For M16C Series, R8C Family C compiler Package

V.5.45 Release 01

Guidebook

(Rev.2.00)

Renesas Solutions Corporation

Jun 1, 2010

Abstract

This document provides a guide to the introduction of M16C Series, R8C Family C compiler Package V.5.45 Release 01. When you install this software package or create a project or want to know about the compiler, please refer to this guidebook.

A. Guidebook for V.5.45 Release 01

This section describes the precautions to be taken when you upgrade the projects you created with old versions of the compiler to V.5.45 Release 01, and the points to be noted when you create new projects with V.5.45 Release 01.

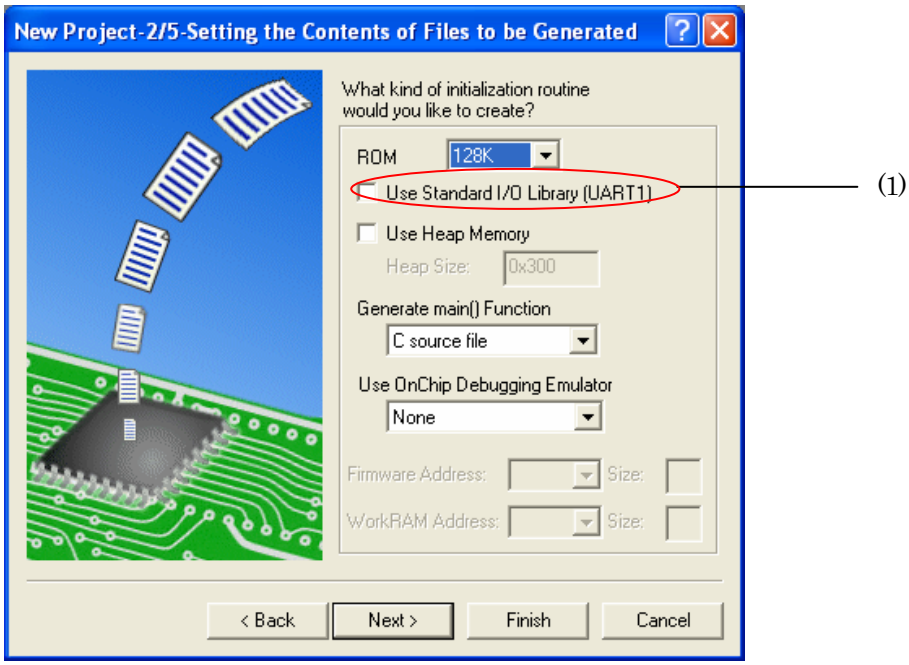
A.1. Points to be noted when upgrading V.5.30 Release 02 or earlier versions of the compiler to V.5.45 Release 01

A.1.1. Modify startup file

・ _init() function

Beginning with V.5.40 Release 00(A), the name of the library function init() has been changed to _init().
Therefore, if you attempt to build without modification, an error message *'_init' value is undefined* may be generated during a link process. This error occurs in the following cases:

・ If you created the project with an old version prior to V.5.40 Release 00, select the check box (1)
・ When init function calls are enabled by altering ncrt0.a30 directly



If this error occurs, alter a part of ncrt0.a30 that is shown below

[When you are using the startup file (ncrt0.a30) supplied with the compiler]

Before modification

```
;======================================================
; Initialize standard I/O
;------------------------------------------------------------
.if __STANDARD_IO__ == 1
        .glb    _init
        .call   _init,G
        jsr.a   _init
.endif
```

After modification

```
;======================================================
; Initialize standard I/O
;------------------------------------------------------------
.if __STANDARD_IO__ == 1
        .glb    __init
        .call   __init,G
        jsr.a   __init
.endif
;------------------------------------------------------------
```

[When you are using the startup file (crt0mr.a30) supplied with the Real Time OS]

For M3T-MR308

Before modification

```
; +-------------------------------------------------+
; |   User Initial Routine ( if there are )         |
; +-------------------------------------------------+
; Initialize standard I/O
        .GLB    _init
        JSR.A   _init
```

After modification

```
; +-------------------------------------------------+
; |   User Initial Routine ( if there are )         |
; +-------------------------------------------------+
; Initialize standard I/O
        .GLB    __init
        JSR.A   __init
;-------------------------------------------------
```

For M3T-MR30

* In several versions including the latest version (V.3.30 Release 2), the jump process shown below is 'commented out.'

Before modification

```
;=============================================================================
; Initialize standard I/O
;-----------------------------------------------------------------------
        .glb    _init
        jsr.a   _init
```

After modification

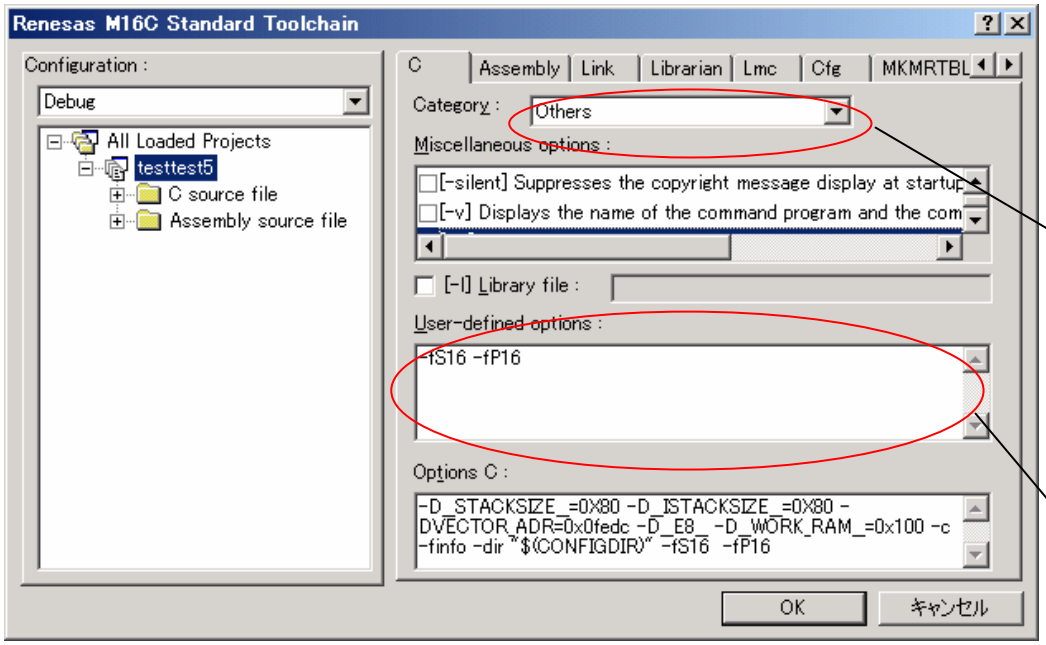
```
;=============================================================================
; Initialize standard I/O
;-----------------------------------------------------------------------
        .glb    __init
        jsr.a   __init
```

A.1.2. Change size of size_t, ptrdiff_t

Beginning with this version, size_t and ptrdiff_t have been changed in size from 16 bits to 32 bits.
If you need to use size_t and ptrdiff_t in 16 bits because you are using a size_t and ptrdiff_t type based user library created with an old version, for example, make the following settings.

- Set the compile options -fsize_t16 (-fS16) and -fptrdiff_t (-fP16).
- Change the libraries to be linked from nc30lib.lib and r8clib.lib to nc30s16.lib and r8cs16.lib when you're using NC30WA, or from nc308lib.lib and nc382lib.lib to nc308_16.lib and nc382_16.lib when you're using NC308WA.

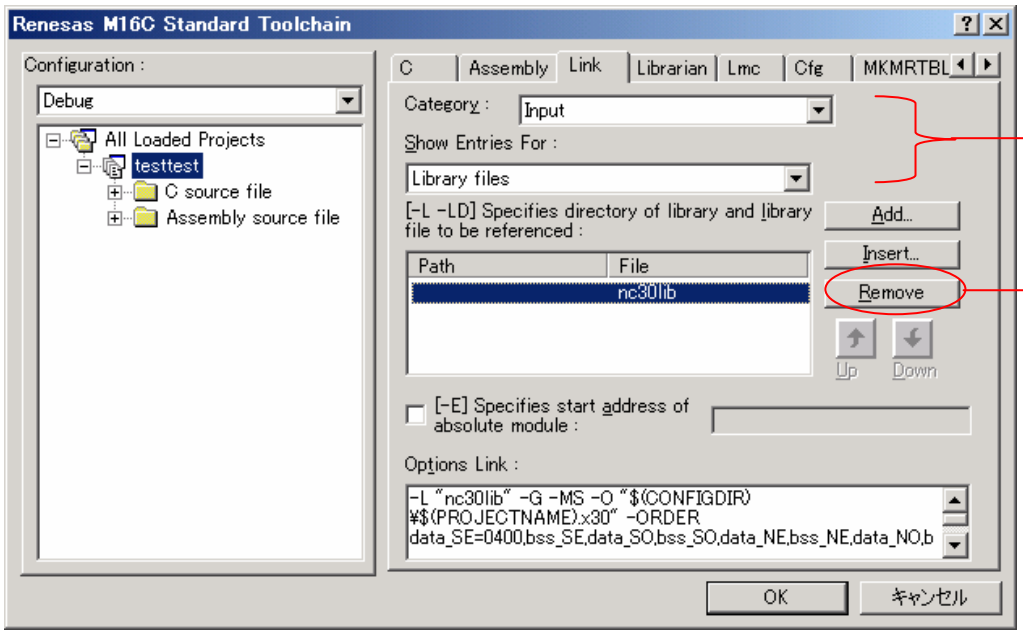
[Procedure for setup in HEW]
Set the compile options -fsize_t16 (-fS16) and -fptrdiff_t (-fP16).
From the Build menu of HEW, select [Renesas M16C Standard Toolchain] -> C tab.



(1)For Category,select Other
(2)For User-Defined Options, enter -fsize_t16 (or -fS16) and -fptrdiff_t16 (or -fP16).

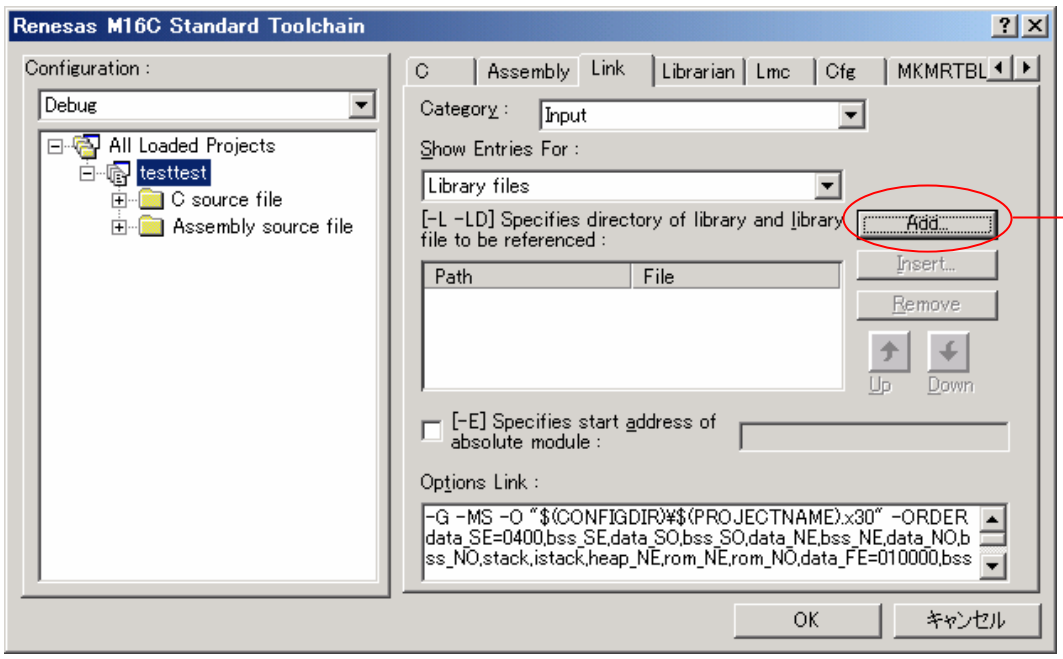
Change the libraries to be linked.

From the Build menu of HEW, select [Renesas M16C Standard Toolchain] -> Link tab.



(3) For Category,select Input
For Show Entries-For select Library files.

(4) Click the Remove button to remove nc30lib.lib temporarily.
Remove r8clib.lib when you're using R8C/Tiny.
Remove nc308lib.lib when you're using M16C/8X
Remove nc382lib.lib when you're using M32C/8X



(5) Click the Add button to select [Library files].

(6) Enter a library name usable for size_t and ptrdiff_t in 16-bit size.
Input nc30s16.lib when you're using M16C
Input r8cs16.lib when you're using R8C/Tiny.
Input nc308_16.lib when you're using M16C/8X
Input nc382_16.lib when you're using M32C/8X
Click the OK button to finish.

[When you're using the makefiles generated by the configurator of the real-time OS]
Correct the following part of statements in the makefile. The following shows an example for the case where nc308_16.lib is linked.

Before modification

```
# Use the following macro when you use C-libraries for M32C/80 series.
#NEWLIB          = -l nc382lib
```

After modification

```
# Use the following macro when you use C-libraries for M32C/80 series.
#NEWLIB          = -l nc308_16.lib
```

[when you're using M3T-MR30]
Since the "LIBS" macro in the makefile is rewritten to "nc30lib.lib" by the configurator, the problem cannot be solved by correcting the "LIBS" macro.
Therefore, correct a process during "$(LINKLIST)" generation to solve the problem.
Correct the following part of statements in the makefile.

Before modification

```
$(LINKLST): makefile
    @mrecho "-o $(PROGRAM)" $(LINKLST)
    @mrecho -a "-ld $(LIB30)" $(LINKLST)
    @mrecho -a "-l $(LIBS)" $(LINKLST)
```

After modification

```
$(LINKLST): makefile
    @mrecho "-o $(PROGRAM)" $(LINKLST)
    @mrecho -a "-ld $(LIB30)" $(LINKLST)
    @mrecho -a "-l nc30s16.lib -l $(LIBS)" $(LINKLST)
```

A.1.3. Interrupt vector table

V.5.40 Release 00 and later versions of the compiler automatically generates a vector table when an interrupt function is declared with specification of a vector number.
Thus it is no longer necessary to specify the -fmake_vector_table (-fMVT) option. Since the vector table is now managed in section **vector**, setting of section **__NC_rvector** is also not required.

If the "Can't generate automatically the variable interrupt vector table" error is reported at the time of linkage, modify the variable-vector-table setting as shown below.

Before modification

```
.if     __MVT__==0
;------------------------------------------------------------
; variable vector section
;------------------------------------------------------------
        .section    vector,ROMDATA  ; variable vector table
        .org        VECTOR_ADR

.if     M60TYPE == 1
        .lword      dummy_int               ; vector 0 (BRK)
        .lword      dummy_int               ; vector 1
                :
            (omitted)
                :
        .lword      dummy_int               ; vector 62
        .lword      dummy_int               ; vector 63
.else ; __MVT__

        .section    __NC_rvector,ROMDATA
        .org                VECTOR_ADR

.endif ; __MVT__
```

After modification

```
        .section    vector,ROMDATA  ; variable vector table
        .org        VECTOR_ADR
```

A.1.4. Special page table

V.5.40 Release 00 and later versions of the compiler automatically generates a vector table when **#pragma SPECIAL** is declared with specification of a vector number.
Thus it is no longer necessary to specify the -fmake_special_table (-fMST) option. Since the special table is now managed in section **svector** that is automatically created at the time of assembly, setting of section **__NC_svector** and specification of a vector number with macro definition **SPECIAL** are also not required (these are now prohibited).

If the "Can't generate automatically the variable interrupt vector table" error is reported at the time of linkage, modify the special-page-table setting as shown below.

Before modification

```
.if         __MST__ == 0
;=============================================================
; fixed vector section
;-------------------------------------------------------------
.if          SVECTOR_ADR < 0fffdcH
             .section   svector,ROMDATA          ; specialpage vector table
             .org                SVECTOR_ADR
.endif
;=============================================================
; special page defination
;-------------------------------------------------------------
;            macro is defined in ncrt0.a30
;            Format: SPECIAL number
;
;-------------------------------------------------------------
;            SPECIAL 255
;            SPECIAL 254
                  :
               (omitted)
                  :
;            SPECIAL 19
;            SPECIAL 18
.else
.if          (SVECTOR_ADR < 0fffdcH)
             .section   __NC_svector,ROMDATA
             .org                SVECTOR_ADR
.endif
.endif ; __MST
```

After modification

All of the code shown above must be deleted.

A.2. Points to be noted when creating a new project

A.2.1. Select CPU

When creating a new project you can select the type of microcomputer in CPU Group. However, the selection of microcomputer types is enabled when

- Registration of the sfr header file
- Registering the variable vector interrupt table entry function registration file (intprg.c) to the workspace
- Link address settings

A.2.2. To create a new workspace with a microcomputer that is not listed in CPU Group

[In the case of R8C/Tiny series]

(1) Choose R8C/Tiny from CPU Series.
(2) Choose Other from CPU Group.

When you create a new project using V.5.45 Release 01, please check whether the compile option and the library file suit the ROM size of the microcomputer type you use on Renesas M16C Standard Toolchain Dialog Box of HEW.
From the Build menu of HEW, select [Renesas M16C Standard Toolchain]

ROM size	compile option	library
Less than 64 Kbytes	-R8C	r8clib.lib
64 Kbytes or more	-R8CE	nc30lib.lib

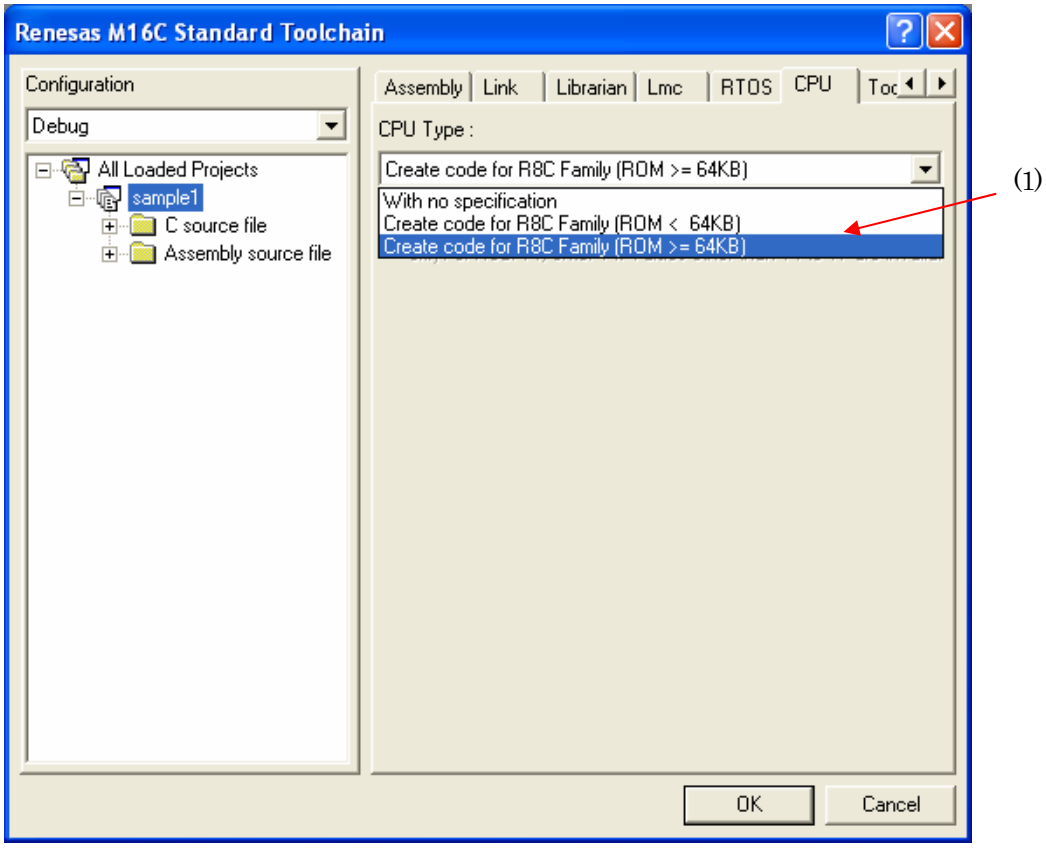
When you create a new project using V.5.43 Release 00 or earlier, please make a change shown below.

If the ROM space of the microcomputer type you use exceeds the 64 Kbyte boundary, make the following settings.

(3) Change the compile option from -R8C to -R8CE.
(4) Change the library to link from r8clib.lib to nc30lib.lib.

Change the compile option from -R8C to -R8CE.

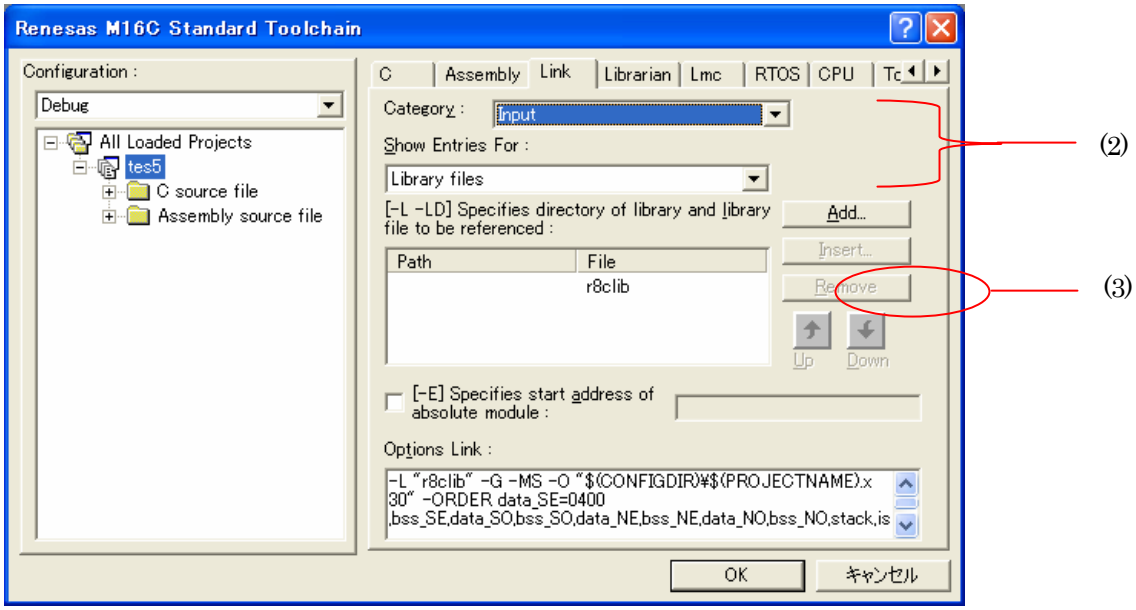
From the Build menu of HEW, select [Renesas M16C Standard Toolchain] -> CPU tab.



(1) From the CPU Type pulldown menu, select Create code for R8C Family (ROM >= 64KB).

Change the linked library from r8clib.lib to nc30lib.lib

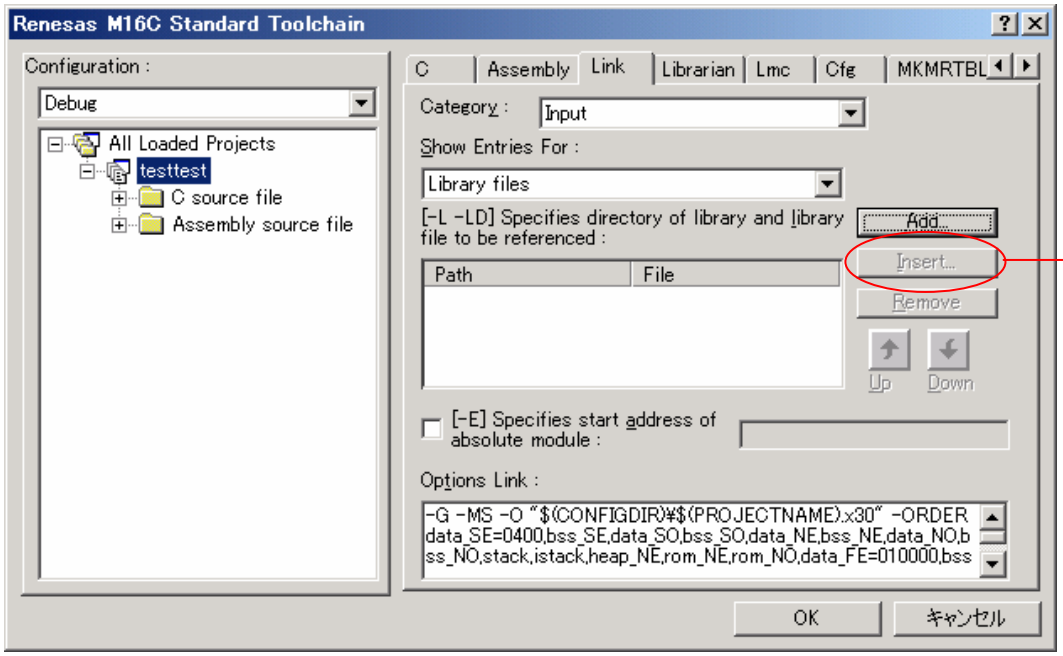
From the Build menu of HEW, select [Renesas M16C Standard Toolchain] -> Link tab.



(2) For Category,select Input

For Show Entries-For select Library files.

(3) Click the Remove button to remove r8clib.lib temporarily.



(4) Click the Add button to select [Library files].

(5)Input nc30lib.lib and click the OK button to finish.

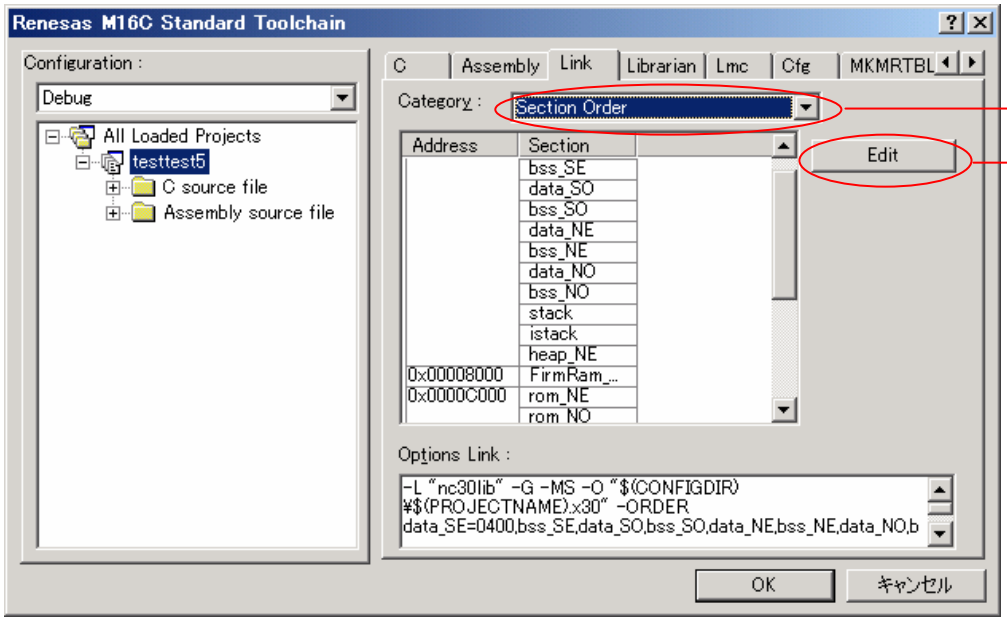
[To create a new workspace using microcomputers other than the R8C/Tiny]

- Choose the relevant CPU series from CPU Series.
- Choose Other from CPU Group.

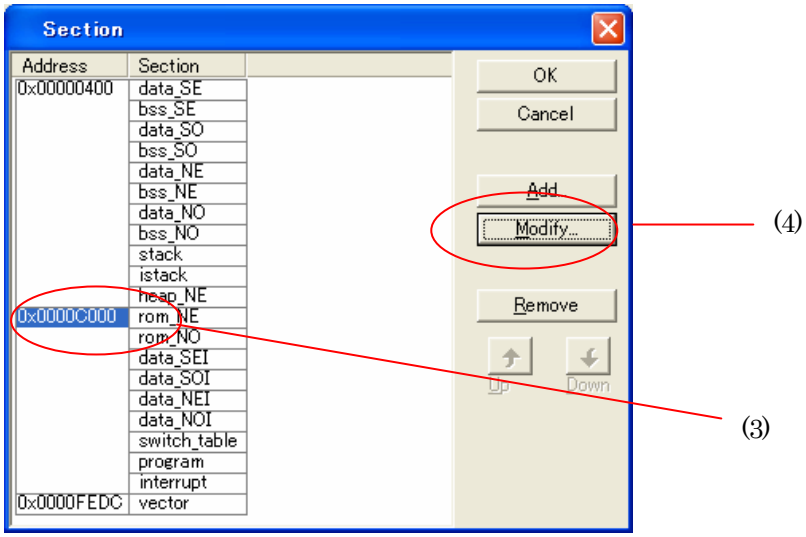
In addition to the above selections, the following requires caution.
・No sfr header files are registered.
Acquire the sfr header file corresponding to the microcomputer you use from the Web site, or create one if necessary.
・The order of sections is inaccurate.
Change the address of each section according to the ROM and RAM spaces of the microcomputer you use.
・No variable vector interrupt entry functions (intprg.c) are registered.
The interrupt functions should be created in a user file.

Change the section's address at linking

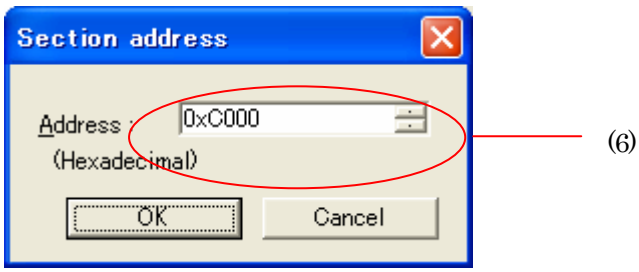
From the Build menu of HEW, select [Renesas M16C Standard Toolchain] -> Link tab.



(2) For Category,select Section-Order
Click the Edit button



(3) select address
(4) click the Modify botton



(6)input the address

A.2.3. When using an assembler startup

Instead of using a C language startup program, if you want to use the startups written in assembler ncrt0.a30 , sect30.inc (for NC308WA, sect308.inc) or nc_define.inc to create a project and the following applies, you need to make corrections.

- No vector numbers are specified when interrupt functions are declared.

If you build a project without making this correction, an error "Can't generate automatically the variable interrupt vector table" may be output when linking.

Content of correction

When using NC30WA

[sect30.inc]

```
.if     __MVT__==0

;------------------------------------------------------------
; variable vector section
;------------------------------------------------------------
        .section    vector,ROMDATA   ; variable vector table
        .org        VECTOR_ADR
.if  0  <-------------------------------------------  Deletes .if 0 to enable .lword.
        .lword      dummy_int                ; vector 0 (BRK)
        .lword      dummy_int                ; vector 1
        ……
        .lword      dummy_int                ; vector 63
.endif  <-------------------------------------------  Deletes .endif
```

When using NC308WA

[sect308.inc: near the 428th line]

```
;------------------------------------------------------------
; variable vector section
;------------------------------------------------------------
        .section    vector,ROMDATA           ; variable vector table
        .org        VECTOR_ADR
.if  0  <-------------------------------------------  Deletes .if 0 to enable .lword.
        .lword      dummy_int                ; BRK (software int 0)
        .lword      dummy_int                ;
        ……..
        .lword      dummy_int                ; software int 63
.endif  <-------------------------------------------  Deletes .if 0 to enable .lword.
```

B. A Guide to Porting Projects Created with TM to High-performance Embedded Workshop V.4

This document explains how to port projects created with TM V.2.xx or V.3.xx into High-performance Embedded Workshop V.4.

B.1. Summary

To port projects created using TM V.2.xx or V.3.xx into High-performance Embedded Workshop V.4, the Import Makefile function of High-performance Embedded Workshop is used. This function can create projects from such items of information as source files and build options described in the specified makefile files.

In TM, project files are created in the makefile format executable in GNU make format. When project files created with TM are selected as makefile files using High-performance Embedded Workshop Import Makefile function, they are converted to files that can run in High-performance Embedded Workshop. In addition to TM project files, the Import Makefile function can also convert files in the makefile formats for hmake, nmake, and gmake to High-performance Embedded Workshop projects.

B.2. Porting Procedure

To port projects created using TM into High-performance Embedded Workshop, perform the following steps:

1. Open the File menu and select the New Workspace command.
2. The New Project Workspace dialog box opens.

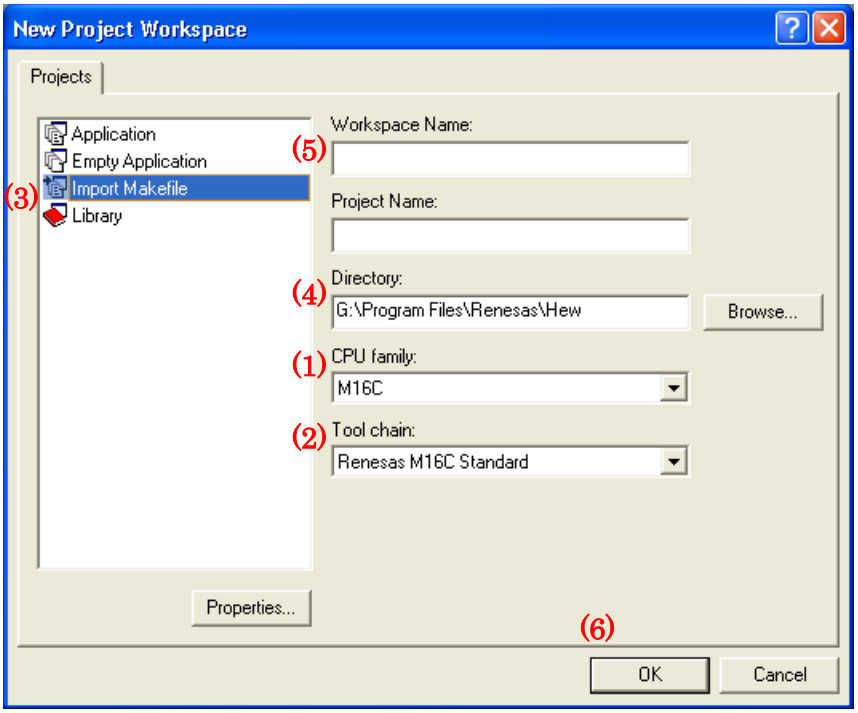


Figure 1 New Project Workspace Dialog Box

- Select the type of CPU used in the TM project from the Type of CPU drop-down list.
- Select the tool chain (cross tool) used for the TM project from the Toolchain drop-down list. The names of tool chains and corresponding cross tools are shown in Table 1.

Table 1 Tool Chains and Corresponding Cross Tools

Tool Chain	Cross Tool
Renesas M16C Standard	NC30WA
Renesas R8C Standard	NC8C
Renesas M32C Standard	NC308WA
Renesas M32R Standard	CC32R

- Select Import Makefile from the Project list.
- Type the directory path in the Directory text box.
- Type the workspace name in the Workspace Name text box. The same name will be automatically entered as the project name in the Project Name text box.
- Click **OK**.

3. You should now be able to see the New Project-1/4-Import Makefile wizard.

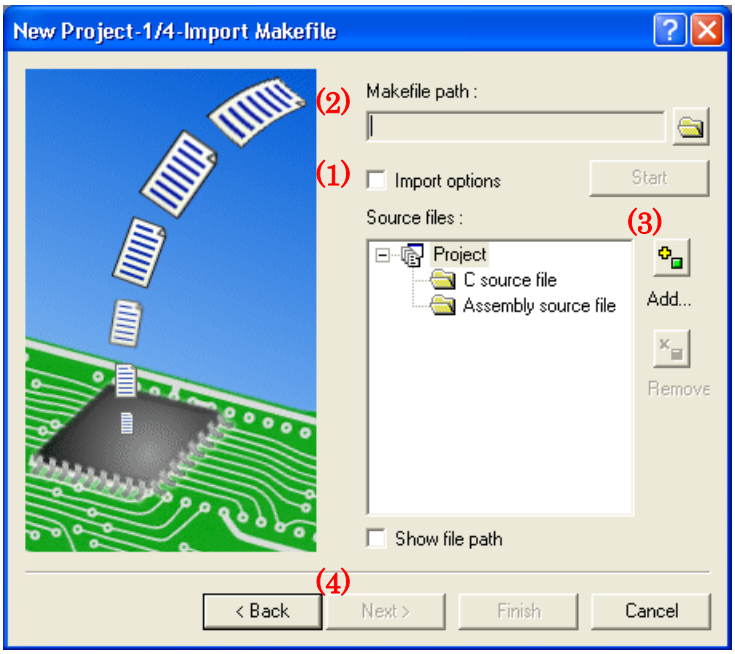


Figure 2 New Project-1/4-Import Makefile Wizard

- Select the Import options check box; this will enable information on build options (compiling and assembling options etc.) to be used to create High-performance Embedded Workshop projects. If you clear the Import options check box, the above information is neglected and not used in High-performance Embedded Workshop.
- Type the name of the TM project file (with extension .tmk) in the Makefile path text box. As soon the name is input, the specified file is analyzed, and upon analysis completion, the analyzed source files are displayed in a tree structure in the Source files box. Click the Start button to analyze the specified file again.
- If there are any errors in the analysis results (tree structure in the Source files box), rectify the tree structure with the Add and Remove buttons.
- Click **Next**.

4. Follow the instructions according to the Wizard as it continues in the procedure.

B.3. Usage Notices

B.3.1. TM-to-High-performance Embedded Workshop Portable and Non-Portable Information

When you port a project created using TM into High-performance Embedded Workshop, not all the components of the project can be ported.

Portable information is as follows:

- Paths of assembler source files
- Paths of C-language source files
- Assembling options
- C-compiling options
- Linking options (except linkage order)

Non-Portable Information:

- Linkage order
- Tool configurations, dependencies, and options other than Assembler, C Compiler, Linker

To transfer these items, edit the High-performance Embedded Workshop project as described in Section 3.4 and further after processing the Import Makefile.

B.3.2. Cross Tools

Import Makefile cannot enable all cross tool versions for use in High-performance Embedded Workshop projects regardless of whether they are used with TM or not; only the following cross tools versions are valid for High-performance Embedded Workshop projects:

NC30WA :	V.5.20 Release1 or later
NC8C :	V.5.30 Release1 or later
NC308WA :	V.5.20 Release1 or later
CC32R :	V.4.20 Release1 or later

B.3.3. High-performance Embedded Workshop Versions

When TM projects are ported into High-performance Embedded Workshop, information portable to High-performance Embedded Workshop varies according to the High-performance Embedded Workshop version. The information that can be ported from each cross tool to various High-performance Embedded Workshop versions are shown in Table 2.

Table 2 Portable Information and Corresponding High-performance Embedded Workshop Versions

		High-performance Embedded Workshop				
		~V.3.01.02	V.3.01.04	V.3.01.05	V.3.01.06	V.4.00
NC30WA	V.5.20 Release1	B	B	B	B	A
	V.5.30 Release1	B	B	B	B	A
NC8C	V.5.30 Release1	B	B	B	B	A
NC308WA	V.5.20 Release1	B	B	B	B	A
CC32R	V.4.20 Release1	B	B	B	B	A
	V.4.20 Release1A	B	B	B	B	A
	V.4.30 Release 00(A)	B	B	B	B	A

A: All the items of information listed in Section 3.1 are portable.
B: Only the paths of assembler and C-language source files are portable.

B.3.4. Generated Project Workspace

Because the project workspace created for a TM project ported to the High-performance Embedded Workshop environment is simply the contents of the makefile itself, its configuration (object output directory) will be different than that of a newly generated project workspace in High-performance Embedded Workshop.
To validate the configuration, modify the output directory file names for the compiler, assembler and linker as follows:

Output Directory (compiler, assembler):	$(CONFIGDIR)
Output Directory (linker):	$(CONFIGDIR)¥$(PROJECTNAME).x30

B.3.5. Load Module Converter

Import Makefile cannot port the information contained in any load module converter (for example, information on options, command executions, or dependencies) into the High-performance Embedded Workshop project. If using a load module converter to create projects in TM, change the settings of the load module converter as follows after completing the Makefile processing:

1. Open the Build menu and select the Build Phases command.
2. The Build Phases dialog box will open.

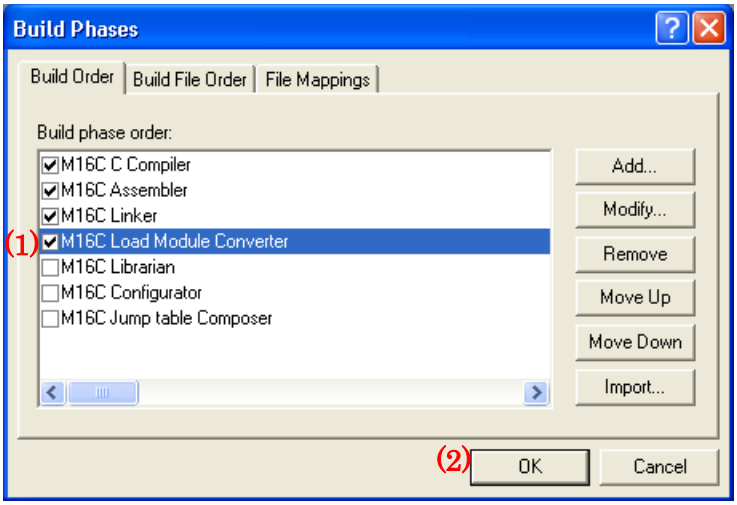


Figure 3 Build Phases Dialog Box

- Select the Mxxx Load Module Converter check box from the Order of Build Phases list.
- Click **OK**.

3. Open the Build menu and select Renesas Mxxx Standard Toolchain.

4. The Renesas Mxxx Standard Toolchain dialog box appears.

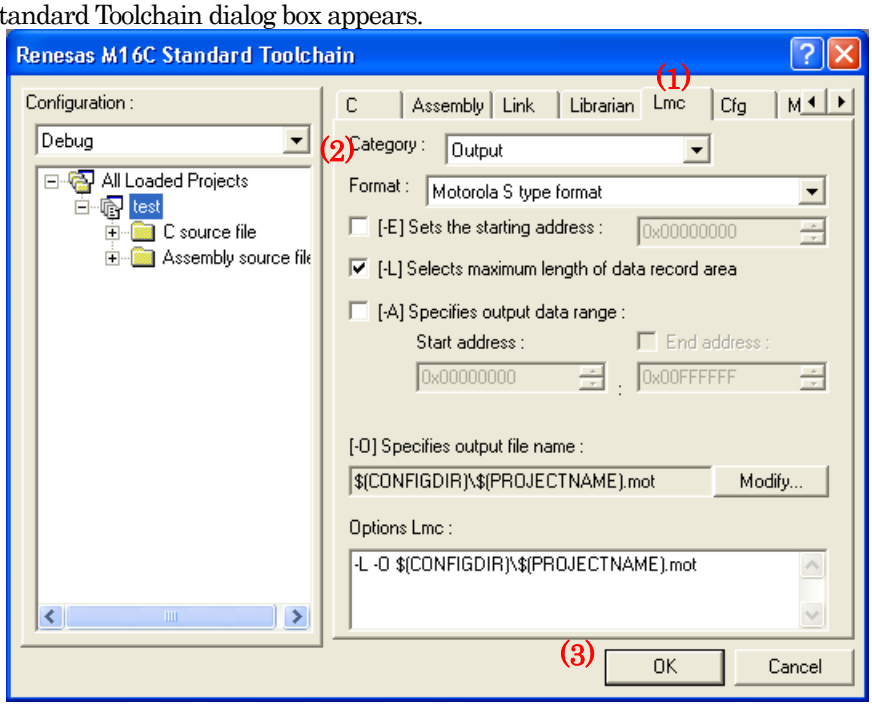


Figure 4 Renesas M16C Standard Toolchain Dialog Box

- Click the Lmc tab.
- Select the Category type from the Category drop-down list.
- Click **OK**.

B.3.6. Other Tools

Import Makefile cannot port any information (options, command executions, dependencies) contained in tools other than the assembler, C compiler, and linker. If any tools other than the assembler, C compiler, linker, and load module converter are used to create projects in TM, custom build phases must be created in High-performance Embedded Workshop. Custom build phases are specifically for operating other tools before, after, or during standard builds (in the assembler, C compiler, and linker).

For more details, see Section 3.2 "Creating Custom Build Phases" in the High-performance Embedded Workshop 4 User's Manual. The following is provided as an example of how to register the cross-reference generation tool xrf30 with High-performance Embedded Workshop.

1. Open the Build menu and select the Build Phases command.
2. The Build Phases dialog box appears; click **Add**.

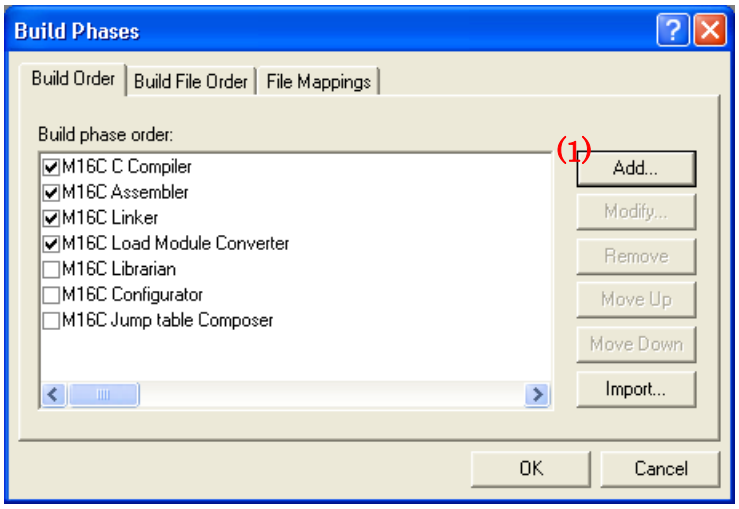


Figure 5 Build Phases Dialog Box

3. The New Build Phase- Step 1/4 wizard opens. Follow the instructions to register the tool as follows:

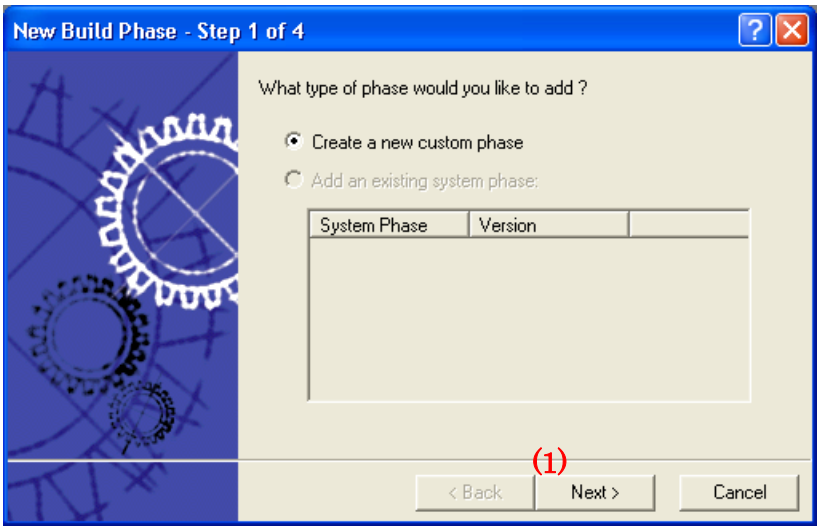


Figure 6 New Build Phase- Step 1/4 Wizard

- Click **Next** (the Create a New Custom Phase check box is selected by default); the New Build Phase-2/4 Step wizard opens.

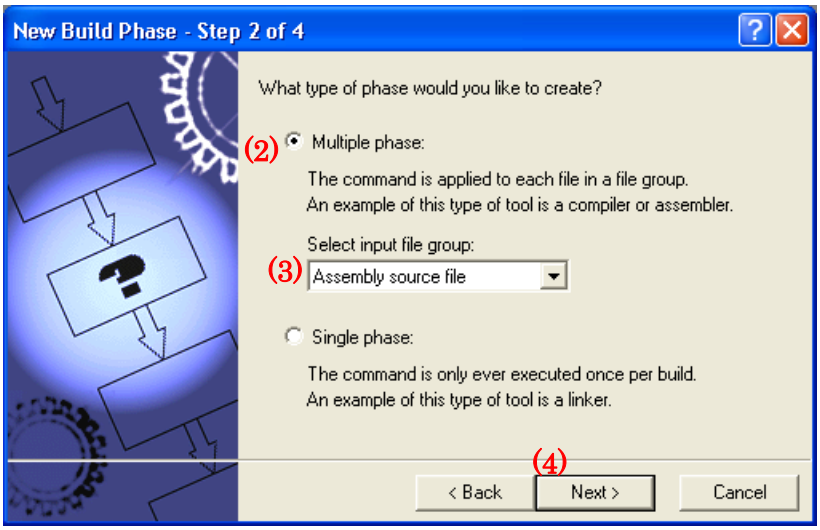


Figure 7 New Build Phase- Step 2/4 Wizard

- In this wizard, select the Multiple Phase check box.
- Select Assembly Source file from the Select input file group.
- Click **Next**; the New Build Phase- Step 3/4 wizard opens.

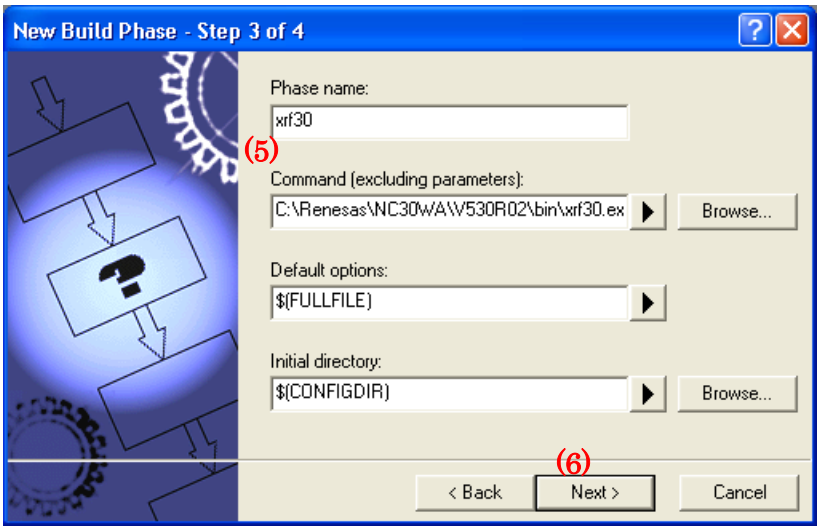


Figure 8 New Build Phase- Step 3/4 Wizard

- Type xrf30 and its fullpath name in the Phase Name and the Command text box.
- Click **Next**; the New Build Phase- Step 4/4 wizard opens.

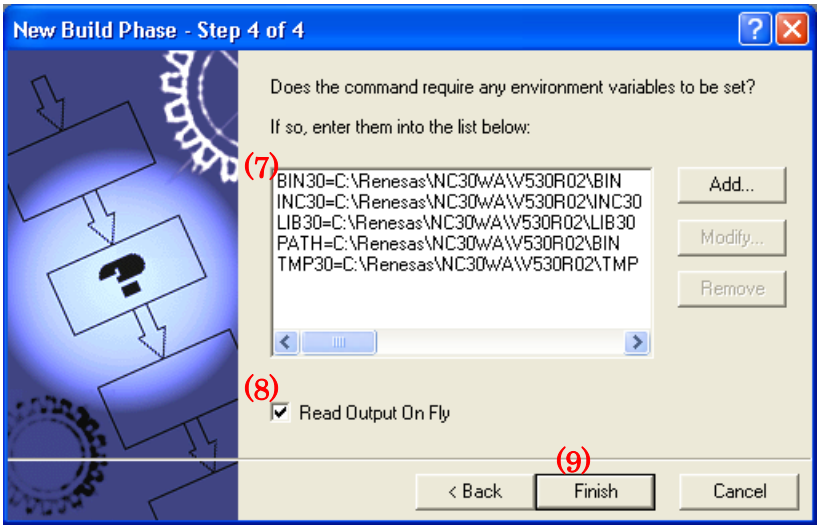


Figure 9 New Build Phase- Step 4/4 Wizard

- In this wizard, enter the necessary environment variables in the list.
- Select "Read Output On Fly" check box.
- Click **Finish**.

4. You return to the Build Phases dialog box at this point, where you can see that xrf30 has been registered as a build phase at the end of the Order of Build phase order.

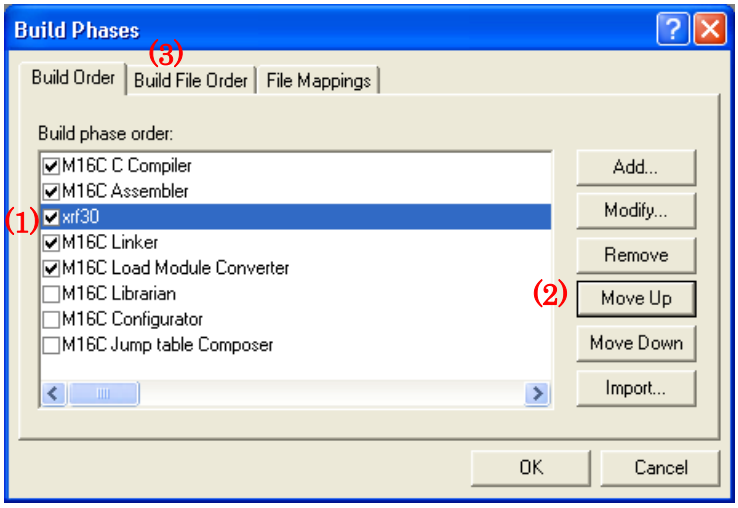


Figure 10 Build Phases Dialog Box (Build Order Tab)

- Select xrf30 from the Order of Build phase order.
- Click **Move Up** to move xrf30 next to the assembler name (see Figure 10).
- Click the Build File Order tab.

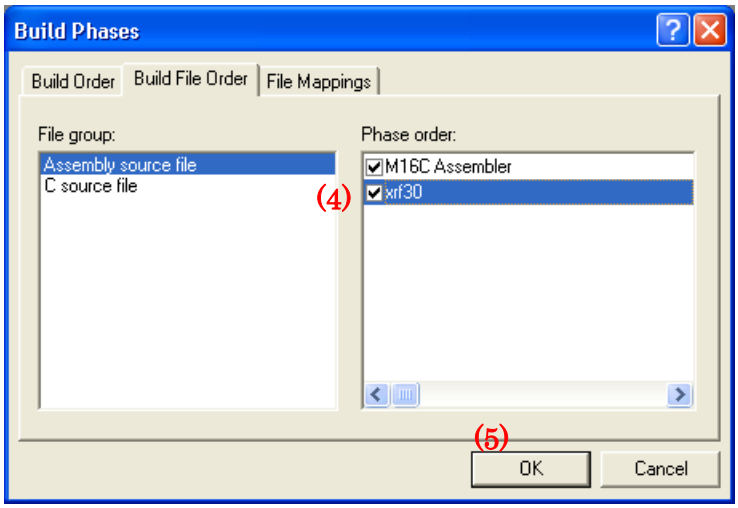


Figure 11 Build Phases Dialog Box (Build File Order Tab)

- Select the xrf30 check box in the Order of Phase order.
- Click **OK**.

5. Open the Options menu and select the xrf30 command.

6. The xrf30 Options dialog box appears; select options as necessary. This setting executes xrf30 for all assembler source files after assemble is completed at a build (before linking files).

B.3.7. Linkage order

Import Makefile cannot port the linking order information to High-performance Embedded Workshop. High-performance Embedded Workshop arranges the linking order alphabetically. To change this order, go through the following steps:

1. Open the Build menu and select the Linkage Order command.
2. The Linkage Order dialog box opens.

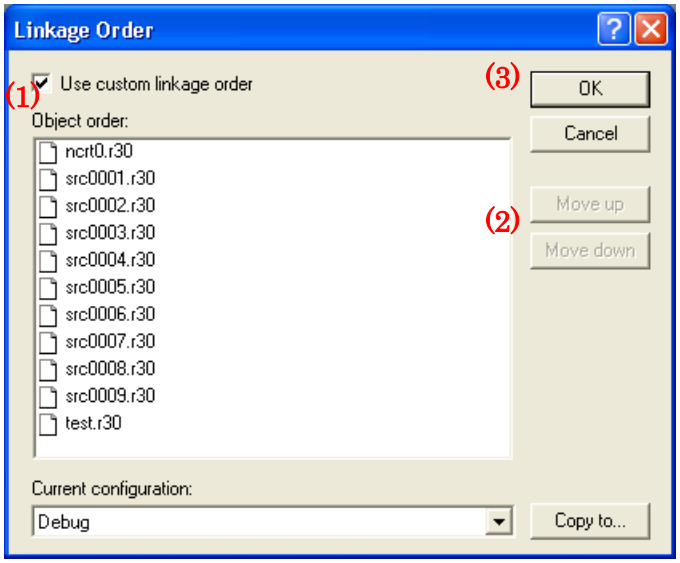


Figure 12 Linkage Order Dialog Box

- Select "Use custom linkage order" check box.
- Select a file from the Object order list, and click **Move up** or **Move down** to move the file. Repeat this step for all files that need to be rearranged.
- Click **OK**.

B.3.8. Placing the Start Up program at the top of Linkage Order

As the Import Makefile cannot port linking order information to High-performance Embedded Workshop, and links are order alphabetically, the start up program may not be placed at the top of the linking order. To place it at the top, follow the steps described previously in Section C.3.7 "Linkage Order."