M16C/64 Group

High-performance Embedded Workshop Start-up Program in C

R01AN0043EJ0100
Rev.1.00
Nov. 30, 2010

1. Abstract

This document describes the source files created when a new workspace is created by the High-performance Embedded Workshop (HEW).

2. Introduction

The application example described in this document applies to the following microcomputer (MCU) and parameters:

- MCU: M16C/64 Group
- HEW version: Version 4.07
- C Compiler Package for M16C Series and R8C Family [M3T-NC30WA]: V. 5.45 Release 01

This application note can be used with other M16C Family MCUs which have the same special function registers (SFRs) as the above group. Check the user's manual for any modifications to functions. Careful evaluation is recommended before using the program described in this application note.

3. Outline

This application note describes source files created when "C source startup Application" is selected and a new workspace is created by HEW.

3.1 Source Files Created by HEW

Table 3.1 lists the C Source Files Created by HEW, and Table 3.2 lists the Header Files Created by HEW.

Table 3.1 C Source Files Created by HEW

Source File Name	Outline	Content
project_name.c (1)	Main file	Source file for main function
firm.c	Definition file for firmware	Maintains program and workspace areas for FoUSB/E8 firmware when the OnChipDebugger is selected.
firm_ram.c	RAM definition file for firmware	
fvector.c	Definition file for fixed vector table	Defines fixed vector tables. Sets OFS1 address and ID code. Modify fixed vector tables when performing non-maskable interrupt handling.
heap.c	Definition file for heap area	Defines the heap area used. The heap size value set (__HEAPSIZE__) when a project is created is defined in cstartdef.h. Modify the value of __HEAPSIZE__ when changing the heap size.
initsct.c	Initialize RAM file	Describes the RAM initial setting. When the user wants to initialize added areas, add the areas to be initialized.
intprg.c	Definition file for variable vector table	Defines relocatable vector tables and interrupt functions (empty functions). When maskable interrupts are used, add processes in interrupt functions.
resetprg.c	C source start-up program file	Executes the initialization processes before the main function is executed. When using the user boot function, set the user boot area in this file.

Note:

1. Insert user project name for project_name.

Table 3.2 Header Files Created by HEW

Header File Name	Outline	Content
cstartdef.h	Definition header file for stack size	Defines stack size and heap size. Select either enable/disable for the watchdog timer auto start after reset.
initsct.h	Macro definition header file to initialize sections	Defines the macro used in initsct.c.
resetprg.h	Header file for Initialization settings	Initializes the stack size, CPU register setting, and heap area.
sfrXX.h (1)	SFR register header file	Defines the SFR registers used for the MCU.
typedefine.h	Definition header file for data style	Defines data styles.

Note:

1. XX indicates the MCU group.

4. Settings of Source Files Created by HEW

4.1 Fixed Vector Table Setting

Fixed vector tables are defined in the fvector.c file. The program is executed from the address set in the reset vector of the fixed vector table after reset start. The address of the start function is set in the reset vector of the file created. Other non-maskable interrupts can be set in the same file.

Figure 4.1 shows the Fixed Vector Table Setting.

```
#pragma sectaddress     fvector,ROMDATA 0xfffdc
///////////////////////////////////////////////////////////////////////////////
#pragma interrupt/v _dummy_int  //udi
#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
```

Fixed vector tables

Refer to User's Manual: Hardware for details on individual interrupt sources.

Example of modifying watchdog timer interrupt

```
extern void wdt_int_func(void);
#pragma interrupt/v wdt_int_func    //wdt
```

Reset vector

```
#if __WATCH_DOG__ != 0
_asm("  .ofsreg 0FEH");
#else
_asm("  .ofsreg 0FFH");
#endif
```

Optional function select address 1 (OFS1) setting

Refer to "4.9 Optional Function Select Address 1 (OFS1) Setting" for details.

```
_asm("  .id  ""¥"#FFFFFFFFFFFFFF¥""");
```

ID code setting

Refer to "4.10 ID Code Writing" for details.

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

Dummy function definition

Figure 4.1 Fixed Vector Table Setting

4.2 cstartdef.h

When changing the project generator wizard setting values, modify them in the cstartdef.h file.
Figure 4.2 shows the cstartdef.h.

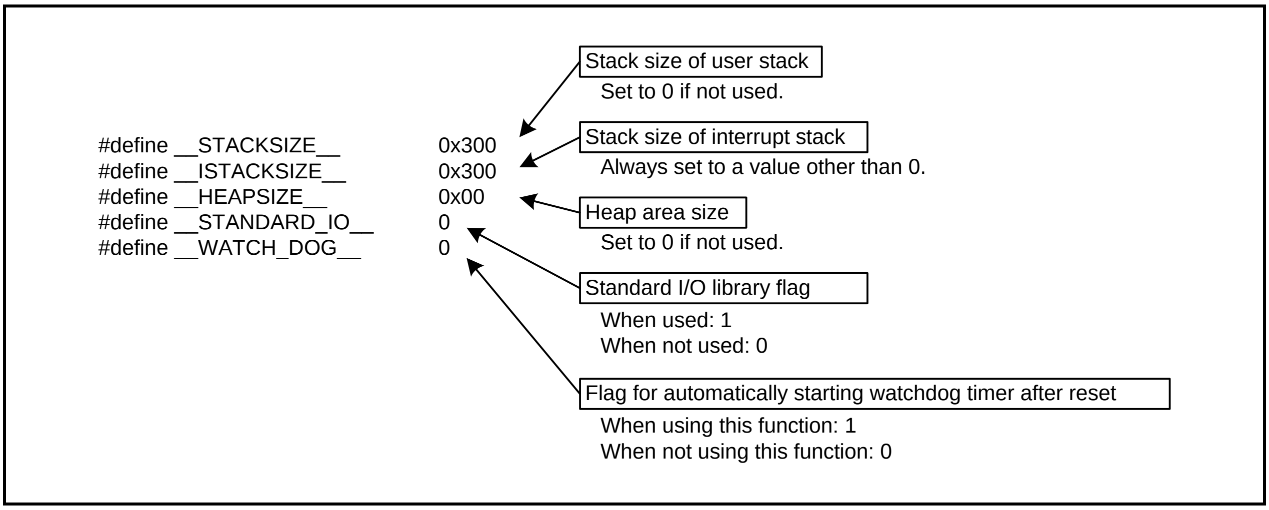
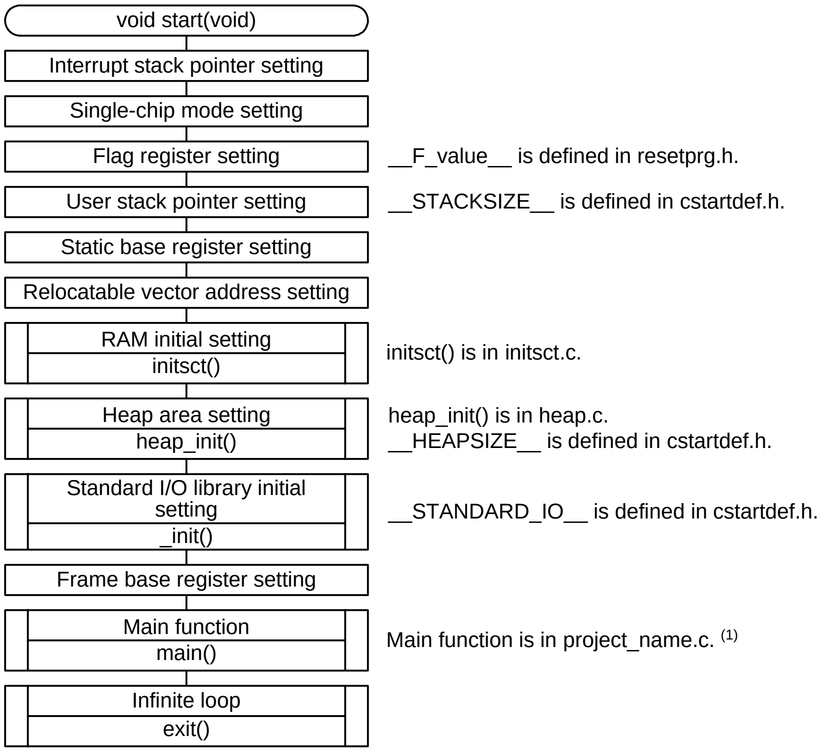


Figure 4.2 cstartdef.h

4.3 start Function Setting

The start function is created in the resetprg.c file.

Figure 4.3 shows the start Function Setting.



Note:
1. Insert user project name for project_name.

Figure 4.3 start Function Setting

4.4 RAM Initial Setting

The RAM initial setting process is created in the initsct.c file. The initsct.c file sets zero-clear to RAM. The initial values in ROM are transferred to RAM.
Figure 4.4 shows an Example of Initializing Added Area Using initsct.c.

```
sclear("bss_SE","data","align");
sclear("bss_SO","data","noalign");
sclear("bss_NE","data","align");
sclear("bss_NO","data","noalign");
sclear_f("bss_FE","data","align");
sclear_f("bss_FO","data","noalign");
// add new sections
// bss_clear("new section name");
sclear("test_bss_NE","data","align");
sclear("test_bss_NO","data","noalign");
scopy("data_SE","data","align");
scopy("data_SO","data","noalign");
scopy("data_NE","data","align");
scopy("data_NO","data","noalign");
scopy_f("data_FE","data","align");
scopy_f("data_FO","data","noalign");
```

Additional area is initialized using initsct.c.
When the test_bss_NE and test_bss_NO sections are added, add two lines if initsct.c is used for initialization.

Figure 4.4 Example of Initializing Added Area Using initsct.c

4.5 Heap Area Setting

Processing to allocate the heap area is created in the heap.c file.
Figure 4.5 shows the Heap Area Setting.

```
#pragma SECTION bss heap
_UBYTE heap_area[__HEAPSIZE__];
```

Variables without initial values described after this definition are assigned to the heap section.
Heap area is allocated.

Figure 4.5 Heap Area Setting

4.6 Main Function Setting

The main function is created in project_name.c file. Add programming in this file.

4.7 Interrupt Function Setting

Relocatable vector tables are created in the intprg.c file. Add interrupt handling to the intprg.c file if needed.

4.8 User Boot Function Setting

The user boot function setting is created in the resetprg.c file. Addresses 13FF0h to 13FFFh are the user boot area. If "UserBoot" in ASCII code is set to these addresses, the user boot function is enabled. Refer to User's Manual: Hardware when using the user boot function.

Figure 4.6 shows the User Boot Function Setting.

```
#pragma sectaddress _UB_section_FE,ROMDATA 0x13FF0
#pragma section rom _UB_section
struct _UB_struct {
        unsigned char code[8];
        unsigned char _near *addr;
        unsigned char bit;
        unsigned char level;
        unsigned char reserved[4];
} const _far _UB_data = {
        {0xffU,0xffU,0xffU,0xffU,0xffU,0xffU,0xffU,0xffU},
        (unsigned char _near *)0xffffU,
        0xffU,
        0xffU,
        {0xffU,0xffU,0xffU,0xffU}
};
#pragma section rom rom
```

_UB_section is assigned to 13FF0h.

Characters described after this definition are assigned to _UB_section.

User boot code area is defined.

ASCII code setting

Port information setting for entry

Reserved area

Characters described after this are assigned to the rom section.

Figure 4.6 User Boot Function Setting

4.8.1 When Not Using the User Boot Function

When the user boot function is not used, delete or comment out the coding area written in Figure 4.6. Addresses 13FF0h to 13FFFh are used for the user boot area. Do not assign programs to these addresses.

4.9 Optional Function Select Address 1 (OFS1) Setting

Optional function select address 1 (OFS1) setting is created in the fvector.c file.
Figure 4.7 shows the Optional Function Select Address 1 (OFS1) Setting.
The default value is FFh. When changing the OFS1 address, modify the value below.

```
_asm("  .ofsreg 0FFH");
```

Set FFh to the OFS1 address.

Figure 4.7 Optional Function Select Address 1 (OFS1) Setting

4.10 ID Code Writing

ID code writing process is created in the fvector.c file.
Figure 4.8 shows the ID Code Writing Process.
The default value is FFFFFFFFFFFFFFh. When changing the ID code setting, modify the value below.

```
_asm(" .id ""¥"#FFFFFFFFFFFFFF¥"");
```

Set the ID code.

Figure 4.8 ID Code Writing Process

5. Other Settings

5.1 Watchdog Timer Setting

The watchdog timer setting is created in the cstartdef.h file. Set __WATCH_DOG__ to select watchdog timer status after reset.
When using the watchdog timer in the source files created by HEW, note the following restrictions.

- The CPU clock must be set to 125 kHz on-chip oscillator divided-by-8 or higher.
- Do not use the watchdog timer count source protection mode.

When restrictions above will be problems, modify the program that the refresh process of watchdog timer in the initsct.h file start before watchdog timer reset occurs.

Figure 5.1 shows the Flag for Automatically Starting Watchdog Timer After Reset.

```
#define __WATCH_DOG__          0  <----  Flag for automatically starting watchdog timer after reset
                                          Function used: 1
                                          Function not used: 0
```

Figure 5.1 Flag for Automatically Starting Watchdog Timer After Reset

5.2 Section Setting

This section shows the section setting method by HEW. Usually, the section location is set in order of “Build” → “Renesas M16C Standard Toolchain...” → “Link” → “Category” in HEW, but this application note shows an example using the map window.
“C source startup Application” must be selected for the project type to use map window when a new project is created.
The following describes how to add the test_bss_NO and test_bss_NE sections, and set reset_times. Figure 5.3 to Figure 5.12 show the Added Sections.

(1) Show map window. Select “Map...” from the “View” menu.

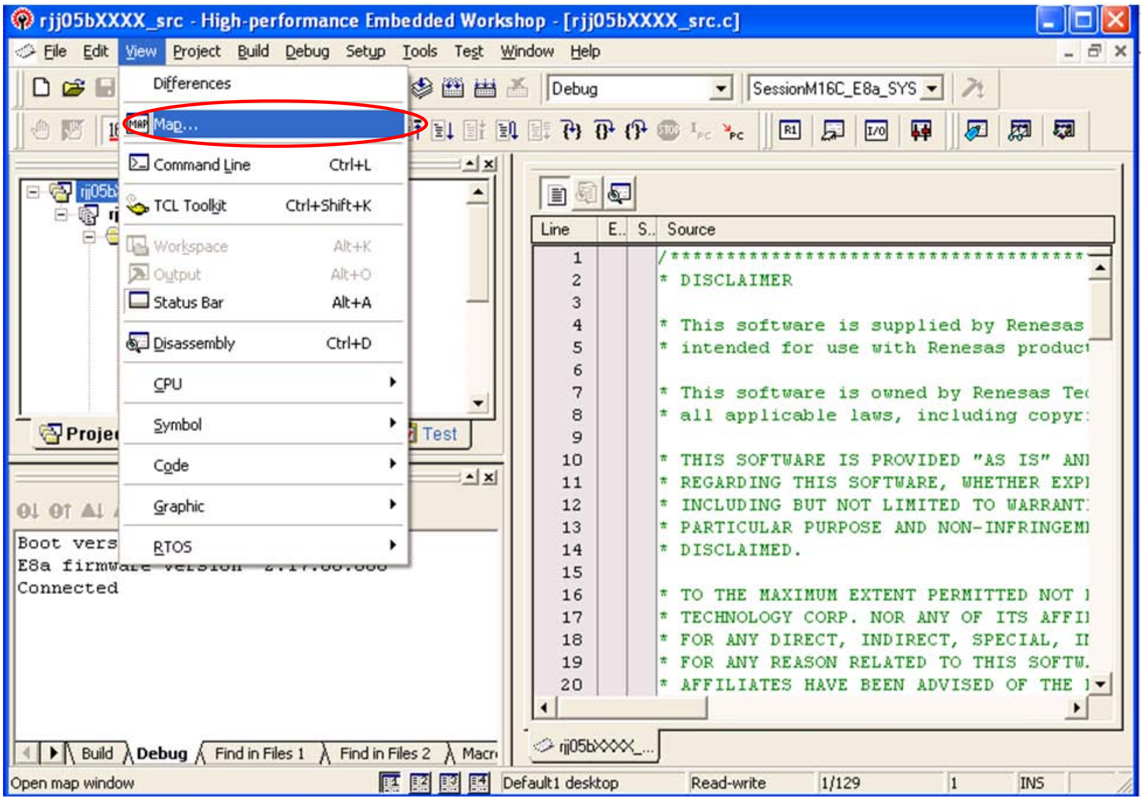


Figure 5.2 Added Section (1/9)

(2) Select “Map Section Information” from “Map”.
Use the same procedure to select “Map Symbol Information”.

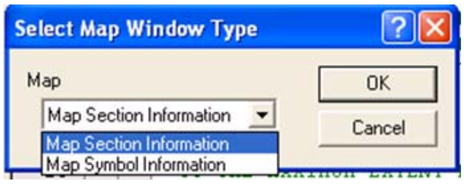


Figure 5.3 Added Section (2/9)

(3) Click the “Edit Mode” button, and edit the section name.

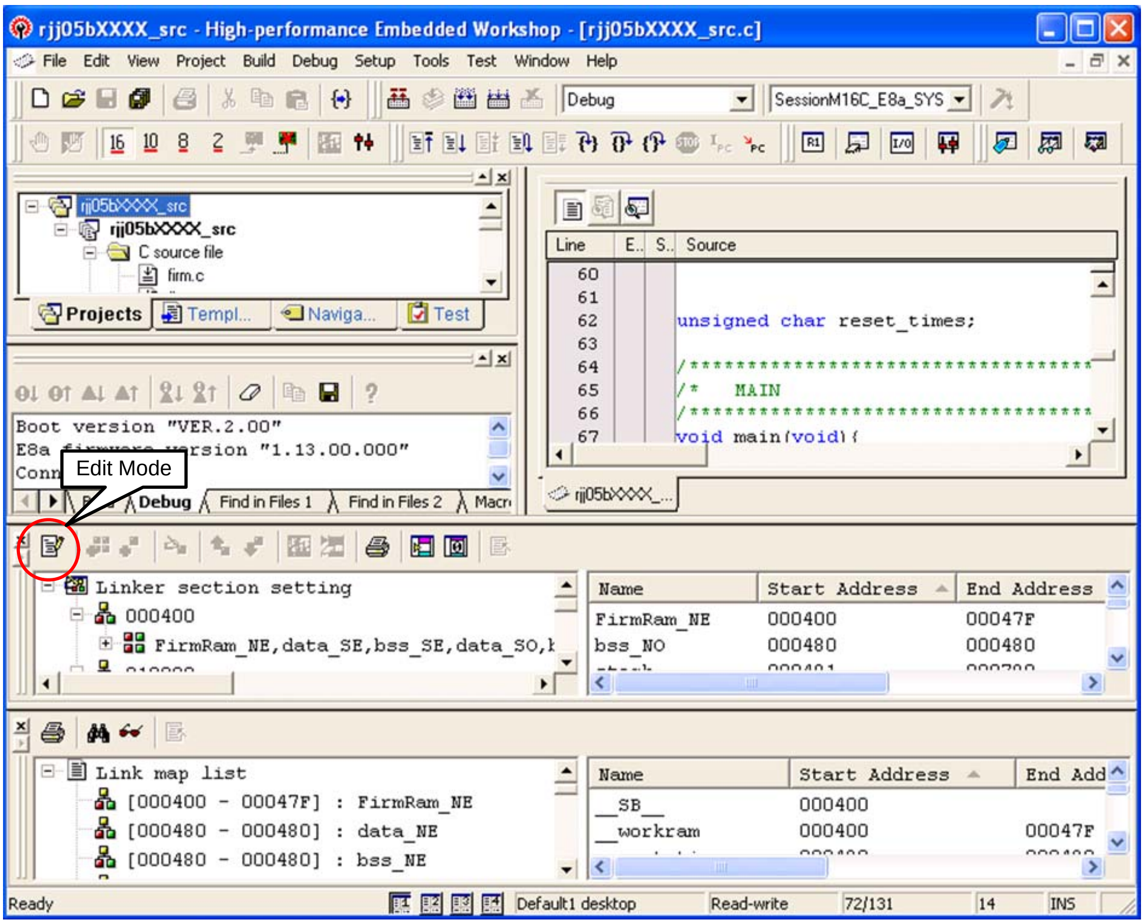


Figure 5.4 Added Section (3/9)

(4) Click the “Add Section...” button and add test_bss_NO.

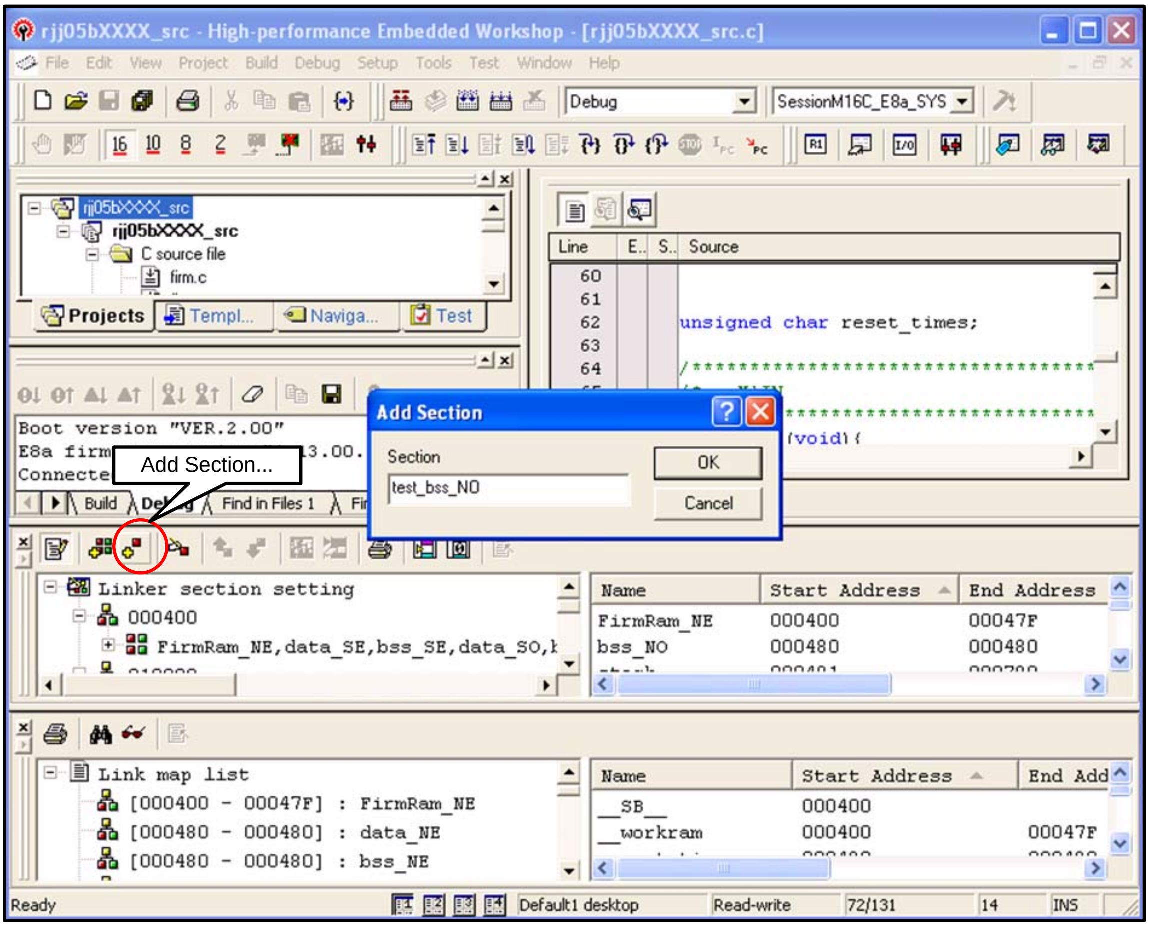


Figure 5.5 Added Section (4/9)

(5) Add test_bss_NE using same method in step (4).

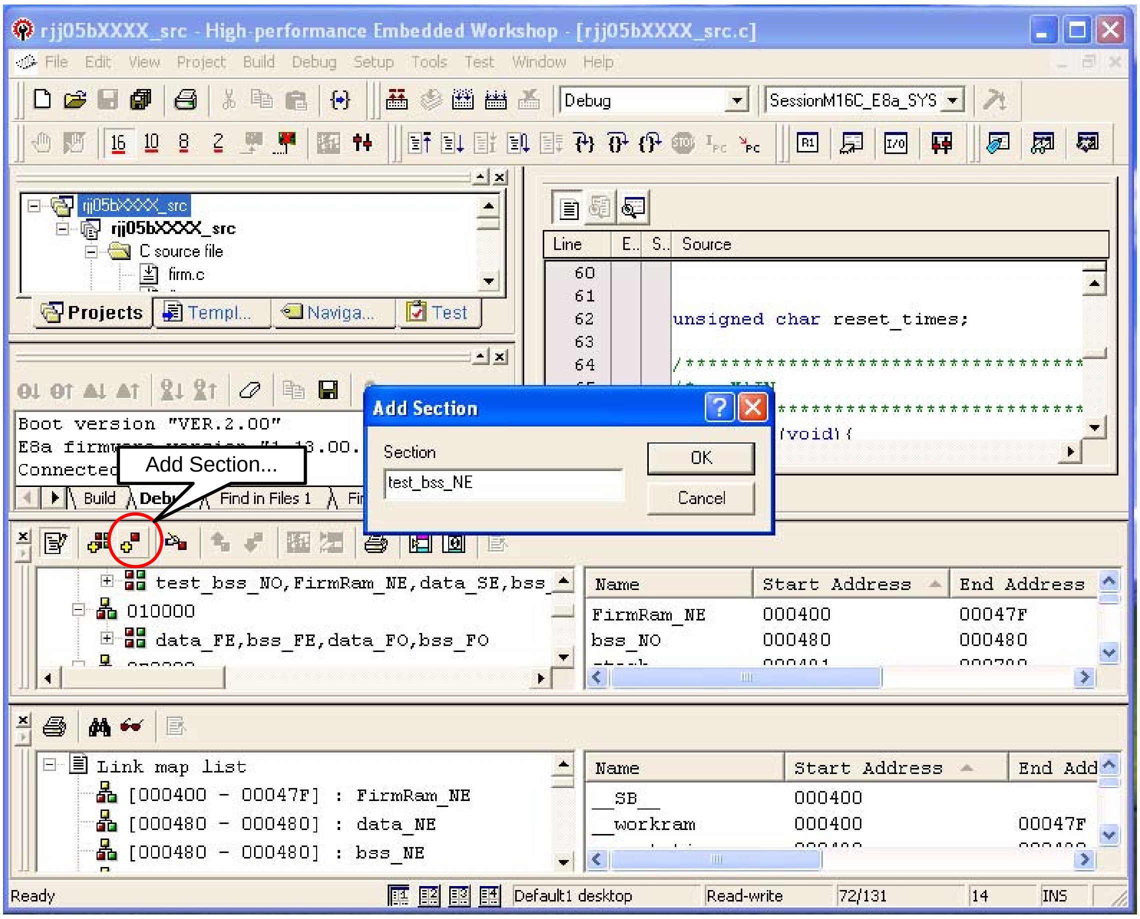


Figure 5.6 Added Section (5/9)

(6) Click the “Edit Mode” button, and confirm the changed linker section information.

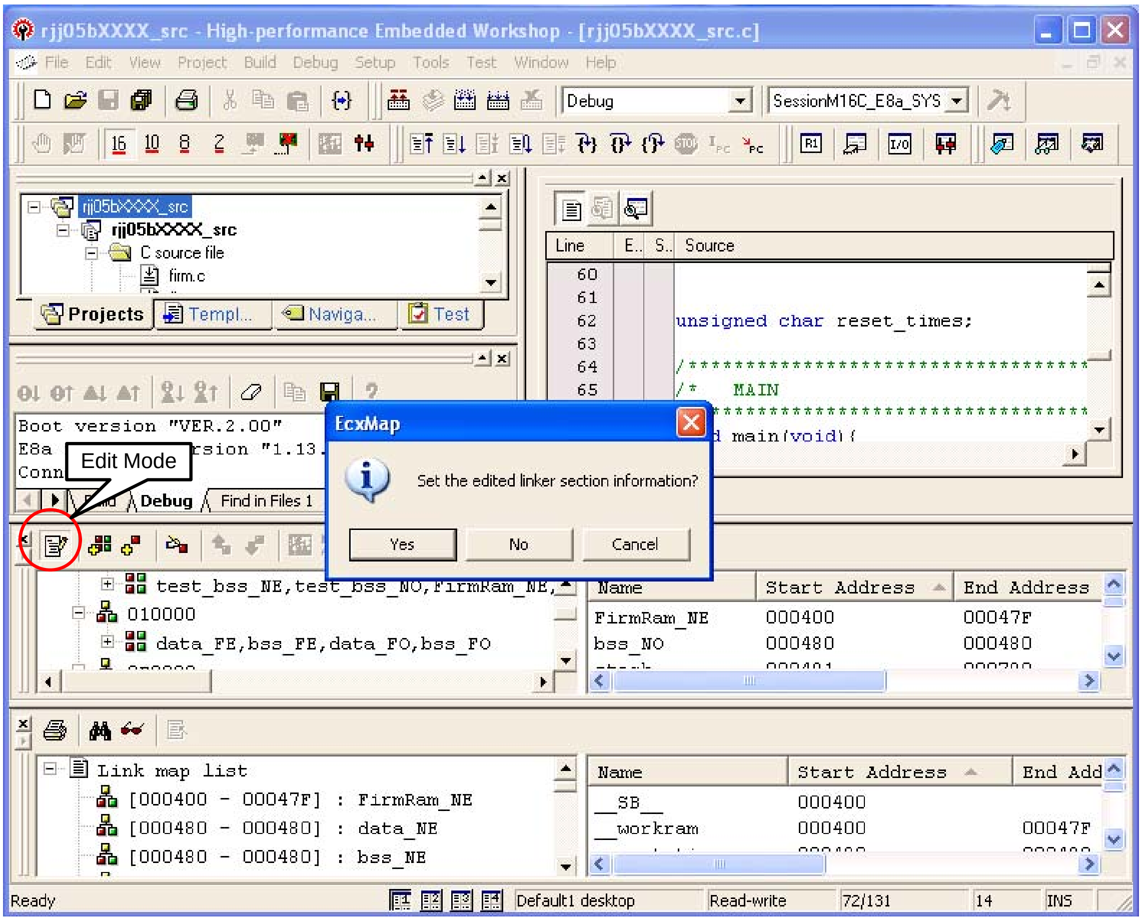


Figure 5.7 Added Section (6/9)

(7) The test_bss_NE and test_bss_NO sections are assigned.

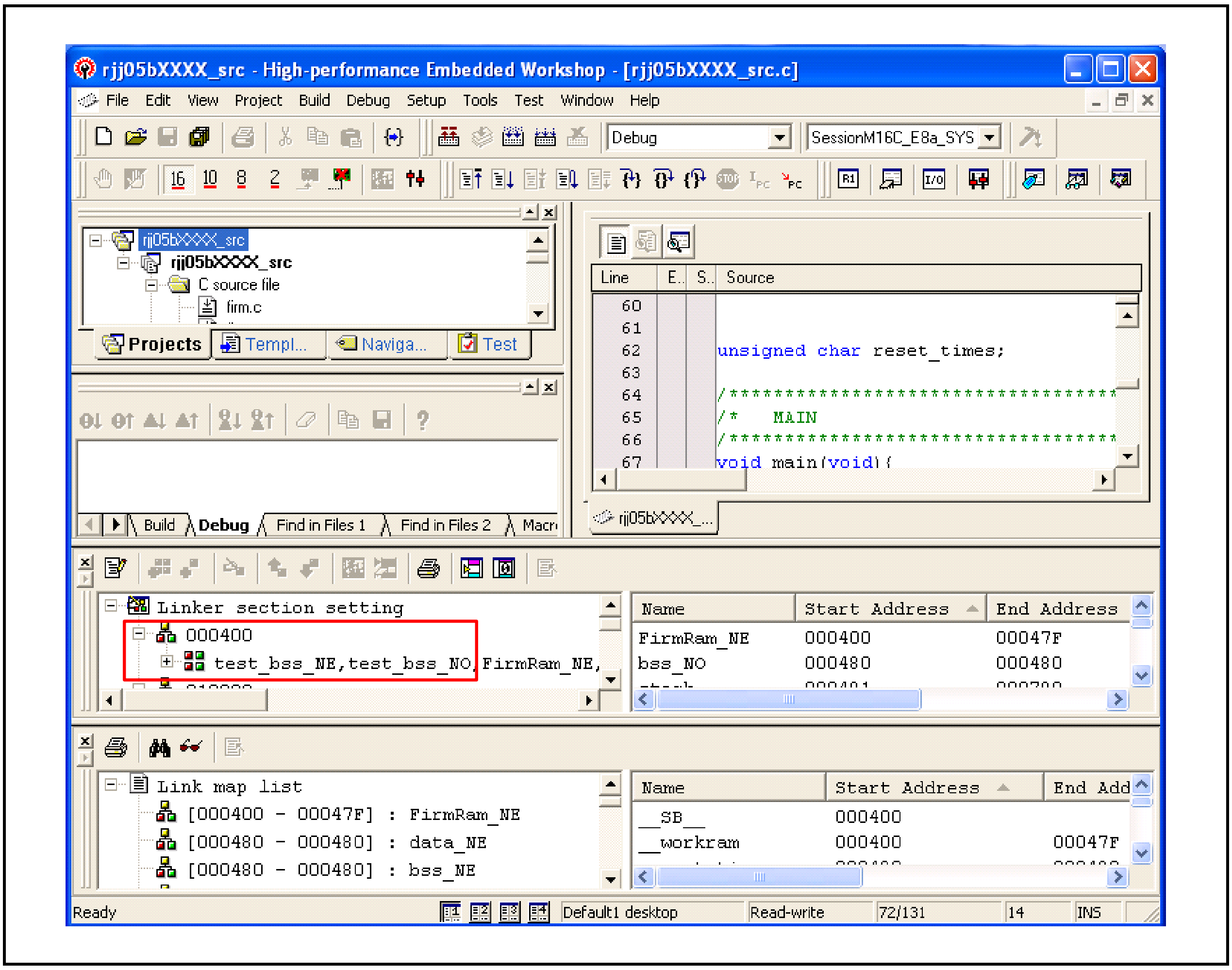


Figure 5.8 Added Section (7/9)

(8) Assign variable reset_times to the test_bss section by a program.

```
#pragma SECTION bss test_bss
unsigned char reset_times;
```

Figure 5.9 Method of Setting Variable to test_bss Section

Variables without default values after the #pragma SECTION declaration are assigned to the test_bss section.

```
int i;                          <- It is assigned to the bss section.

#pragma SECTION bss test_bss    <- Variables without default values described after this are
                                   assigned to the test_bss section.

int j;                          <- It is assigned to the test_bss section.
int k = 10;                     <- Variables that have initial values are assigned to the data
                                   section.
```

Figure 5.10 Section Assignment Based on Variable Declaration Placement

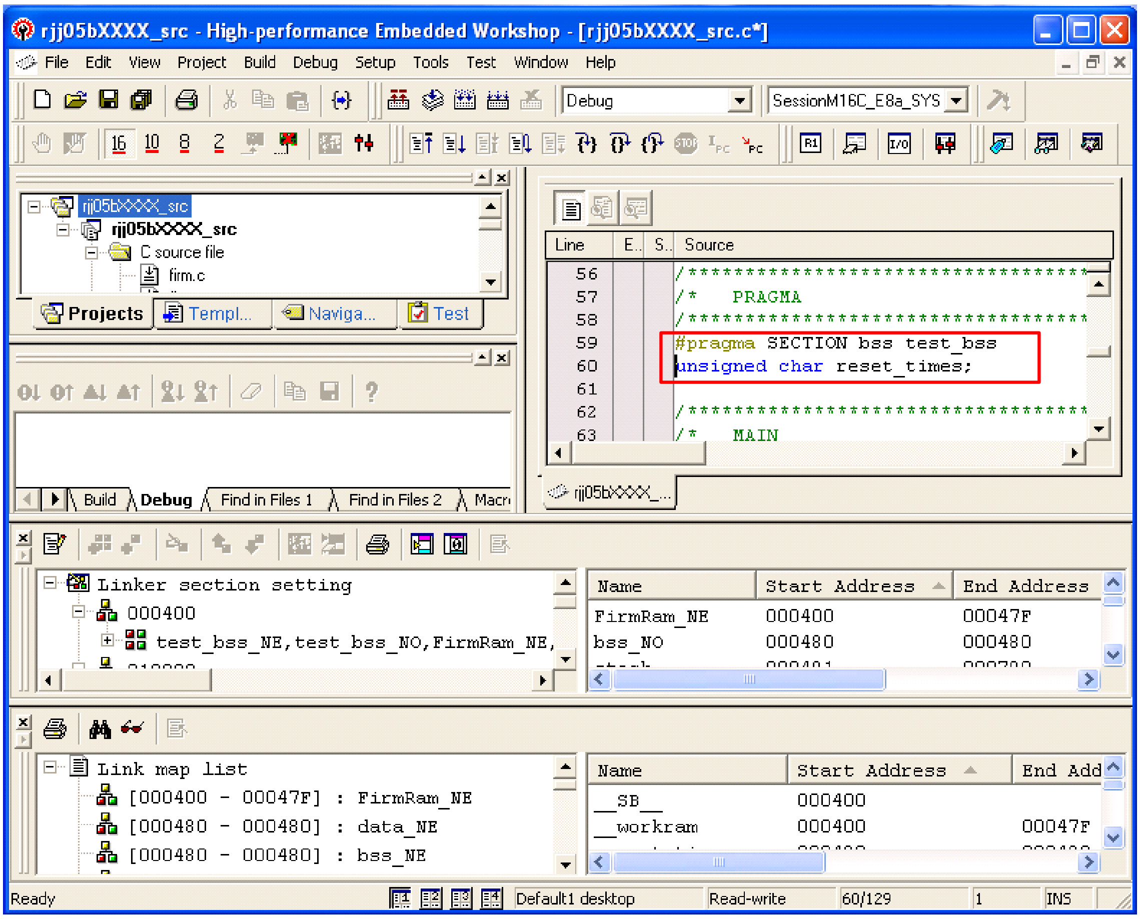


Figure 5.11 Added Section (8/9)

(9) When executing “Build”, variable reset_times is assigned to the test_bss_NO section.

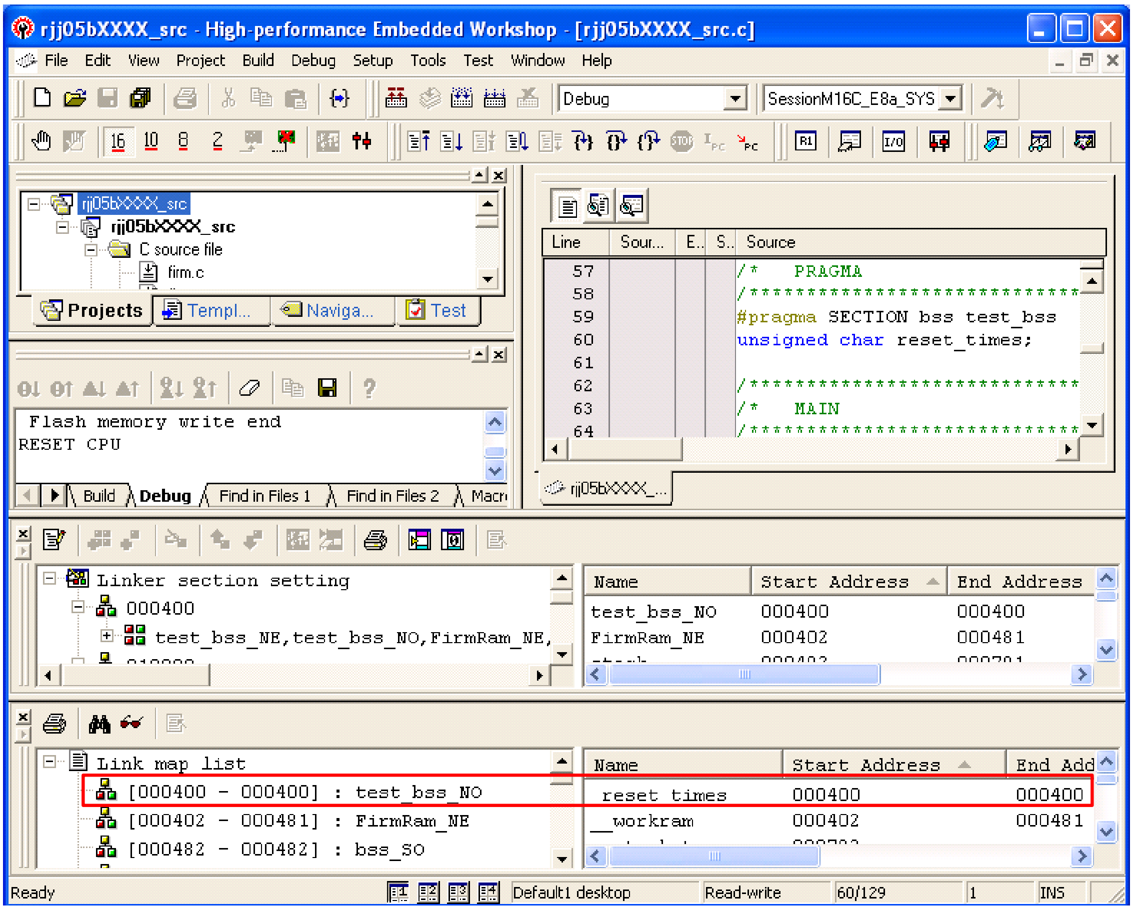


Figure 5.12 Added Section (9/9)

5.3 Using Standard I/O Functions in the Standard Library

The standard I/O functions (1) in the standard library used in the M16C Series have the following restrictions:

- The Address of the U1TB register is defined as 3AAh in the standard I/O functions of the nc30lib.lib. Therefore, the MCUs (M16C/64, 65, etc.) which the address of that are defined as 25Ah are not supported.
- The size which read the receiving buffer register is defined wrongly in the low-level functions for the standard I/O functions (scanf, etc.).

To withdraw the above restrictions, compile the device.c file found in the sample code with the user program, and then link them. (2)

Notes:

1. Standard I/O functions indicate the following the standard input functions and standard output functions:
   Standard input functions: fgetc, getc, getchar, fgets, gets, fread, scanf, fscanf
   Standard output functions: fputc, putc, putchar, fputs, puts, fwrite, printf, fprintf, vfprintf, vprintf
2. Register the device.c file to the project when using the High-performance Embedded Workshop. Replace the device.c file, if it has already existed.

6. Reference Documents

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

C Compiler User's Manual
M16C Series, R8C Family C Compiler Package V.5.45
C Compiler User's Manual Rev.2.00
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

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

Inquiries
http://www.renesas.com/inquiry

Revision History	M16C/64 Group High-performance Embedded Workshop Start-up Program in C

Rev.	Date	Description	
		Page	Summary
1.00	Nov. 30, 2010	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
     In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
     In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different part number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0