コンパイラパッケージ
アプリケーションノート：< STL ライブラリ V.1.00.00 >
ユーザーズマニュアル

RJJ10J2802-0100
Rev.1.00
2010.07.30

本ドキュメントでは、Renesas C/C++コンパイラ向け STL ライブラリ V.1.00.00 の使用方法を説明します。

目次

1. はじめに

本マニュアルは、以下のルネサス エレクトロニクス社（以下 弊社）製 C/C++コンパイラ向けに提供する STL ライブラリ V.1.00.00 (以下 本 STL ライブラリ)の機能を追加するための手順と、コンパイラの仕様による使用上の制限事項を説明します。

・ RX ファミリ用 C/C++コンパイラ V.1.00.00 以降 （以下 RXC）
・ SuperH ファミリ用 C/C++コンパイラ V.9.04.00 以降（以下 SHC）

1.1. STL 母体

本 STL ライブラリは、以下のオープンソース STLport を母体として利用しています。

・ STLport-5.2.1

上記 STLport ライブラリの使用については、
http://www.stlport.org の STLport のライセンスポリシーに基づいています。

1.2. 対象外の機能

本STLライブラリは、「1.1. STL母体」で示す機能のうち、表 1に示す機能を対象外としています。
尚、対象としている機能一覧についてはアプリケーションノート「STL ライブラリ サポート機能一覧」を参照下さい。

表 1. 対象外の機能

対象外の機能
fstream、iomanip、ios、iosfwd、iostream、istream、ostream、sstream、streambuf、strstream、hash_map、hash_set、pthread_alloc、rope、slist、type_traits、unordered_map、unordered_set、clocale, csignal, ctime、(*1) ciso646, (*1)cwchar, (*1)cwctype、(*2)complex のストリーム入出力、(*3)string のストリーム入出力、(*4)wstring のストリーム入出力

(*1) SHC のみ対象外
(*2) char型を除く。詳細は「3.1 complexの制限事項」を参照。
(*3) char型を除く。詳細は「6.3ストリーム入出力処理利用時の制限事項」を参照。
(*4) 詳細は「6.3ストリーム入出力処理利用時の制限事項」を参照。

1.3. 免責

本 STL ライブラリは、弊社サポートの対象外となります。利用につきましては、弊社は一切の責任を負いません。お客様の責任において、十分テストしてご使用頂きますようお願い致します。

2. STL 機能組み込み手順

RXC/SHC に、STL 機能を組み込むための手順を示します。

2.1. STL ライブラリの配置

本 STL ライブラリに含まれるディレクトリ[stlport]を任意の場所に配置してください。
本 STL ライブラリを同梱しているコンパイラパッケージをインストールした場合は、図 1 のようにディレクトリが展開されます。以降は、アクティブな High-performance Embedded Workshop(以下 HEW)のディレクトリが”C:¥Program Files¥Renesas¥Hew”で、ディレクトリ[stlport]が以下に展開されているものとして説明します。

<stlport ディレクトリ> = C:¥Program Files¥Renesas¥Hew¥EXAMPLES¥STL¥1_0_0¥stlport

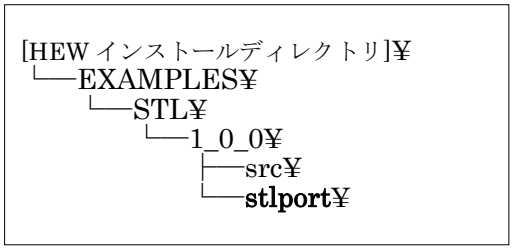

図 1. コンパイラパッケージインストール時の STL ライブラリの配置

2.2. 環境設定

RXC/SHCにおいてSTLを使う場合、ビルド時に利用するインクルードファイルディレクトリに、2.1で配置した<stlportディレクトリ>を追加する必要があります。その方法として、HEWによるツールチェインのオプションダイアログで設定する方法と、コマンドラインで設定する方法があります。本節ではそれぞれの方法を説明します。

2.2.1. HEW による設定（RXC/SHC）

ここではHEWによるツールチェインのオプションダイアログから、インクルードファイルディレクトリを指定する方法を説明します。以下の手順を実施してください。RXCの場合として、図 2 ～ 図 5 にその様子を示します。

(1) メニューのビルドから、下記を選択してください。
RXCの場合：RX Standard Toolchain...（図 2）
(2) 開いたダイアログから、「コンパイラ」タブの
オプション項目(S): インクルードファイルディレクトリ」を開いてください。（図 3）
(3) 「追加(A)...」ボタンを押してください。
(4) 相対パス(R): HEW installation directory を選択してください。
(5) サブディレクトリ(S): に、「EXAMPLES¥STL¥1_0_0¥stlport」
と入力し、OKボタンを押してください。（図 4）
(6) Toolchain ダイアログに、「$(HEWDIR)¥EXAMPLES¥STL¥1_0_0¥stlport」
が追加されていることを確認してください。
(7) OKボタンを押してください。（図 5）

以上で STL が利用可能になります。

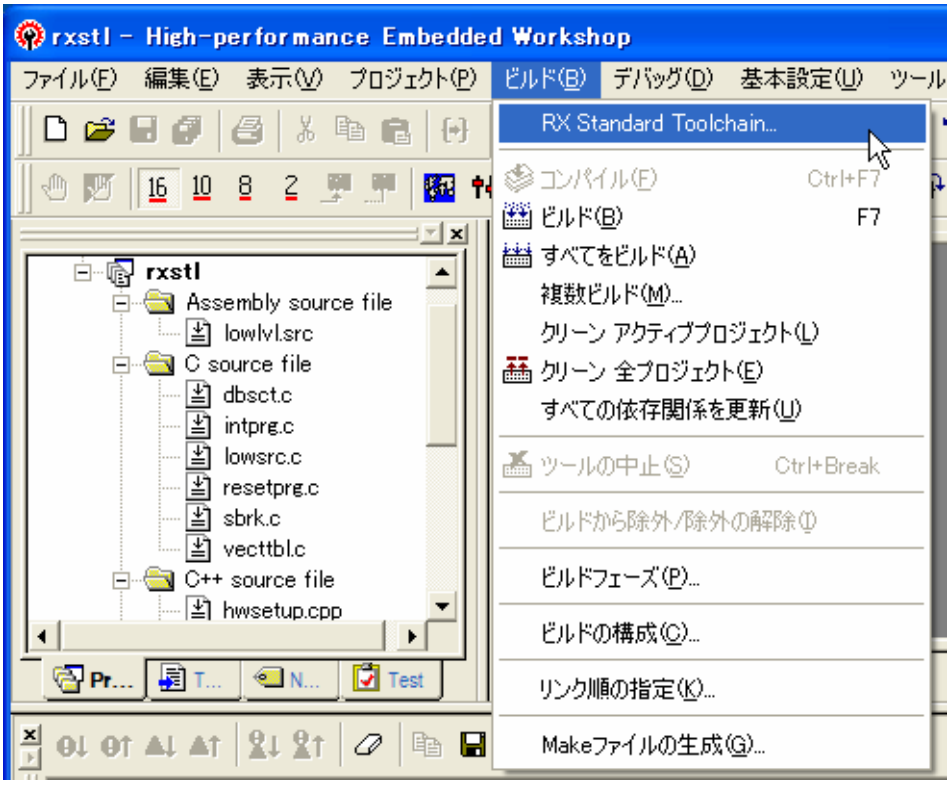

図 2. RX Standard Toolchain... の場所

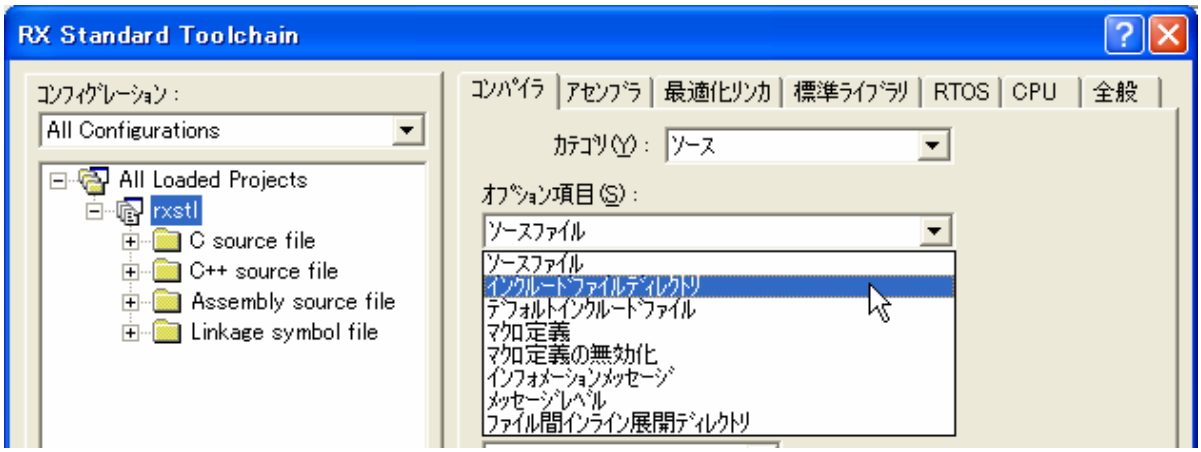

図 3. インクルードファイルディレクトリ設定の場所

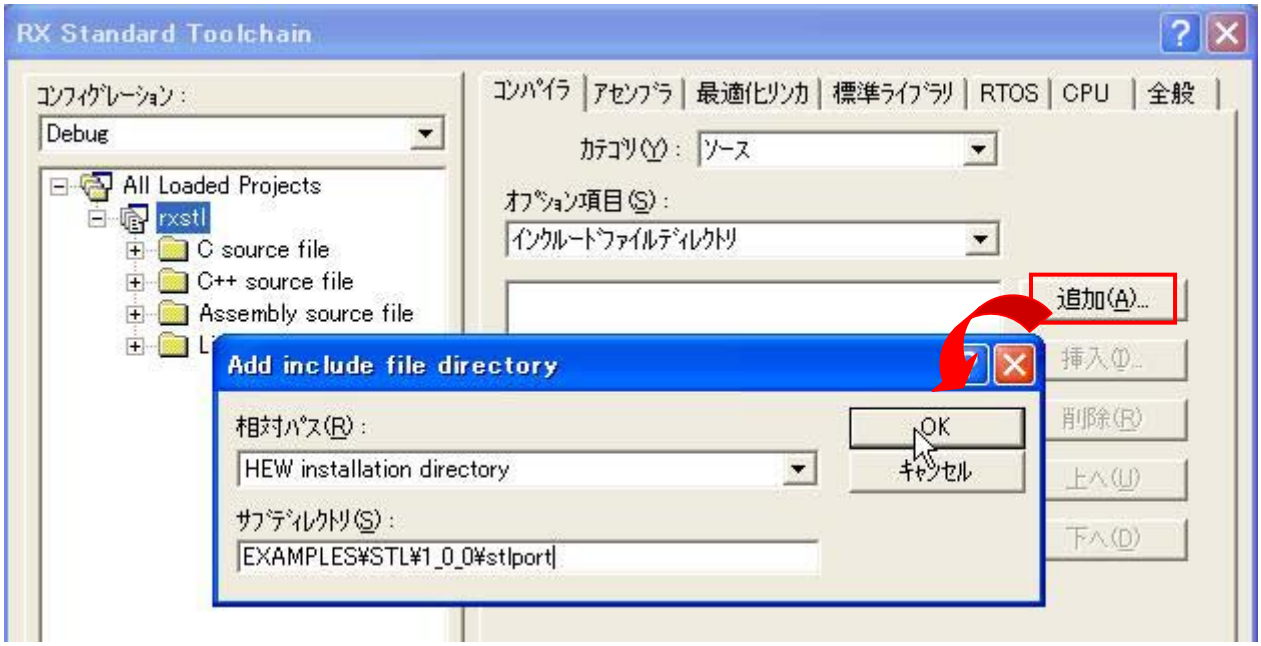

図 4. ディレクトリの指定例

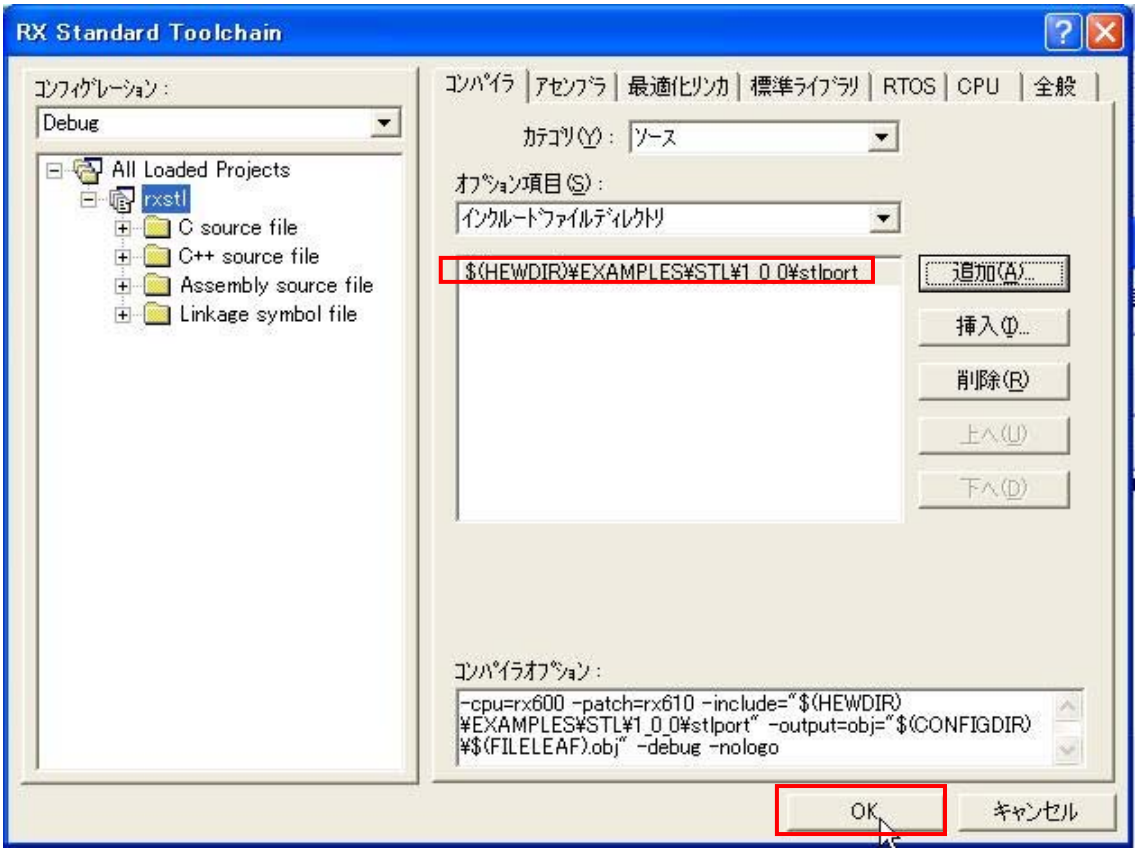

図 5. stlport ディレクトリを追加した状態

2.2.2. コマンドラインによる設定（RXC）

ここでは RXC の場合の、環境変数設定によるインクルードファイルディレクトリ追加の手順を説明します。

INC_RX 環境変数 にて標準インクルードディレクトリをセットする際に、**<stlport ディレクトリ>を既存のディレクトリ指定の左に記述してください。**

INC_RX=<stlport ディレクトリ>;<標準 include のディレクトリ>

のように、セミコロンで区切ることで連続して指定できます。

標準的な例を以下に示します。

```
INC_RX=C:¥Program Files¥Renesas¥Hew¥EXAMPLES¥STL¥1_0_0¥stlport
      ;C:¥Program Files¥Renesas¥Hew¥Tools¥Renesas¥RX¥1_0_0¥Include
```

環境変数を変更できない環境の場合、コンパイラの include オプションで指定することも可能です。

コンパイラの include オプションで、下記のように指定してください。

-include=<stlport ディレクトリ>

具体的な例として、下記のようなコマンドになります。

```
> ccrx tp.cpp -cpu=rx600 -include="C:¥Program Files¥Renesas¥Hew¥EXAMPLES¥STL¥1_0_0¥stlport"
```

以上で STL が利用可能になります。

2.2.3. コマンドラインによる設定（SHC）

ここでは SHC の場合の、環境変数設定によるインクルードファイルディレクトリ追加の手順を説明します。

SHC_INC 環境変数 または SHC_LIB 環境変数 に標準インクルードディレクトリをセットする際、**<stlport ディレクトリ>を既存のディレクトリ指定の左に記述してください。**

SHC_INC= <stlport ディレクトリ>;<標準 include のディレクトリ>

のように、セミコロンで区切ることで連続して指定できます。

具体的には、下記のようになります。

```
SHC_INC=C:¥Program Files¥Renesas¥Hew¥EXAMPLES¥STL¥1_0_0¥stlport
      ;C:¥Program Files¥Renesas¥Hew¥Tools¥Renesas¥SH¥9_3_2¥Include
```

環境変数を変更できない環境の場合、コンパイラの include オプションで指定することも可能です。

コンパイラの include オプションで、下記のように指定してください。

-include=<stlport ディレクトリ>

具体的な例として、下記のようなコマンドになります。

```
> shc tp.cpp -cpu=sh2a -include="C:¥Program Files¥Renesas¥Hew¥EXAMPLES¥STL¥1_0_0¥stlport"
```

以上で STL が利用可能になります。

3. complex の利用手順

RXC/SHC には、EC++ライブラリによる complex クラスが実装されています。しかしこれは float_complex クラス および double_complex クラス の実装であり、C++の complex クラステンプレートとは異なるものです。STL 機能を組み込むことにより、C++の complex クラステンプレートを使うことができます。本章では、本 STL ライブラリの complex クラスの利用方法及び制限事項を説明します。

3.1. complex の制限事項

RXC/SHC で complex クラスを使う上での制限事項は、以下になります。

a) complex クラスはテンプレート引数を一つ持ち、実数部および虚数部の精度を指定することが可能ですが、扱える型は下記の 3 つのみです。

- complex<float>
- complex<double>
- complex<long double>

上記以外の型を指定した場合、その動作は言語仕様上未規定です。

b) RXC/SHC の EC++ライブラリに含まれる関数の名前空間はグローバルですが、complex クラステンプレート、および数学関数の名前空間は std です。

```
// complex の宣言には std:: が必要
std::complex<float> f1(4.0f, 3.0f);
// abs(complex)には std:: が必要だが、cout, endl には std:: を付けてはならない
cout << std::abs(f1) << endl;
// RXC/SHC 標準ライブラリには std:: を付けてはならない。
// 従って、abs(long) には std:: をつけてはならない。
cout << abs(-10L) << endl;
```

c) ストリーム入出力は char 型のみです。

ストリーム入出力には、RXC/SHC の持つストリーム入出力ライブラリを使いますが、このライブラリは char 型のみで使用可能となっています。ワイド文字（wchar_t 型）によるストリーム入出力はサポートしていません。

3.2. ソースファイルの組み込み

本 STL ライブラリの complex の演算の一部では、RXC/SHC が持つ数学関数ライブラリ、およびストリーム入出力ライブラリを利用します。また、RXC/SHC が持つ数学関数ライブラリに加え、追加のソースファイルが必要となります。

表 2. complex を使う上で必要なライブラリと、追加のソースファイル

利用する機能	EC++ライブラリ	追加のソースファイル
四則演算、比較	なし	なし
実部および虚部の取得	なし	なし
数学関数	・数値計算用ライブラリ(C99) ・数値計算用ライブラリ(C99)(float 型関数)	・complex.cpp ・complex_trig.cpp
ストリーム入出力	・ストリーム入出力用クラスライブラリ	・complex_io.cpp

追加のソースファイルは、下記のディレクトリに格納されています。
「C:¥Program Files¥Renesas¥Hew¥EXAMPLES¥STL¥1_0_0¥src」

3.2.1. complex の四則演算、比較

complex を四則演算や比較のみで利用するのであれば、追加のソースファイルは不要です。

```
std::complex<float> f1(2.0f, 4.0f), f2(1.0f, 0.0f);
f1 += f2;
```

3.2.2. complex の実部および虚部の取得

complex の実部および虚部を取得するだけであれば、追加のソースファイルは不要です。

```
f1.real();
f1.imag();
```

のように、complex クラスのメンバ関数 real() および imag() を使うことができます。

3.2.3. complex を数学関数に渡す

complex を数学関数に渡し、各種の演算を行う場合、RXC/SHC の持つライブラリと、追加のソースファイルを組み込む必要があります。

・ RXC/SHC の持つライブラリ

図 6を参考に、「math.h」および「mathf.h」にチェックを入れてください。また、RXCの場合、ライブラリ構成は「C99」にしてください。

※RXC において「C99」にしない場合、「L2310 (E) Undefined external symbol "_hypot"」が発生します。

・ 追加のソースファイル

図 7、図 8、図 9を参考に、「complex.cpp」および「complex_trig.cpp」をプロジェクトに追加してください。

以上で、数学関数を使えるようになります。

ただし、complex クラスを扱う数学関数は名前空間 std 内に属するので、

```
std::pow(f1, 5);
```

のように、スコープ解決演算子が必要です。

3.2.4. complex をストリーム入出力に渡す

complex をストリーム入出力に渡し、値の入出力を行う場合、RXC/SHC の持つライブラリと、追加のソースファイルを組み込む必要があります。

・ RXC/SHC の持つライブラリ

図 6を参考に、「ios(EC++)」にチェックを入れてください。

・ 追加のソースファイル

図 7、図 8、図 9を参考に、「complex_io.cpp」をプロジェクトに追加してください。

以上で、ストリーム入出力を使えるようになります。

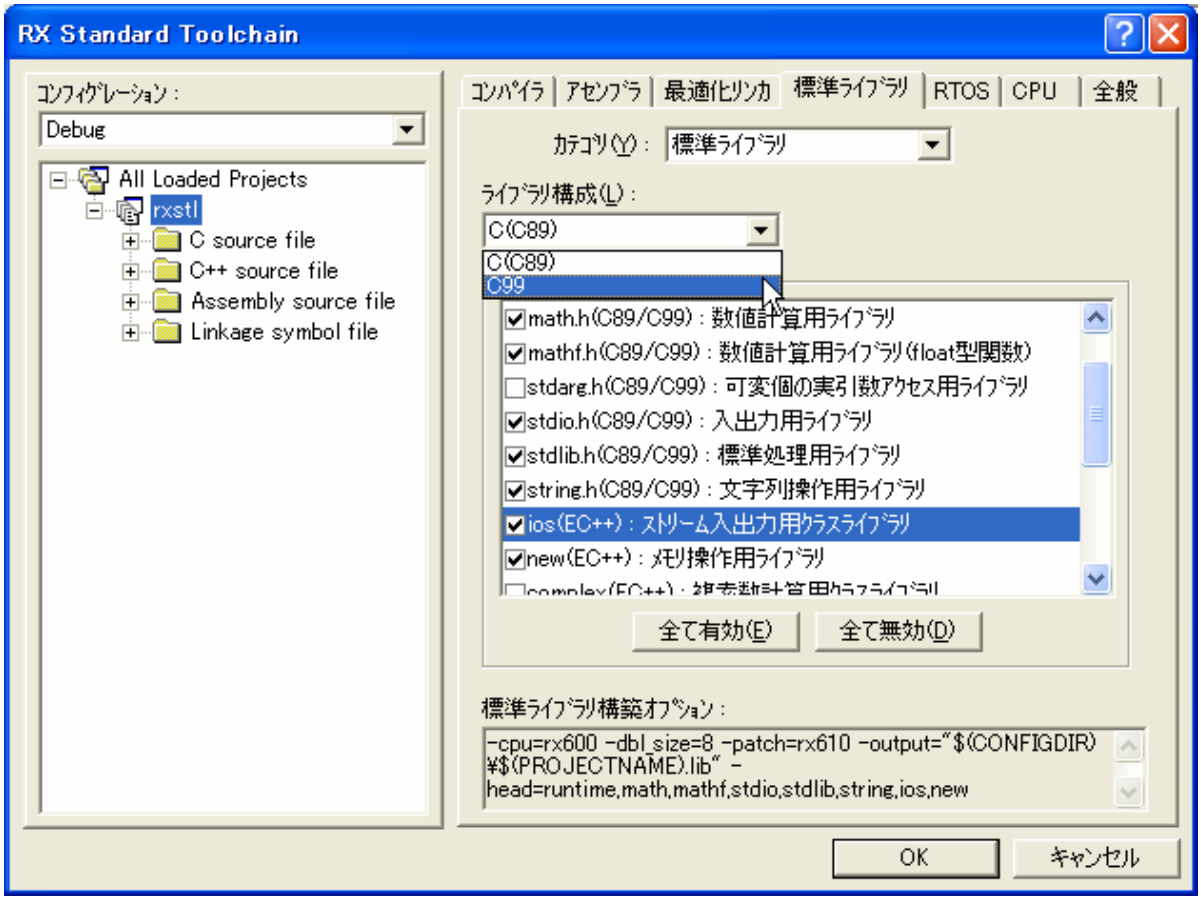

図 6. RXC/SHC の持つライブラリ

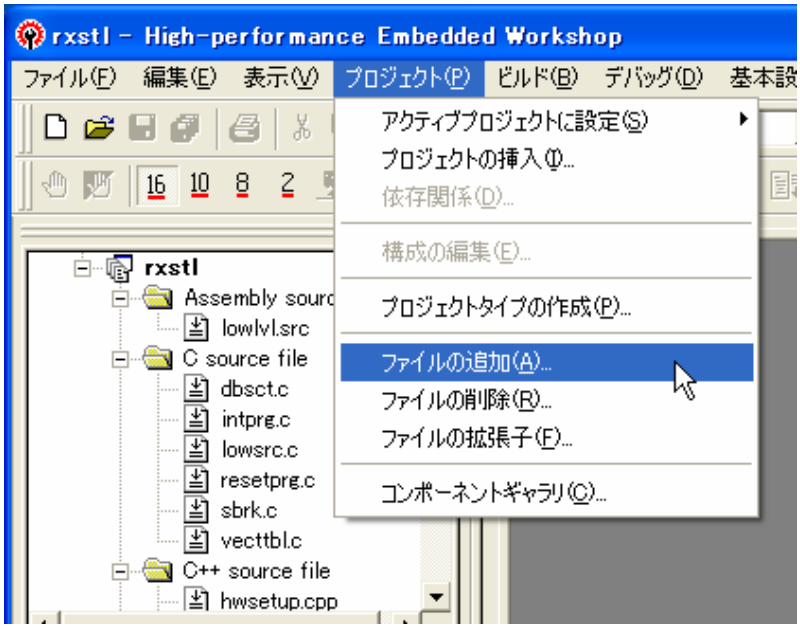

図 7. プロジェクトへのファイルの追加

図 8. 追加するソースファイルの選択

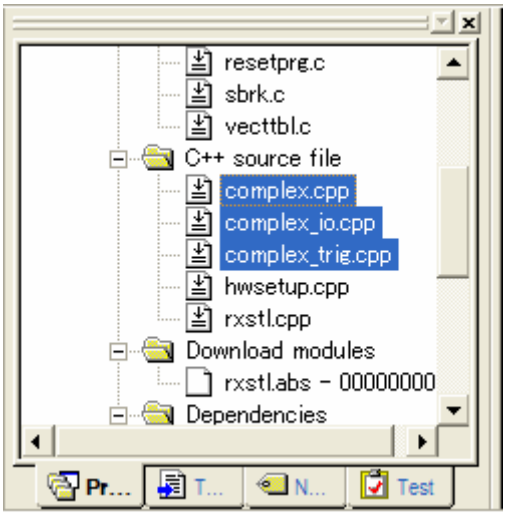

図 9. ソースファイル追加後のプロジェクトファイル一覧

4. ワイド文字の利用手順

ワイド文字(wstring, wchar_t)を使用する場合は、図 10のように、ライブラリ構成は「C99」にし、「wchar.h(C99)：ワイド文字入出力ライブラリ」をチェックしてください。

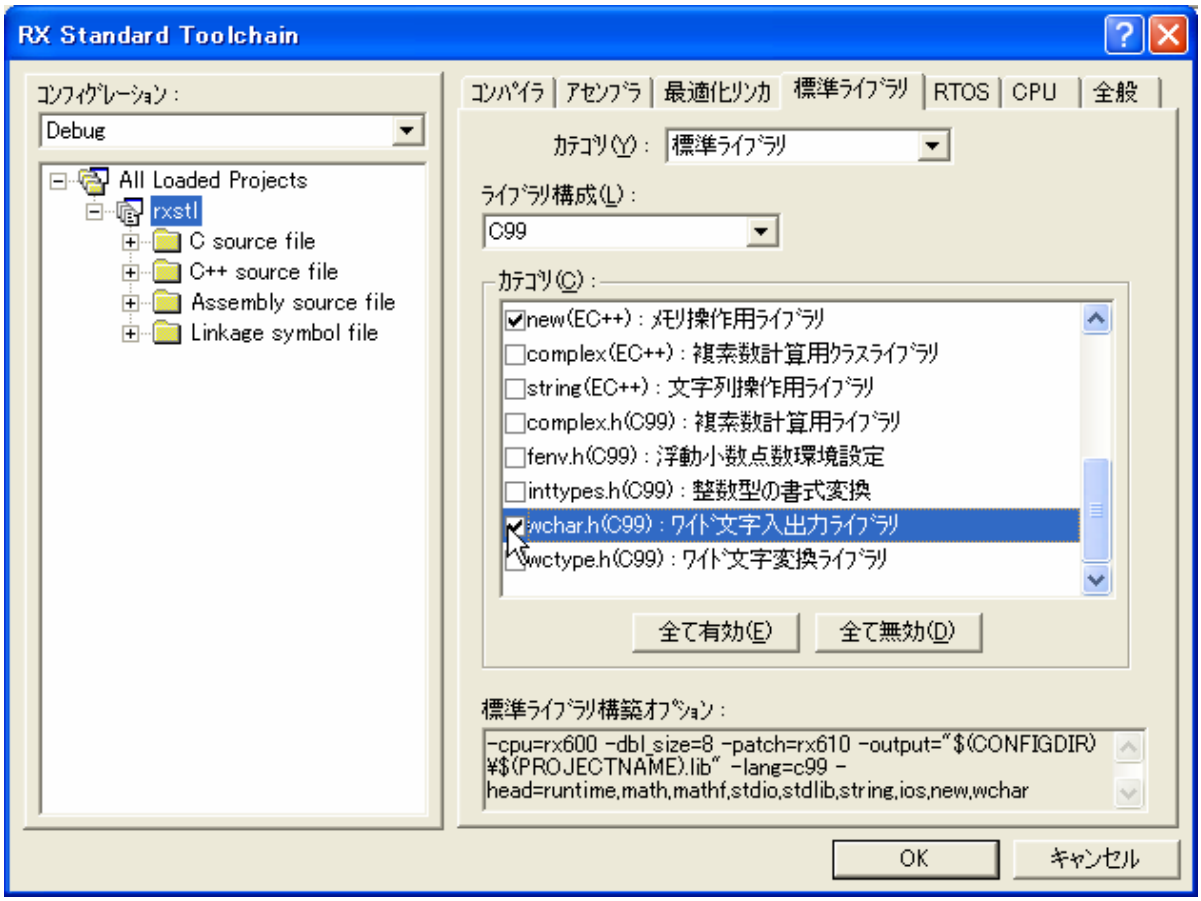

図 10. ワイド文字を使用する場合のライブラリ指定

5. 例外クラス（exception, stdexcept）の利用手順

SHC /RXC で例外クラス（exception, stdexcept）を利用するには、本章に記す以下の手順を全て実行する必要があります。

5.1. コンパイラオプションの設定

SHCは、図 11, RXCは、図 12のように、「C++のtry、throw、catchを有効にする」、「C++のdynamic_cast,typeidを有効にする」をチェックしてください。

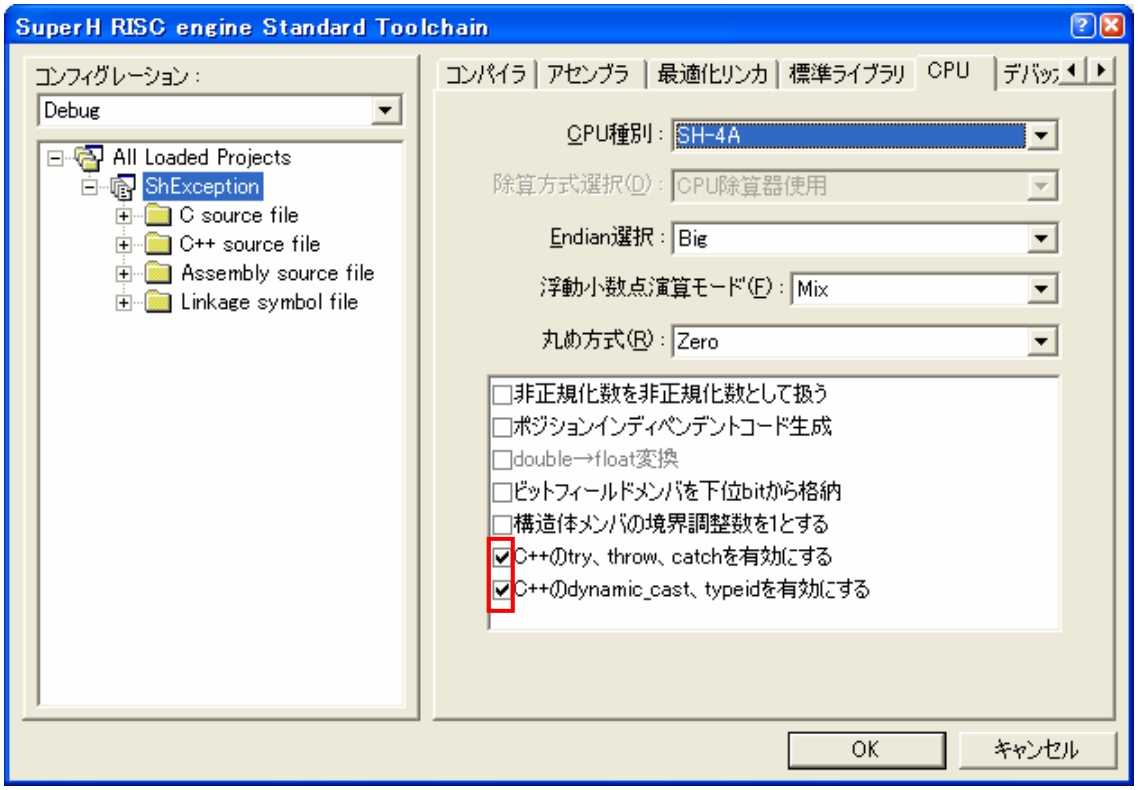

図 11. SHC 例外クラス使用時のオプション設定

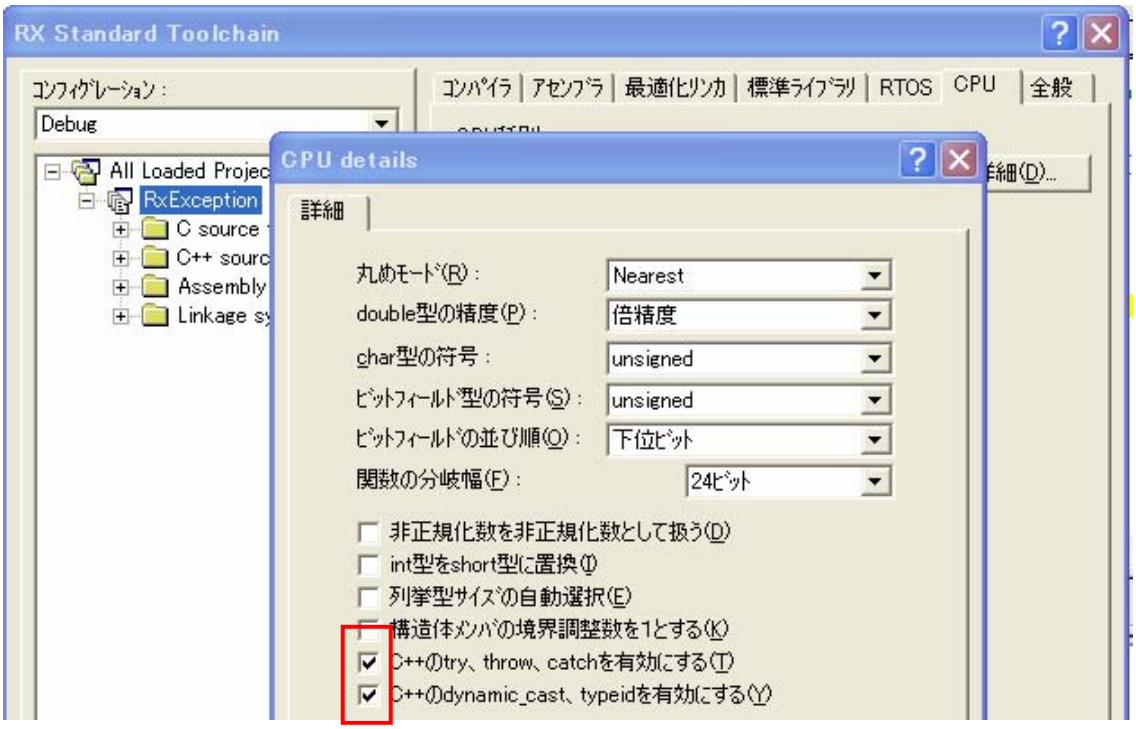

図 12. RXC 例外クラス使用時のオプション設定

5.2. 最適化リンカオプションの設定

SHC /RXCともに、図 13のように「プレリンカ制御」を「使用」にします。

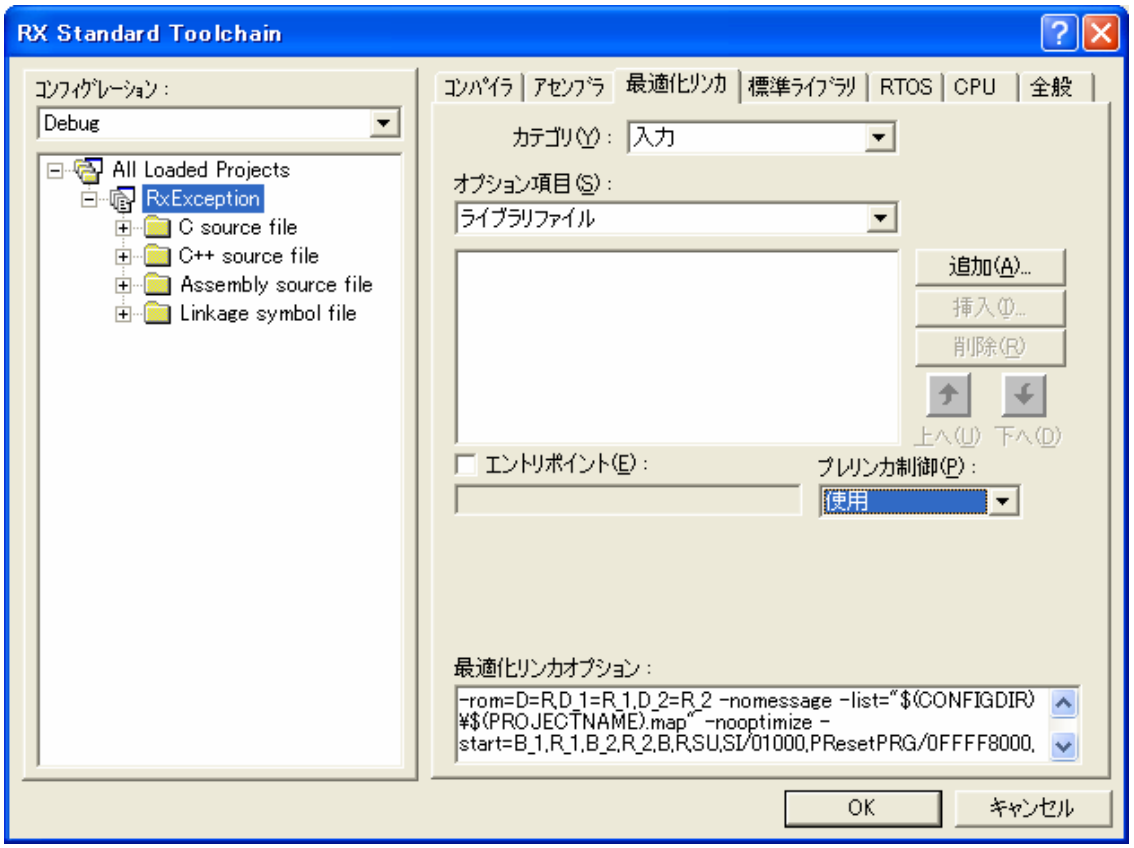

図 13. 例外クラス使用時の最適化リンカオプションオプション設定

5.3. 標準ライブラリの設定

SHCは図 14のように、その他カテゴリのその他オプションから、「EC++言語に基づいたチェック」のチェックを外してください。RXCはこのオプションがありませんので、標準ライブラリの設定は不要です。

図 14. 例外クラス使用時の、標準ライブラリその他のオプション設定(SHC)

5.4. ソースファイルの組み込み

SHC /RXCともに、図 15のように提供ソース「stdexcept_base.cpp」をプロジェクトに追加してください。
追加のソースファイルは、下記のディレクトリに格納されています。
「C:¥Program Files¥Renesas¥Hew¥EXAMPLES¥STL¥1_0_0¥src」

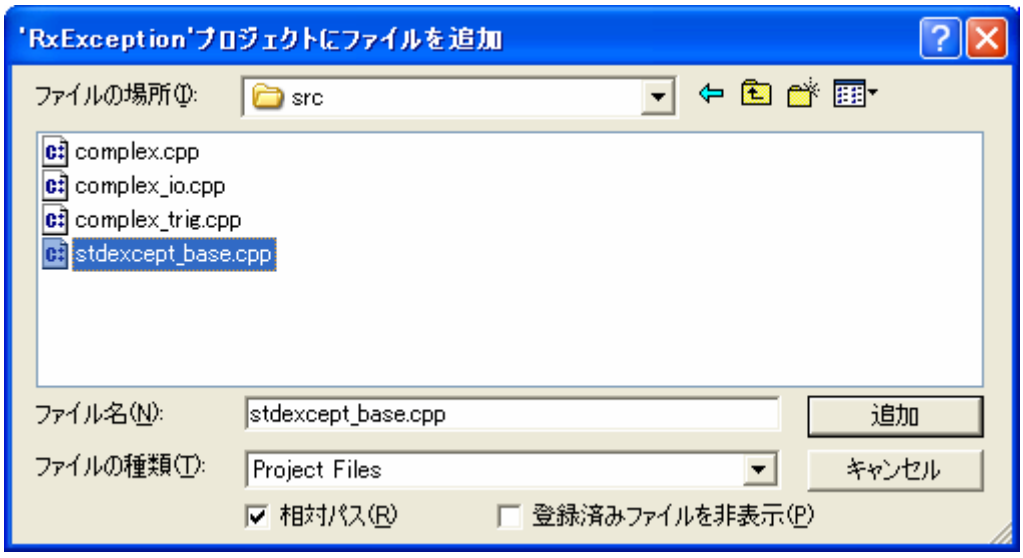

図 15. 例外クラス使用時の提供ソース追加

図 16のように「stdexcept_base.cpp」がワークスペースに追加されていることを確認してください。

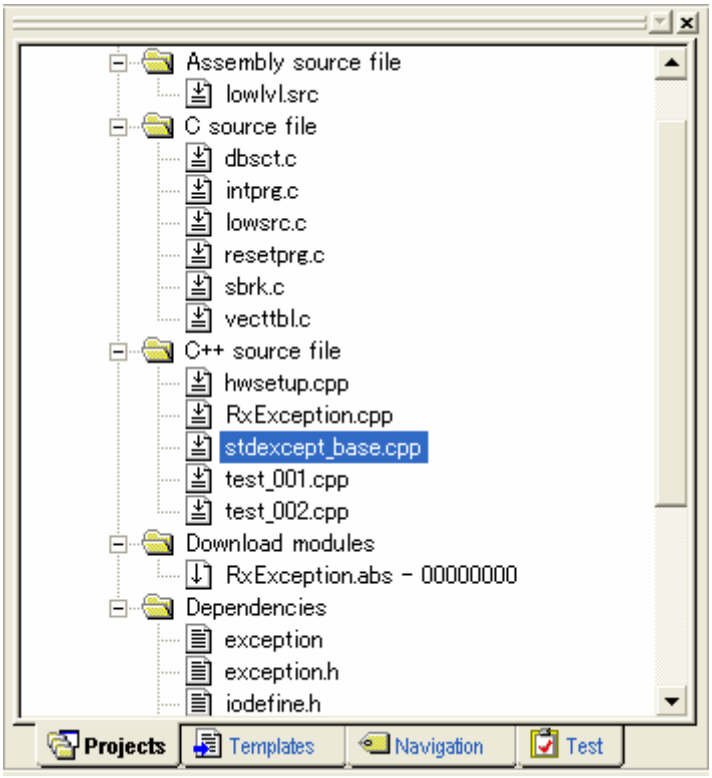

図 16. 例外クラス使用時の提供ソース追加後のワークスペース表示

6. string の利用手順

6.1. ストリーム入出力処理利用時の必須事項

string と、RXC/SHC の iostream を組み合わせて利用する場合、RXC/SHC が提供する標準ライブラリのうち、下記ライブラリを組み込む必要があります。

- ios(EC++) :ストリーム入出力用クラスライブラリ
- ctype.h： 文字操作用ライブラリ

また、下記のソースファイルをユーザプロジェクトに組み込む必要があります。

- string_io.cpp

6.2. ソースファイルの組込み

ストリーム入出力処理利用時は、図 17、図 18、図 19を参考に、「string_io.cpp」をプロジェクトに追加してください。

追加のソースファイルは、下記のディレクトリに格納されています。

「C:¥Program Files¥Renesas¥Hew¥EXAMPLES¥STL¥1_0_0¥src」

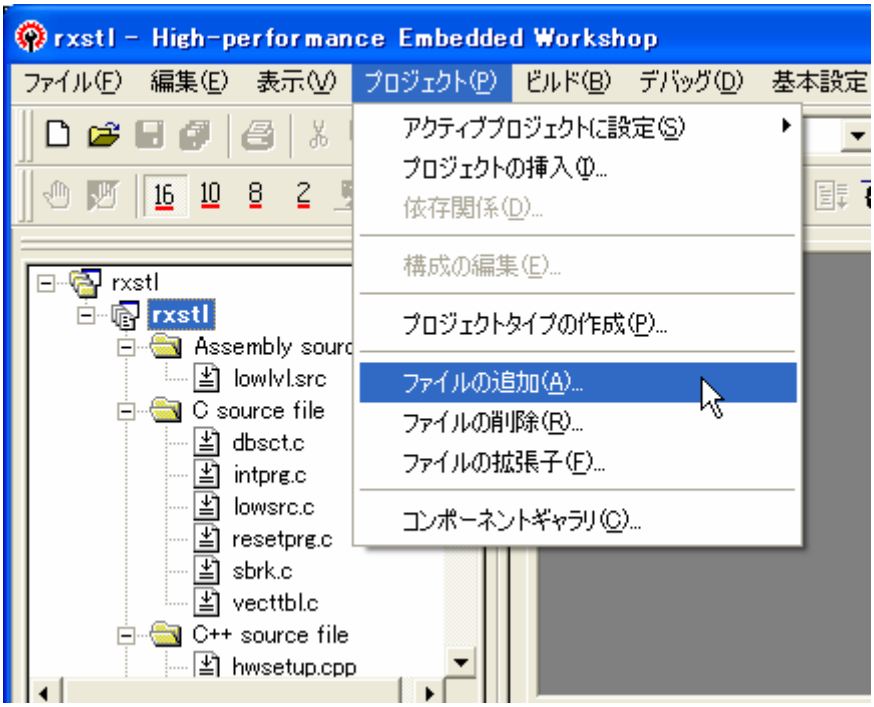

図 17. プロジェクトへのファイルの追加

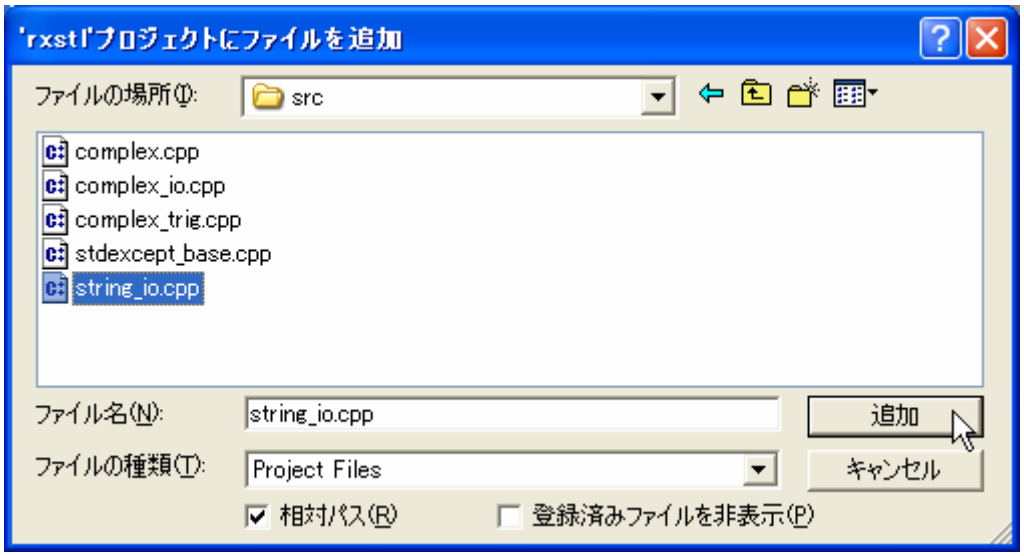

図 18. 追加するソースファイルの選択

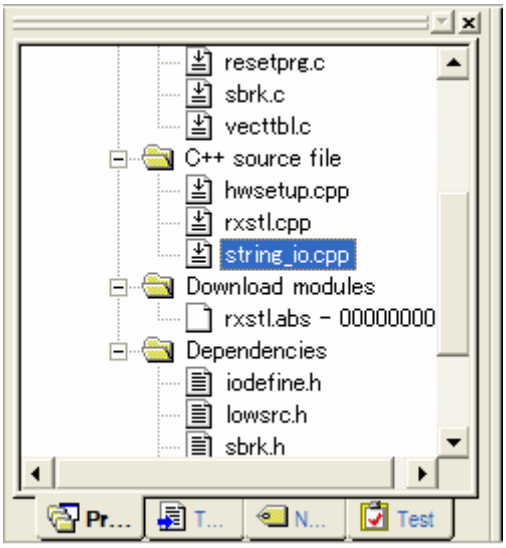

図 19. ソースファイル追加後のプロジェクトファイル一覧

6.3. ストリーム入出力処理利用時の制限事項

RXC/SHC には、EC++ライブラリによる標準ストリーム入出力 cin, cout が定義されています。
ただし、標準 C++に規定されるものと比較し、以下の違いがあります。

表 3 STL と EC++の、標準ストリーム入出力の違い

	標準 C++	EC++
名前空間	std	グローバル
標準ストリーム入出力	cin, cout, cerr, clog wcin, wcout, wcerr, wclog	cin, cout

たとえば「std::cout << str1;」のように記述した場合、コンパイルエラーとなります。
この場合は、「::cout << str1;」と記述しなければなりません。

EC++ライブラリでは、cin, cout をサポートしていますが、cerr, clog は非サポートです。

また、RXC が持つ標準 C ライブラリはワイド文字をサポートしていますが、
EC++ライブラリにおいては、ワイド文字は非サポートです。
従いまして、ワイド文字入出力のための wcin, wcout も非サポートです。
「std::wcout << wstr1;」のように記述していた場合、下記のような代替案を
ご検討ください。
・wprintf()を用いた出力に切り替える
・wstring 型の文字列を string 型に変換した上で、cout/cin を使う

7. 注意事項

7.1. グローバルクラスオブジェクトの初期処理と後処理 (SHC/RXC)

C++言語でグローバルクラスオブジェクトを使用する場合、初期処理関数(_CALL_INIT)と後処理関数(_CALL_END)を main 関数の前後で呼び出す必要があります。

これは、グローバルなクラスオブジェクト宣言は、関数を実行しても宣言が実行される事がないため、明示的に該当クラスのコンストラクタを呼び出す初期処理関数(_CALL_INIT)と、デストラクタを呼び出す後処理関数(_CALL_END)を呼び出す必要があるためです。

High-performance Embedded Workshopでプロジェクト生成時に、スタートアップルーチンを生成し、main関数をCソースで生成するよう指定した場合は(図 20)、resetprg.c内で初期処理関数(_CALL_INIT)と後処理関数(_CALL_END)はコメントアウトされているので、必要な場合にはコメントアウトを解除してください。

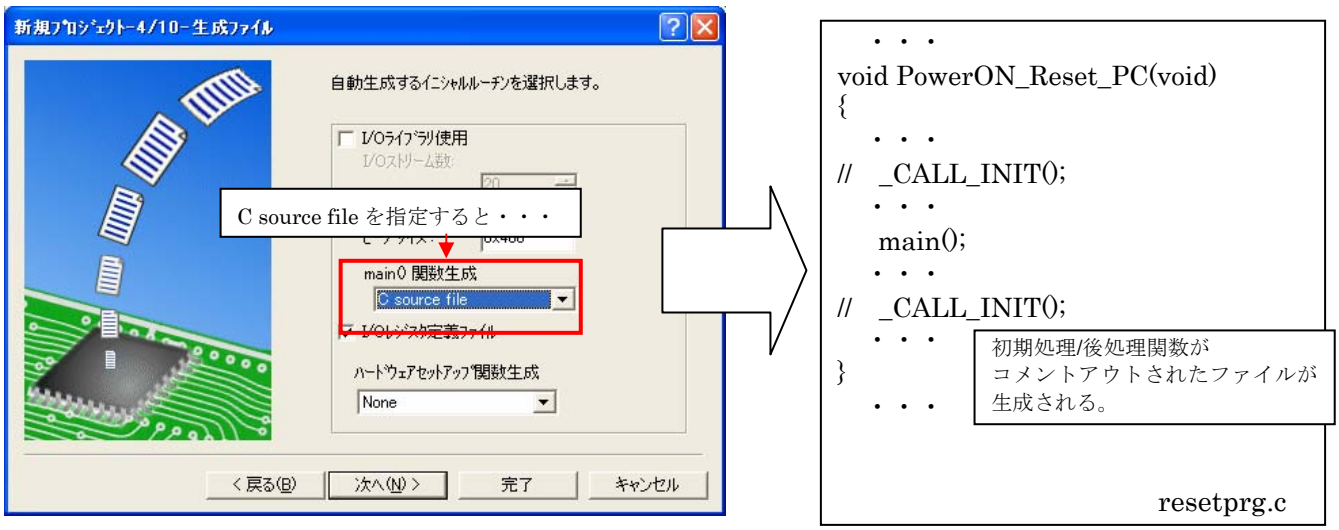

図 20. プロジェクト生成時の注意シーン

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

http://japan.renesas.com/

お問合せ先

http://japan.renesas.com/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2010.07.30	—	初版発行
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

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/inquiry

Colophon 1.0