

統合環境Hew (Ver 4.04) 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

(H8/3664 E8版)

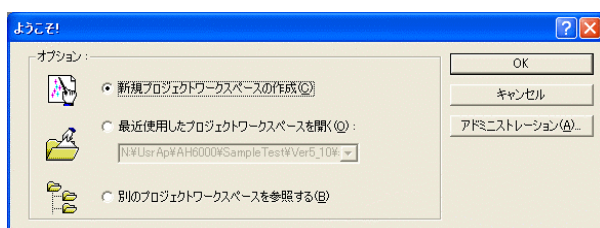
ルネサスC言語用統合環境「Hew Ver 4.04」で H-debugger 用に新ワークスペース／プロジェクトを登録する手順方法を説明します。
説明を明確にするために、名前等を仮に決めて例に沿って説明を進めます。

ワークスペース名	ReneH83664_Hew4		
プロジェクト名	Project		
登録モジュール名	H8_3664.c	Cファイル	メインモジュール (アプリ用)
	HewDebugH8.h	ヘッダファイル	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。)
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		割込みベクターモジュール
	Dbsect.c		定数転送用セクション管理宣言
	iodef.h	ヘッダファイル	I/O 定義ビットフィールド記述用
CPUタイプ	H8/3664F		

1. 新ワークスペースの登録方法

“HEW” 起動させます。

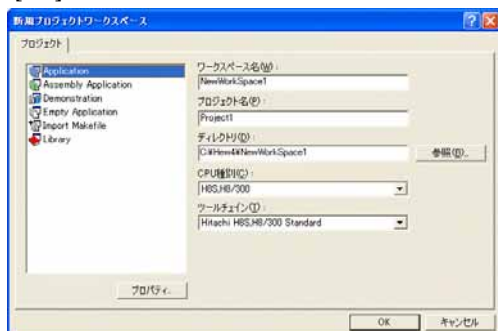
[1-1]



“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。

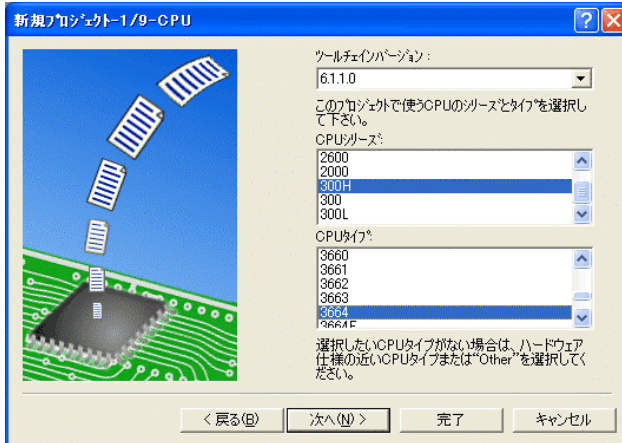
[1-2]



ワークスペース名	“ReneH83664_Hew4”
プロジェクト名	“Project”
デレクトリ	“C:\Hew4\Renesas”
CPU 種別	“H8S, H8/300”
ツールチェーン	“Hitachi H8S, H8/300 Standard”
プロジェクト	Application

この項目を確認後、OKをクリックして下さい。

[1-3]



CPU シリーズを“300H”に選択する。
CPU タイプを“3664F”に選択する。
確認後、

次へ>をクリックします。

[1-4]



CPU スペックを確認後、

次へ>をクリックして下さい。

[1-5]



- ① I/O ライブラリを使用しませんのでチェックを外して下さい。
- ② ヒープメモリを使用しませんのでチェックを外して下さい。
main() 関数生成は“None”に選択する。
- ③ I/O レジスタ定義ファイルは使用しますのでチェックして下さい。
ハードウェアセットアップ関数生成は“None”に選択する。

確認後、次へ>をクリックして下さい。

[1-6]



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。

次へ>をクリックして下さい。

[1-7]



スタックボトムの設定です。
RAM位置を示す為「HFF80」にする。
スタックサイズはデフォルト値にします。

次へ>をクリックして下さい。

[1-8]



ここで明示されたHeap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、

次へ>をクリックして下さい。

[1-9]



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、

次へ>をクリックして下さい。

[1-10]

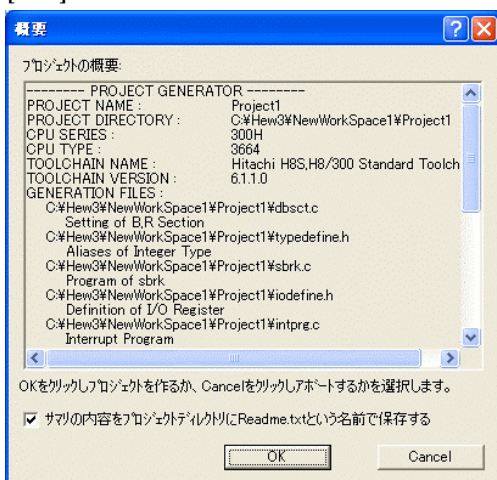


ここで最終になります。

使用するCモジュールを表示します。

この状態で完了をクリックして下さい。

[1-11]



確認画面が表示されますので、

OKをクリックして下さい。

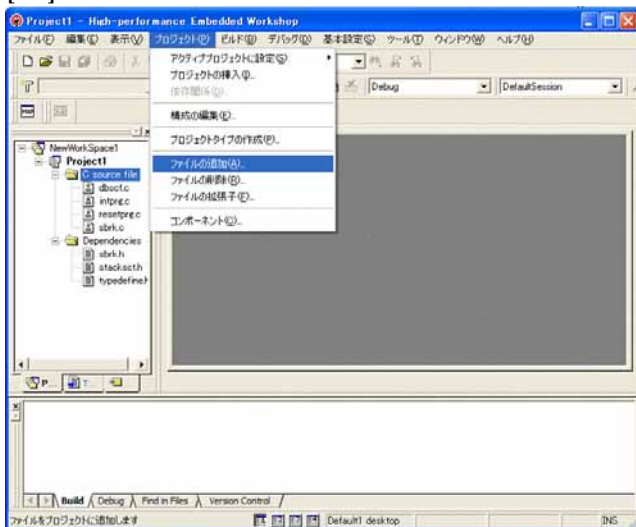
ここまでの操作が新規プロジェクトの登録方法です。

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2ファイルを”C:\Hew4\Renesas\ReneH83664_Hew4\Project”にコピーします。

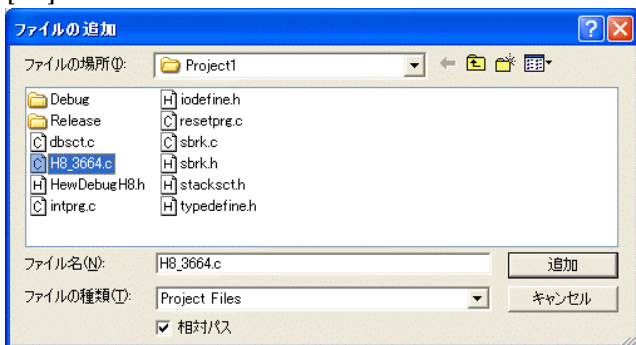
H8_3664.c HP よりダウンロードします。(ルネサスC)
HewDebugH8.h ReneH83664_Hew4.LZH

[2-1]



[プロジェクト]-
[ファイルの追加]をクリックします。

[2-2]



下記1ファイルを指定して下さい。

h8_3664.c

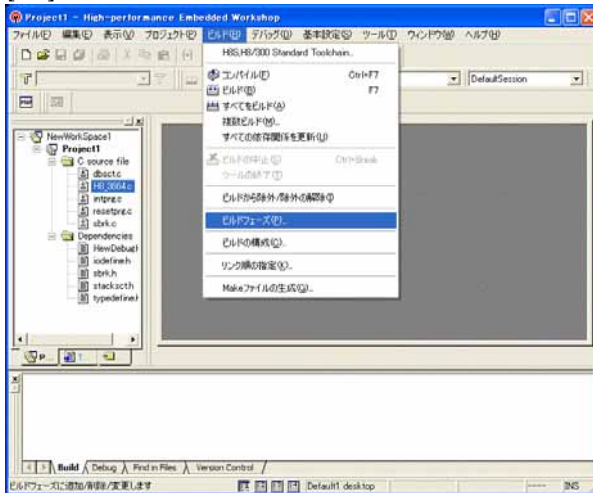
選択後

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。

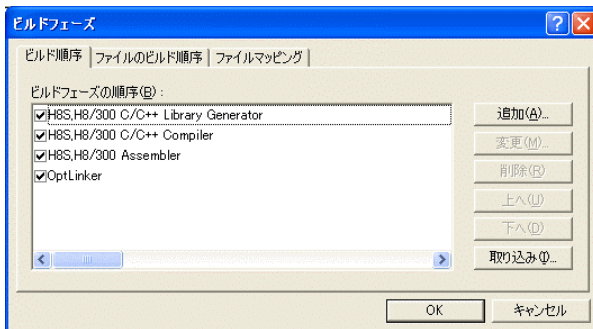
3. シンボルコンバータ「HC symconv」を登録する。

[3-1]



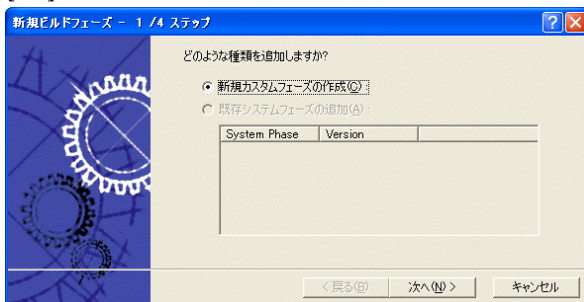
[ビルド]-
[ビルドフェーズ]をクリックします。

[3-2]



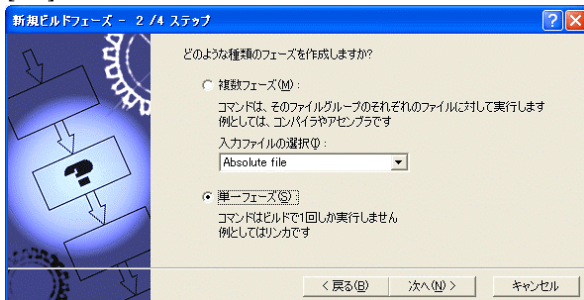
追加をクリックします。

[3-3]



次へ>をクリックします。

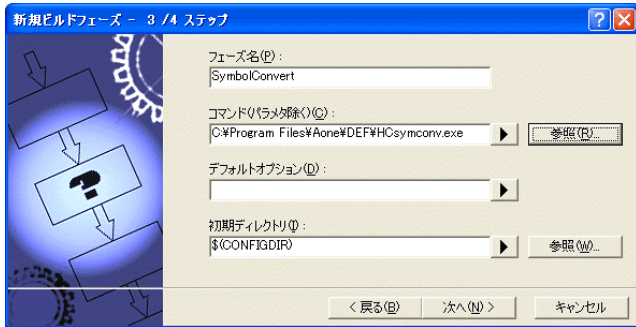
[3-4]



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。

[3-5]



①フェーズ：SymbolConvert

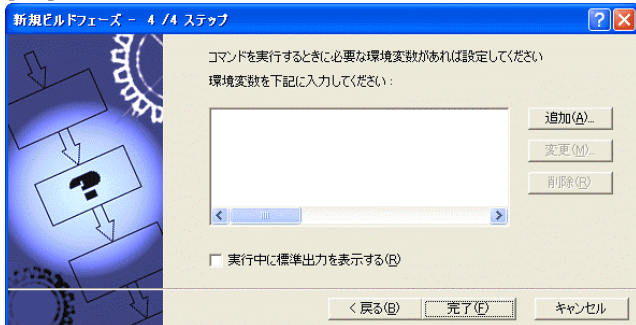
②コマンド：

C:\ProgramFiles\Aone\DEF\HCsymconv.exe

③初期ディレクトリ：\$(CONFIGDIR)

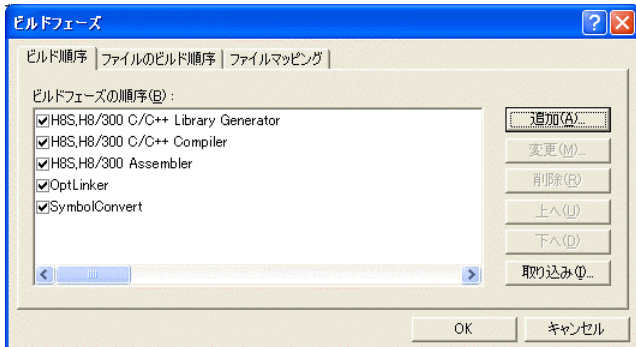
次へ>をクリックします。

[3-6]



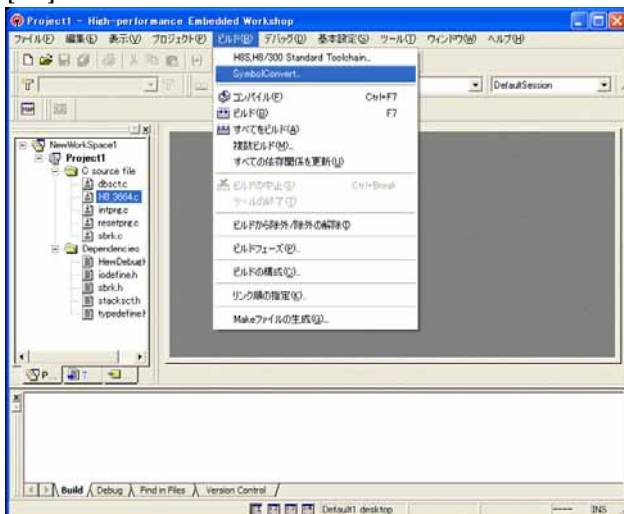
完了をクリックします。

[3-7]



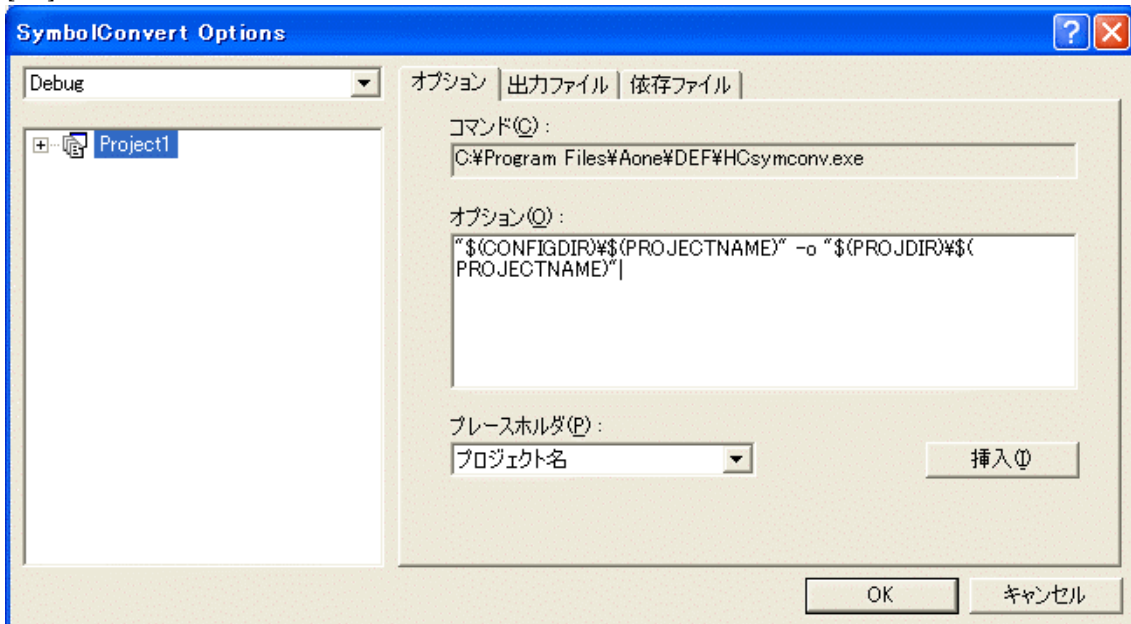
OKをクリックします。

[3-8]



[ビルド]-
[SymbolConvert]をクリック
します。

[3-9]



オプションに下記内容を設定する。

"\$(CONFIGDIR)\\$(PROJECTNAME)" -o "\$(PROJDIR)\\$(PROJECTNAME)"
(入力ファイル名) (出力先名)

注意事項

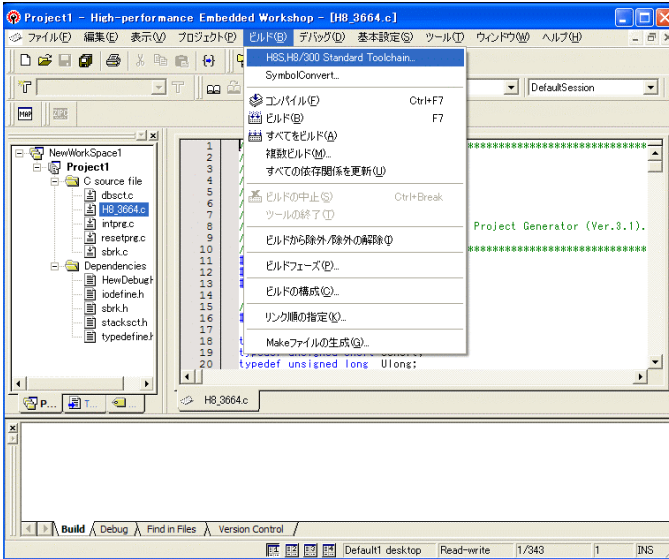
- ① ディレクトリ名に ' ' スペースを使用している場合は、“ダブルクォートで囲んで下さい。
“\$(CONFIGDIR)\\$(PROJECTNAME)" -o "\$(PROJDIR)\\$(PROJECTNAME)"
- ② \$(PROJECTNAME)の先頭に「¥」記号を挿入して下さい。(手入力)
- ③ オプションSW「-o」の両端には、スペースを入れてください。(手入力)
- ④ この設定例は、後説明の「*.mot」ファイルの生成されるディレクトリと同じ場所にシンボルコンバータが生成する「*.sym/*.lin」を置く為の指定です。
<コンフィグレーションDIR>に生成させたい場合は、
“\$(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(Ver 3. 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。(Ver 3. 2xxから)
- 4) [-i] (省略可) 重複モジュール情報を削除する。(Ver 3. 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。(Ver 3. 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。(Ver 3. 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。(Ver 3. 80Bから)

5. ツール(リンカ)の設定

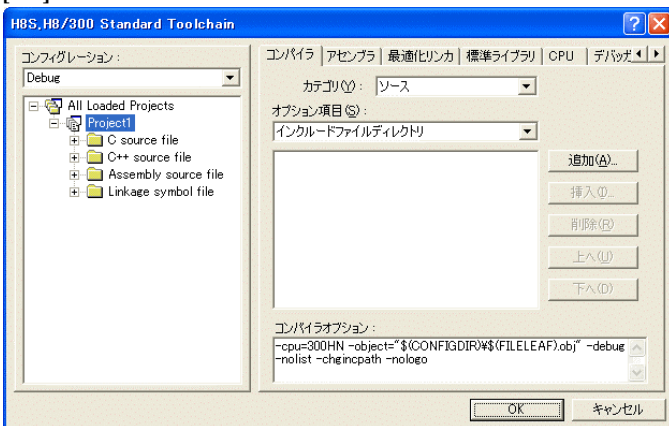
[5-1]



[ビルド] -

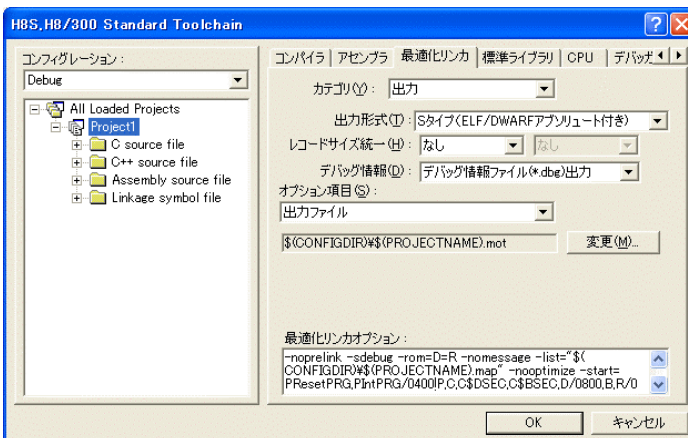
[H8S, H8/300Standard Toolchain]をクリックします。

[5-2]



「最適化リンカ」タグをクリックする。

[5-3]



①カテゴリ:

「出力」を選択する。

②出力形式:

「Sタイプ (ELF/DWARFアブソリュート付き)」を選択する。

③デバッグ情報:

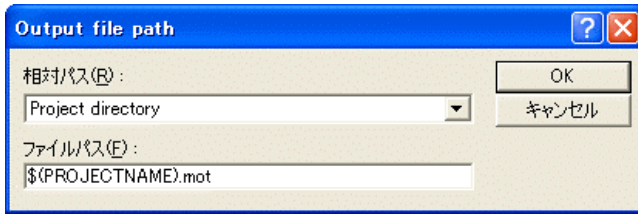
「デバッグ情報ファイル (*.dbg)出力」を選択する。

④オプション項目: 「出力ファイル」の

変更をクリックします。

(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。

[5-4]



相対パス: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。

[5-5]



カテゴリ:の「セクション」を選択する。

下図のようにセクション指定をする。
 (モニタエリア確保のため)

OKをクリックする。

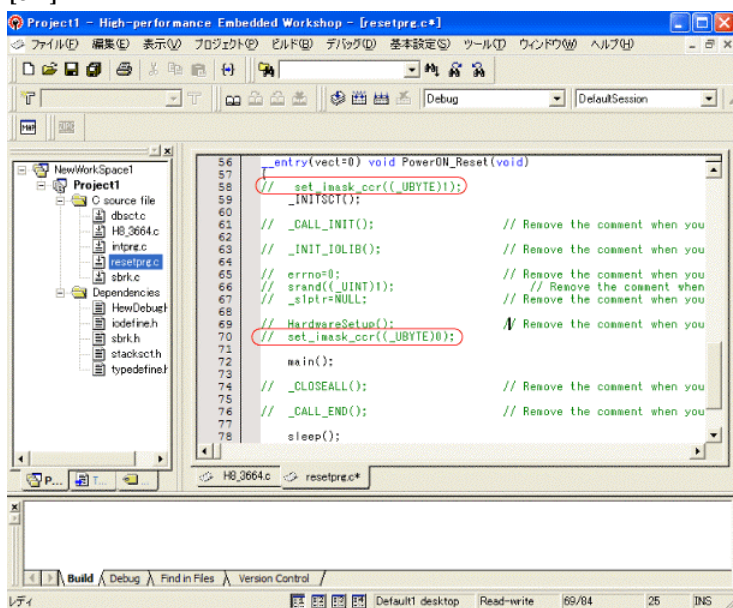
0x0000800	PRresetPRG
	PIntPRG
0x0000C00	P
	C
	C\$DSEC
	C\$BSEC
	D
0x0000F7C0	B
	R
0x0000FE80	S

(重要) 「Bセクション」0xF7C0 (0xF780) は、ソースブレイク使用時の例です。DEFバージョン6.50xから、ソースブレイ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

[6-1]



① `set_imask_ccr((_UBYTE)1)` をコメントにします。

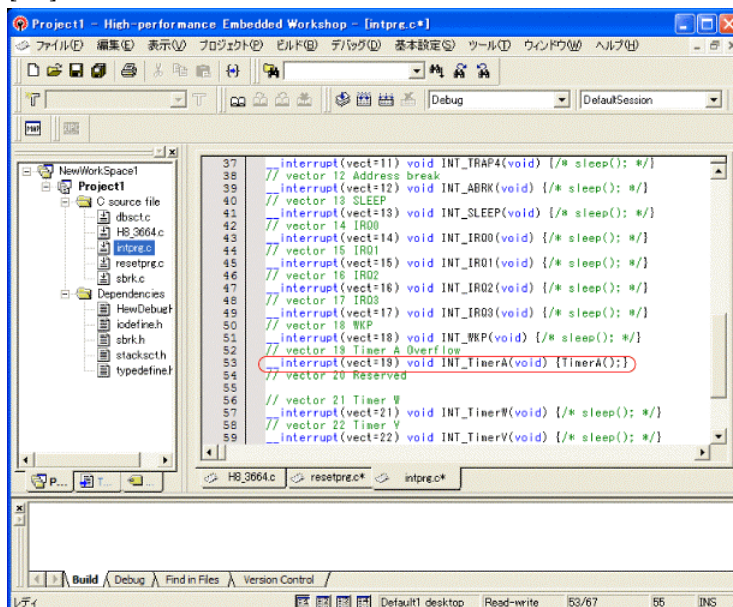
② `set_imask_ccr((_UBYTE)0)` をコメントにします。

(理由)

この例ですと `main()` 関数内で I/O 初期化および割り込み許可をしていますので、ここでは割り込み禁止状態にしておきます。

2) <intprg.c>の変更

[6-2]



今回使用の「H8_3664.c」は、Timer A の Overflow 割り込みを使用した例ですのでベクターを設定をします。

① 「vector 19」に「TimerA();」関数を登録します。

3) コンパイラの「最適化」を外す

[6-3]



①[ビルド] -

[H8S, H8/300Standard Toolchain]をクリックします。

②「コンパイラ」を選択

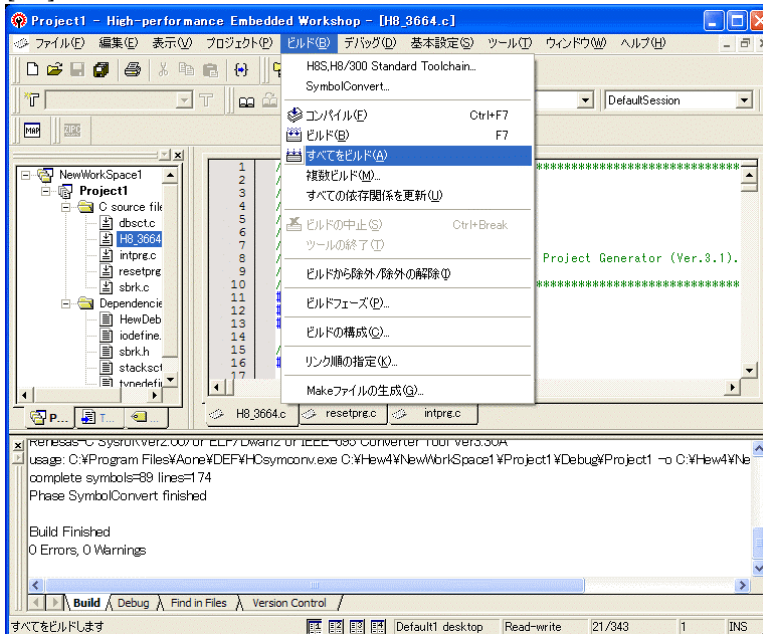
③カテゴリ「最適化」を選択

④「最適化」のチェックを外す。

⑤OKをクリックする。

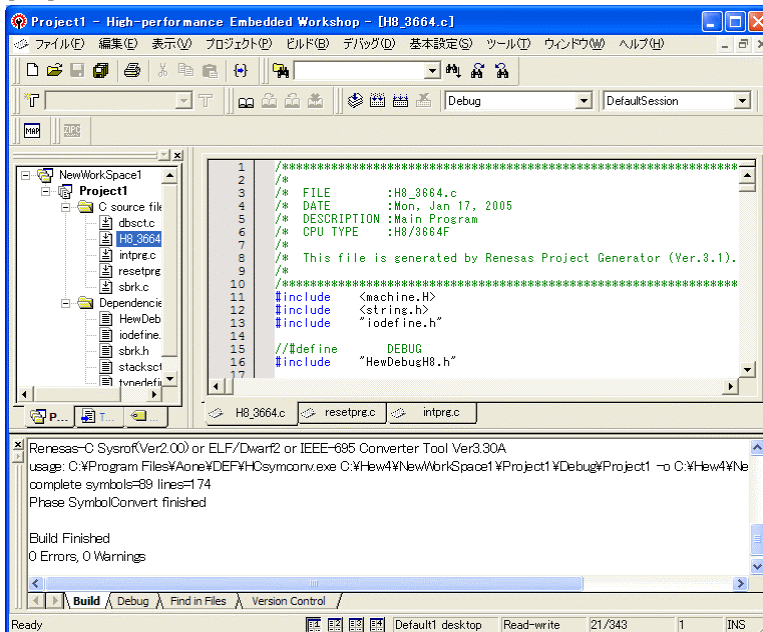
7. ビルドの実行

[7-1]



[ビルド] -
[すべてをビルド]をクリック
します。

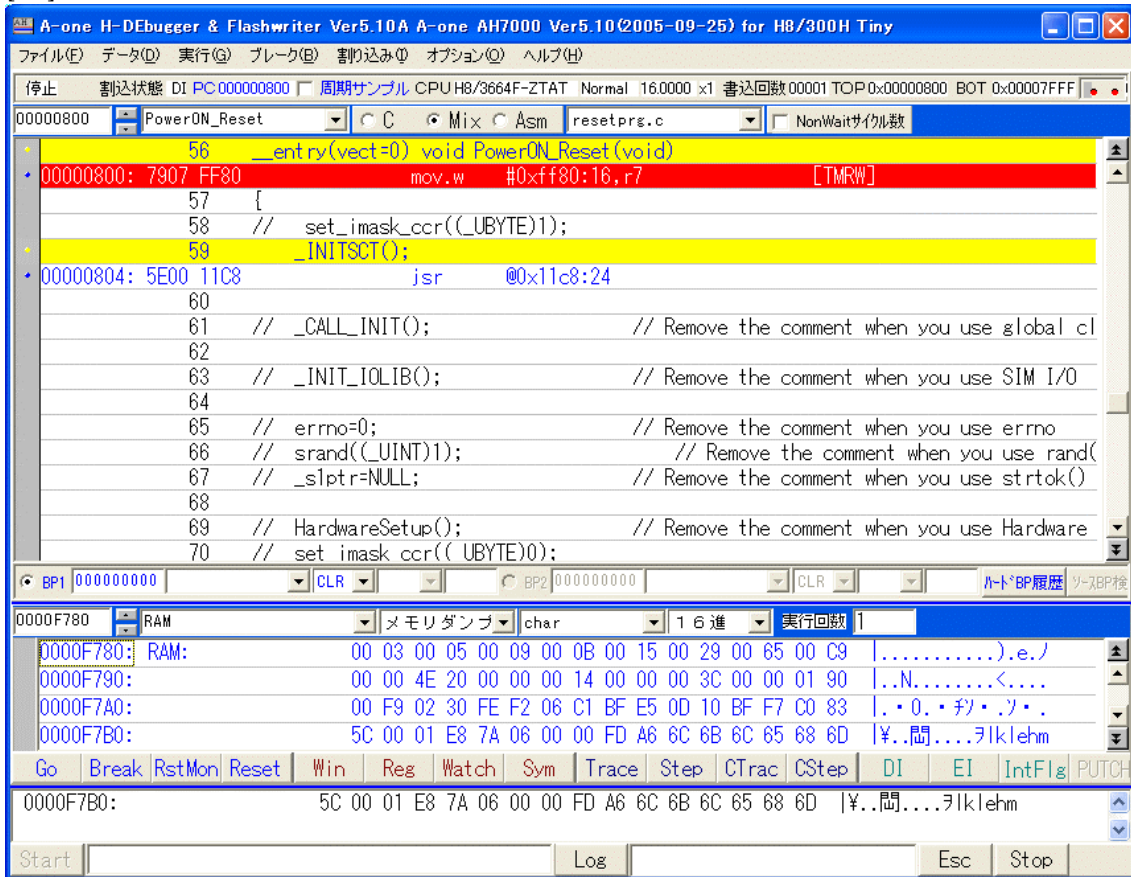
[7-2]



「0Error 0Warnings」
になり作業終了です。

8. DEFでの確認

[8-1]



① 800H番地にスタックポインタの設定プログラムが確認できます。

これで「H-debugger」用の設定作業が終了です。

以上