お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

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。

2003/07/03

[Q]
リンク時に、「L2310 Undefined external symbol “xxxx” referenced in “yyyy”」**が出力される。**

[A]

１．ユーザが作成したシンボル（関数、変数）の場合

１．１　C 言語プログラムから C++言語プログラムの関数をコールしている場合

C++言語で関数定義する際、「extern “C”」を追加して下さい。ただし、「extern “C”」宣言した関数は多重定義できません。

例

C プログラム側

```
void foo(void);
void main(void)
{
    foo();
}
```

C++プログラム側

```
extern “C” void foo(void);
void foo(void)
{
}
```

１．２　C++言語プログラムから C 言語プログラムの関数をコールしている場合

C++言語で関数のプロトタイプ宣言をする際、「extern “C”」宣言を追加して下さい。ただし、「extern “C”」宣言した関数は多重定義できません。

例

C プログラム側

```
void foo(void)
{
}
```

C++プログラム側

```
extern “C” void foo(void);
void main(void)
{
    foo();
}
```

１．３　C/C++言語プログラムからアセンブリプログラムの関数をコール、あるいは、変数の参照を行っている場合

アセンブリプログラムにて、「.EXPORT」制御命令を使用しシンボル名（先頭に下線"_"を付加）を外部定義宣言して下さい。また、C++言語プログラムの関数をコールする場合には、「１．１」、「１．２」と同様、C++関数に「extern "C"」を付加して下さい。

例(1)…変数参照

アセンブリプログラム側

```
      .EXPORT    _a
      .SECTION   D,DATA
_a:   .DATA.L    0
      .END
```

C プログラム側

```
extern int a:
void main(void)
{
    a = 1;
}
```

例(2)…関数コール

アセンブリプログラム側

```
      .EXPORT    _foo
      .SECTION   P,CODE
_foo:
      RTS
      NOP
      .END
```

C プログラム側

```
extern void foo(void);
void main(void)
{
    foo();
}
```

１．４　アセンブリプログラムから C/C++言語プログラムの関数をコール、あるいは、変数の参照を行っている場合

アセンブリプログラムにて、「.IMPORT」制御命令を使用しシンボル名（先頭に下線"_"を付加）を外部参照宣言して下さい。また、C++言語プログラムの関数をコールする場合には、「１．１」、「１．２」と同様、C++関数に「extern "C"」を付加して下さい。

例(1)…変数参照

SH の例　アセンブリプログラム側

```
      .IMPORT    _a
      .SECTION   P,CODE
_main:
      MOV.L      A_a, R1
      MOV.L      @R1, R0
      ADD        #1,  R0
      RTS
      MOV.L      R0,  @R1

      .ALIGN     4
A_a:  .DATA.L    _a
      .END
```

C プログラム側

```
int a;
```

H8 の例　アセンブリプログラム側

```
      .IMPORT    _a,_b
      .SECTION  P,CODE
_main:
      MOV.B     @_a, R5L
      MOV.B     @R5L, @_b
      RTS
      .END
```

C プログラム側

```
char a,b;
```

例(2)…関数コール

SH の例　アセンブリプログラム側

```
      .IMPORT    _foo
      .SECTION  P,CODE
_main:
      STS.L     PR, @-R15
      MOV.L     A_foo, R0
      JSR       @R0
      NOP
      LDS.L     @R15+, PR
      RTS
      NOP

      .ALIGN    4
A_foo: .DATA.L   _foo
      .END
```

C プログラム側

```
void foo(void)
{
}
```

H8 の例　アセンブリプログラム側

```
      .IMPORT    _foo
      .SECTION  P,CODE
_main:
      JSR       @_foo
      RTS
```

C プログラム側

```
void foo(void)
{
}
```

2．ユーザが作成していないシンボル（関数、変数）の場合

2．1　組込み関数を使用している場合

組込み関数を使用するソースファイル内で、必ず<machine.h>をインクルードして下さい。

例

```
#include <machine.h>

void main(void)
{
    nop();
}
```

２．２　標準ライブラリを使用している場合

標準ライブラリを使用するソースファイル内で、必ず使用するライブラリのヘッダファイルをインクルードして下さい。

例

```
#include <math.h>

void main(void)
{
    sin(1.0);
}
```

また、必ず標準ライブラリをリンクして下さい。

(1) 標準ライブラリのリンク方法（コマンドライン使用時）

lbgsh.exe（SH の場合）、または、lbg38.exe（H8 の場合）を使用し、標準ライブラリファイル（*.lib）を作成して下さい。

例

```
> lbgsh.exe –cpu=sh4
> lbg38.exe –cpu=2000a
```

また、リンク時に lbgsh.exe または lbg38.exe で作成した標準ライブラリファイルを追加するよう、-library オプションを指定して下さい。

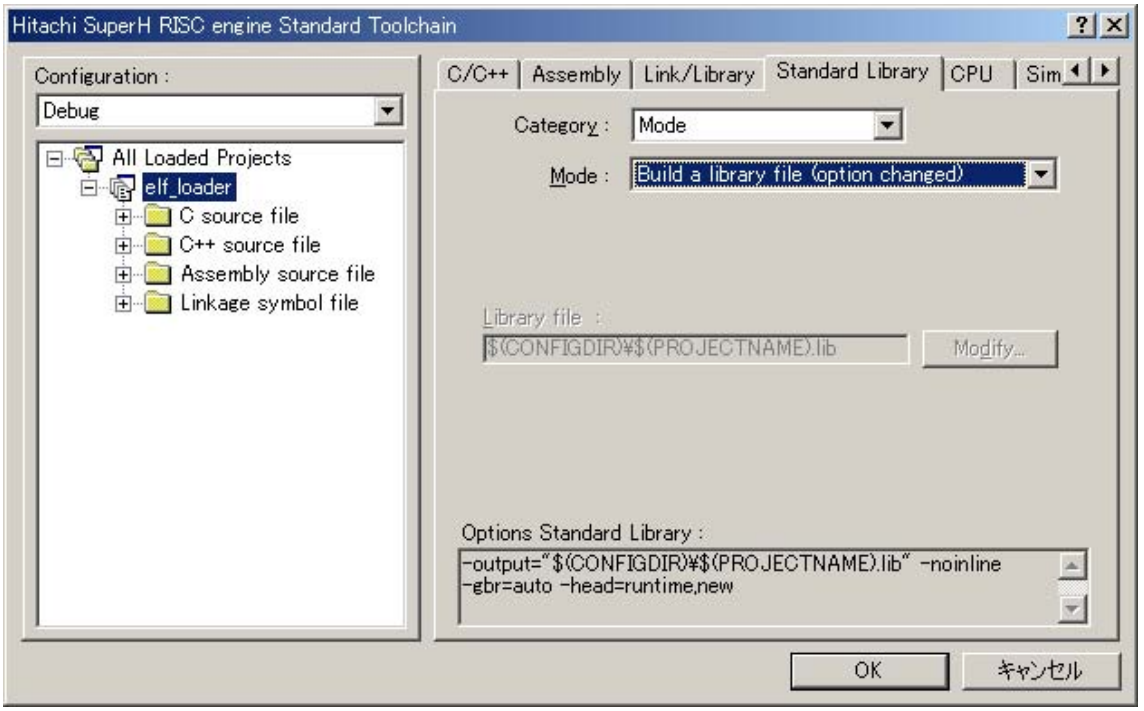
例

```
> optlnk.exe *.obj –library=stdlib.lib
```

(2) 標準ライブラリのリンク方法（HEW 使用時）

下記ダイアログにて、標準ライブラリを構築するよう設定して下さい。

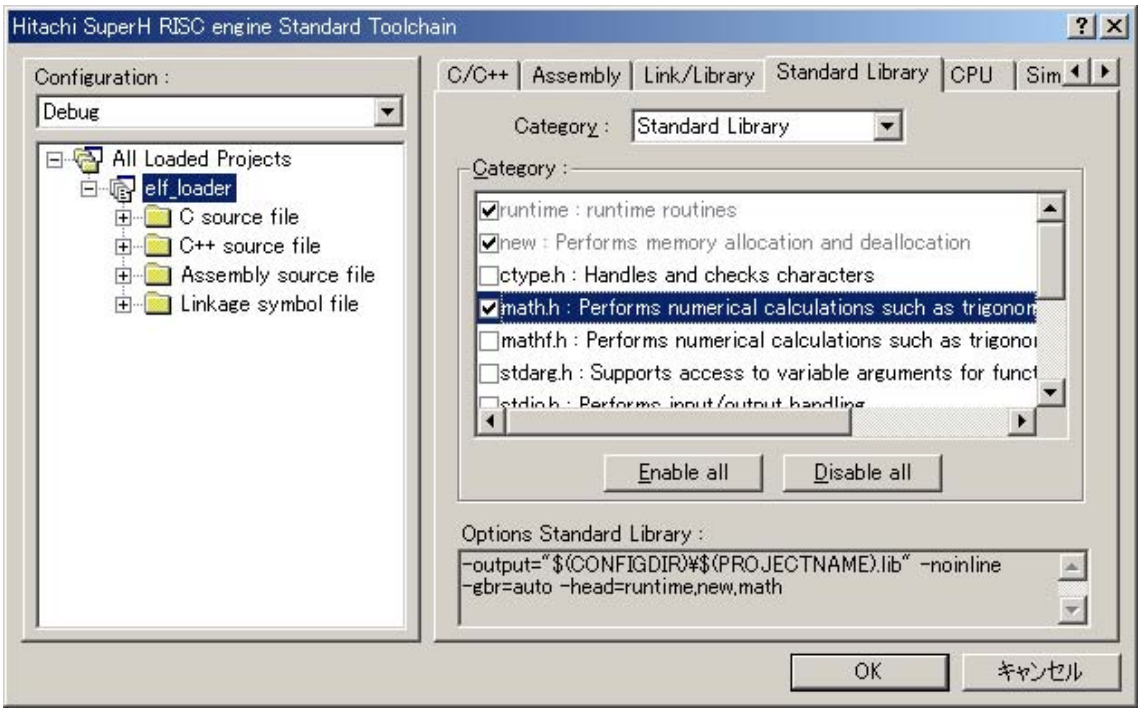
HEW メニュー：[オプション(Options) → *** Standard Toolchain]
→ [Standard Library]タブ
→ [Category:]に”Mode”を選択
→ [Mode:]に”Build a library file (Option changed)”または、”Build a library file (anytime)”を選択して下さい。



また、同タブにて、

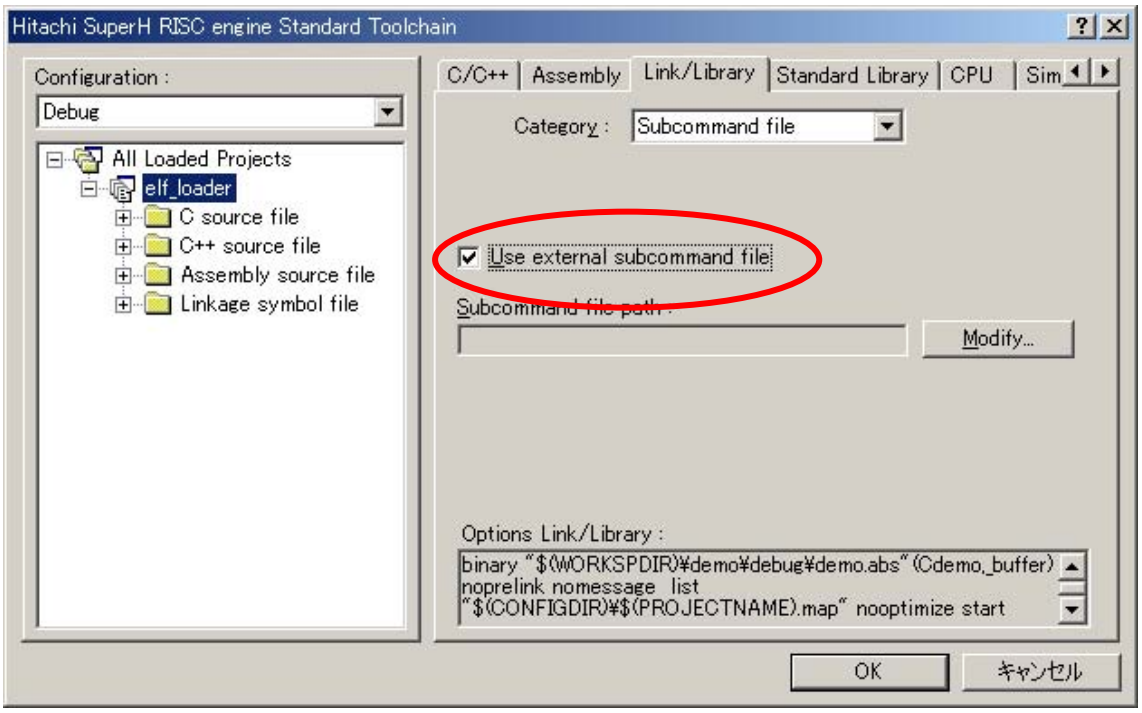
→ [Category:]**に”**Standard Library**”を選択**

→ [Category:]**にて、ヘッダファイル単位で、必要なモジュールにチェックを付けて下さい。**



なお、同ダイアログの[Link/Library]**タブにて、サブコマンドファイルを使用するように指定している場合（下記がチェックされている場合）には、サブコマンドファイルに**-library **を使用して標準ライブラリをリンクするようにして下さい。サブコマンドファイルを使用しない場合（下記**

がチェックされていない場合）には、標準ライブラリは自動的にリンクされます。



2．3　“xxxx”が open、close、read、write、lseek、sbrk、errno_addr、wait_sem、signal_sem の何れかの場合

これらのシンボルは低水準インターフェースルーチンと呼ばれ、標準入出力、メモリ管理ライブラリを使用したときに必要になる関数です。例えば、標準出力と言っても、その出力先は LCD、LED、HDD、FDD、プリンタ、CD-R/RW など様々であり、その機器により操作は異なります。従って、低水準インターフェースルーチンはコンパイラでは提供しておらず、ユーザにて実装頂く必要があります（ただし、HEW ご使用の場合はサンプルとして生成する事も可能です）。実装方法はコンパイラのユーザーズマニュアルをご参照下さい。なお、errno_addr、wait_sem、siglan_sem は、SHC コンパイラ Ver.7 以降で、かつ、標準ライブラリ構築時に-reent を指定した場合（リエントラントライブラリを指定した場合）に必要になる関数です。