High-performance Embedded Workshop V.4.09 リリースノート

R20UT2059JJ0100
Rev.1.00
2012.05.16

弊社製品をご使用いただき厚く御礼申し上げます。High-performance Embedded Workshop を使用するにあたり注意事項がございます。ご留意いただけますようお願い申し上げます。

目次

1. 注意事項

1.1 ネットワークドライブ

1.1.1 PC 間の時計のずれ

ソースファイルや出力ファイルの更新日付は、保存した PC の時間で更新されます。
ネットワークで共有しているプロジェクトやソースファイルをビルドするとき、PC 間の時計のずれが原因で再ビルドが正確にできない場合があります。
このような場合、PC 間の時計を合わせるか、または[ビルド -> すべてをビルド]でプロジェクトをビルドしてください。

1.1.2 ネットワーク上プロジェクトのビルド

ネットワーク経由で開いているプロジェクトは、ビルド中にネットワークの状態に依存して、ファイルが開けないなどのエラーが発生することがあります。
例えば、C/C++コンパイラは次に示すエラーメッセージを出力する場合があります。

```
C3019(F) Cannot open source file
```

このような場合、エラーが発生したファイルを再度ビルドしてください。

1.1.3 プロジェクトへのファイルの追加

ネットワーク経由で、リモートドライブ上にあるファイルをプロジェクトに[プロジェクト -> ファイルの追加]などで追加するときに、ネットワークの状態に依存して High-performance Embedded Workshop がアプリケーションエラーで終了することがあります。
このような場合、再度追加を試みてください。アプリケーションエラーが何度も発生する場合は、ファイルをローカルドライブにコピーしてからプロジェクトに追加することをご検討ください。

1.2 エディタのシンタックスカラーリング機能

High-performance Embedded Workshop のエディタには、ソースプログラムファイルのコメント文を色付けするシンタックスカラーリング機能があります。ソースプログラムの書き方によって、シンタックスカラーリング機能が正しく動作しないことがあります。
このような場合、シンタックスカラーリング機能を外してください。シンタックスカラーリング機能を外すには、[基本設定 -> オプション]を選択し、[オプション]ダイアログボックスの[エディタ]タブで[シンタックスカラーリング]のチェックを外し、[OK]ボタンをクリックしてください。

C/C++言語のソースファイルは、以下のような場合、シンタックスカラーリング機能が正しく動作しません。

(i) コメントをネストした場合

例:
```
/* /*    */ */  ← 下線部がコメントの色になる。
```

(ii) 文字列内に"/*"や"*/"を記述した場合

例:
```
/*                ← 下線部がコメントの色になる。
char A[]="*/";  ← 下線部がコメントの色になる。
*/
```

この問題を回避するために、C/C++言語のソースファイルで、* 文字や、/ 文字をコメント以外の目的で隣接して記述することを避けてください。

1.3 ファイル依存関係

High-performance Embedded Workshop は、ソースファイルがインクルードしているファイルを検索し、ワークスペースウィンドウの[Projects]タブに依存ファイルとして表示します。
この依存ファイルは、ファイル依存関係としてビルド可否の判断に利用されます。
ソースプログラムがシンタックス上正しく記述されていても、依存ファイルの検索中にアウトプットウィンドウの[Build]タブにエラーメッセージが出力された場合は、インクルードしているファイルが依存ファイルとして正しく検索されていないことがあります。
このような場合、(a) [ビルド -> すべてをビルド]を選択してすべてのファイルをビルドするか、または、(b) ワークスペースウィンドウの[Projects]タブで当該ソースファイル名を選択し、[ビルド -> コンパイル ファイル名]で当該ソースファイルをコンパイルしてください。

C/C++言語のソースファイルに記述されているインクルードファイルは、以下のような場合、依存ファイルとして正しく検索されません。

(i) 文字列内に"/*"や"*/"がある場合

```
例:   char A[] = "/*";          ← コメント開始とみなされる。
      #include "file.h"        ← このファイルは依存ファイルとして検索されない。
      char B[] = "*/";          ← コメント終了とみなされる。
```

High-performance Embedded Workshop で、#if、#ifdef、#define などのプリプロセッサ文をサポートするには、[ビルド]メニューからツールチェインのオプションダイアログボックスを表示し、[全般]タブで[依存関係検索にてプリプロセッサ文サポート]をチェックします。
表 1.1 にサポートしているプリプロセッサ文を示します。

表 1.1 プリプロセッサ文

プリプロセッサ文	説明
#define	識別子を定義する。プリプロセッサ文が識別子を含んでいると、識別子を定義された文字列で置き換える。 例: #define NICE_FILE “nice.h” #include NICE_FILE
#undef	定義された識別子の置き換えを無効にする。
#include	指定されたファイルが現在のソースファイルに依存していることを示す。
#if~#else~#endif #elif	#if <式>は、式を満たすソースコードの範囲のみ評価する。 式の結果が 0 以外のときは、#if から#else の範囲。 式の結果が 0 のときは、#else から#endif の範囲。
Defined マクロ	defined(<識別子>)文は、識別子が定義されていれば 1 を返す。 識別子が定義されていなければ 0 を返す。 通常、#if defined(MACRO)として記述される。
#ifdef #ifndef	#ifdef <識別子>文は、識別子が定義されていれば#if 1 と等価になる。 識別子が定義されていなければ#if 0 と等価になる。 #ifndef <識別子>文は、識別子が定義されていれば#if 0 と等価になる。 識別子が定義されていなければ#if 1 と等価になる。
#line	無視する。
#error	無視する。
#pragma	無視する。
#	無視する。

プリプロセッサ文に<式>がある場合、依存関係の検索では以下の 10 種類の演算子をサポートしています。これ以外の演算子を使用すると、依存関係の検索は正しく動作しません。

( ) 、 !、 <、 <=、 >、 >=、 ==、 !=、 &&、 ||

[依存関係検索にてプリプロセッサ文サポート]がチェックされていると、C/C++言語のソースファイルに記述されているインクルードファイルは、以下のような場合、依存ファイルとして正しく検索されません。

(ii) 以下に示すコンパイラパッケージに含まれるツールチェインを使用し、プリプロセッサ文の<式>が文字列を比較する条件判断文を含む場合

- SuperH ファミリ用 C/C++コンパイラパッケージ V.5.1 ～ V.9.03 Release 02
- H8SX,H8S,H8 ファミリ用 C/C++コンパイラパッケージ V.3.0A ～ V.7.00 Release 00
- RX ファミリ用 C/C++コンパイラパッケージ V.1.00 Release 00
- M16C シリーズ,R8C ファミリ用 C コンパイラパッケージ V.5.30 Release 0 ～ V.5.45 Release 01
- M32C シリーズ用 C コンパイラパッケージ V.5.40 Release 0 ～ V.5.42 Release 00
- R32C シリーズ用 C コンパイラパッケージ V.1.01 Release 00 ～ V.1.02 Release 01
- M32R ファミリ用 C/C++コンパイラパッケージ V.5.00 Release 00 ～ V.5.01 Release 01
- 740 ファミリ用 C コンパイラパッケージ V.1.00 Release 1 ～ V.1.01 Release 02

例:
```
#define A 'a'
#if (A == 'a')          ← この記述はエラーになり、これ以降の行に記述
                          されているインクルードファイルは依存ファイル
                          として検索されない。
   #include "file.h"    ← このファイルは依存ファイルとして検索されない。
#endif
```

(iii) 以下に示すコンパイラパッケージに含まれるツールチェインを使用し、アセンブリ言語の文で、命令のオペランドにイミディエイト値がある場合

- SuperH ファミリ用 C/C++コンパイラパッケージ V.5.1 ～ V.9.03 Release 02
- H8SX,H8S,H8 ファミリ用 C/C++コンパイラパッケージ V.3.0A ～ V.7.00 Release 00
- M16C シリーズ,R8C ファミリ用 C コンパイラパッケージ V.5.30 Release 0 ～ V.5.45 Release 00
- M32C シリーズ用 C コンパイラパッケージ V.5.40 Release 0 ～ V.5.41 Release 01
- R32C シリーズ用 C コンパイラパッケージ V.1.01 Release 00 ～ V.1.02 Release 00
- M32R ファミリ用 C/C++コンパイラパッケージ V.5.00 Release 00 ～ V.5.01 Release 01
- 740 ファミリ用 C コンパイラパッケージ V.1.00 Release 1 ～ V.1.01 Release 02

例:
```
static void Change_PSW_PW_to_UserMode(void)
{
   MVFC   PSW,R1
   OR     #00100000h,R1  ← この記述はエラーになり、これ以降の行に記述
      :                    されているインクルードファイルは依存ファイル
}                          として検索されない。
```

(iv) 以下に示すコンパイラパッケージに含まれるツールチェインを使用し、アセンブリ言語の文で、命令のオペランドに基数が付いた整数定数がある場合

- SuperH ファミリ用 C/C++コンパイラパッケージ
- H8SX,H8S,H8 ファミリ用 C/C++コンパイラパッケージ

例:
```
void func(void)
{
#pragma asm
   AD1:  .EQU H'7FFF     ← この記述はエラーになり、これ以降の行に記述
      :                    されているインクルードファイルは依存ファイル
#pragma endasm             として検索されない。
      :
}
```

アセンブリ言語のソースファイルに記述されているインクルードファイルは、以下のような場合、依存ファイルとして正しく検索されません。

(v) AS30、AS308、または AS100 対応のソースファイルに、インクルードファイルを指定するための指示命令"..FILE"や"@"を含む場合

例:
```
<sample.a30 ファイル>
.INCLUDE ..FILE@.inc     ← この記述はエラーになり、sample.inc は
                           依存ファイルとして検索されない。
```

1.4 ナビゲーション機能

1.4.1 C 言語関数定義ナビゲーション

High-performance Embedded Workshop は、C/C++言語のソースファイルから C 言語関数定義を検索してワークスペースウィンドウの Navigation タブに表示します。
検索された C 言語関数定義をダブルクリックすると、該当ファイルの該当行がエディタで開きます。
C 言語関数定義の検索は、プリプロセッサ文を無視しています。そのため、C 言語関数定義を正しく解析できない場合があります。
例えば、以下のような場合、2 つの func()関数定義をワークスペースウィンドウの Navigation タブに表示します。

```
#define DEF 1
#ifdef DEF
void func(void)
{
}
#else
int func(int a)
{
}
```

1.4.2 ナビゲーション機能の"C++ Classes"のデフォルト設定

ナビゲーション機能の"C++ Classes"のデフォルト設定は無効（チェックなし）です。
設定状態は、[カテゴリ選択]ダイアログボックスで確認できます（図 1.1 参照）。
[カテゴリ選択]ダイアログボックスは、[Navigation]タブウィンドウ内で右クリックし、[カテゴリの選択]を選択すると開きます。

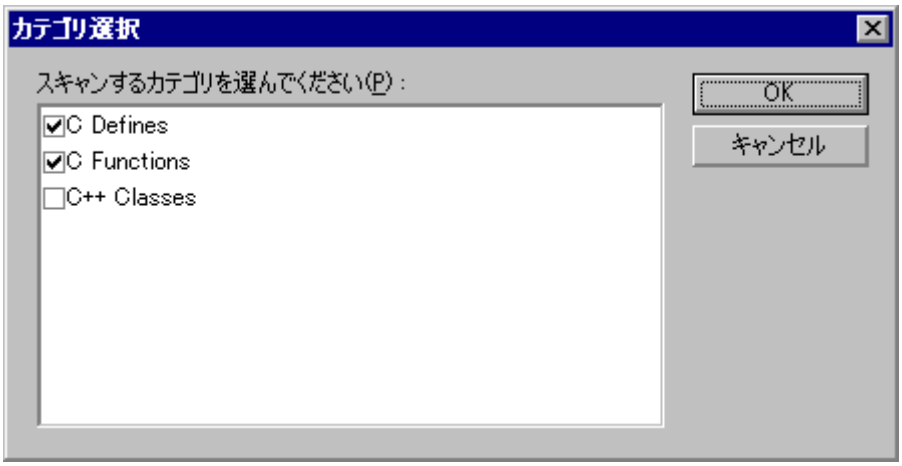

図 1.1 [カテゴリ選択]ダイアログボックス

1.4.3 ナビゲーション機能の"C++ Classes"のウォーニングメッセージ

ナビゲーション機能の"C++ Classes"の設定を有効（チェックあり）に変更すると、以下のウォーニングメッセージを表示します（図 1.2 参照）。
本機能を有効にするとバックグランドでナビゲーション機能が動作するため、PC の CPU 占有率が 100%近くになる場合があり（PC の性能による）、High-performance Embedded Workshop の反応が遅くなることがあります。

図 1.2 ウォーニングメッセージ

1.5 式の指定

(1) 式のシンボルに C++プログラムの関数名は指定できません。

(2) 関数名として多重定義演算子は指定できません。

1.6 パラレルビルド

パラレルビルドを実行したときに、M16C, R8C C/C++コンパイラ V.6.00 でアウトプットウィンドウに出力されるソースコードメッセージ（===> void ...の部分）が、下記のようにエラーメッセージと対で出力されない場合があります。

```
C:¥Workspace¥sample¥initsct.c(31) :  C2570 (E) invalid function declare
C:¥Workspace¥sample¥fvector.c(39) :  C2766 (E) parse error at near '_asm'
===> void initsct(void)
===> _asm(" .id ""¥"#FFFFFFFFFFFFFF¥""");
```

1.7 エラーが発生したファイルを開く

ビルドで C/C++ Compiler、Assembler がアウトプットウィンドウに出力したエラー／ウォーニングメッセージをダブルクリックすると当該ファイルを開き、当該行にカーソルを移動します。
しかし、当該ファイルウィンドウがエディタウィンドウエリアで最小化されている場合、当該エラー／ウォーニング メッセージをダブルクリックしても開きません。
このような場合、当該ファイルを元のサイズに戻すか、または、最大化してください。

1.8 オプションダイアログボックス上でのディレクトリパス指定

High-performance Embedded Workshop の[ビルド]メニューからコンパイラなど各ツールのオプションダイアログボックスを表示できます。
図 1.3 に示すようなオプションダイアログボックスで"Custom directory"を選択した場合、"Directory"フィールドには必ず絶対パスを指定してください。

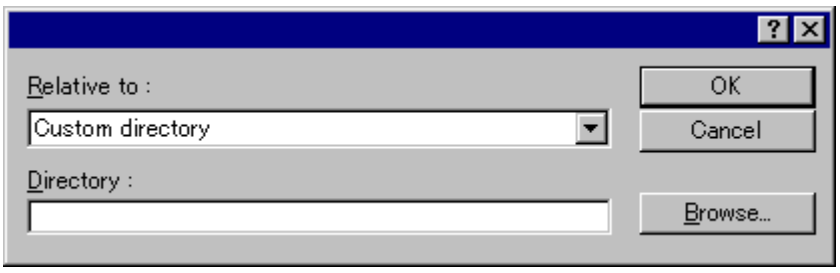

図 1.3 ディレクトリパス指定ダイアログボックス

1.9 プロジェクトに追加するファイル

1 つのプロジェクトに同じ名前（パスとファイル名の拡張子を除く）の複数のファイルを追加できません。
プロジェクトに追加する前に別の名前でファイルを保存してください。

1.10 文字列の置換および検索機能

1.10.1 2 バイト文字コードの置換

メニューの[編集 -> 置換]を選択して開くダイアログボックスで、「置換文字列」または「置換後の文字列」テキストボックスに 2 バイト文字コードを入力し、「正規表現」チェックボックスをチェックして「置換」ボタンをクリックすると、誤った文字列に置換されます。
また、「すべてを置換」ボタンでも正しく置換できません。
この問題を回避するために、2 バイト文字コードを置換する前に「正規表現」チェックボックスのチェックを外してください（正規表現を無効にする）。

1.10.2 2 バイト文字コードの検索

メニューの[編集 -> 検索]を選択して開くダイアログボックスで、「検索文字列」テキストボックスに 2 バイト文字コードを入力し、「正規表現」チェックボックスをチェックして「次を検索」ボタンをクリックすると、誤った検索結果になる場合があります。
見つかった文字列は、ソースウィンドウ内でハイライト表示されます。このときにハイライト表示された文字列とその前後の文字列が誤った文字列に置き換わる場合があります。このままソースファイルを保存すると、これらの文字列が誤ったまま保存されます。
この問題を回避するために、2 バイト文字コードを検索する前に「正規表現」チェックボックスのチェックを外してください（正規表現を無効にする）。

1.11 エディタウィンドウ

(1) 文字セット
エディタウィンドウは Unicode をサポートしていません。

(2) 日本語版のフォント
日本語環境の場合、エディタのフォントに英語フォントを指定しても日本語フォント（MS ゴシック）が使用されます。

1.12 ワークスペースの作成言語

High-performance Embedded Workshop V.4.09 をインストールした時に指定した言語（日本語か英語）と異なる言語の High-performance Embedded Workshop V.3.01 で作成したワークスペースを開くと、ツールバーが表示されません。
その場合、[基本設定 -> カスタマイズ]の[ツールバー]タブでツールバーを選択して表示してください。

1.13 日本語版の表示言語

メニューおよびダイアログボックス内の文字列が一部英語で表示される場合があります。

1.14 ツールアドミニストレーション

1.14.1 コンポーネントおよびツールのアンインストール

High-performance Embedded Workshop V.4.05 以降、[ツールアドミニストレーション]ダイアログボックスの[アンインストール]ボタンを削除しました。
これにより、コンポーネントおよびツールは個別にアンインストールできなくなりました。
アンインストールが必要な場合は、Windows® のコントロールパネルから High-performance Embedded Workshop セット全体をアンインストールしてください。

1.14.2 コンポーネントおよびツールの登録解除

[ツールアドミニストレーション]ダイアログボックスの[登録解除]ボタンを使用して、コンポーネント、またはツールの High-performance Embedded Workshop への登録を解除すると、その後、これらを含む該当製品をインストールしても、登録解除したコンポーネントまたはツールは High-performance Embedded Workshop へ登録されません。
これらを登録するには、[ツールアドミニストレーション]ダイアログボックスの[登録]ボタンを使用してください。
[ツールアドミニストレーション]ダイアログボックスは、ワークスペースを開く前に、[ツール]メニューから[アドミニストレーション]を選択して開いてください。

1.15 旧バージョン形式で保存したワークスペースファイルを開く

High-performance Embedded Workshop V.4.04 以降、ワークスペースファイルを旧バージョン形式で保存することが可能です。ワークスペースファイルを別ファイルとして保存するために、プレフィックスを使用できます。このワークスペースファイルを開くときには、以下の点に注意してください。

- ワークスペースに複数のプロジェクトを登録していた場合、最初に出現するプロジェクトがアクティブプロジェクトになります。
- メインウィンドウで開いていたファイルは自動的に開きません。

1.16 デバッグ機能（共通）

1.16.1 ソースレベル実行

ステップインを実行すると標準 C ライブラリにも移行します。上位関数に戻るにはステップアウトを使用してください。

また、for および while 文では、1 回のステップで次の行に進まないことがあります。進める場合はもう一度ステップしてください。

インライン関数内からステップアウトを実行した場合は、インライン関数の呼び出し元関数ではなく、インライン関数を呼び出した（非インライン）関数の呼び出し元関数まで実行します。

例：`main()`関数から関数（`Func()`）を呼び出し、`Func()`関数からインライン関数（`InFunc()`）を呼び出している場合

```
10  int a = 0;
11  main()
12  {
13    Func();
14  }
15
16  void Func()
17  {
18    InFunc();
19  }
20
21  inline void InFunc()
22  {
23     a++;
24  }
```

23 行目でステップアウトを実行すると、ステップは 14 行目で停止します（Func()関数内でステップアウトを実行した場合と同じ結果となります。19 行目では停止しません）。

1.16.2 ロードモジュール作成後のソースファイル位置移動

ロードモジュール作成後にソースファイルを移動させた場合、作成したロードモジュールのデバッグ中にソースファイルを指定するための[ファイルを開く]ダイアログボックスが表示されることがあります。

対応するソースファイルを選択し[開く]ボタンをクリックしてください。

1.16.3 エディタウィンドウ

エディタウィンドウに表示中のプログラムを変更し、ソースファイルとロードモジュールを再ロードしたときは、一旦エディタウィンドウを閉じて、開き直してください。

そのまま使用すると、エディタウィンドウの表示が不正になる場合があります。

1.16.4 逆アセンブリウィンドウ

逆アセンブリウィンドウのソースモードを表示し、さらにエディタウィンドウの混合モードを表示している場合に、ステップ実行、またはプログラム実行を行なうと、逆アセンブリウィンドウのソースモードで、プログラムカウンタ（PC）のアイコン（➪）が正しい位置に表示されない場合があります。
逆アセンブリウィンドウのソースモードでデバッグする場合は、エディタウィンドウをソースモードに切り替えてご使用ください。

1.16.5 スタックトレースウィンドウ

割込み関数を実行した状態でスタックトレースウィンドウを表示させると、割込み関数以前の表示が不正となります。

1.16.6 メモリウィンドウ

(1) メモリデータの検索
メモリデータの検索を開始する前に、メモリウィンドウの[Data]または[Code]カラム上で、マウスにより検索範囲を選択するか、または任意の位置をクリックしてください。
メモリウィンドウ上で右クリックしただけでは、検索を開始しても見つかったデータは強調表示されません。

(2) 表示フォント
表示フォントをプロポーショナルフォントに設定している場合、表示が欠ける場合があります。
表示フォントを固定幅のフォントに変更してください。

1.16.7 エントリポイント

リンケージエディタの ENTRY オプション等でエントリポイントを指定した場合でも、プログラムのダウンロード時 PC へのエントリポイントアドレスの設定は行ないません。
プログラム実行前に PC の設定を行なってください。

1.16.8 レジスタウィンドウ

High-performance Embedded Workshop V.4.04 以降は、レジスタ値をセッションに保存しないようにしました。

1.17 デバッグ機能（SuperHファミリ、H8SX,H8S,H8 ファミリ、およびRXファミリ用デバッガ）

1.17.1 ラインアセンブル

ラインアセンブル時の入力基数のデフォルトは、基数の設定に関係なく 10 進数です。
16 進数を入力する場合は、H' または 0x を指定してください。

1.17.2 SYSROF 形式ファイルのロード

SYSROF 形式のデバッグ対象プログラムはロードできません。
デバッグ対象プログラムは ELF/DWARF2 形式で作成してください。

1.17.3 Hitachi Debugging Interface セッションファイル

Hitachi Debugging Interface で保存したセッションファイルは使用できません。
High-performance Embedded Workshop のプロジェクトワークスペースとして作成し直してください。
なお、Hitachi Debugging Interface のコマンドファイルは使用可能です。

1.17.4 プロファイル

プロファイルは、オーバーレイ機能には対応していません。

1.17.5 スタックトレースウィンドウ

インライン展開された関数は呼び出し履歴に表示されません。

1.17.6 変数値の参照

- スタックトレースウィンドウ
- ローカルウィンドウ
- ウォッチウィンドウ
- ツールチップウォッチ機能
- インスタントウォッチ機能

(1) インライン関数の局所変数

PC 値がインライン関数内にある場合、関数パラメータおよび関数内で定義した局所変数の値を上記のウィンドウまたは機能を使用して参照できません。

(2) 最適化後の局所変数

生成されたオブジェクトコードによっては、最適化オプションでコンパイルされた C ソースファイルの局所変数の値を上記のウィンドウまたは機能を使用して正しく表示できないことがあります。逆アセンブリウィンドウで生成されたオブジェクトコードを確認してください。

(3) メモリキャッシュ

メモリキャッシュが有効な場合、リトルエンディアンで作成したプログラムのデバッグ中、変数の値を上記のウィンドウまたは機能を使用して正しく表示できないことがあります。
例えば、以下のコードのように 0x2002 番地から 4 バイト変数の値が正しく表示できません。

```
#pragma address data=0x2002
long data;
```

この問題を回避するために、CACHE コマンドを使用してメモリキャッシュを無効にしてください。CACHE コマンドの注意事項については、「1.21.1 CACHE コマンド」を参照してください。

```
cache off
```

1.17.7 char ポインタ型変数のポインタの指し先の値の参照

- ローカルウィンドウ
- ウォッチウィンドウ

上記のウィンドウでは、char ポインタ型変数のポインタが指すアドレスが、(i)または(ii)のいずれかに該当している場合、"Not available now. "メッセージを表示し、ポインタの指し先の値を参照できません（例 2 参照）。

(i) 0xFFFFFF81 番地以上の場合

(ii) メモリマップの各領域の終端アドレスから 128 バイト以内の場合

メモリマップの各領域と、その終端アドレスから 128 バイト以内の領域の例を示します。

```
0x00000000 ～ 0x00007FFF、0x00007F81 ～ 0x00007FFF
0xF0000000 ～ 0xF000FFFF、0xF000FF81 ～ 0xF000FFFF
0xFFFF8000 ～ 0xFFFFFFFF、0xFFFFFF81 ～ 0xFFFFFFFF
```

この問題を回避するために、ウォッチウィンドウでは、ポインタの指し先を直接示す式を入力してください（例 3 参照）。

例 1: アドレスが(i)および(ii)に該当している場合（6 行目および 8 行目）

```
01    char* cp = (char*)0xFFFFFF80;
02    void main(void)
03    {
04        *cp = '0';        //0xFFFFFF80 番地に 0x30 を代入
05        cp++;
06        *cp = '1';        //0xFFFFFF81 番地に 0x31 を代入
07        cp++;
08        *cp = '2';        //0xFFFFFF82 番地に 0x32 を代入
09    }
```

例 2: ウォッチウィンドウ、ローカルウィンドウでの表示内容

```
- "cp" 0xffffff80 { 00001400 } (char*)
  - "*"  H'30 "0" { FFFFFF80 } (char)   //参照可能
- "cp" 0xffffff81 { 00001400 } (char*)
  - "*"  Not available now.             //アドレスが(i)および(ii)に該当
- "cp" 0xffffff82 { 00001400 } (char*)
  - "*"  Not available now.             //アドレスが(i)および(ii)に該当
```

例 3: ウォッチウィンドウで入力する式

```
- "*cp" H'31 "1" { FFFFFF81 } (char)
- "*cp" H'32 "2" { FFFFFF82 } (char)
```

1.17.8 変数値の変更

- ローカルウィンドウ
- ウォッチウィンドウ

(1) 日本語文字列の入力

変数の値を上記のウィンドウを使用して変更する場合、入力するデータに日本語文字列を指定しないでください。
日本語文字列を入力する場合はメモリウィンドウを使用してください。

(2) レジスタに割り付いた変数

上記のウィンドウでは、変数の型サイズを使用して変数の値を変更します。このため、変数の型サイズと変数の割り付いたレジスタの型サイズが異なる場合、不正な値がレジスタに設定されることがあります。

例: 1 バイトの変数が 4 バイトのレジスタに割り付いている場合
1 バイトの変数に負の値を指定すると、レジスタの上位 3 バイトに符号拡張した値は設定されません。

```
int func() {
   int res;
   signed char ch;          // レジスタ"R2"を参照(*1)
   ch = -1;                 // 変数"ch"に「-1」を代入(*2)
   if (ch == -1) {
      res = 0;
   } else {
      res = 1;
   }
   return res;
}

*1. "R2"の初期値は「0x12345678」とします。
*2. - プログラム実行により変数"ch"に「-1」を代入した場合
       "R2": 0x12345678 -> 0xffffffff
    - ウォッチウィンドウから変数"ch"に「-1」を代入した場合
       "R2": 0x12345678 -> 0x123456ff
```

この問題を回避するために、レジスタウィンドウを使用してレジスタの値を直接変更してください。

1.17.9 ウォッチウィンドウ

(1) 変数の展開

プログラムのデバッグ中、ウォッチアイテムがスコープを外れると、展開中のウォッチアイテムは自動的に展開を閉じます。
その場合、最初の子アイテムに設定されていたリアルタイム設定をウォッチアイテム全体の設定として保持します。

(2) 変数値の自動更新機能

変数をキャストして登録した場合、変数値の自動更新機能を使用できません。

1.17.10 オーバーレイの構成ダイアログボックス

オーバーレイセクションに連結させたセクションがその他のオーバーレイセクションにも連結されているかのように表示してしまう場合があります。
以下の(1)、(2)、(3)のようにオーバーレイセクションを配置し（例 1）、ビルドの結果、各セクションが例 2 のように配置された場合、[オーバーレイの構成]ダイアログボックスには以下のように表示されます（例 3）。

例 1:
```
<Section>
(1) "P11, P12"
(2) "P21"
(3) "P31, P32"
```

例 2:
```
<Sec> <Start - End >
"P11" 0x1000 - 0x10FF （*1）
"P12" 0x1100 - 0x12FF
"P21" 0x1000 - 0x10FF （*1）
"P31" 0x1000 - 0x105F
"P32" 0x1060 - 0x11FF
```
*1. 連結セクションを持つ"P11"とそれを持たない"P21"セクションの終了アドレスが同じ。

```
例 3:  <Section>
       (1) "P11, P12"
       (2) "P21, P12" （*2）
       (3) "P31, P32"
   *2. "P12"セクションが"P21"にも連結されているかのように表示。
```

1.17.11 コマンド

(1) WATCH_EDIT コマンド

ウォッチウィンドウで、異なったスコープを指定した同名の変数を参照している場合、WATCH_EDIT コマンドを使用してその変数の値を編集すると、ウォッチウィンドウにあるその名前を持つ最初の変数のみ値を変更します。

1.17.12 SuperH ファミリ、および H8SX,H8S,H8 ファミリ用シミュレータ・デバッガ

(1) メモリリソース設定

メモリリース変更機能の仕様がエミュレータと異なりますので、ご注意ください。
シミュレータ･デバッガの仕様は以下のとおりです。

- 変更前と変更後のメモリ属性（Read、Write、Read/Write）が同一の場合
  メモリリソースサイズの変更と判断してメモリリソースサイズを指定値に変更します。
- 変更前と変更後のメモリ属性が異なる場合
  メモリリソース属性の変更と判断して、指定範囲のメモリ属性を指定値に変更します。

(2) PC ブレークポイント設定数と[条件を指定して実行]メニューのテンポラリ PC ブレークポイント設定数

PC ブレークポイント設定数と[条件を指定して実行]メニューのテンポラリ PC ブレークポイント設定数の合計は、最大 1,024 個です。
したがって、PC ブレークポイントを 1,024 個設定した状態では、[条件を指定して実行]メニューのテンポラリ PC ブレークポイントでの指定は無効となります。
PC ブレークポイントと[条件を指定して実行]メニューのテンポラリ PC ブレークポイントは、設定数の合計が 1,024 個以下で使用してください。

(3) デバッグの設定ダイアログボックス

- 自動的にターゲットを接続しない
  [オプション]タブの[自動的にターゲットを接続しない]チェックボックスの設定にかかわらず、[デバッグの設定]完了時にターゲットを接続します。

(4) SH-4 シミュレータ、SH-4 with BSC シミュレータ

- SH-4 with BSC シミュレータでは、DMA 転送において、転送元、転送先のアドレスの下位 3 ビットが違う場合、転送最後のデータが不正になります。

  ```
  例: SAR0=2000  DAR0=4004  DMATCR0=2  CHCR0=5491
      2000 番地のメモリ内容: 0102030405060708
      DMA 転送終了後の 4004 番地の内容: 0106
  ```
- SH-4 シミュレータ、SH-4 with BSC シミュレータともに、デコード完了したアドレスの命令を変更してもパイプラインリセット実行からの開始になりません。
- SH-4 with BSC シミュレータでは、ブレークデータの設定で指定したデータのサイズと異なるサイズでメモリアクセスを行なった場合、ブレークデータの条件が成立しても停止しない場合があります。その場合、メモリアクセスのサイズとブレークデータの設定で指定するデータのサイズを同一にしてください。
- SH-4 シミュレータ、SH-4 with BSC シミュレータともに、倍精度 FDIV 命令、倍精度 FSQRT 命令のパイプラインが実機と異なります。F3 ステージのパイプが 1 サイクル多く表示されます。

(5) SH-3DSP シミュレータ

- DSP 繰り返し（ループ）制御中の例外コード
  DSP 繰り返し（ループ）制御中に例外が発生した場合、EXPEVT（例外事象レジスタ）に設定する例外コードがプログラミングマニュアルとは異なります。
  DSP 繰り返し（ループ）制御中の例外コード

一般例外事象	プログラミングマニュアル	シミュレータ
TLB ミス例外/TLB 無効例外（読み出し）	H'070	H'040
TLB ミス例外/TLB 無効例外（書き込み）	H'070	H'060
TLB 保護例外（読み出し）	H'0D0	H'0A0
TLB 保護例外（書き込み）	H'0D0	H'0C0
CPU アドレスエラー（読み出し）	H'070	H'0E0
CPU アドレスエラー（書き込み）	H'070	H'100

- X/Y メモリアクセスの競合
  XRAM メモリ（XROM、YROM、YRAM メモリでも同様）に命令コードとデータを配置した場合、命令コードフェッチによる XRAM メモリアクセスと MOVX、MOVY 命令による XRAM メモリアクセスが同一スロットになっても競合によるストールは発生しません。
  このため、サイクル数が異なります。
- 4 の倍数番地以外から実行した場合のパイプライン
  4 の倍数番地以外から実行した場合のフェッチのパイプラインがプログラミングマニュアルとは異なります。
  このため、4 の倍数番地以外から実行した場合のフェッチで例外が発生した場合の動作が異なります。
  例: 4 の倍数番地以外から実行した場合のパイプライン

```
プログラミングマニュアル      シミュレータ
IF IF ID EX                IF ID EX
      IF ID EX                IF ID EX
         if ID EX                if ID EX
```

(6) トレース機能（SH3、SH3E、SH-3DSP のみ）

- FPU、MAC、DSP レジスタのアクセス情報
  FPU、MAC、DSP レジスタに書き込む命令を実行しても、トレース情報にアクセス情報は表示しません。
- パイプライン表示
  プログラミングマニュアルの表記上ライトバックがない命令でも、命令動作上レジスタへの書き込みが発生する場合、シミュレータ・デバッガのトレース表示では、メモリアクセス，ライトバックのステージとして表示します。サイクル数は正しく表示します。
  例: TRAPA 例外のトレース表示

```
シミュレータ・デバッガでのトレース表示                                プログラミング
                                                                   マニュアルの表記
                     IF DE EX MA SW                                IF DE EX MA SW
05  NOP              06 05                                         06 05
06  TRAPA #H'10      07 06 05                                      -- 06
--  - - - - -        -- -- 06                                      -- -- 06
--  - - - - -        -- -- 06 06                                   -- -- 06
--  - - - - -        -- -- 06 06                                   -- -- 06
--  - - - - -        -- -- 06 06 06 (06):SSR<-60000001             -- -- 06
--  - - - - -        08 -- -- 06 06 (06):SPC<-00001006             07 -- --
--  - - - - -        09 08 -- -- --                                08 07 --
```

(7) H8SX シミュレータ
H8SX シミュレータは、H8SX CPU のミドルモードをサポートしていません。

(8) シミュレータターゲットの選択
プロジェクト作成時のデバッガターゲット選択では、CPU、および動作モードと一致するシミュレータを選択してください。

(9) SH2A-FPU シミュレータ・デバッガの内蔵 RAM 領域
SH2A-FPU シミュレータ・デバッガのメモリマップでは 0xFFF80000 番地から 0xFFFBFFFF 番地を内蔵 RAM 領域として表示しますが、0xFFFA0000 番地から 0xFFFBFFFF 番地は予約領域です。
このため、SH2A-FPU シミュレータ・デバッガご使用時は以下の点にご注意ください。
- 内蔵 RAM 領域は 0xFFF80000 番地から 0xFFF9FFFF 番地以内を使用してください。
- スタック領域を内蔵 RAM 領域に割り当てる場合も、0xFFF80000 番地から 0xFFF9FFFF 番地以内となるようにスタックポインタ初期値を設定してください。

(10) 旧バージョン用ワークスペースの保存
High-performance Embedded Workshop V.4.04 以降では、旧バージョン形式でワークスペースを保存すると、シミュレータ・デバッガの動作が非常に遅くなることがあります。
その場合、トレース容量を 32,768 以下にしてください。

1.18 ウィンドウおよびダイアログボックス

1.18.1 スクロール

(1) インテリジェントマウスで上スクロールができない場合があります。
この場合は、ウィンドウ上のスクロールボタンを使用してください。

(2) タッチパッドの操作面にスクロールゾーンがある PC の場合、逆アセンブリウィンドウ上、または混合モードまたは逆アセンブリモードで開いているエディタウィンドウ上でクリックボタンを押したまま、スクロールゾーンでスクロールすると、アプリケーションエラーが発生する場合があります。
上記のウィンドウでは、クリックボタンを押したまま、スクロールゾーンでスクロールしないでください。

1.18.2 ウィンドウ位置の保存

ウィンドウの表示位置は、以下の操作により移動する場合があります。
- セッションのセーブ後にセッションのリフレッシュ
- ターゲットの切断後の再接続
- バーチャルデスクトップの切り替え

1.19 ヘルプ

英語版 Windows® でヘルプ使用時に以下のメッセージボックスが表示される場合があります。
[Download]ボタンをクリックして"Japanese Text Display Support"をダウンロードするか、[Never download any of these components]チェックボックスをチェックしてから[Cancel]ボタンをクリックしてください。"Japanese Text Display Support"をダウンロードしなくても、ヘルプの表示には影響ありません。

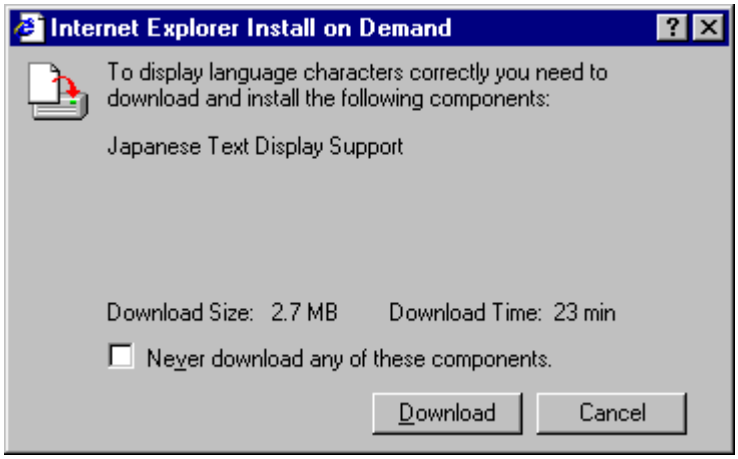

図 1.4 Internet Explorer Install on Demand **メッセージボックス**

1.20 マクロ生成支援機能

マクロ生成支援機能を使用し、[デバッグ -> 実行]、[デバッグ -> プログラムの停止]の選択を記録したマクロを実行すると、[デバッグ -> 実行]は実行されますが、[デバッグ -> プログラムの停止]は実行されません。

1.21 コマンド

1.21.1 CACHE コマンド

エミュレータ・デバッガのマニュアルに CACHE コマンドの記載がない場合、CACHE コマンドを指定しないでください。
キャッシュのメモリアクセス単位は、0x3FF など固定値のため、指定したアクセス範囲より大きい範囲のキャッシュメモリをアクセスする場合があります。

1.21.2 MEMORY_EDIT コマンド

文字を一重引用符（'）で囲っても ASCII 文字列をデータとして入力できません。
文字列をデータとして入力する場合は対応する数値を入力してください。

1.21.3 MEMORY_FIND コマンド

DOUBLE 指定（倍精度浮動小数点数（8 バイト））は<mode>パラメータで使用できません。

1.21.4 コマンドバッチファイル実行タイミング（デバッグの設定ダイアログボックス）

[オプション]タブの[コマンドバッチファイル実行タイミング]でバッチファイルを設定した場合、以下の制限があります。以下のコマンドの代わりにメニュー、ツールバー等、GUI 経由での実行をご検討ください。

(1) [オプション]タブの[コマンドバッチファイル実行タイミング]で[At target connection]を選択したバッチファイル中に"INITIALIZE"コマンドを指定した場合、[コマンドライン]ウィンドウで"INITIALIZE"コマンドを実行しないでください。初期化する場合は、[デバッグ -> 初期化]を選択してください。

(2) [オプション]タブの[コマンドバッチファイル実行タイミング]で[At target connection]を選択したバッチファイル中に以下のコマンドは指定しないでください。
(a) [OPEN_WORKSPACE]コマンド
(b) [CHANGE_PROJECT]コマンド
(c) [CHANGE_CONFIGURATION]コマンド

(3) [オプション]タブの[コマンドバッチファイル実行タイミング]で[Before download of modules]または[After download of modules]を選択した場合、[コマンドライン]ウィンドウで"FILE_LOAD"または"FILE_LOAD_ALL"コマンドを実行しないでください。

(4) [オプション]タブの[コマンドバッチファイル実行タイミング]で[Before download of modules]または[After download of modules]を選択したバッチファイル中に以下のコマンドは指定しないでください。
(a) [OPEN_WORKSPACE]コマンド
(b) [CHANGE_PROJECT]コマンド
(c) [CHANGE_CONFIGURATION]コマンド
(d) [GO]コマンド
(e) [GO_RESET]コマンド
(f) [GO_TILL]コマンド
(g) [STEP]コマンド
(h) [STEP_OUT]コマンド
(i) [STEP_OVER]コマンド

(5) [オプション]タブの[コマンドバッチファイル実行タイミング]で[After reset]を選択した場合、[コマンドライン]ウィンドウで"RESET"コマンドを実行しないでください。

(6) [オプション]タブの[コマンドバッチファイル実行タイミング]で[After reset]を選択したバッチファイル中に以下のコマンドは指定しないでください。
(a) [OPEN_WORKSPACE]コマンド
(b) [CHANGE_PROJECT]コマンド
(c) [CHANGE_CONFIGURATION]コマンド
(d) [RESET]コマンド

1.21.5 コマンドバッチファイル実行順序（デバッグの設定ダイアログボックス）

[オプション]タブの[コマンドバッチファイル実行順序]では相対パス形式でバッチファイルを指定すると、正しくファイルにアクセスできない場合があります。
プレースホルダを適用できないバッチファイルは絶対パス形式で指定してください。

1.21.6 コマンドラインのファイル指定

コマンドラインでファイルを指定する場合、プレースホルダを使用してください（TCL コマンドは除く）。
プレースホルダに含まれていないディレクトリを指定する場合は、絶対パスを指定してください。絶対パスで指定すると、他のマシンやパスの内容が異なる環境に移動する場合、正しくファルを参照できなくなるため、ファイル指定をやり直してください。

```
FILE_LOAD ELF/DWARF2 $(CONFIGDIR)¥¥demo.abs
```

1.21.7 コマンドのパラメータ区切り

TCL が有効の場合、TCL コマンドを使用できます。パラメータが Tab で区切られた TCL コマンドバッチの使用も可能です。

```
for   {set  i  0}    {$i   <  2} {incr i} {
   puts  [memory_display   300   10]
}
```

TCL が無効の場合、TCL コマンドは使用できません。パラメータの区切りとして半角スペースのみ使用できます。そのため、Tab を使用すると以下のようなエラーになります。

```
>TCL
TCL Disabled
>memory_display   300   10
Error: Invalid command
```

1.21.8 コマンド短縮形

- REMOVE_FILE コマンド
  High-performance Embedded Workshop V.3.01 から短縮形"RF"を"REM"に変更しました。

1.22 TCLコマンドとHigh-performance Embedded Workshopコマンドの親和性の拡張

TCL コマンドと High-performance Embedded Workshop コマンドの親和性を拡張（*）したことにより、以下のような記述の場合、High-performance Embedded Workshop コマンド"memory_display 300 10"の実行結果は出力されません。

```
for {set i 0} {$i < 2} {incr i} {
   memory_display 300 10
}
```

High-performance Embedded Workshop コマンドの実行結果を出力する場合は、TCL コマンド"puts"のパラメータとして High-performance Embedded Workshop コマンドを"[]"内に置いてください。

```
for {set i 0} {$i < 2} {incr i} {
   puts [memory_display 300 10]
}
```

*. TCL コマンドと High-performance Embedded Workshop コマンドの親和性を拡張したことにより、TCL コマンド"set"のパラメータとして、High-performance Embedded Workshop コマンドの実行結果を変数へ代入できるようになりました。

以下の例では、High-performance Embedded Workshop コマンド"memory_display 300 10"の実行結果が変数"md_300_10"に代入されます。
TCL コマンド"set"で変数"md_300_10"を指定すると、実行結果を参照できます。

```
set md_300_10 [memory_display 300 10]
```

1.23 TCL/TKコマンド

(1) 対話モードの中止方法

[TCL ツールキット]で対話モードからコマンド入力モードに戻るには、"/."を入力してください。
現在のモードが対話モードかコマンド入力モードかは、"/."を入力して判断してください。

(2) 日本語ディレクトリ

High-performance Embedded Workshop を日本語ディレクトリにインストールした場合、
High-performance Embedded Workshop を実行すると、TCL/TK のライブラリを仮想ドライブ（"u:"ドライブ）に関連付けます。

すでに"u:"ドライブを使用している場合、High-performance Embedded Workshop の開始時に"u:"ドライブとの接続を解除し、High-performance Embedded Workshop の TCL/TK ライブラリと接続します。
High-performance Embedded Workshop の終了時に、TCL/TK のライブラリは接続を解除します。
元のパスとは、自動で再接続しません。
仮想ドライブを使いたくない場合、High-performance Embedded Workshop を英語ディレクトリにインストールしてください。

(3) ログファイルの内容の消去方法

[TCL ツールキット]を使用している場合、[Console]画面の内容がログファイルに出力されます。
ログファイルは、以下のディレクトリにテキスト形式ファイルで生成されます。

例 1: Windows® XP Operating System
C:¥Documents and Settings¥<ユーザ名>¥Local Settings¥Temp¥log.txt
例 2: Windows Vista® または Windows® 7 Operating System
C:¥Users¥<ユーザ名>AppData¥Local¥Temp¥log.txt

[TCL ツールキット]を終了すると、ログファイルの内容は自動的に消去されます。
[TCL ツールキット]の使用中にログファイルの内容を消去したい場合、以下のコマンドを実行してください。

```
set dir $env(TEMP)
set dataFile [open $dir/log.txt {RDWR TRUNC}]
close $dataFile
```

1.24 TCLツールキットとコマンドラインのコマンド

(1) "trace"コマンド

(a) [TCL ツールキット]で TCL の"trace"コマンドを実行する場合、コマンド名をすべて小文字で指定してください。

(b) [TCL ツールキット]で High-performance Embedded Workshop の"TRACE"コマンドを実行する場合、コマンド名をすべて大文字で指定してください。

(c) [コマンドライン]ウィンドウで TCL の"trace"コマンドを実行する場合、コマンド名を"tcl_trace"に置き換えて指定してください。

(2) "clock"コマンド

(a) [TCL ツールキット]で TCL の"clock"コマンドを実行する場合、コマンド名をすべて小文字で指定してください。

(b) [TCL ツールキット]でエミュレータの"CLOCK"コマンドを実行する場合、コマンド名をすべて大文字で指定してください。

(c) [コマンドライン]ウィンドウで TCL の"clock"コマンドを実行する場合、コマンド名を"tcl_clock"に置き換えて指定してください。

(3) "event"コマンド

(a) [TCL ツールキット]で TK の"event"コマンドを実行する場合、コマンド名をすべて小文字で指定してください。

(b) [TCL ツールキット]でエミュレータの"EVENT"コマンドを実行する場合、コマンド名をすべて大文字で指定してください。

"CLOCK"コマンドと"EVENT"コマンドは、一部のエミュレータではサポートしていません。

1.25 High-performance Embedded Workshopユーザーズマニュアルの訂正

ドキュメント番号: R20UT0372JJ0100

P.44「2.3.2 ファイルおよびフォルダをドラッグアンドドロップする」内の「ファイルをユーザフォルダ上にドロップすると」の記述を以下のとおり訂正します。

High-performance Embedded Workshop オンラインヘルプの該当箇所も同様に訂正します。

誤:

ファイルをユーザフォルダ上にドロップすると、ファイルはそのフォルダ下に直接追加されます。
同じ名前のファイルでも異なるパスのファイルは追加されます。

正:

ファイルをユーザフォルダ上にドロップすると、ファイルはそのフォルダ下に直接追加されます。

2. ツールチェイン仕様補足

2.1 ファイル拡張子

High-performance Embedded Workshop がツールチェインの各ツールをビルド実行したとき、コンフィグレーションディレクトリにサブコマンドファイルを残します。
C/C++ Compiler と Assembler のサブコマンドファイルの拡張子は、入力ファイル名の拡張子を表 2.1 のように変更したファイル名になります。
C/C++ Library Generator、OptLinker のサブコマンドファイルは、プロジェクト名に表 2.1 の拡張子を付加したファイル名になります。
サブコマンドファイルは隠しファイルの属性を持ちます。隠しファイルを表示する場合、ディレクトリを表示するウィンドウのプロパティをすべてのファイルを表示するように設定する必要があります。

表 2.1 ツールチェインサブコマンドファイル拡張子

拡張子	ファイルグループ
shg	SuperH RISC engine C/C++ Library Generator
shc	SuperH RISC engine C/C++ Compiler
sha	SuperH RISC engine Assembler
h8g	H8S, H8/300 C/C++ Library Generator
h8c	H8S, H8/300 C/C++ Compiler
h8a	H8S, H8/300 Assembler
hlk	OptLinker
m16cl	M16C Linker
m16ci	M16C Librarian
m16cc	M16C C/C++ Compiler
m16ca	M16C Assembler
m16ct	M16C mkmrtbl (M3T-MR30/4)
r8ct	R8C mr8ctbl (MR8C/4)
m32cl	M32C Linker
m32ci	M32C Librarian
m32cc	M32C C Compiler
m32ct	M32C mr308tbl (M3T-MR308/4)
m32rl	M32R C Compiler
m32ri	M32R Librarian
m32rm	M32R Load module converter
m32rc	M32R C Compiler
m32ra	M32R Assembler
74lk	740 Linker
74lb	740 Librarian
100l	R32C/100 Linker
100i	R32C/100 Librarian
100c	R32C/100 C Compiler
100t	R32C/100 mr100tbl (M3T-MR100/4)
rxg	RX C/C++ Library Generator
rxc	RX C/C++ Compiler
rxa	RX Assembler
r600t	RX mkritbl (RI600/4)

3. ツールチェインアップグレード

以下に示すツールチェインをプロジェクトで使用している場合、ツールチェインの新しいバージョンをインストールすることで、使用しているツールチェインをアップグレード（変更）できます。

- SuperH ファミリ用 C/C++コンパイラパッケージ V.5.1 およびそれ以降
- H8SX,H8S,H8 ファミリ用 C/C++コンパイラパッケージ V.3.0A およびそれ以降
- RX ファミリ用 C/C++コンパイラパッケージ V.1.00 Release 00 およびそれ以降
- M16C シリーズ,R8C ファミリ用 C コンパイラパッケージ V.5.20 Release 1 およびそれ以降
- M32C シリーズ用 C コンパイラパッケージ V.5.20 Release 1 およびそれ以降
- R32C シリーズ用 C コンパイラパッケージ V.1.01 Release 00 およびそれ以降
- M32R ファミリ用 C/C++コンパイラパッケージ V.4.20 Release 1 およびそれ以降
- 740 ファミリ用 C コンパイラパッケージ V.1.01 Release 01 およびそれ以降
- 740 ファミリ用アセンブラパッケージ V.4.10 Release 02 およびそれ以降

使用しているツールチェインをアップグレードするには

[ツール -> ツールチェインバージョンを変更]で[ツールチェインのバージョンの変更]ダイアログボックスを開き、[ツールチェインバージョン]でバージョン番号を選択してください。

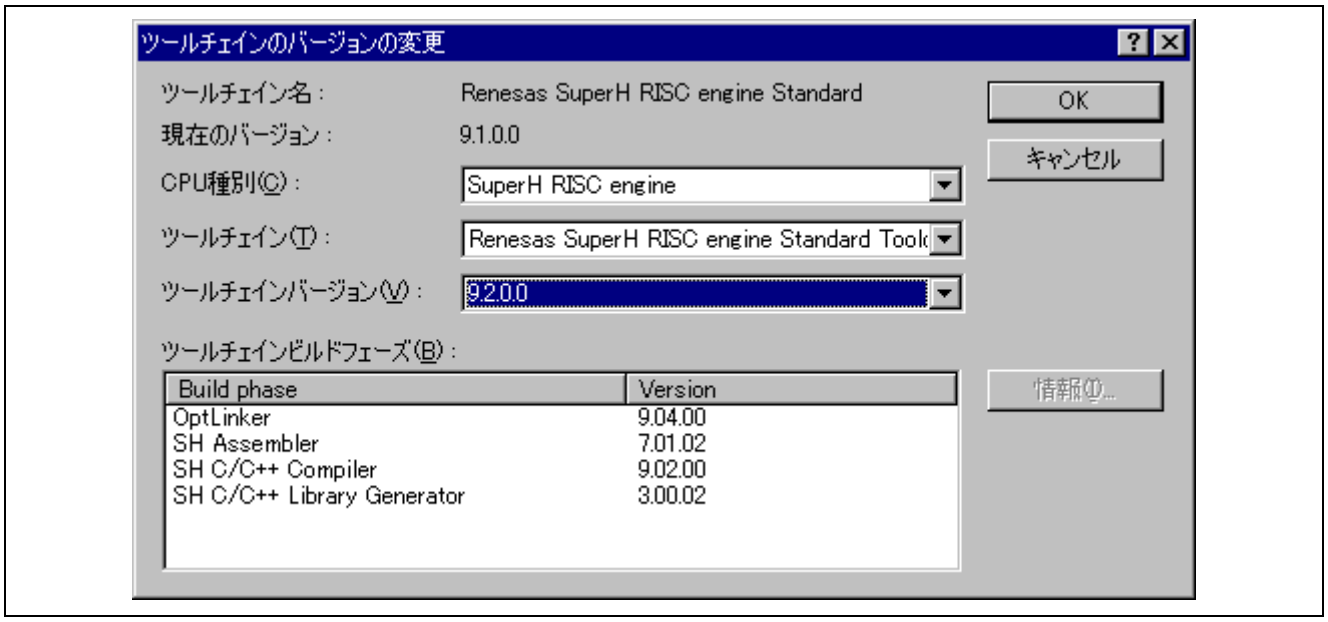

図 3.1 [ツールチェインのバージョンの変更]ダイアログボックス

(1) SuperH ファミリ用 C/C++コンパイラパッケージ V.5.1x、または H8SX,H8S,H8 ファミリ用 C/C++コンパイラパッケージ V.3.0x からのアップグレードの場合

(a) 標準ライブラリ構築ツールの設定

標準ライブラリ構築ツールが追加されました。

High-performance Embedded Workshop はアップグレード時、IM Optlnker のオプション情報をもとに、標準ライブラリ入力指定があればライブラリを生成するオプション（モード：標準ライブラリファイル作成（常に作成））を設定します。

標準ライブラリ入力指定がなければ、ライブラリを生成しないオプション（モード：標準ライブラリファイル指定なし）を設定します。

(b) 最適化リンカの設定

IM Optlinker、Librarian、Stype Converter を統合し、OptLinker になりました。

アップグレード前の各ツールの登録状態により、アップグレード後の OptLinker のオプション設定が異なります。表 3.1 にアップグレード時のオプション継承を示します。

表 3.1 アップグレード時のオプション継承

No	アップグレード前			アップグレード後			
	IM OptLinker の状態	Librarian の状態	Stype Converter の状態	OptInker の状態	IM OptLinker オプション情報	Librarian オプション情報	Stype Converter オプション情報
1	—	—	—	—	×	×	×
2	—	—	□	□	×	×	○
3	—	—	■	■	×	×	○
4	—	□	—	□	×	○	×
5	—	□	□	□	×	○	×
6	—	□	■	■	×	×	○
7	—	■	—	■	×	○	×
8	—	■	□	■	×	○	×
9	—	■	■	■	×	○	×
10	□	—	—	□	○	×	×
11	□	—	□	□	○	×	○
12	□	—	■	■	×	×	○
13	□	□	—	□	○	×	×
14	□	□	□	□	○	×	○
15	□	□	■	■	×	×	○
16	□	■	—	■	×	○	×
17	□	■	□	■	×	○	×
18	□	■	■	■	×	○	×
19	■	—	—	■	○	×	×
20	■	—	□	■	○	×	×
21	■	—	■	■	○	×	○
22	■	□	—	■	○	×	×
23	■	□	□	■	○	×	×
24	■	□	■	■	○	×	○
25	■	■	—	■	○	×	×
26	■	■	□	■	○	×	×
27	■	■	■	■	○	×	○

—：未登録、□：登録チェック無し、■：登録チェック有り、○：継承、×：継承しない

(2) SuperH ファミリ用 C/C++コンパイラパッケージのアップグレードの場合

(a) MAP 最適化用ビルドの自動化

High-performance Embedded Workshop V.2.00（SH Ver.7.0B / Ver.7.0.01 / Ver.7.0.02）では、C コンパイラで最適化リンケージエディタが出力した外部シンボル割り付け情報を活用した最適化を実行するために、再ビルド用のカスタムフェーズを提供していました。

High-performance Embedded Workshop V.2.01 以降では、MAP 最適化実行時に自動的に再ビルド実行が可能になりました。

再ビルド用のカスタムフェーズを無効にするには

再ビルド用のカスタムフェーズは不要になるため、[ビルド -> ビルドフェーズ]で[ビルドフェーズ]ダイアログボックスを開き、"Map optimize"フェーズのチェックを外してください。

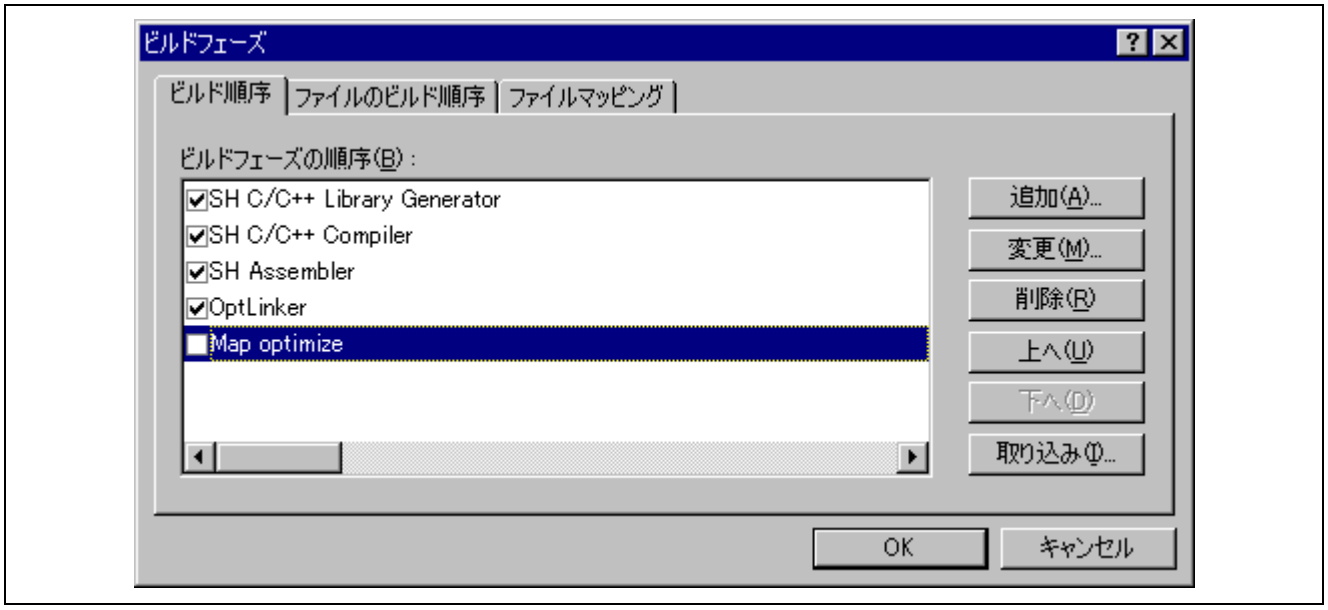

図 3.2 [ビルドフェーズ]ダイアログボックス

(3) M16C シリーズ,R8C ファミリ用 C コンパイラパッケージ V.5.x からのアップグレードの場合

(a) オブジェクトフォーマットの変更

M16C シリーズ,R8C ファミリ用 C/C++コンパイラパッケージ V.6.00 Release 00 以降のオブジェクトフォーマットは ELF/DWARF2 です。

[デバッグ -> デバッグの設定]で[デバッグの設定]ダイアログボックスを開き、[デバッグフォーマット]で ELF/DWARF2 を選択してください。次に、[変更]ボタンをクリックして[ダウンロードモジュール]ダイアログボックスを開いてください。[ファイルフォーマット]で ELF/DWARF2 を選択し、ファイル名の拡張子を.x30 から.abs に変更してください。

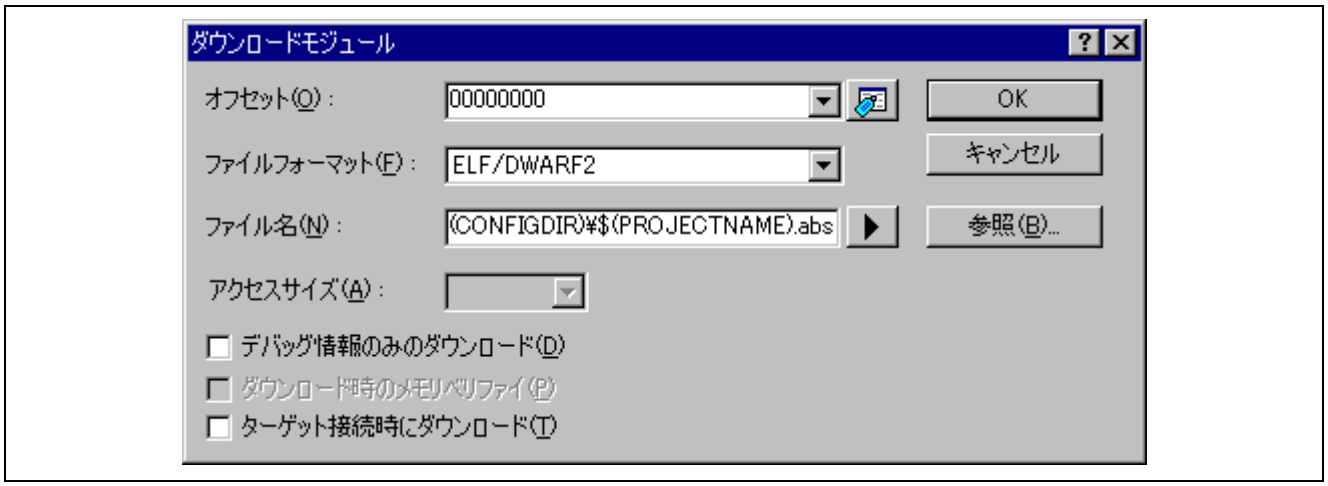

図 3.3 [ダウンロードモジュール]ダイアログボックス

4. RXのプロジェクトへ変換

以下に示すツールチェインをプロジェクトで使用している場合、RX ファミリ用 C/C++コンパイラパッケージ V.1.01 Release 00 以降をインストールすることで、プロジェクトを RX のプロジェクトへ変換できます。

- SuperH ファミリ用 C/C++コンパイラパッケージ V.9.04 Release 00
- H8SX,H8S,H8 ファミリ用 C/C++コンパイラパッケージ V.6.02 Release 00 ～ V.6.02 Release 02
- H8SX,H8S,H8 ファミリ用 C/C++コンパイラパッケージ V.7.00 Release 00
- M16C シリーズ,R8C ファミリ用 C コンパイラパッケージ V.5.45 Release 00 および V.5.45 Release 01
- M16C シリーズ,R8C ファミリ用 C/C++コンパイラパッケージ V.6.00 Release 00

このプロジェクト変換は、以下の設定にのみ適用され、現在のプロジェクトジェネレータを RX 用プロジェクトジェネレータへ移行します。

- ツールチェインビルドフェーズのオプション
  [ビルド -> RX Standard Toolchain]で設定できます。
- ビルドフェーズのシステムフェーズ（ツールチェインビルドフェーズ）
  [ビルド -> ビルドフェーズ]で設定できます。

この他のすべての設定、プロジェクトに登録されたファイル（ソースファイルなど）、コンフィグレーションフォルダや make フォルダの出力ファイルは変換しません。

現在の環境をバックアップする

プロジェクト変換後は、プロジェクトを元に戻すことはできません。
したがって、プロジェクト変換を始める前に、エクスプローラなどでプロジェクトが登録されているワークスペースフォルダをコピーし、別名で保存してください。

デバッガを使用する

プロジェクト変換後にデバッガを使用する場合は、RX 用プロジェクトジェネレータを選択した新規セッションを追加してください。
新規セッションの追加方法は、メインメニューから[ヘルプ->トピック]を選択して High-performance Embedded Workshop オンラインヘルプを開き、「セッションを追加する」トピック（デバッグ機能->デバッグの準備をする->デバッグセッション）を参照してください。

出力ファイルの内容を更新する

プロジェクト変換後は、必要に応じてセッションを追加し、元の出力ファイルの内容を更新するため以下を実行してください。

- [ビルド -> すべての依存関係を更新]
- [ビルド -> Make ファイルの生成] *
- [ビルド -> すべてをビルド]

注： *. Make ファイルを使用している場合のみ必要です。

プロジェクトを RX プロジェクトへ変換するには

[ツール -> ツールチェインバージョンを変更]で[ツールチェインのバージョンの変更]ダイアログボックスを開き、[CPU 種別]から「RX」を選択してください。

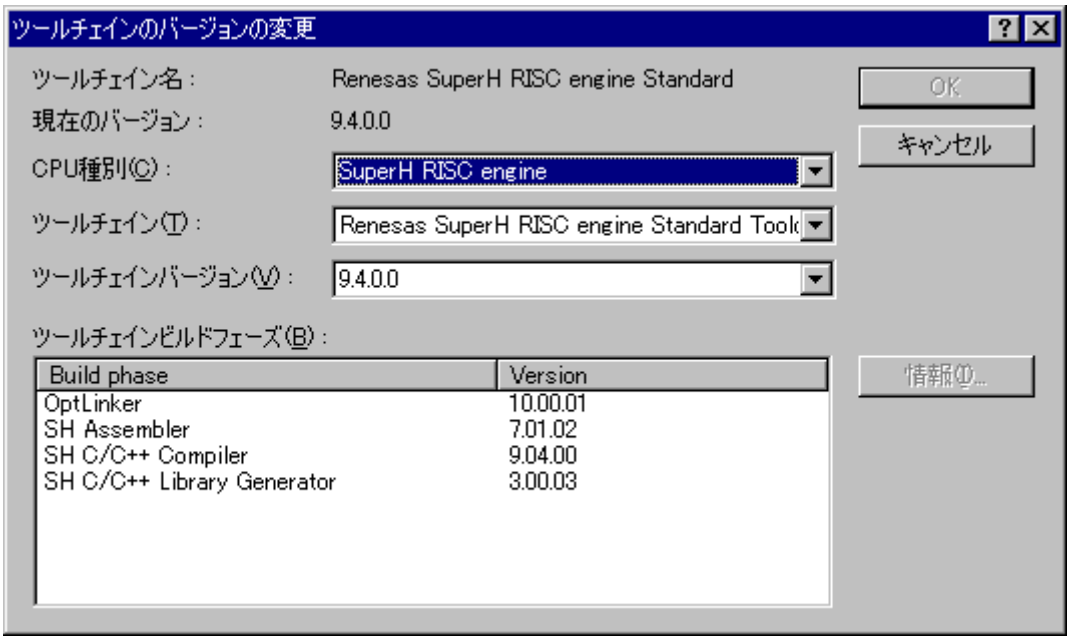

図 3.4 [ツールチェインのバージョンの変更]ダイアログボックス

以下の表に示すツールチェインビルドフェーズのオプションは、プロジェクト変換後も引き継がれます。[全般]の[依存関係検索エラーメッセージ抑止]、および[依存関係検索にてプリプロセッサ文サポート]も引き継がれます。

この他のツールチェインビルドフェーズのオプション、およびビルドフェーズのシステムフェーズは、RX 用のデフォルト値になります。

(a) SuperH ファミリ用 C/C++コンパイラパッケージ V.9.04 Release 00 -> RX

フェーズ	変換元オプション		変換先オプション	
Compiler	**コンパイラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]**	-include	**コンパイラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]**	-include
	コンパイラ <ソース> [オプション項目 :] [マクロ定義]	-define	**コンパイラ <ソース> [オプション項目 :] [マクロ定義]**	-define
	コンパイラ <ソース> [オプション項目 :] [デフォルトインクルードファイル]	-preinclude	**コンパイラ <ソース> [オプション項目 :] [デフォルトインクルードファイル]**	-preinclude
	コンパイラ <ソース> [オプション項目 :] [インフォメーションメッセージ] [インフォメーションレベルメッセージの表示]	-message	**コンパイラ <ソース> [オプション項目 :] [インフォメーションメッセージ] [インフォメーションレベルメッセージ抑止]**	-message
	コンパイラ <ソース> [オプション項目 :] [ファイル間インライン展開ディレクトリ]	-file_inline_path	**コンパイラ <ソース> [オプション項目 :] [ファイル間インライン展開ディレクトリ]**	-file_inline_path
	GUI **なし。入力ファイルの拡張子で言語を判断します。**	-lang	**コンパイラ <ソース> [オプション項目:] [ソースファイル] [言語 :]**	-lang

フェーズ	変換元オプション		変換先オプション	
	コンパイラ <オブジェクト> [出力ファイル形式 :]	-code=asmcode	コンパイラ <オブジェクト> [出力ファイル形式 :]	-output=src
	コンパイラ <オブジェクト> [出力ファイル形式 :]	-preprocessor	コンパイラ <オブジェクト> [出力ファイル形式 :]	-output=prep
	コンパイラ <オブジェクト> [出力ファイル形式 :]	-noline	コンパイラ <オブジェクト> [出力ファイル形式 :]	-noline
	コンパイラ <オブジェクト> [出力ディレクトリ:]		コンパイラ <オブジェクト> [出力ディレクトリ:]	
	コンパイラ <オブジェクト> [デバッグ情報出力]	-debug	コンパイラ <オブジェクト> [デバッグ情報出力]	-debug
	コンパイラ <オブジェクト> [詳細...] [セクション :]	-section	コンパイラ <オブジェクト> [詳細...] [セクション :]	-section
	コンパイラ <最適化> [最適化]	-optimize	コンパイラ <最適化> [最適化レベル :]	-optimize
	コンパイラ <最適化> [switch 文展開 :]	-case	コンパイラ <最適化> [詳細...] [その他] [switch 文展開 :]	-case
	コンパイラ <最適化> [詳細...] [インライン展開] [インライン展開ファイル]	-file_inline	コンパイラ <最適化> [詳細...] [インライン展開] [インライン展開ファイル]	-file_inline
	コンパイラ <最適化> [詳細...] [インライン展開] [自動インライン展開]	-inline	コンパイラ <最適化> [詳細...] [インライン展開] [自動インライン展開]	-inline
	コンパイラ <最適化> [モジュール間最適化]	-goptimize	コンパイラ <最適化> [モジュール間最適化]	-goptimize
	CPU [C++の try、throw、catch を有効にする]	-exception	CPU [詳細...] [C++の try、throw、catch を有効にする]	-exception
	CPU [C++の dynamic_cast、typeid を有効にする]	-rtti=on	CPU [詳細...] [C++の dynamic_cast、typeid を有効にする]	-rtti=on
	CPU [構造体メンバの境界調整数を 1 とする]	-pack=1	CPU [詳細...] [構造体メンバの境界調整数を 1 とする]	-pack
	CPU [Endian 選択 :]	-endian	CPU [エンディアン :]	-endian
Assembler	アセンブラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include	アセンブラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include
	アセンブラ <ソース> [オプション項目 :] [シンボル定義]	-define	アセンブラ <ソース> [オプション項目 :] [シンボル定義]	-define
	アセンブラ <オブジェクト> [デバッグ情報出力 :]	-debug	アセンブラ <オブジェクト> [デバッグ情報出力]	-debug
	アセンブラ <オブジェクト> [オブジェクト出力ディレクトリ :]		アセンブラ <オブジェクト> [出力ディレクトリ :]	
OptLinker	最適化リンカ <入力> [オプション項目 :] [ライブラリファイル]	-library	最適化リンカ <入力> [オプション項目 :] [ライブラリファイル]	-library
	最適化リンカ <入力> [オプション項目 :] [リロケータブルファイル/オブジェクトファイル]	-input	最適化リンカ <入力> [オプション項目 :] [リロケータブルファイル/オブジェクトファイル]	-input
	最適化リンカ <入力> [オプション項目 :] [バイナリファイル]	-binary	最適化リンカ <入力> [オプション項目 :] [バイナリファイル]	-binary

フェーズ	変換元オプション		変換先オプション	
	最適化リンカ <入力> [オプション項目 :] [シンボル定義]	-define	最適化リンカ <入力> [オプション項目 :] [シンボル定義]	-define
	最適化リンカ <入力> [エントリポイント :]	-entry	最適化リンカ <入力> [エントリポイント :]	-entry
	最適化リンカ <入力> [プレリンカ制御 :]	-noprelink	最適化リンカ <入力> [プレリンカ制御 :]	-noprelink
	最適化リンカ <出力> [出力形式 :]	-form	最適化リンカ <出力> [出力形式 :]	-form
	最適化リンカ <出力> [レコードサイズ統一 :]	-record	最適化リンカ <出力> [レコードサイズ統一 :]	-record
	最適化リンカ <出力> [データレコード長 :]	-byte_count	最適化リンカ <出力> [データレコード長 :]	-byte_count
	最適化リンカ <出力> [デバッグ情報 :]	-debug	最適化リンカ <出力> [デバッグ情報 :]	-debug
	最適化リンカ <出力> [デバッグ情報 :]	-sdebug	最適化リンカ <出力> [デバッグ情報 :]	-sdebug
	最適化リンカ <出力> [オプション項目 :] [出力ファイル]	-output	最適化リンカ <出力> [オプション項目 :] [出力ファイル]	-output
	最適化リンカ <出力> [オプション項目 :] [ROM から RAM へマップするセクション]	-rom	最適化リンカ <出力> [オプション項目 :] [ROM から RAM へマップするセクション]	-rom
	最適化リンカ <出力> [オプション項目 :] [出力ファイルの分割]	-output	最適化リンカ <出力> [オプション項目 :] [出力ファイルの分割]	-output
	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [インフォメーションレベルメッセージ抑止]	-message	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [インフォメーションレベルメッセージ抑止]	-message
	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [インフォメーションレベルメッセージ抑止]	-nomessage	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [インフォメーションレベルメッセージ抑止]	-nomessage
	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [参照されない定義シンボルの通知]	-msg_unused	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [参照されない定義シンボルの通知]	-msg_unused
	最適化リンカ <出力> [オプション項目 :] [空きエリア出力指定] [空きエリア出力]	-space	最適化リンカ <出力> [オプション項目 :] [空きエリア出力指定] [空きエリア出力]	-space
	最適化リンカ <出力> [オプション項目 :] [CRC コード]	-crc	最適化リンカ <出力> [オプション項目 :] [CRC コード]	-crc
	最適化リンカ <リスト> [リンケージリスト出力]	-list	最適化リンカ <リスト> [リンケージリスト出力]	-list
	最適化リンカ <リスト> [リスト内容 :]	-show	最適化リンカ <リスト> [リスト内容 :]	-show
	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-optimize	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-optimize
	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-nooptimize	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-nooptimize

フェーズ	変換元オプション		変換先オプション	
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [未参照シンボル削除抑止シンボル]	-symbol_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [未参照シンボル削除抑止シンボル]	-symbol_forbid
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [共通コード統合抑止シンボル]	-samecode_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [共通コード統合抑止シンボル]	-samecode_forbid
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止セクション]	-section_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止セクション]	-section_forbid
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止アドレス範囲]	-absolute_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止アドレス範囲]	-absolute_forbid
	最適化リンカ <セクション> [設定項目 :] [セクション]	-start	最適化リンカ <セクション> [設定項目 :] [セクション]	-start
	最適化リンカ <セクション> [設定項目 :] [シンボルアドレスファイル]	-fsymbol	最適化リンカ <セクション> [設定項目 :] [シンボルアドレスファイル]	-fsymbol
	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu
	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu=stride	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu=stride
	最適化リンカ <ベリファイ> [分割対象がセクション :]	-contiguous_section	最適化リンカ <ベリファイ> [分割対象がセクション :]	-contiguous_section
	最適化リンカ <その他> [その他のオプション :] [スタック情報ファイル(sni)出力]	-stack	最適化リンカ <その他> [その他のオプション :] [スタック情報ファイル(sni)出力]	-stack
	最適化リンカ <その他> [ユーザ指定オプション :]	-total_size	最適化リンカ <その他> [ユーザ指定オプション :]	-total_size
	最適化リンカ <その他> [その他のオプション :] [S9 レコードを終端に出力]	-s9	最適化リンカ <その他> [その他のオプション :] [S9 レコードを終端に出力]	-s9
	最適化リンカ <その他> [その他のオプション :] [入力ファイルロード時のメモリ使用量削減]	-memory	最適化リンカ <その他> [その他のオプション :] [入力ファイルロード時のメモリ使用量削減]	-memory
	最適化リンカ <その他> [その他のオプション :] [デバッグ情報圧縮]	-compress	最適化リンカ <その他> [その他のオプション :] [デバッグ情報圧縮]	-compress
	最適化リンカ <サブコマンドファイル> [サブコマンドファイルを指定]	-subcommand	最適化リンカ <サブコマンドファイル> [サブコマンドファイルを指定]	-subcommand

(b) H8SX,H8S,H8 ファミリ用 C/C++コンパイラパッケージ V.6.02 Release 00 ～ V.6.02 Release 02 -> RX

フェーズ	変換元オプション		変換先オプション	
Compiler	コンパイラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include	コンパイラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include
	コンパイラ <ソース> [オプション項目 :] [マクロ定義]	-define	コンパイラ <ソース> [オプション項目 :] [マクロ定義]	-define

フェーズ	変換元オプション		変換先オプション	
	コンパイラ <ソース> [オプション項目 :] [デフォルトインクルードファイル]	-preinclude	コンパイラ <ソース> [オプション項目 :] [デフォルトインクルードファイル]	-preinclude
	コンパイラ <ソース> [オプション項目 :] [インフォメーションメッセージ] [インフォメーションレベルメッセージの表示]	-message	コンパイラ <ソース> [オプション項目 :] [インフォメーションメッセージ] [インフォメーションレベルメッセージ抑止]	-message
	コンパイラ <ソース> [オプション項目 :] [ファイル間インライン展開ディレクトリ]	-file_inline_path	コンパイラ <ソース> [オプション項目 :] [ファイル間インライン展開ディレクトリ]	-file_inline_path
	コンパイラ <オブジェクト> [出力ファイル形式 :]	-code=asmcode	コンパイラ <オブジェクト> [出力ファイル形式 :]	-output=src
	コンパイラ <オブジェクト> [出力ファイル形式 :]	-preprocessor	コンパイラ <オブジェクト> [出力ファイル形式 :]	-output=prep
	コンパイラ <オブジェクト> [出力ディレクトリ:]		コンパイラ <オブジェクト> [出力ディレクトリ:]	
	コンパイラ <オブジェクト> [デバッグ情報出力]	-debug	コンパイラ <オブジェクト> [デバッグ情報出力]	-debug
	コンパイラ <オブジェクト> [出力文字コード :]	-outcode	コンパイラ <オブジェクト> [出力文字コード :]	-outcode
	コンパイラ <オブジェクト> [セクション :]	-section	コンパイラ <オブジェクト> [詳細...] [セクション :]	-section
	コンパイラ <最適化> [最適化設定] [設定]	-optimize	コンパイラ <最適化> [最適化レベル :]	-optimize
	コンパイラ <最適化> [Switch 文展開 :]	-case	コンパイラ <最適化> [詳細...] [その他] [switch 文展開 :]	-case
	コンパイラ <最適化> [詳細...] [インライン展開] [インライン展開ファイル]	-file_inline	コンパイラ <最適化> [詳細...] [インライン展開] [インライン展開ファイル]	-file_inline
	コンパイラ <最適化> [モジュール間最適化]	-goptimize	コンパイラ <最適化> [モジュール間最適化]	-goptimize
	CPU [C++の try、throw、catch を有効にする]	-exception	CPU [詳細...] [C++の try、throw、catch を有効にする]	-exception
	CPU [C++の dynamic_cast、typeid を有効にする]	-rtti=on	CPU [詳細...] [C++の dynamic_cast、typeid を有効にする]	-rtti=on
	CPU [メンバの境界調整数を 1 とする]	-pack=1	CPU [詳細...] [構造体メンバの境界調整数を 1 とする]	-pack
Assembler	アセンブラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include	アセンブラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include
	アセンブラ <ソース> [オプション項目 :] [シンボル定義]	-define	アセンブラ <ソース> [オプション項目 :] [シンボル定義]	-define
	アセンブラ <オブジェクト> [デバッグ情報出力 :]	-debug	アセンブラ <オブジェクト> [デバッグ情報出力]	-debug
	アセンブラ <オブジェクト> [モジュール間最適化]	-goptimize	アセンブラ <オブジェクト> [モジュール間最適化]	-goptimize
	アセンブラ <オブジェクト> [オブジェクト出力ディレクトリ :]		アセンブラ <オブジェクト> [出力ディレクトリ :]	

フェーズ	変換元オプション		変換先オプション	
OptLinker	最適化リンカ <入力> [オプション項目 :] [ライブラリファイル]	-library	最適化リンカ <入力> [オプション項目 :] [ライブラリファイル]	-library
	最適化リンカ <入力> [オプション項目 :] [リロケータブルファイル/オブジェクトファイル]	-input	最適化リンカ <入力> [オプション項目 :] [リロケータブルファイル/オブジェクトファイル]	-input
	最適化リンカ <入力> [オプション項目 :] [バイナリファイル]	-binary	最適化リンカ <入力> [オプション項目 :] [バイナリファイル]	-binary
	最適化リンカ <入力> [オプション項目 :] [シンボル定義]	-define	最適化リンカ <入力> [オプション項目 :] [シンボル定義]	-define
	最適化リンカ <入力> [エントリポイント :]	-entry	最適化リンカ <入力> [エントリポイント :]	-entry
	最適化リンカ <入力> [プレリンカ制御 :]	-noprelink	最適化リンカ <入力> [プレリンカ制御 :]	-noprelink
	最適化リンカ <出力> [出力形式 :]	-form	最適化リンカ <出力> [出力形式 :]	-form
	最適化リンカ <出力> [レコードサイズ統一 :]	-record	最適化リンカ <出力> [レコードサイズ統一 :]	-record
	最適化リンカ <出力> [データレコード長 :]	-byte_count	最適化リンカ <出力> [データレコード長 :]	-byte_count
	最適化リンカ <出力> [デバッグ情報 :]	-debug	最適化リンカ <出力> [デバッグ情報 :]	-debug
	最適化リンカ <出力> [デバッグ情報 :]	-sdebug	最適化リンカ <出力> [デバッグ情報 :]	-sdebug
	最適化リンカ <出力> [オプション項目 :] [出力ファイル]	-output	最適化リンカ <出力> [オプション項目 :] [出力ファイル]	-output
	最適化リンカ <出力> [オプション項目 :] [ROM から RAM へマップするセクション]	-rom	最適化リンカ <出力> [オプション項目 :] [ROM から RAM へマップするセクション]	-rom
	最適化リンカ <出力> [オプション項目 :] [出力ファイルの分割]	-output	最適化リンカ <出力> [オプション項目 :] [出力ファイルの分割]	-output
	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [インフォメーションレベルメッセージ抑止]	-message	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [インフォメーションレベルメッセージ抑止]	-message
	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [インフォメーションレベルメッセージ抑止]	-nomessage	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [インフォメーションレベルメッセージ抑止]	-nomessage
	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [参照されない定義シンボルの通知]	-msg_unused	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [参照されない定義シンボルの通知]	-msg_unused
	最適化リンカ <出力> [オプション項目 :] [空きエリア出力指定] [空きエリア出力]	-space	最適化リンカ <出力> [オプション項目 :] [空きエリア出力指定] [空きエリア出力]	-space
	最適化リンカ <出力> [オプション項目 :] [CRC コード]	-crc	最適化リンカ <出力> [オプション項目 :] [CRC コード]	-crc

フェーズ	変換元オプション		変換先オプション	
	最適化リンカ <リスト> [リンケージリスト出力]	-list	最適化リンカ <リスト> [リンケージリスト出力]	-list
	最適化リンカ <リスト> [リスト内容 :]	-show	最適化リンカ <リスト> [リスト内容 :]	-show
	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-optimize	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-optimize
	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-nooptimize	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-nooptimize
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [未参照シンボル削除抑止シンボル]	-symbol_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [未参照シンボル削除抑止シンボル]	-symbol_forbid
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [共通コード統合抑止シンボル]	-samecode_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [共通コード統合抑止シンボル]	-samecode_forbid
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止セクション]	-section_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止セクション]	-section_forbid
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止アドレス範囲]	-absolute_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止アドレス範囲]	-absolute_forbid
	最適化リンカ <セクション> [設定項目 :] [セクション]	-start	最適化リンカ <セクション> [設定項目 :] [セクション]	-start
	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu
	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu=stride	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu=stride
	最適化リンカ <ベリファイ> [分割対象がセクション :]	-contiguous_section	最適化リンカ <ベリファイ> [分割対象がセクション :]	-contiguous_section
	最適化リンカ <その他> [その他のオプション :] [スタック情報ファイル(sni)出力]	-stack	最適化リンカ <その他> [その他のオプション :] [スタック情報ファイル(sni)出力]	-stack
	最適化リンカ <その他> [ユーザ指定オプション :]	-total_size	最適化リンカ <その他> [ユーザ指定オプション :]	-total_size
	最適化リンカ <その他> [その他のオプション :] [S9 レコードを終端に出力]	-s9	最適化リンカ <その他> [その他のオプション :] [S9 レコードを終端に出力]	-s9
	最適化リンカ <その他> [その他のオプション :] [入力ファイルロード時のメモリ使用量削減]	-memory	最適化リンカ <その他> [その他のオプション :] [入力ファイルロード時のメモリ使用量削減]	-memory
	最適化リンカ <その他> [その他のオプション :] [デバッグ情報圧縮]	-compress	最適化リンカ <その他> [その他のオプション :] [デバッグ情報圧縮]	-compress
	最適化リンカ <サブコマンドファイル> [サブコマンドファイルを指定]	-subcommand	最適化リンカ <サブコマンドファイル> [サブコマンドファイルを指定]	-subcommand

(c) H8SX,H8S,H8 ファミリ用 C/C++コンパイラパッケージ V.7.00 Release 00 -> RX

フェーズ	変換元オプション		変換先オプション	
Compiler	コンパイラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include	コンパイラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include
	コンパイラ <ソース> [オプション項目 :] [マクロ定義]	-define	コンパイラ <ソース> [オプション項目 :] [マクロ定義]	-define
	コンパイラ <ソース> [オプション項目 :] [デフォルトインクルードファイル]	-preinclude	コンパイラ <ソース> [オプション項目 :] [デフォルトインクルードファイル]	-preinclude
	コンパイラ <ソース> [オプション項目 :] [インフォメーションメッセージ] [インフォメーションレベルメッセージの表示]	-message	コンパイラ <ソース> [オプション項目 :] [インフォメーションメッセージ] [インフォメーションレベルメッセージ抑止]	-message
	コンパイラ <ソース> [オプション項目 :] [ファイル間インライン展開ディレクトリ]	-file_inline_path	コンパイラ <ソース> [オプション項目 :] [ファイル間インライン展開ディレクトリ]	-file_inline_path
	コンパイラ <ソース> [C 言語の選択:]	-lang	コンパイラ <ソース> [オプション項目:] [ソースファイル] [言語 :]	-lang
	コンパイラ <オブジェクト> [出力ファイル形式 :]	-code=asmcode	コンパイラ <オブジェクト> [出力ファイル形式 :]	-output=src
	コンパイラ <オブジェクト> [出力ファイル形式 :]	-preprocessor	コンパイラ <オブジェクト> [出力ファイル形式 :]	-output=prep
	コンパイラ <オブジェクト> [出力ディレクトリ:]		コンパイラ <オブジェクト> [出力ディレクトリ:]	
	コンパイラ <オブジェクト> [デバッグ情報出力]	-debug	コンパイラ <オブジェクト> [デバッグ情報出力]	-debug
	コンパイラ <オブジェクト> [出力文字コード :]	-outcode	コンパイラ <オブジェクト> [出力文字コード :]	-outcode
	コンパイラ <オブジェクト> [詳細...] [セクション :]	-section	コンパイラ <オブジェクト> [詳細…] [セクション :]	-section
	コンパイラ <最適化> [最適化]	-optimize	コンパイラ <最適化> [最適化レベル :]	-optimize
	コンパイラ <最適化> [Switch 文展開 :]	-case	コンパイラ <最適化> [詳細…] [その他] [switch 文展開 :]	-case
	コンパイラ <最適化> [詳細…] [インライン展開] [インライン展開ファイル]	-file_inline	コンパイラ <最適化> [詳細…] [インライン展開] [インライン展開ファイル]	-file_inline
	コンパイラ <最適化> [モジュール間最適化]	-goptimize	コンパイラ <最適化> [モジュール間最適化]	-goptimize
	CPU [C++の try、throw、catch を有効にする]	-exception	CPU [詳細…] [C++の try、throw、catch を有効にする]	-exception
	CPU [C++の dynamic_cast、typeid を有効にする]	-rtti=on	CPU [詳細…] [C++の dynamic_cast、typeid を有効にする]	-rtti=on
	CPU [メンバの境界調整数を 1 とする]	-pack=1	CPU [詳細…] [構造体メンバの境界調整数を 1 とする]	-pack
Assembler	アセンブラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include	アセンブラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include

フェーズ	変換元オプション		変換先オプション	
	アセンブラ <ソース> [オプション項目 :] [シンボル定義]	-define	アセンブラ <ソース> [オプション項目 :] [シンボル定義]	-define
	アセンブラ <オブジェクト> [デバッグ情報出力 :]	-debug	アセンブラ <オブジェクト> [デバッグ情報出力]	-debug
	アセンブラ <オブジェクト> [モジュール間最適化]	-goptimize	アセンブラ <オブジェクト> [モジュール間最適化]	-goptimize
	アセンブラ <オブジェクト> [オブジェクト出力ディレクトリ :]		アセンブラ <オブジェクト> [出力ディレクトリ :]	
OptLinker	最適化リンカ <入力> [オプション項目 :] [ライブラリファイル]	-library	最適化リンカ <入力> [オプション項目 :] [ライブラリファイル]	-library
	最適化リンカ <入力> [オプション項目 :] [リロケータブルファイル/オブジェクトファイル]	-input	最適化リンカ <入力> [オプション項目 :] [リロケータブルファイル/オブジェクトファイル]	-input
	最適化リンカ <入力> [オプション項目 :] [バイナリファイル]	-binary	最適化リンカ <入力> [オプション項目 :] [バイナリファイル]	-binary
	最適化リンカ <入力> [オプション項目 :] [シンボル定義]	-define	最適化リンカ <入力> [オプション項目 :] [シンボル定義]	-define
	最適化リンカ <入力> [エントリポイント :]	-entry	最適化リンカ <入力> [エントリポイント :]	-entry
	最適化リンカ <入力> [プレリンカ制御 :]	-noprelink	最適化リンカ <入力> [プレリンカ制御 :]	-noprelink
	最適化リンカ <出力> [出力形式 :]	-form	最適化リンカ <出力> [出力形式 :]	-form
	最適化リンカ <出力> [レコードサイズ統一 :]	-record	最適化リンカ <出力> [レコードサイズ統一 :]	-record
	最適化リンカ <出力> [データレコード長 :]	-byte_count	最適化リンカ <出力> [データレコード長 :]	-byte_count
	最適化リンカ <出力> [デバッグ情報 :]	-debug	最適化リンカ <出力> [デバッグ情報 :]	-debug
	最適化リンカ <出力> [デバッグ情報 :]	-sdebug	最適化リンカ <出力> [デバッグ情報 :]	-sdebug
	最適化リンカ <出力> [オプション項目 :] [出力ファイル]	-output	最適化リンカ <出力> [オプション項目 :] [出力ファイル]	-output
	最適化リンカ <出力> [オプション項目 :] [ROM から RAM へマップするセクション]	-rom	最適化リンカ <出力> [オプション項目 :] [ROM から RAM へマップするセクション]	-rom
	最適化リンカ <出力> [オプション項目 :] [出力ファイルの分割]	-output	最適化リンカ <出力> [オプション項目 :] [出力ファイルの分割]	-output
	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [インフォメーションレベルメッセージ抑止]	-message	最適化リンカ <出力> [オプション項目 :] [出力ファイル/インフォメーション抑止] [インフォメーションレベルメッセージ抑止]	-message

フェーズ	変換元オプション		変換先オプション	
	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [インフォメーションレベルメッセージ抑止]	-nomessage	最適化リンカ <出力> [オプション項目 :] [出力ファイル/インフォメーション抑止] [インフォメーションレベルメッセージ抑止]	-nomessage
	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [参照されない定義シンボルの通知]	-msg_unused	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [参照されない定義シンボルの通知]	-msg_unused
	最適化リンカ <出力> [オプション項目 :] [空きエリア出力指定] [空きエリア出力]	-space	最適化リンカ <出力> [オプション項目 :] [空きエリア出力指定] [空きエリア出力]	-space
	最適化リンカ <出力> [オプション項目 :] [CRC コード]	-crc	最適化リンカ <出力> [オプション項目 :] [CRC コード]	-crc
	最適化リンカ <リスト> [リンケージリスト出力]	-list	最適化リンカ <リスト> [リンケージリスト出力]	-list
	最適化リンカ <リスト> [リスト内容 :]	-show	最適化リンカ <リスト> [リスト内容 :]	-show
	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-optimize	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-optimize
	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-nooptimize	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-nooptimize
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [未参照シンボル削除抑止シンボル]	-symbol_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [未参照シンボル削除抑止シンボル]	-symbol_forbid
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [共通コード統合抑止シンボル]	-samecode_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [共通コード統合抑止シンボル]	-samecode_forbid
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止セクション]	-section_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止セクション]	-section_forbid
	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止アドレス範囲]	-absolute_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止アドレス範囲]	-absolute_forbid
	最適化リンカ <セクション> [設定項目 :] [セクション]	-start	最適化リンカ <セクション> [設定項目 :] [セクション]	-start
	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu
	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu=stride	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu=stride
	最適化リンカ <ベリファイ> [分割対象がセクション :]	-contiguous_section	最適化リンカ <ベリファイ> [分割対象がセクション :]	-contiguous_section
	最適化リンカ <その他> [その他のオプション :] [スタック情報ファイル(sni)出力]	-stack	最適化リンカ <その他> [その他のオプション :] [スタック情報ファイル(sni)出力]	-stack
	最適化リンカ <その他> [ユーザ指定オプション :]	-total_size	最適化リンカ <その他> [ユーザ指定オプション :]	-total_size
	最適化リンカ <その他> [その他のオプション :] [S9 レコードを終端に出力]	-s9	最適化リンカ <その他> [その他のオプション :] [S9 レコードを終端に出力]	-s9

フェーズ	変換元オプション		変換先オプション	
	最適化リンカ <その他> [その他のオプション :] [入力ファイルロード時のメモリ使用量削減]	-memory	最適化リンカ <その他> [その他のオプション :] [入力ファイルロード時のメモリ使用量削減]	-memory
	最適化リンカ <その他> [その他のオプション :] [デバッグ情報圧縮]	-compress	最適化リンカ <その他> [その他のオプション :] [デバッグ情報圧縮]	-compress
	最適化リンカ <サブコマンドファイル> [サブコマンドファイルを指定]	-subcommand	最適化リンカ <サブコマンドファイル> [サブコマンドファイルを指定]	-subcommand

(d) M16C シリーズ,R8C ファミリ用 C コンパイラパッケージ V.5.45 Release 00 および V.5.45 Release 01 -> RX

フェーズ	変換元オプション		変換先オプション	
Compiler	コンパイラ <ソース> [オプション項目 :] [インクルードファイル検索ディレクトリ]	-I	コンパイラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include
	コンパイラ <ソース> [オプション項目 :] [識別子定義]	-D	コンパイラ <ソース> [オプション項目 :] [マクロ定義]	-define
	コンパイラ <オブジェクト> [ファイルフォーマット :]	-S	コンパイラ <オブジェクト> [出力ファイル形式 :]	-output=src
	コンパイラ <オブジェクト> [ファイルフォーマット :]	-P	コンパイラ <オブジェクト> [出力ファイル形式 :]	-output=prep
	コンパイラ <オブジェクト> [ファイルフォーマット :]	-E	コンパイラ <オブジェクト> [出力ファイル形式 :]	-output=prep
	コンパイラ <オブジェクト> [出力先ディレクトリ指定 :]	-dir	コンパイラ <オブジェクト> [出力ディレクトリ:]	
	コンパイラ <オブジェクト> [デバッグオプション :] [デバッグ情報をアセンブリ言語ソースファイルに出力する]	-g	コンパイラ <オブジェクト> [デバッグ情報出力]	-debug
Assembler	アセンブラ <ソース> [オプション項目 :] [インクルードファイル検索ディレクトリ] [インクルードファイルの検索… :]	-I	アセンブラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include
	アセンブラ <ソース> [オプション項目 :] [識別子定義]	-D	アセンブラ <ソース> [オプション項目 :] [シンボル定義]	-define
	アセンブラ <オブジェクト> [出力先ディレクトリ指定 :]	-O	アセンブラ <オブジェクト> [出力ディレクトリ :]	-output
	アセンブラ <オブジェクト> [ローカルシンボル情報を出力する]	-S	アセンブラ <オブジェクト> [デバッグ情報出力]	-debug
	アセンブラ <オブジェクト> [システムシンボル情報を出力する]	-SM	アセンブラ <オブジェクト> [デバッグ情報出力]	-debug
Optlnk	リンカ <入力> [オプション項目 :] [ライブラリファイル]	-L	最適化リンカ <入力> [オプション項目 :] [ライブラリファイル]	-library
	リンカ <入力> [オプション項目 :] [リロケータブルファイル]		最適化リンカ <入力> [オプション項目 :] [リロケータブルファイル/オブジェクトファイル]	-input

フェーズ	変換元オプション		変換先オプション	
	リンカ <入力> [アブソリュートモジュールの開始アドレスを指定する :]	-E	最適化リンカ <入力> [エントリポイント :]	-entry
	リンカ <出力> [ソースデバッグ情報をアブソリュートモジュールファイルに出力する]	-G	最適化リンカ <出力> [デバッグ情報 :]	-debug
			最適化リンカ <その他> [その他のオプション :] [スタック情報ファイル(sni)出力]	-stack
	リンカ <出力> [出力ファイルパス指定]	-O	最適化リンカ <出力> [オプション項目 :] [出力ファイル]	-output
	リンカ <コマンドファイル> [コマンドファイルの記述内容に従ってリンケージエディタを実行する]	@	最適化リンカ <サブコマンドファイル> [サブコマンドファイルを指定]	-subcommand
	ロードモジュールコンバータ <出力> [フォーマット :]	-H	最適化リンカ <出力> [出力形式 :]	-form
	ライブラリアン <操作> [操作 :]	-C	最適化リンカ <出力> [出力形式 :]	-form
RTOS	RTOS <Mkmrtbl> [MRC ファイル検索ディレクトリ :]		RTOS <テーブル生成> [MRC ファイル検索ディレクトリ :]	

(e) R8C,M16C ファミリ用 C/C++コンパイラパッケージ V.6.00 Release 00 -> RX

フェーズ	変換元オプション		変換先オプション	
Compiler	コンパイラ <ソース> [オプション項目:] [ソースファイル] [言語 :]	-lang	コンパイラ <ソース> [オプション項目:] [ソースファイル] [言語 :]	-lang
	コンパイラ <ソース> [オプション項目 :] [インクルードファイル検索ディレクトリ]	-I	コンパイラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include
	コンパイラ <ソース> [オプション項目 :] [識別子定義]	-D	コンパイラ <ソース> [オプション項目 :] [マクロ定義]	-define
	コンパイラ <ソース> [オプション項目 :] [デフォルトインクルードファイル]	-preinclude	コンパイラ <ソース> [オプション項目 :] [デフォルトインクルードファイル]	-preinclude
	コンパイラ <オブジェクト> [ファイルフォーマット :]	-S	コンパイラ <オブジェクト> [出力ファイル形式 :]	-output=src
	コンパイラ <オブジェクト> [ファイルフォーマット :]	-P	コンパイラ <オブジェクト> [出力ファイル形式 :]	-output=prep
	コンパイラ <オブジェクト> [出力先ディレクトリ指定 :]	-dir	コンパイラ <オブジェクト> [出力ディレクトリ:]	
	コンパイラ <オブジェクト> [デバッグオプション :] [デバッグ情報をアセンブリ言語ソースファイルに出力する]	-g	コンパイラ <オブジェクト> [デバッグ情報出力]	-debug
	コンパイラ <最適化> [モジュール間最適化]	-goptimize	コンパイラ <最適化> [モジュール間最適化]	-goptimize
	CPU [C++の try、throw、catch を有効にする]	-exception	CPU [詳細…] [C++の try、throw、catch を有効にする]	-exception

フェーズ	変換元オプション		変換先オプション	
	CPU [C++の dynamic_cast、typeid を有効にする]	-rtti=on	CPU [詳細...] [C++の dynamic_cast、typeid を有効にする]	-rtti=on
Assembler	アセンブラ <ソース> [オプション項目 :] [インクルードファイル検索ディレクトリ]	-I	アセンブラ <ソース> [オプション項目 :] [インクルードファイルディレクトリ]	-include
	アセンブラ <ソース> [オプション項目 :] [識別子定義]	-D	アセンブラ <ソース> [オプション項目 :] [シンボル定義]	-define
	アセンブラ <オブジェクト> [ローカルシンボル情報を出力する]	-S	アセンブラ <オブジェクト> [デバッグ情報出力]	-debug
	アセンブラ <オブジェクト> [システムシンボル情報を出力する]	-SM	アセンブラ <オブジェクト> [デバッグ情報出力]	-debug
	アセンブラ <オブジェクト> [モジュール間最適化]	-goptimize	アセンブラ <オブジェクト> [モジュール間最適化]	-goptimize
	アセンブラ <オブジェクト> [出力先ディレクトリ指定 :]	-O	アセンブラ <オブジェクト> [出力ディレクトリ :]	
OptLinker	リンカ <入力> [オプション項目 :] [ライブラリファイル]	-library	最適化リンカ <入力> [オプション項目 :] [ライブラリファイル]	-library
	リンカ <入力> [オプション項目 :] [リロケータブルファイル/オブジェクトファイル]	-input	最適化リンカ <入力> [オプション項目 :] [リロケータブルファイル/オブジェクトファイル]	-input
	リンカ <入力> [オプション項目 :] [バイナリファイル]	-binary	最適化リンカ <入力> [オプション項目 :] [バイナリファイル]	-binary
	リンカ <入力> [オプション項目 :] [シンボル定義]	-define	最適化リンカ <入力> [オプション項目 :] [シンボル定義]	-define
	リンカ <入力> [エントリポイント :]	-entry	最適化リンカ <入力> [エントリポイント :]	-entry
	リンカ <入力> [プレリンカ制御 :]	-noprelink	最適化リンカ <入力> [プレリンカ制御 :]	-noprelink
	リンカ <出力> [出力形式 :]	-form	最適化リンカ <出力> [出力形式 :]	-form
	リンカ <出力> [レコードサイズ統一 :]	-record	最適化リンカ <出力> [レコードサイズ統一 :]	-record
	リンカ <出力> [データレコード長 :]	-byte_count	最適化リンカ <出力> [データレコード長 :]	-byte_count
	リンカ <出力> [デバッグ情報 :]	-debug	最適化リンカ <出力> [デバッグ情報 :]	-debug
	リンカ <出力> [デバッグ情報 :]	-sdebug	最適化リンカ <出力> [デバッグ情報 :]	-sdebug
	リンカ <出力> [オプション項目 :] [出力ファイル]	-output	最適化リンカ <出力> [オプション項目 :] [出力ファイル]	-output
	リンカ<出力> [オプション項目 :] [ROM から RAM へマップするセクション]	-rom	最適化リンカ <出力> [オプション項目 :] [ROM から RAM へマップするセクション]	-rom

フェーズ	変換元オプション		変換先オプション	
	リンカ <出力> [オプション項目 :] [出力ファイルの分割]	-output	最適化リンカ <出力> [オプション項目 :] [出力ファイルの分割]	-output
	リンカ <出力> [オプション項目 :] [出力ファイル/インフォメーション抑止] [インフォメーションレベルメッセージ抑止]	-message	最適化リンカ <出力> [オプション項目 :] [出力ファイル/インフォメーション抑止] [インフォメーションレベルメッセージ抑止]	-message
	リンカ<出力> [オプション項目 :] [出力ファイル/インフォメーション抑止] [インフォメーションレベルメッセージ抑止]	-nomessage	最適化リンカ <出力> [オプション項目 :] [出力ファイル/インフォメーション抑止] [インフォメーションレベルメッセージ抑止]	-nomessage
	リンカ <出力> [オプション項目 :] [メッセージ出力指定] [参照されない定義シンボルの通知]	-msg_unused	最適化リンカ <出力> [オプション項目 :] [メッセージ出力指定] [参照されない定義シンボルの通知]	-msg_unused
	リンカ <出力> [オプション項目 :] [空きエリア出力指定] [空きエリア出力]	-space	最適化リンカ <出力> [オプション項目 :] [空きエリア出力指定] [空きエリア出力]	-space
	リンカ <出力> [オプション項目 :] [CRC コード]	-crc	最適化リンカ <出力> [オプション項目 :] [CRC コード]	-crc
	リンカ <出力> [オプション項目 :] [ベクタ] [特定ベクタ :]	-vectn	最適化リンカ <出力> [オプション項目 :] [ベクタ] [特定ベクタ :]	-vectn
	リンカ<出力> [オプション項目 :] [ベクタ] [空きベクタ :]	-vect	最適化リンカ <出力> [オプション項目 :] [ベクタ] [空きベクタ :]	-vect
	リンカ <リスト> [リンケージリスト出力]	-list	最適化リンカ <リスト> [リンケージリスト出力]	-list
	リンカ <リスト> [リスト内容 :]	-show	最適化リンカ <リスト> [リスト内容 :]	-show
	リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-optimize	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-optimize
	リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-nooptimize	最適化リンカ <最適化> [最適化方法 :] [最適化設定] [設定 :]	-nooptimize
	リンカ <最適化> [最適化方法 :] [最適化部分抑止] [未参照シンボル削除抑止シンボル]	-symbol_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [未参照シンボル削除抑止シンボル]	-symbol_forbid
	リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止セクション]	-section_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止セクション]	-section_forbid
	リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止アドレス範囲]	-absolute_forbid	最適化リンカ <最適化> [最適化方法 :] [最適化部分抑止] [最適化抑止アドレス範囲]	-absolute_forbid
	リンカ <セクション> [設定項目 :] [セクション]	-start	最適化リンカ <セクション> [設定項目 :] [セクション]	-start
	リンカ <ベリファイ> [アドレス整合チェック :]	-cpu	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu
	リンカ <ベリファイ> [アドレス整合チェック :]	-cpu=stride	最適化リンカ <ベリファイ> [アドレス整合チェック :]	-cpu=stride

フェーズ	変換元オプション		変換先オプション	
	リンカ <ベリファイ> [分割対象がセクション :]	-contiguous_section	最適化リンカ <ベリファイ> [分割対象がセクション :]	-contiguous_section
	リンカ <その他> [その他のオプション :] [スタック情報ファイル(sni)出力]	-stack	最適化リンカ <その他> [その他のオプション :] [スタック情報ファイル(sni)出力]	-stack
	リンカ <その他> [ユーザ指定オプション :]	-total_size	最適化リンカ <その他> [ユーザ指定オプション :]	-total_size
	リンカ <その他> [その他のオプション :] [S9 レコードを終端に出力]	-s9	最適化リンカ <その他> [その他のオプション :] [S9 レコードを終端に出力]	-s9
	リンカ <その他> [その他のオプション :] [入力ファイルロード時のメモリ使用量削減]	-memory	最適化リンカ <その他> [その他のオプション :] [入力ファイルロード時のメモリ使用量削減]	-memory
	リンカ <その他> [その他のオプション :] [デバッグ情報圧縮]	-compress	最適化リンカ <その他> [その他のオプション :] [デバッグ情報圧縮]	-compress
	リンカ <サブコマンドファイル> [サブコマンドファイルを指定]	-subcommand	最適化リンカ <サブコマンドファイル> [サブコマンドファイルを指定]	-subcommand
RTOS	RTOS <Mkmrtbl> [MRC ファイル検索ディレクトリ :]		RTOS <テーブル生成> [MRC ファイル検索ディレクトリ :]	

5. Microsoft® Windows Vista® およびWindows® 7 互換性対応

5.1 Windows Vista® およびWindows® 7 における標準権限での使用について

Windows Vista® および Windows® 7 における標準権限での使用について High-performance Embedded Workshop V.4.05.00 以降の使用上の注意事項を以下に示します。

5.1.1 内容

High-performance Embedded Workshop および High-performance Embedded Workshop システム下のすべてのソフトウェアツール製品が Windows Vista® または Windows® 7 対応版であるにもかかわらず、管理者権限の要求画面が表示される（標準権限で動作しない）場合があります。
これは、V.4.04.01 以前の High-performance Embedded Workshop を Windows Vista® または Windows® 7 で動作させるために管理者権限で実行するプログラムの互換設定をして、その後 Windows Vista® 対応版の High-performance Embedded Workshop V.4.05.00 以降、または Windows® 7 対応版の High-performance Embedded Workshop V.4.08.00 以降にアップデートした場合に起こります。

補足：
High-performance Embedded Workshop は、以下の 2 点を満たす場合に、Windows Vista® または Windows® 7 において標準権限で使用できます。
(1) Windows Vista® 対応版（V.4.05.00 以降）、または Windows® 7 対応版（V.4.08.00 以降）である。
(2) High-performance Embedded Workshop システム下で使用するすべてのソフトウェア製品が Windows Vista® または Windows® 7 対応版である。

上記以外の場合、およびソフトウェア製品のインストール時には管理者権限を必要とします。

5.1.2 回避策

High-performance Embedded Workshop システム下のすべてのソフトウェア製品を Windows Vista® または Windows® 7 対応版にしているにもかかわらず、管理者権限の要求画面が表示されている場合には、以下の方法で、互換モードを適用しない設定にしてください。

High-performance Embedded Workshop のショートカットのプロパティを開き、「互換性」タブで以下の 2 つのチェックボックスのチェックを外して無効にする。
- 互換モードでこのプログラムを実行する
- 管理者としてこのプログラムを実行する

ご注意書き

1. 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・システムの設計において、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て、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。
2. 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。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。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。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   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、産業用ロボット等
   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、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図し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損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の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。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、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、社会的損害等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、お客様の機器・システムとしての出荷保証を行ってください。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、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の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。
9. 本資料に記載されている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・システムに使用することはできません。また、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に使用しないでください。当社製品または技術を輸出する場合は、「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。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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禁じます。

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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/