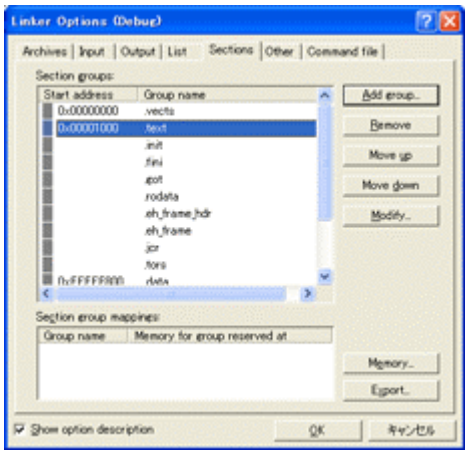
① vects.o
② start.o
③ inthandler.o
④ hwinit.o
⑤ SH7051.o

[6-4]



［Ｏｕｔｐｕｔ］タグ
①Output Directory : (Default)
②Endian : Big endian(Default)
③Output format :Both(Default and S-Record)
すべて、デフォルトです。

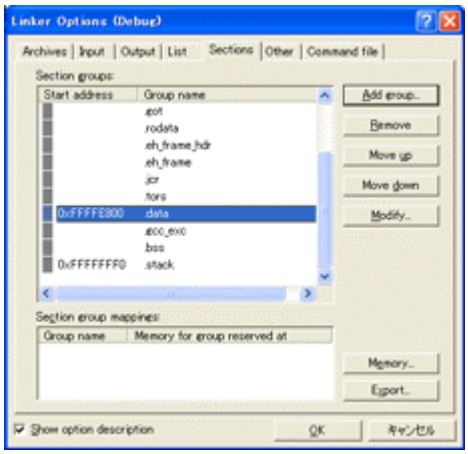
[6-5]



［Ｓｅｃｔｉｏｎｓ］タグ
.text セクションの開始アドレスを確認します。
（モニターエリアを空ける為）

①.text セクションの開始アドレスが【0x1000】になっているのを確認します。.

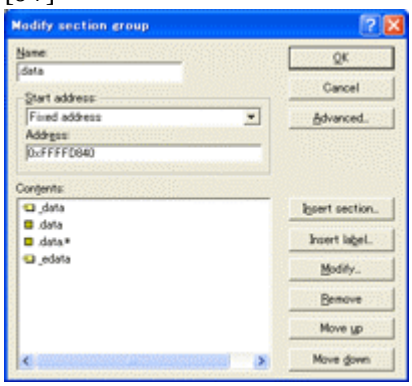
[6-6]



［Ｓｅｃｔｉｏｎｓ］タグ
.data セクションのアドレスを変更します。

①.data セクションを選択します。
②【Modify】PB をクリックします。

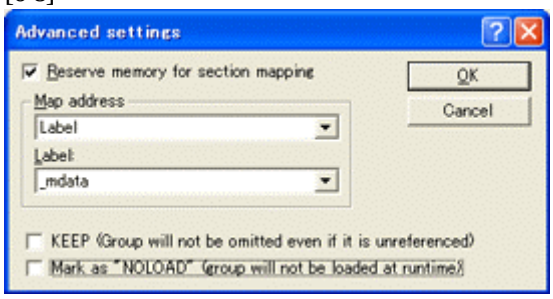
[6-7]



③Start address:【Fixed address】に選択します。
④Address: .data セクションの先頭アドレスを指定します。
ソースブレークを使用する場合　　【0xFFFFD840】
ソースブレークを使用しない場合　【0xFFFFD800】
＜DEF バージョン 6.50A より＞
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する必要があります。。

⑤Advanced をクリックします。

[6-8]

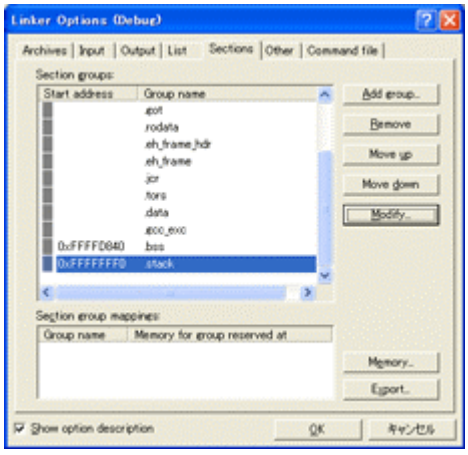


⑥「Reserve memory for section mapping」をチェックする。
⑦Map address:【Label】にする。
⑧Label:【_mdata】

⑨OK をクリックします。

初期値を ROM->RAM にコピーする為の重要な設定です。

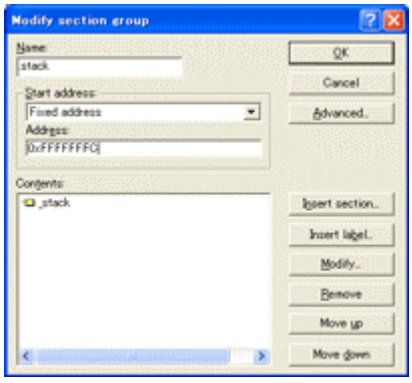
[6-9]



［Ｓｅｃｔｉｏｎｓ］タグ
.stack セクションのアドレスを指定します。
ここでの指定値は、スタックポインタへの初期設定値になります。

①.stack セクションを選択します。
②【Modify】PB をクリックします。

[6-10]



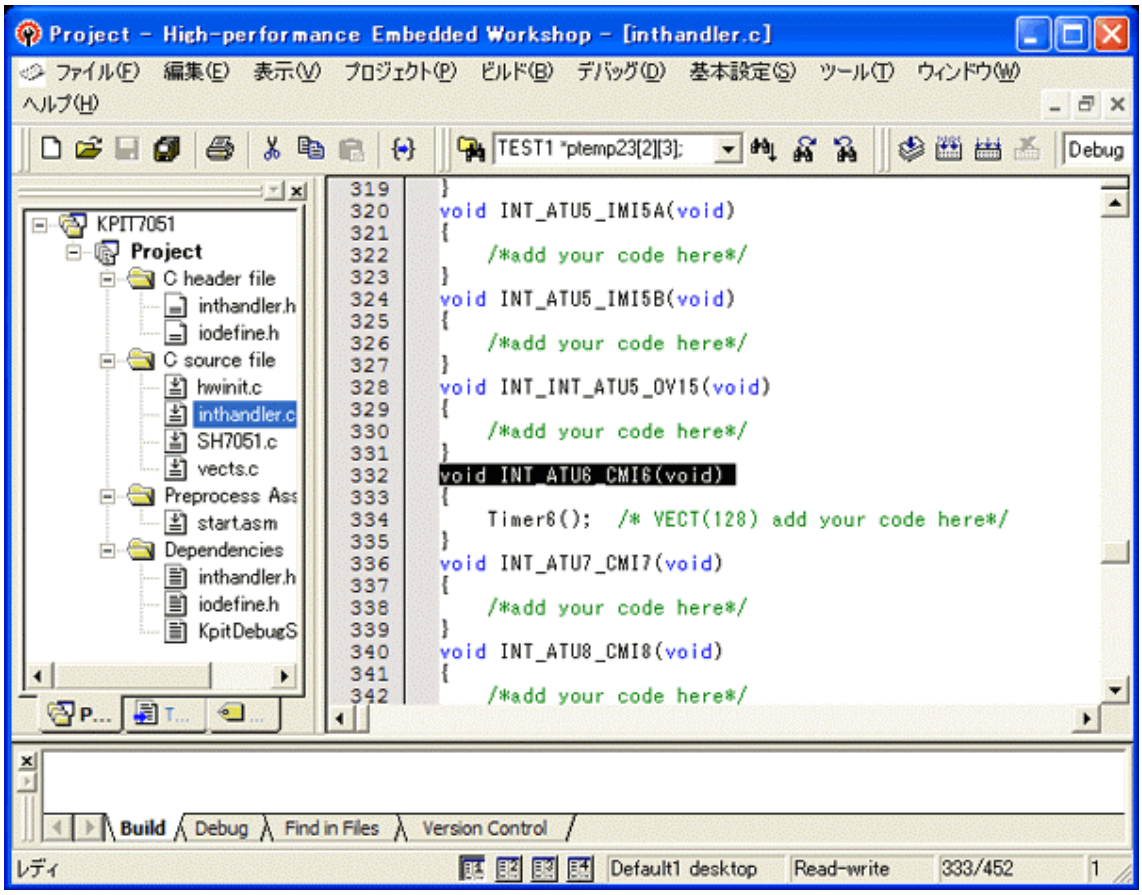
③Address: を「0xFFFFFFFC」に変更します。

④OK をクリックします。

７．割り込みハンドラへ登録します。

目的：　今回説明に使用したモジュール「SH7051.c」は、Timer6（ベクター１２８）の割り込みを使用していますので、割り込みハンドラへ登録します。

[7-1]



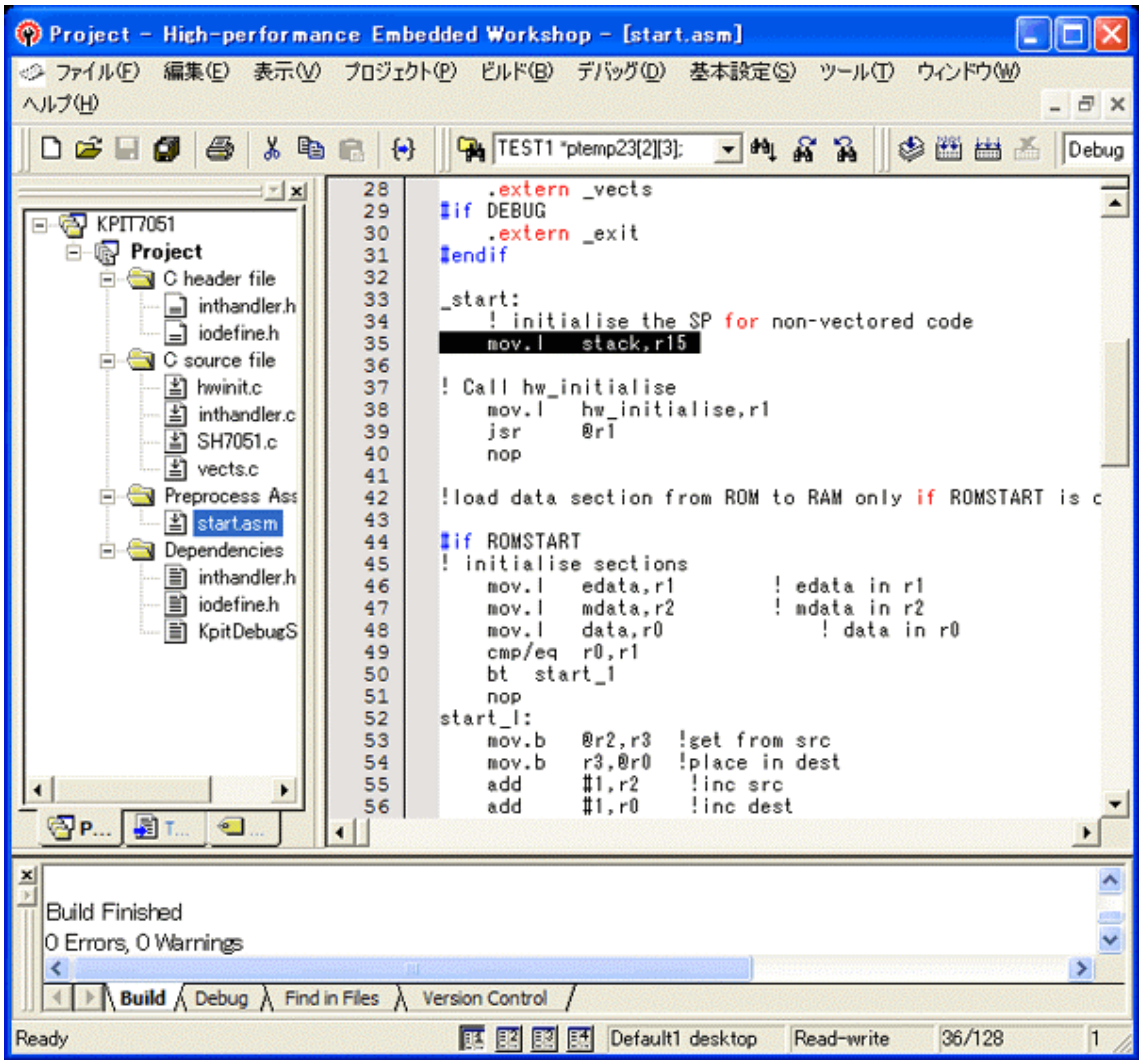
①inthandler.c を選択します。
②void INT_ATU6_CMI6(void){ Timer6();}の関数を記述します。

【注意】
①「vects.c」の２５６ベクターに「INT_Dummy」が登録されています。これは 0x400 番地を超えますし、意味の無い登録ですので削除します。（このサンプルでは、削除済みです。）

８．スタートアップ「start.asm」の説明です。

目的：　スタートアップ「start.asm」に実際は不要なソースコードがありますが、
その補足説明です。

[10-1]



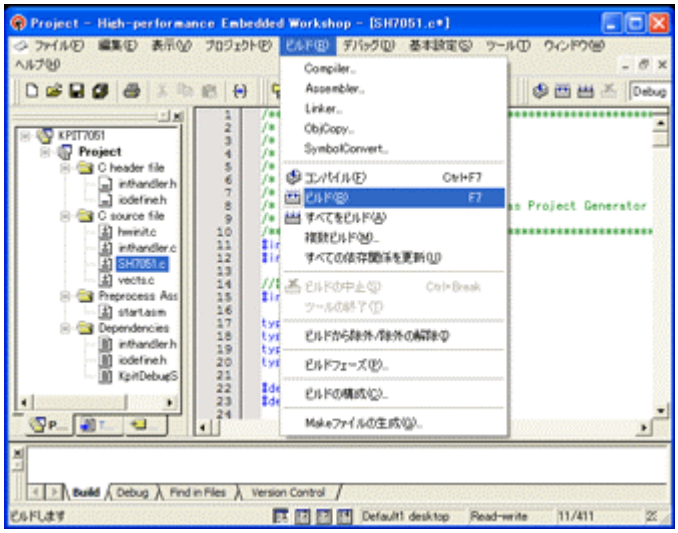
①35行目に「ｍｏｖ．ｌ　ｓｔａｃｋ，ｒ１５」と記述してあります。本来SH-2の場合は、ベクタ１にスタックポインタ値が格納されており、リセット解除時にハード側にて設定されます。
しかし、デバッガ側でリセットコマンドを発行した場合、リセット解除後、NMIを起動し、PC値をベクター０値に書き換えています。つまりリセット解除後、NMI起動までにタイムラグがある為、ターゲットCPUは若干実行しますのでスタックポインタ値が変わってしまいます。
この行にように再設定するのはＳＰ値の復帰になりますので意味のあることになります。
又、デバッグ終了時でも、このソース行を残しておいても問題ありません。

②他の方法としては、今迄弊社が推奨しているソフトタイマー１～２００ｍｓを入れる方法もあります。この方法ですとNMI起動まではソフトタイマ処理中ですのでスタックポインタも内部I/Oレジスタの内容も書き変りません。
どちらにするかは、それぞれの事情によって選択して下さい。

９．ビルドを実行します。

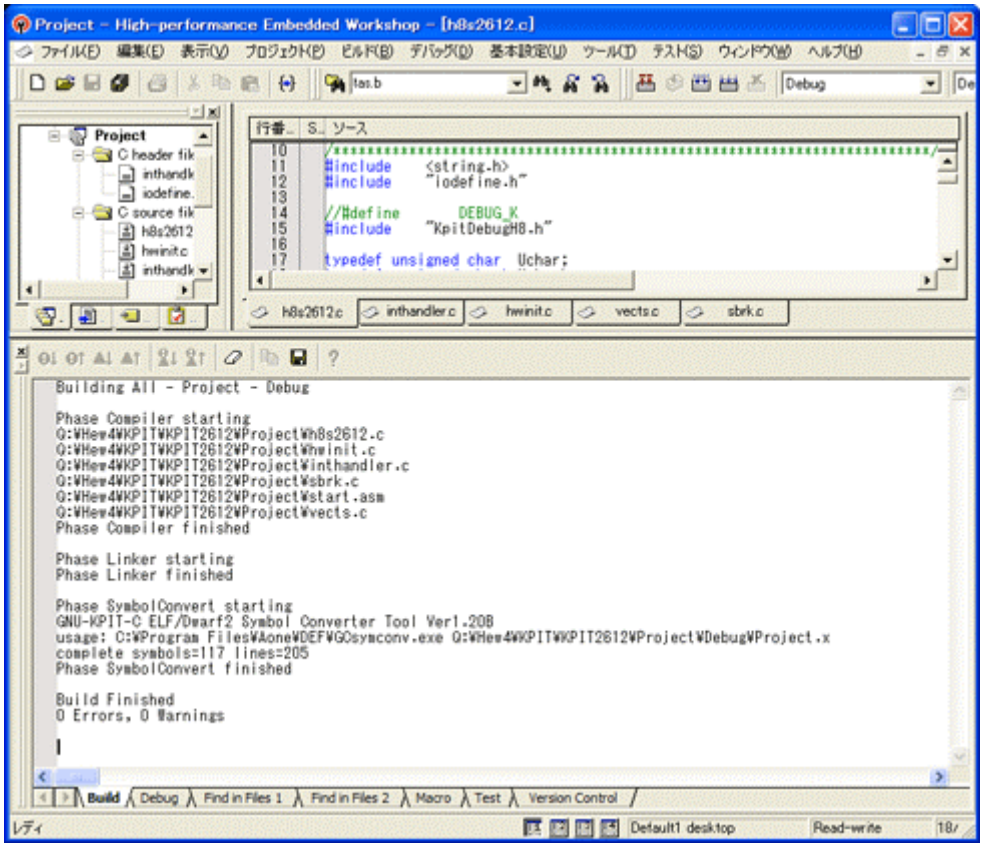
目的：　コンパイル／アセンブリ／リンクロケート／GCsymconv を実行させる為、
ビルドを実行します。

[9-1]



[ビルド]-
[ビルド]をクリックします。

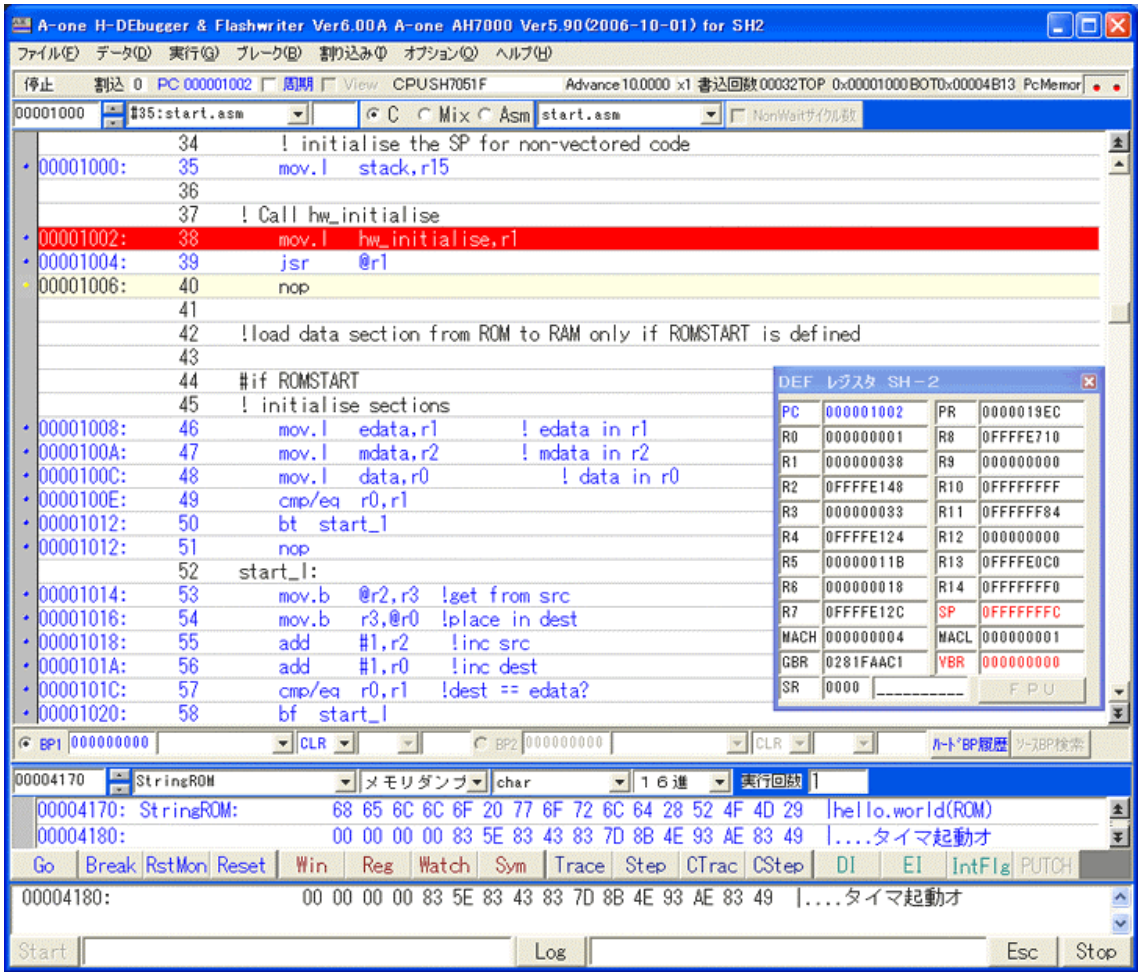
[9-2]



↑のように「0 Errors, 0 Warnings」になれば成功です。

１０．ＤＥＦでの確認

[12-1]



①１０００Ｈ番地にスタックポインタの設定コードがあります。
②ＳＰ値が「0xFFFFFFFC」値になっているのが確認できます。

これで「Ｈ－ｄｅｂｕｇｇｅｒ」用の設定作業が終了です。

以上