To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

"Standard":	Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
"High Quality":	Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
"Specific":	Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

REJ10J1250-0200

===== Be sure to read this note. =====

C Compiler Package for M32C/90,/80, M16C/80,/70 Series

V.5.40 Release 00 Release note (Rev.2.0)

Renesas Solutions Corporation

Feb 16, 2006

Abstract

Welcome to C Compiler Package for M32C/90, /80, M16C/80, /70 Series V.5.40 Release 00. This document contains supplementary descriptions to User's Manual. When you read certain items in the User's manual, please read this document as well.

1. Precautions on Product

When using the compiler, please be sure to follow the precautions and suggestions described below.

1.1. Precautions about C Compiler

1.1.1. On nesting inline functions

- Description
  When an inline function that takes a parameter is nested, it may refer to an incorrect argument (a variable, not an argument).
- Conditions
  This problem occurs if the following conditions are both satisfied:
  (1) An inline function is nested in another.
  (2) Inline function A as a calling source and inline function B as the destination take the same parameter.
- Example

```
inline B(int aaa, char ccc)                /* Condition (2) */
 {
         .....
 }
 inline A(int c, int aaa, char *ccc)       /* Condition (2) */
 {
 int   i;
 char c;
       B(i,c);                             /* Condition (1) */
 }
```

- Workaround
  This problem can be circumvented any of the following ways:
  (1) Change the name of the parameter taken by the destination function (inline function B in the above example).
  (2) Don't nest any inline function.
  (3) Compile the program using the -Oforward_function_to_inline(-OFFTI) option.

1.1.2. Precautions about command option I

The number of directories that can be specified by the command option "-I" is 50 or less.

1.1.3. Precautions about the search of an include file

If you give a file to include together with a drive name in the #include line, and attempt to compile the file from a directory different from the one in which the file to compile is present, instances may occur in which the file to include cannot be searched.

1.1.4. Precautions to be taken when using #pragma ASM/ENDASM and asm()

- Regarding debug information when using #pragma ASM outside functions, if you write #pragma ASM anywhere outside functions, no C source line information will be output. For this reason, information regarding descriptions in #pragma ASM to #pragma ENDASM, such as error message lines when assembling or linking and line information when debugging, may not be output normally.
- C compilers generate code of arguments to be passed via registers and of register variables by analyzing their scopes. However, if manipulations of register values are described using inline assemble functions (such as #pragma ASM / #pragma ENDASM directives and asm function), C compilers cannot hold information on the scopes of the above-mentioned arguments and register variables. So, be sure to save and recover register contents on and from the stack when registers are loaded using inline assemble functions described above.

1.1.5. Precautions about debugging of a program using _Bool type

When you debug the program which uses the BOOL type, please confirm whether the debugger is supporting the

BOOL type.
In using the debugger which is not supporting the BOOL type, please use a debugging option"-gbool_to_char (-gBTC)" at the time of compile.

1.1.6. Precautions regarding the preprocessing directive #define

To define a macro which will be made the same value as the macro ULONG_MAX, always be sure to add the prefix UL.

1.2. Precaution of MCU-Dependent Code

1.2.1. M16Precautions regarding the M16C interrupt control register

When the "-O5" optimizing option is used, the compiler generates in some cases BTSTC or BTSTS bit manipulation instructions. In M16C, the BTSTC and BTSTS bit manipulation instructions are prohibited from rewriting the contents of the interrupt control registers. However, the compiler does not recognize the type of any register, so, should BTSTC or BTSTS instructions be generated for interrupt control registers, the assembled program will be different from the one you intend to develop. For detailed information about this, see below "Precautions for Interrupts" Described in Related Documents (Excerpts).
When using any of the products concerned, ensure that no incorrect code is generated.

- Example

  When the -O5 optimizing option is used in the program shown below, a BTSTC instruction is generated at compilation, which prevents an interrupt request bit from being processed correctly, resulting in the assembled program performing improper operations.

```
#pragma ADDRESS   TA0IC   006ch                    // M16C/80 MCU's Timer A0 interrupt control register
struct {
            char      ILVL : 3;
            char      IR    : 1;                    // An interrupt request bit
            char      dmy : 4;
} TA0IC;

void   WaitUntillRisON( void )
{
            while( TA0IC.IR == 0 )                  // Waits for TA0IC.IR to become 1
            {
                        ;
            }                                       // Returns 0 to TA0IC.IR when is become 1
}
```

- Workaround

  (1) Suppress the generation of the BTSTC and BTSTS instructions resulting from using an optimizing option by selecting the -ONA (or -Ono_asmopt) option together with "-O5" optimizing option.

  (2) Add an asm function to disable optimization locally, as shown in the example below.

```
void   WaitUntillRisON( void )
{
            while( TA0IC.IR == 0 )
            {
                        asm( );
            }
}
```

- Notes

  Make sure that no BTSTC and BTSTS instructions are generated after these side-steppings.

1.2.2. Precautions about access of SFR area

You may need to use specific instructions when writing to or reading registers in the SFR area. Because the specific instruction is different for each model, see the User's Manual for the specific Machine. These instructions should be used in your program using the asm function.

1.3. Precautions about MS-Windows

1.3.1. Precautions about environment of operation

(1) C Compiler Package operates under Windows 98, Windows NT 4.0 or later. It does not work under Windows 95 and Windows NT 3.5x or earlier.

(2) If in Windows NT environment the command prompt size is set to other than "80 x 25", the command prompt size will change frequently as you start the compiler. Make sure the command prompt size is set to "80 x 25".

1.3.2. Suggestions Concerning File Names

The file names that can be specified are subject to the following restrictions:

- Directory and file names that contain kanji cannot be used.
- Only one period (.) can be used in a file name.
- Network path names cannot be used. Assign the path to a drive name.
- Keyboard shortcuts cannot be used.
- The "..." symbol cannot be used as a means of specifying two or more directories.
- A file name in length of 128 characters or more including path specification cannot be used.

1.3.3. Precautions about virus check programs

If the virus check program is memory-resident in your computer, C Compiler Package may not start up normally. In such a case, remove the virus check program from memory before you start C Compiler Package.

1.3.4. Precautions when upgrading

To upgrade C Compiler Package, uninstall the currently installed C Compiler Package first before you install the new version.

- Procedure for uninstalling C Compiler Package

  To uninstall C Compiler Package, launch Add/Remove Programs in Control Panel and then execute Uninstall.

2. Contents of upgrade

2.1. Function addition and revision of C Compiler

- The compilation option "- M90" to generate the code corresponding to the M32C/90 series was added.
- Optimizing options -OR_MAX and -OS_MAX introduced in the compiler.
- The enhanced feature "#pragma MONITOR[n]" was added.
- The en-size blank space usable for names of directories (folders) under which tool products are installed; and of the files and paths.
- The start-up program in the C language introduced in addition to the assembly language.
- The type of the type definition size_t changed from unsigned int to unsigned long, and that of ptrdiff_t from signed int to signed long.
- Code-optimizing capabilities enhanced. Examples of optimization are as follows:
  (1) In the case where any statement or expression is repeatedly used in a program, it is made into a subroutine, to which calls are made instead of the statement or expression.
  (2) The continuation conditional expression not generated of the "for" construct that executes its iteration statement only once.

2.2. Problem correction of C Compiler

Improvements have been made to all of the following precaution that had been informed to you by tool news:

- On using if-else constructs
  http://tool-support.renesas.com/eng/toolnews/n050401/tn13.htm
- On passing a pointer or address to a function as an argument
  http://tool-support.renesas.com/eng/toolnews/n050601/tn8.htm
- On referencing array-type variables qualified with "const"
  http://tool-support.renesas.com/eng/toolnews/n050601/tn9.htm
- On defining more than one void function
  http://tool-support.renesas.com/eng/toolnews/n050716/tn9.htm
- On using optimizing option "-OR"
  http://tool-support.renesas.com/eng/toolnews/n050716/tn10.htm
- On compiling programs where the real-time OS M3T-MR308 used
  http://tool-support.renesas.com/eng/toolnews/n051001/tn6.htm
- On incorrect description of do statements
  http://tool-support.renesas.com/eng/toolnews/n051116/tn7.htm
- On performing a subtraction containing a constant as a subtrahend
  http://tool-support.renesas.com/eng/toolnews/n051116/tn8.htm

2.3. Functional revision of Assembler

2.3.1. as308

- The en-size blank space usable for names of directories (folders) under which tool products are installed; and of the files and paths.
- C language startup programs are now supported. Pursuant to this revision, following changes have been made:
  (1) The assemble options "-fMVT" and "-fMST" are removed.
  (2) A new directive command ".INITSCT" is added.
  (3) The directive commands ".RVECTOR" and ".SVECTOR" have their functionalities expanded.
- The assembler option "- M90" to generate the code corresponding to the M32C/90 series was added.

2.3.2. ln308

- The en-size blank space usable for names of directories (folders) under which tool products are installed; and of the files and paths.
- C language startup programs are now supported. Pursuant to this revision, following changes have been made:

(1) The link options "-fMVT" and "-fMST" are removed.
(2) The link options "-ORDER" have their functionalities expanded.
(3) The link options "-VECT" have their functionalities expanded.
(4) The link options "-VECTN" is added.

- The link option "- M90" to generate the code corresponding to the M32C/90 series was added.

2.3.3. lb308

The en-size blank space usable for names of directories (folders) under which tool products are installed; and of the files and paths.

2.3.4. xrf308

The en-size blank space usable for names of directories (folders) under which tool products are installed; and of the files and paths.

2.4. Problem correction of Assembler Systems

2.4.1. as308

Improvements have been made to all of the following precaution that had been informed to you by tool news:

- On using assembler options "-mode60" and "-mode60p"
  http://tool-support.renesas.com/eng/toolnews/n050301/tn12.htm

2.4.2. ln308

Improvements have been made to all of the following precaution that had been informed to you by tool news:

- On using the linker's option "-JOPT"
  http://tool-support.renesas.com/eng/toolnews/n051201/tn7.htm

2.4.3. lmc308

Improvements have been made to all of the following precaution that had been informed to you by tool news:

- On machine-language files created by the load module converter (lmc308 or lmc30) with the "-F" option selected
  http://tool-support.renesas.com/eng/toolnews/n050816/tn2.htm

3. Installing C Compiler Package

3.1. System requirements

Host computer	IBM[1] PC compatible machine
OS	Windows[2] 98, Windows Me, Windows NT 4.0, Windows 2000 or Windows XP
Memory capacity	128 MB at least; 256 MB or more recommended.
Hard disk drive	150 MB or more
Display	SVGA or higher
I/O device	CD-ROM drive
Others	Pointing device (e.g. a mouse)

3.2. How to install

Installation is executed according to the following instructions. Be sure to exit all applications being executed before installation.

3.2.1. Installation procedure

(1) Insert the CD-ROM into the CD-ROM drive. (Hereafter, the D drive is assumed.)
(2) Click [Run...] on the Windows [Start] menu.
(3) Using the [Run] dialog box, select Autorun.exe in the CD-ROM root directory (e.g., D:¥ Autorun.exe), and click [OK].
(4) Follow the instruction of the installer displayed on the screen.
(5) After installing the HEW system, the [Configure AutoUpdate] dialog opens. The initial setting must be made. For detailed in information about the initial setting, please refer to the Explanatory Information on AutoUpdate Tool.

3.2.2. Restrictions on installation of High-performance Embedded Workshop

(1) You need to input a license ID in the middle of installation. Before you start installing C Compiler Package, check your license ID.
(2) Do not install HEW V.4.00.03 in the same directory as Hitachi Integration Manager or Hitachi Embedded Workshop.
(3) If [Renesas] does not appear in the [Program] folder of the [Start] menu of Windows, restart the operating system.
(4) If your installation terminates abnormally during the installation, restart the host computer and install HEW V.4.00.03 again.
(5) Do not install HEW V.4.00.03 in the same directory as HEW Ver.1.x.
(6) When you would like to use the project created by HEW Ver. 1.0x, upgrade the version to HEW1.1x (open and save the project in HEW1.1x) and then open it with HEW V.4.00.03.
(7) Please carefully read the "License Agreement" and “Release Note” included with your product before using C Compiler Package. If you've installed this product in your computer, it is assumed that you've agreed to the provisions stipulated in the License Agreement.
(8) For the notice regarding HEW, see the following
http://tool-support.renesas.com/eng/toolnews/hew/hew.html

3.3. Uninstalling programs

The installed programs can be uninstalled according to the following instructions. Begin the un-installation after closing all the applications.

[1] IBM is a registered trademark of International Business Machines Corporation.
[2] Windows, Windows NT are registered trademarks of Microsoft Corporation.

3.3.1. Uninstallation of High-performance Embedded Workshop

(1) Select [Control Panel] on the Windows [Start] menu.
(2) Select the [Add or Remove Programs] icon.
(3) Click [High-performance Embedded Workshop] on the [Change or Remove Programs] tab and click the [Remove] button.
(4) Follow the instruction displayed on the screen.

3.3.2. Uninstallation of C Compiler

(1) Select [Control Panel] on the Windows [Start] menu.
(2) Select the [Add or Remove Programs] icon.
(3) Click [Renesas C Compiler Package for M32C/90, 80 and M16C/80, 70 Series V.5.40 Release 00] on the [Change or Remove Programs] tab and click the [Remove] button.
(4) Follow the instruction displayed on the screen.

3.4. Startup or termination of program

3.4.1. Startup and termination of the High-performance Embedded Workshop

- Startup

  Click [High-performance Embedded Workshop] in the [High-performance Embedded Workshop] folder in the [Renesas] folder in the [Program] folder of the Windows [Start] menu.
- Termination

  Click [Exit] on the [File] menu.

3.4.2. Launch Manual Navigator

- Startup

  Click [Manual Navigator] in the [High-performance Embedded Workshop] folder in the [Renesas] folder in the [Program] folder of the Windows [Start] menu.
- Termination

  Termination: Click [Exit] on the [File] menu.

3.4.3. Displaying the online manuals and attached documents

Select the manual on the Navigator window in Manual Navigator.

- Note

  (1) Manual Navigator requires Adobe Reader.[3].
  (2) If Manuals folder is moved, Manual Navigator cannot show them.

3.5. Setting when compiler is used on DOS prompt and command prompt

The environment variable of the C compiler is set to setnc308.bat that exists in the installation directory of the C compiler.
Please execute setnc308.bat when you use the compiler on the DOS prompt and the command prompt.

[3] Adobe and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

4. Software version list of C Compiler Package V.5.40 Release 00

The following lists the software items and their versions include with C Compiler Package.

- nc308 V.1.08.00.000
- cpp308 V.4.08.00.000
- ccom308 V.5.04.00.000
- aopt308 V.1.04.00.000
- as308 V. 4.02.02.000
- mac308 V. 2.22.00.000
- pre30 V.1.10.12
- asp308 V. 4.02.05.000
- ln308 V. 4.03.03.000
- lb308 V. 2.02.01.000
- lmc308 V. 3.00.02.000
- xrf308 VV. 2.02.00.000
- abs308 V. 1.03.00.000
- stk V.1.00.04
- utl308 V.1.00.10
- Stk Viewer V.1.00.01
- MapViewer V.3.00.00

5. Conformance with MISRA C Rule in Standard Function Library

In C-Source code of standard function library C Compiler Package, it is found that 52 rules[4] are against the MISRA C Rule NOTE, but these violations do not constitute a drawback to any operation.

5.1. Cause of Rule Violation

In C-Source code of standard function library C Compiler Package, the major causes for rule violation are as follows:

- C-Compiler specifications (near/far modifier, asm () function and #pragma)
- Declaration of function based on ANSI Standard
- The evaluation sequence in the conditional statement is not described explicitly, using a parenthesis.
- Implicit type conversion

5.2. Inspection No. running counter to the rule

The following are Inspection Nos. that run counter to the Rule:

1	12	13	14	18	21	22	28	34	35
36	37	38	39	43	44	45	46	48	49
50	54	55	56	57	58	59	60	61	62
65	69	70	71	72	76	77	82	83	85
99	101	103	104	105	110	111	115	118	119
121	124								

5.3. Evaluation Environment

C Compiler	C Compiler Package for M32C/80, M16C/80, /70 Series V.5.20 Release 1
Compile Option	-O -c -as308 "-DOPTI=0" -gnone -finfo -fNII -misra_all -r $*.csv
MISRA C Checker	SQMlint V.1.00 Release 1A

[4] These results were produced after inspection using MISRAC Rule Checker for M32C/90, M32C/80, M16C/80 Series.

6. C language Startup Program

Startup programs written in C language are supported beginning with the latest version described V.5.40 Release 00. This C language startup program can only be used in combination with V.5.40 Release 00.
This C language startup program can only be used in combination with V.5.40 Release 00.

6.1. File composition of C language Startup Program

C language Startup Program contains the following 13 files.

(1) resetprg.c
Initializes the microcomputer.

(2) initsct.c
Initializes each section (by clearing them to 0 and transferring initial values).

(3) heap.c
Reserves storage for the heap area.

(4) fvector.c
Defines the fixed vector table.

(5) intprg.c
Declares the entry function for variable vector interrupts.

(6) firm.c / firm_ram.c
Reserves storage for the program and workspace areas used by firm of FoUSB/NSD as dummy areas when OnChipDebugger is selected.

(7) cregdef.h
Declares the internal registers of the microcomputer.
Please do not alter the file.

(8) heapdef.h
Initializes the heap area.

(9) initsct.h
Contains statements for the processes (assembler macros) that initialize each section.
Please do not alter the file.

(10) resetprg.h
Include does each header file for C language Startup Program.

(11) stackdef.h
Defines the stack size.

(12) vector.h
Defines the variable vector address.

(13) typedef.h
Declares each type by typedef.

6.2. Processing of C language startup program

6.2.1. resetprg.c

he content of this file varies with the selected MCU.
This file is necessary for C language Startup Program.

```
#include "resetprg.h"
////////////////////////////////////////////
// declare sfr register
#pragma ADDRESS   protect    0AH
#pragma ADDRESS   pmode0   04H
#pragma ADDRESS   _SB__       0400H
_UBYTE protect,pmode0;
_UBYTE _SB__;

#pragma entry start
void start(void);
extern void initsct(void);
extern void _init(void);
void exit(int);

#pragma section program    interrupt                                                    → (1)
#pragma inline set_cpu()
void set_cpu(void)                                                                      → (2)
{
            _isp_       = &_istack_top;      // set interrupt stack pointer              → (3)
            protect = 0x02;                  // change protect mode register             → (4)
            pmode0   = 0x00;                 // set processor mode register              → (5)
            protect = 0x00;                  // change protect mode register             → (6)
            _flg_       = 0x0080;            // set flag register                        → (7)
            _sp_        = &_stack_top;       // set user stack pointer                   → (8)
            _sb_        = (char _far *)0x400;  // 400H fixation (Do not change)          → (9)
            _asm("      fset          b");
            _sb_        = (char _far *)0x400;
            _asm("      fclr          b");
            _intb_ = (char _far *)VECTOR_ADR; // set variable vector's address          → (10)

}

void start(void)
{
            set_cpu();                       // initialize mcu                           → (11)
            initsct();                       // initlalize each sections                 → (12)
#ifdef __HEAP__
            heap_init();                     // initialize heap                          → (13)
#endif
#ifdef __STANDARD_IO__
            _init();                         // initialize standard I/O                  → (14)
#endif
            _fb_ = 0;                        // initialize FB registe for debugger
            main();                          // call main routine                        → (15)

            exit(0);                         // infinite loop
}

void exit(int rc)
{
            while(1);
}
```

(1) The startup function is located in the interrupt section.
(2) Declares the function body of the CPU initialization function set_cpu().
(3) Initializes the interrupt stack pointer.
(4) Write enables the protect mode register.
(5) Sets the processor mode register to "single-chip mode". If modes need to be changed, this expression must be altered.
(6) Write protects the protect mode register.
(7) Sets the U flag to 1 (stack pointer changed for the user stack).

(8) Initializes the user stack pointer.
(9) Sets the SB register to address 0x400 (which sets the start address of RAM).
(10) Sets the variable vector address in the INTB register. The VECTOR_ADR that defines the variable vector address is defined in vector.h. Note also that if the variable vector address is altered by a link option for section order under the HEW environment, resetprg.c must always be recompiled.
(11) Calls the CPU initialization function.
(12) Initializes each section (by clearing them to 0 and transferring initial values).
(13) Initializes the heap area.If memory management functions are used, call to this function must be enabled.
(14) Initializes the standard input/output device.If standard input/output functions are used, call to this function must be enabled.
(15) Calls the main function.

6.2.2. resetprg.h

Include does each header file for C language Startup Program.
This file is necessary for C language Startup Program.

6.2.3. initsct.c

The content of this file varies with the selected MCU.
This file is necessary for C language Startup Program.

```
#include "initsct.h"
void initsct(void);

void initsct(void)
{
        sclear("bss_SE","data,align");                              (1)
        sclear("bss_SO","data,noalign");
        sclear("bss_NE","data,align");
        sclear("bss_NO","data,noalign");
        sclear("bss_FE","data,align");                              (2)
        sclear("bss_FO","data,noalign");

        /* clear bss for NSD */
        sclear("bss_MON1_E","data,align");
        sclear("bss_MON2_E","data,align");
        sclear("bss_MON3_E","data,align");
        sclear("bss_MON4_E","data,align");
        sclear("bss_MON1_O","data,noalign");
        sclear("bss_MON2_O","data,noalign");
        sclear("bss_MON3_O","data,noalign");
        sclear("bss_MON4_O","data,noalign");

        // when add new sections
        // bss_clear("new section's name");

        scopy("data_SE","data,align");                              (3)
        scopy("data_SO","data,noalign");
        scopy("data_NE","data,align");
        scopy("data_NO","data,noalign");

        /* copy data section for NSD */
        scopy("data_MON1_E","data,align");
        scopy("data_MON2_E","data,align");
        scopy("data_MON3_E","data,align");
        scopy("data_MON4_E","data,align");
        scopy("data_MON1_O","data,noalign");
        scopy("data_MON2_O","data,noalign");
        scopy("data_MON3_O","data,noalign");
        scopy("data_MON4_O","data,noalign");
        scopy("data_FE","data,align");                              (4)
        scopy("data_FO","data,noalign");
}
```

(1) sclear: Clears the bss section of the near area to zero.
If the bss section name is altered or a new bss section name is added using the #pragma SECTION bss feature, NE and NO must be altered or added in pairs.

```
sclear(“section name_NE,” “data.align”);
sclear(“section name_NO,” “data.noalign”);
```

Example: When a section is added by #pragma section bss bss2, the following must be added to init.sct.c

```
sclear( “bss2_NE” , “data.align” );
sclear( “bss2_NO” , “data,noalign” );
```

(2) sclear_f: Clears the bss section of the far area to zero.
If an external variable without initial values is declared using the far qualifier, this macro function must be enabled.

(3) scopy: Transfers initial values to the data section of the near area.
If the data section name is altered or a new dada section name is added using the #pragma SECTION data feature, NE and NO must be altered or added in pairs.

```
scopy(“section name_NE,” “data.align”);
scopy(“section name_NO,” “data.noalign”);
```

Example: When a section is added by #pragma section data data2, the following must be added to initsct.c

```
sclear( “data2_NE” , “data.align” );
sclear( “data2_NO” , “data,noalign” );
```

(4) scopy_f: Transfers initial values to the data section of the far area.
If an external variable with initial values is declared using the far qualifier, this macro function must be enabled.

6.2.4. initsct.h

This file is necessary for C language Startup Program.
Please do not alter the file.

6.2.5. heap.c

Only when memory management functions such as malloc are used.

```
#include "typedefine.h"
#include "heapdef.h"
#pragma SECTION    bss        heap                                    → (1)

_UBYTE heap_area[__HEAPSIZE__];                                       → (2)
```

(1) Locates the heap area in the heap_NE section.
If the heap size consists of an odd number of bytes, the heap_NO section is assumed by default.

(2) Reserves storage for the heap area by an amount equal to the size defined in __HEAPSIZE__.

6.2.6. heapdef.h

This file is necessary for memory management functions.

```
extern        _UBYTE _far * _mnext;
extern        _UDWORD _msize;
////////////////////////////////////////
// It's size of heap
// When you want to change size of heap,
// please change this line.
// When you change this line,
// you must modify the value using hex character.

#ifndef __HEAPSIZE__
#define __HEAPSIZE__ 0x300
#endif
extern _UBYTE heap_area[__HEAPSIZE__];

#pragma inline heap_init()
void heap_init(void)
{
              _mnext = &heap_area[0];                                        → (1)
              _msize = __HEAPSIZE__;                                         → (2)
}
```

(1) Initializes the heap management area.

(2) Initializes the heap size.

6.2.7. fvector.c

This file is necessary for C language Startup Program.

```
#include "vector.h"
#pragma sectaddress          fvector,ROMDATA Fvectaddr                       → (1)

/////////////////////////////////////////////////////////////////////////

#pragma interrupt/v _dummy_int         //udi                                 → (2)
#pragma interrupt/v _dummy_int         //over_flow
#pragma interrupt/v _dummy_int         //brki
#pragma interrupt/v _dummy_int         //address_match
#pragma interrupt/v _dummy_int         //single_step
#pragma interrupt/v _dummy_int         //wdt
#pragma interrupt/v _dummy_int         //dbc
#pragma interrupt/v _dummy_int         //nmi
#pragma interrupt/v start                                                    → (3)


#pragma interrupt _dummy_int()
void _dummy_int(void){}
```

(1) Outputs the section and address of a fixed vector table.
  This pragma is used exclusively for startup and cannot normally be used.

(2) Fills fixed vectors other than reset with a dummy function (_dummy_int).
  This pragma is used exclusively for startup and cannot normally be used.

(3) Defines the entry function.
  The function to be executed upon reset is registered in a fixed vector.

6.2.8. intprg.c

The content of this file depends on the MCU.

```
// BRK (software int 0)
#pragma interrupt     _brk(vect=0)
void _brk(void){}

// vector 1 reserved
// vector 2 reserved
// vector 3 reserved
// vector 4 reserved
// vector 5 reserved
// vector 6 reserved

// A/D1 (software int 7)
#pragma interrupt     _ad1(vect=7)
void _ad1(void){}

// DMA0 (software int 8)
#pragma interrupt     _dma0(vect=8)                                    → (1)
void _dma0(void){}

// DMA1 (software int 9)
#pragma interrupt     _dma1(vect=9)
void _dma1(void){}

// DMA2 (software int 10)
#pragma interrupt     _dma2(vect=10)
void _dma2(void){}

// DMA3 (software int 11)
#pragma interrupt     _dma3(vect=11)
void _dma3(void){}

// TIMER A0 (software int 12)
#pragma interrupt     _timer_a0(vect=12)
void _timer_a0(void){}

// TIMER A1 (software int 13)
#pragma interrupt     _timer_a1(vect=13)
void _timer_a1(void){}

              :
          (Omission)
              :
```

(1) Declares the variable vector interrupt function.
The functions corresponding to each variable vector interrupt function are declared. A variable vector table is generated at the same time.

6.2.9. firm.c / firm_ram.c

The content of this file is altered depending on the microcomputer type and selected OnChipDebugger.

```
#include "typedefine.h"
#pragma section bss FirmRam                                            → (1)
_UBYTE _workram[0x4];          // for Firmware's workram               → (2)
```

(1) Allocates the work ram area to be used by the E8 firmware in the FirmRam_NE section.
(2) Reserves storage for the work ram area by an amount equal to the size defined in __WORK_RAM__.

6.2.10. cregdef.h

This file is necessary for C language Startup Program.
Please do not alter the file.

6.2.11. stackdef.h

This file is necessary for C language Startup Program.

```
#ifndef __STACKSIZE__
#pragma STACKSIZE  0x300                              → (1)
#else
#pragma STACKSIZE  __STACKSIZE__                      → (2)
#endif
#ifndef ISTACKSIZE
#pragma ISTACKSIZE 0x300                              → (3)
#else
#pragma ISTACKSIZE __ISTACKSIZE__                     → (4)
#endif
extern _UINT _stack_top,_istack_top;
```

(1) Indicates the default size of the user stack.
(2) Outputs a user stack section and reserves storage for it.
(3) Indicates the default size of the interrupt stack.
(4) Outputs an interrupt stack section and reserves storage for it.

6.2.12. vector.h

This file is necessary for C language Startup Program.

```
#define Fvectaddr        0xffffdc                     → (1)
#ifndef VECTOR_ADR
#define VECTOR_ADR             0x0fffd00              → (2)
#endif
```

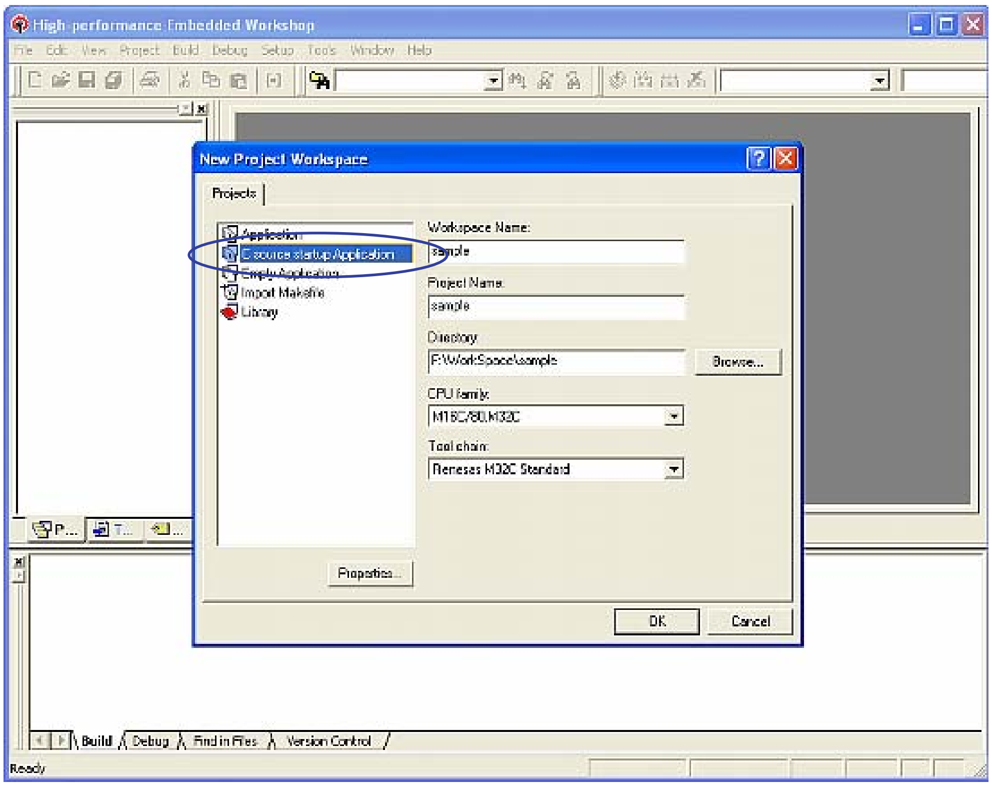
(1) Indicates the start address of a fixed vector table.
(2) Indicates the start address of a variable vector table.
If the start address of a variable vector table is changed, the address that is set in the INTB register in resetprg.c must also be changed at the same time.

6.2.13. typedefine.h

This file is necessary for C language Startup Program.
Please do not alter the file.

6.3. C language Startup Program is used on High-performance Embedded Workshop.

(1) “Csource startup Application “ of “New Project Workspace” is selected, and Workspace is made.


New Project Workspace

Projects

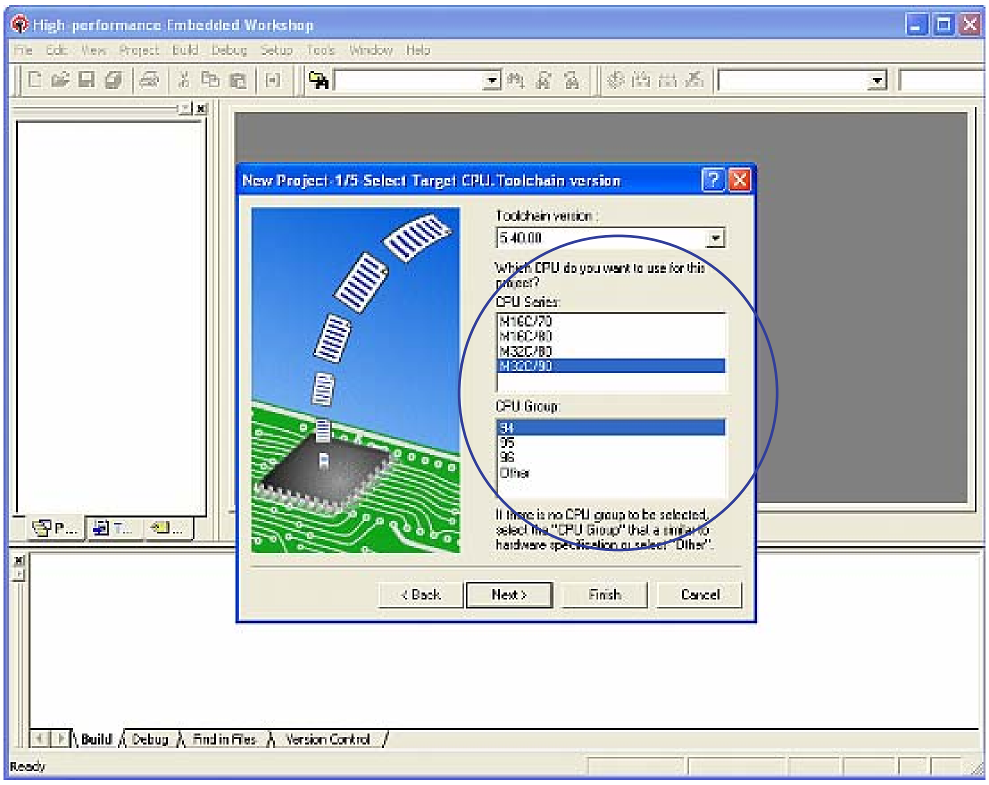
Application
C source startup Application
Empty Application
Import Makefile
Library

Workspace Name: sample
Project Name: sample
Directory: F:\WorkSpace\sample
CPU family: M16C/80,M32C
Tool chain: Renesas M32C Standard

High-performance Embedded Workshop
File Edit View Project Build Debug Setup Tools Window Help
Browse...
Properties...
OK
Cancel
Build
Debug
Find in Files
Version Control
Ready

If while multiple compilers are installed in your computer you select another microcomputer for the CPU type after selecting C source startup Application, the focus for C source startup Application will move to Application, with the result that the selected C source startup has no effect. In such a case, therefore, select “C source startup Application” again.

(2) The target microcomputer is selected from "CPU Series" and "CPU Group".


New Project-1/5 Select Target CPU,Toolchain version

Toolchain version : 5.40.00

Which CPU do you want to use for this project?

CPU Series:
M16C/70
M16C/80
M32C/80
M32C/90

CPU Group:
94
95
96
Other

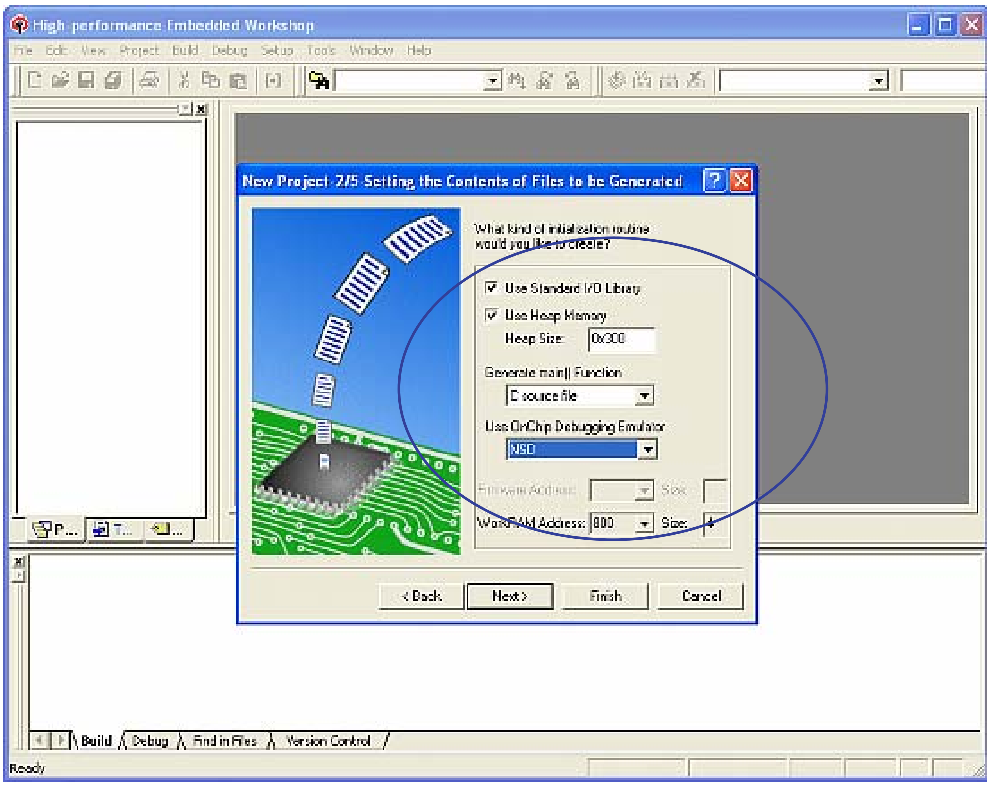
If there is no CPU group to be selected, select the "CPU Group" that a similar to hardware specification or select "Other".

High-performance Embedded Workshop
File Edit View Project Build Debug Setup Tools Window Help
< Back
Next >
Finish
Cancel
Build
Debug
Find in Files
Version Control
Ready

When a type of microcomputer is selected, its corresponding sfr header file is copied to the workspace.

Furthermore, a variable vector table (intprg.c) is registered.

(3) Settings for the case where the standard function and memory management function libraries are used


New Project-2/5-Setting the Contents of Files to be Generated

What kind of initialization routine would you like to create?

- [✓] Use Standard I/O Library
- [✓] Use Heap Memory
  - Heap Size: 0x300
- Generate main() Function: C source file
- Use OnChip Debugging Emulator: NSD
- Firmware Address: Size:
- WorkRAM Address: 800 Size: 4

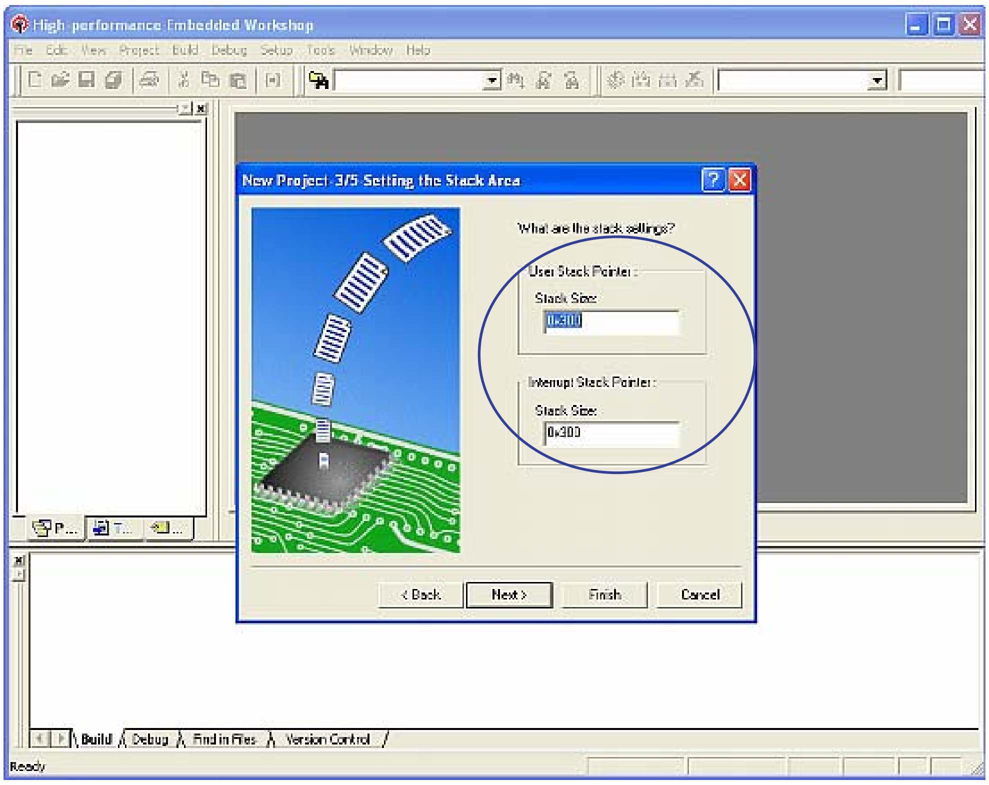
High-performance Embedded Workshop
File Edit View Project Build Debug Setup Tools Window Help
< Back
Next >
Finish
Cancel
Build
Debug
Find in Files
Version Control
Ready

(1) Select this check box when you use the standard function library.
When this check box is selected (flagged with a check mark), function calls to _init() in resetprg.c are enabled.
Furthermore, device.c and init.c are registered to the project.

(2) Select this check box when you use the memory management function library.
When this check box is selected (flagged with a check mark), function calls to heap_init() in resetprg.c are enabled.
Furthermore, heapdef.h and heap.c are registered to the project.

(3) Select the appropriate debugger when you use OnChip Debugging Emulator.
The selectable debuggers are FoUSB and NSD.
Note, however, that you cannot select either one of the two or both depending on the selected type of microcomputer.
When this selection is made, firm.c is registered.

(4) Selecting the stack size


New Project-3/5 Setting the Stack Area

What are the stack settings?

User Stack Pointer:
Stack Size: 0x300

Interrupt Stack Pointer:
Stack Size: 0x300

High-performance Embedded Workshop
File Edit View Project Build Debug Setup Tools Window Help
< Back
Next >
Finish
Cancel
Build
Debug
Find in Files
Version Control
Ready

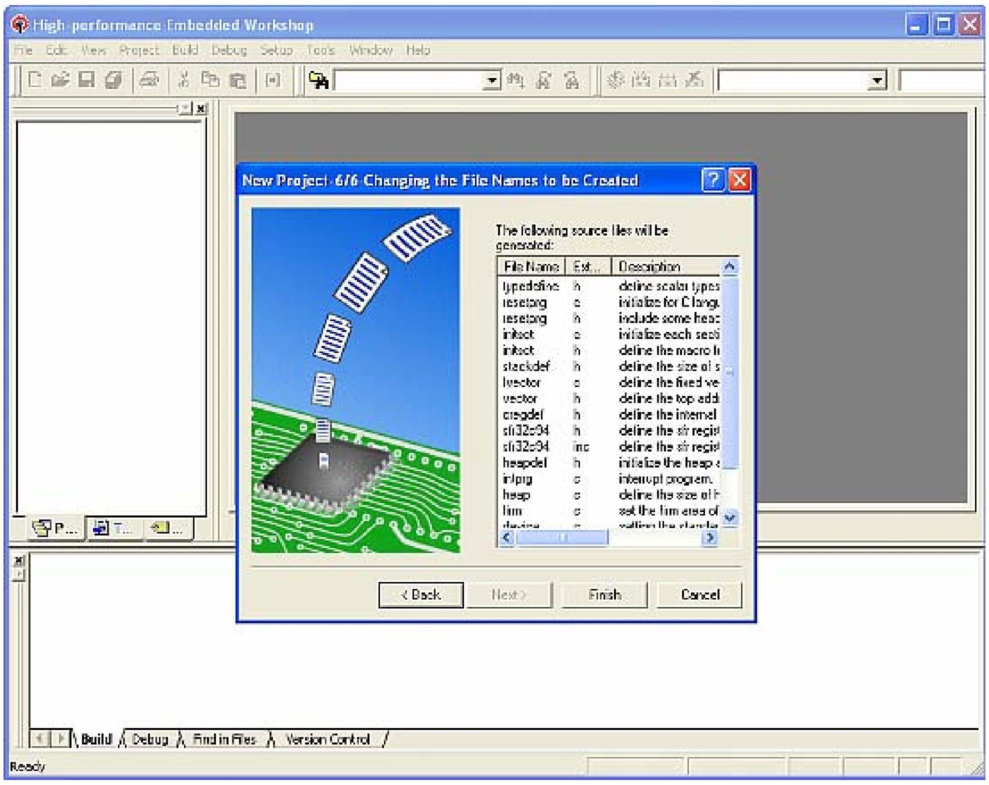
(1) Set the user stack size.
When this stack size is set, stackdef.h is registered.

(2) Set the interrupt stack size.
When this stack size is set, stackdef.h is registered.

To change the stack and HEAP sizes after creating a project, alter the value of each of the following in compile option settings:

```
-D__STACKSIZE__=xxxx
-D__ISTACKSIZE__=xxxx
-D__HEAPSIZE__=xxxx
```

(5) List of registered files


New Project-6/6-Changing the File Names to be Created

The following source files will be generated:

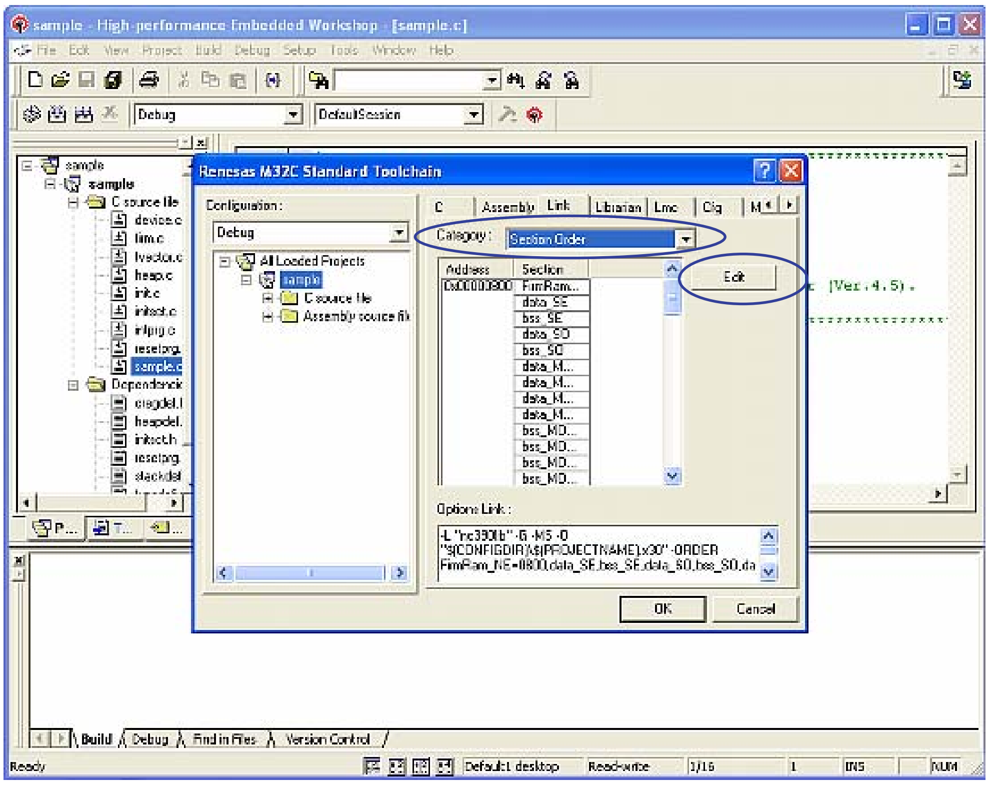
File Name	Ext...	Description
typedefine	h	define scalar types
resetprg	c	initialize for C langu
resetprg	h	include some head
initsct	c	initialize each secti
initsct	h	define the macro fi
stackdef	h	define the size of s
fvector	c	define the fixed ve
vector	h	define the top add
cregdef	h	define the internel
sfr32c94	h	define the sfr regis
sfr32c94	inc	define the sfr regis
heapdef	h	initialize the heap a
intprg	c	interrupt program.
heap	c	define the size of h
firm	c	set the firm area of

Here, you can check the list of files to be registered.

However, since the sfr header (C language header or assembler header) registered for each type of microcomputer is only copied to the workspace, take a look at this list to confirm the file name.

(6) Section Order

To confirm the order in which sections are linked and the addresses to which they are linked, take a look at "Category": Section Order in "Link" of "Renesas M32C Standard Toolchain".


Renesas M32C Standard Toolchain

Configuration: Debug

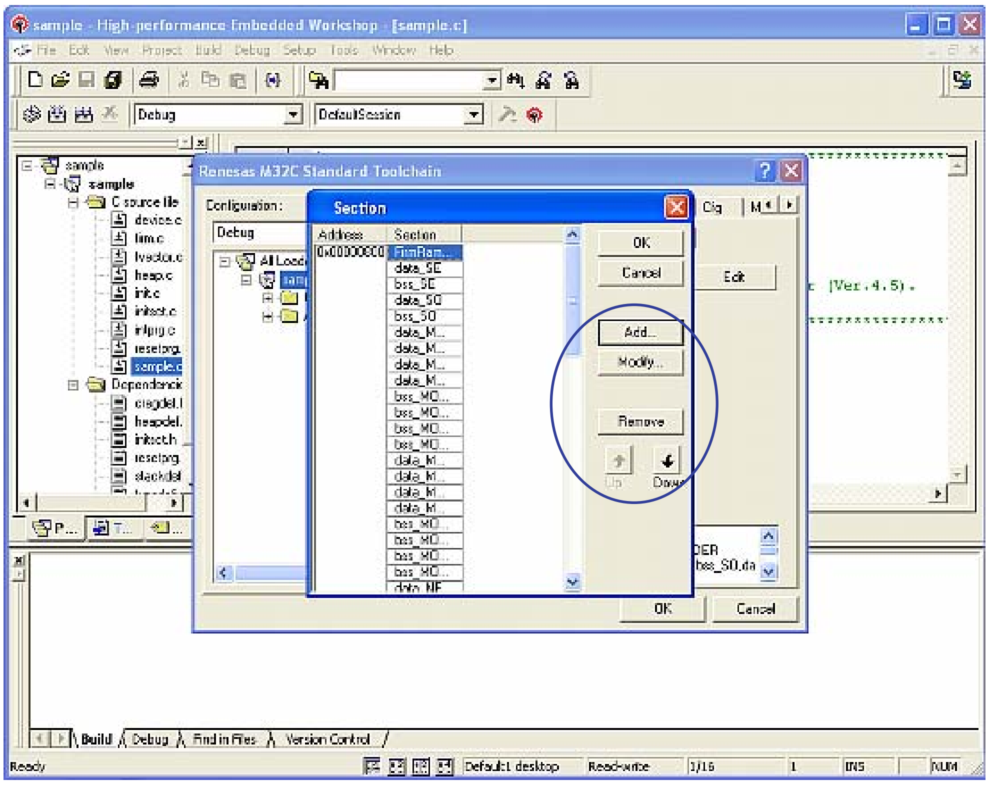
Category : Section Order

Address	Section
0x00000800	FirmRam...
	data_SE
	bss_SE
	data_SO
	bss_SO
	data_M...
	data_M...
	data_M...
	data_M...
	bss_MD...
	bss_MD...
	bss_MD...
	bss_MD...

Options Link :

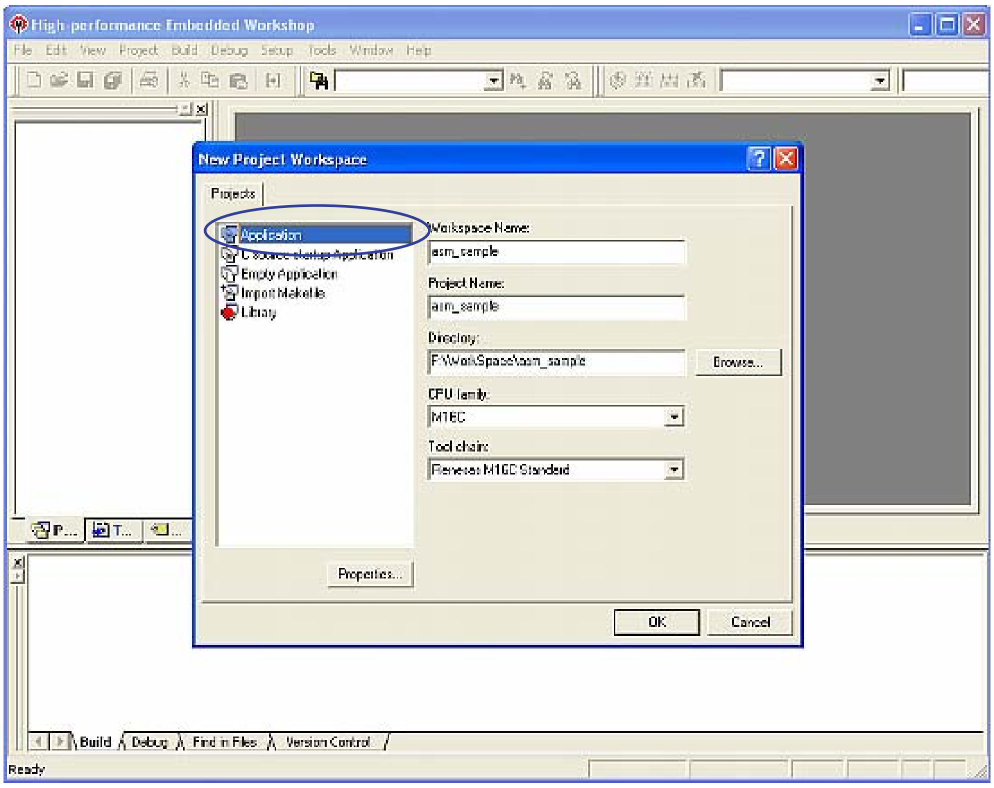
```
-L "nc390lib" -G -MS -O
"$(CONFIGDIR)\$(PROJECTNAME).x30" -ORDER
FirmRam_NE=0800,data_SE,bss_SE,data_SO,bss_SO,da
```

If you added a new section with #pragma SECTION, click the [Edit] button in (1) to open the Section window.



6.4. Assembly language Startup Program is used on High-performance Embedded Workshop.

“Application “ of “New Project Workspace” is selected, and Workspace is made.



Workspace Name: asm_sample
Project Name: asm_sample
Directory: F:\WorkSpace\asm_sample
CPU family: M16C
Tool chain: Renesas M16C Standard

7. Appendix

The following describes the contents not written in the C Compiler User's Manual and Assembler User's Manual in the C Compiler Package V.5.40 for the M32C/90, 80 and M16C/80, 70 series.

7.1. C Compiler User's Manual

7.1.1. Code Generation Change Options

-M90

Change code generation

Description
(1) Generates the code corresponding to the M32C/90 series.
(2) Predefines the M32C/90.

Notes
If the option -M90 is selected when compiling or assembling source files, be sure to use the standard function library nc390lib.lib when linking.

-fsizet_16 -fS16

Change the bit size of type definition

Description
Changes the type definition size_t from type unsigned long to type unsigned int.

Notes
If this option is selected, be sure to use one of the standard function libraries listed below when linking.

- M32C/90series
  nc390_16.lib
- M32C/80series
  nc382_16.lib
- M16C/80, /70series
  nc308_16.lib

-f ptrdifft_16 -fP16

Change the bit size of type definition

Description
Changes the type definition ptrdiff_t from type signed long to type singed int.

Notes
If this option is selected, be sure to use one of the standard function libraries listed below when linking.

- M32C/90series
  nc390_16.lib
- M32C/80 series
  nc382_16.lib
- M16C/80, /70series
  nc308_16.lib

7.1.2. Directive to Specify the Location of the RAM Monitor Area

#pragma MONITOR[n](n = 1–4)

Directive to specify the location of the RAM monitor area

Description Declares that the specified external variable be located in a section used exclusively for the RAM monitor area.

Form #pragma MONITOR[n] external variable name (n = 1–4)

Rules

1. Only external variables and external static variables can be specified.
2. The area for the external variable declared by #pragma MONITOR[n] is allocated to one of the sections listed below.

data_MON[n]_E	External variables in even size that have initial values are located here
data_MON[n]_O	External variables in odd size that have initial values are located here
bss_MON[n]_E	External variables in even size that do not have initial values are located here
bss_MON[n]_O	External variables in odd size that do not have initial values are located here
data_MON[n]_EI	Initial values of external variables in even size that have initial values are located here
data_MON[n]_OI	Initial values of external variables in odd size that have initial values are located here

3. The declaration of #pragma MONITOR[n] must be made before the external variable is defined.
4. The external variable declared by #pragma MONITOR[n] cannot be used in combination with other extended #pragma directives. However, if #pragma SBDATA and #pragma MONITOR[n] are specified at the same time, #pragma SBDATA has priority. At this time, no warnings are output.

Notes

1. #pragma MONITOR[n] does not affect the op-codes generated by the compiler. Please pay attention to the near/far attributes of variables.
2. Even if external variables with different near/far attributes coexist in a section used exclusively for the RAM monitor area, no errors and warnings are assumed. Please pay attention to the near/far attributes of variables.
3. The sections used exclusively for the RAM monitor area are not subject to size limitations.
4. The location address of the section allocated by #pragma MONITOR[n] and a process to set the initial value for the external variable should be written in the startup program.
5. If #pragma MONITOR[n] is declared a number times for one and the same external variable, the #pragma MONITOR[n] declared first is effective.
6. If the compile option -fno_even[-fNE] is specified, the external variable is located in a section with odd size attribute (e.g., data_MON1_O).
7. The external variables declared by #pragma MONITOR[n] are not affected by #pragma SECTION.
8. The declaration of #pragma MONITOR[n] has no effect if 'n' in it is other than 1–4. If the compile option -Wunknown_pragma[-WUP] or -Wall is specified, a warning is output.
9. External variables with ROM attribute cannot be handled by #pragma MONITOR[n]. However, if the compile option -fconst_not_ROM[-fCNR] is specified, these variables can be handled by #pragma MONITOR[n].

```
#pramga   MONITOR1          i
const      int              i;   <==== Has no effect
```

10. #pragma MONITOR[n] is not affected by the specification of compile options -M82 and -M90.