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.

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

C Compiler Package for M32R Family

V.4.30 release 00

Release Notes 4th Edition

Renesas Solutions Corp.

February 27, 2008

Welcome to the C Compiler Package for M32R Family (abbreviated as CC32R) V.4.30 Release 00. This document contains supplementary descriptions to the electronic User's Manual.

Please read this release note while you refer to a corresponding item in electronic User's Manual.

Renesas Technology Corp. and Renesas Solutions Corp. reserve the right to change the contents of this release note without notice for improvements on characteristics, etc.

1. Changes from V.4.20 Release 1

The changes from the previous version are as follows.

64-bit integer arithmetic function library added (standard library):

A set of functions to perform 64-bit integer arithmetic operations have been added. As for the integer type in C language, these functions can perform the four fundamental operations in arithmetic, as well as bitwise, shift and compare operations in the 64-bit range.

For details please see "Chapter-13 The set of 64-bit integer arithmetic functions" in M3T-CC32R User's Manual <C Compiler> .

Section specification for the base register facility supported (C compiler):

The "wildcard specification" in access control files has been made possible to add a section specification in it. As a result, by altering the output destination for a symbol to which the base register is applied by using #pragma SECTION beforehand and then specifying the altered section in a section-specified wildcard, it is now possible to apply the base register to multiple symbols collectively.

For details please see "(3) Object registration line, A.1.7.2 The Access Control File Syntax, A.1 Base Register Function" in M3T-CC32R User's Manual <C Compiler> .

Access control file output facility expanded (map generator):

This facility has been improved so that when creating an access control file from the load module, static-declared objects and local variables will also be output.

(Debugging information is required for the creation of access control files.)

Map output during link error supported (linker):

Even when an error occurs during a link operation where a map generation (-M) operation is specified, a temporary link map file will now be generated. Although this is essentially temporary information because the addresses in it differ from those generated during normal link, it will prove effective when analyzing the cause of a link error that occurred for reasons of section allocation, etc.

csv symbol map file output supported (map generator):

A facility to generate a csv format symbol map file has been added. The option to be specified to use this facility is -c or -c16. Because the file is in csv format, it can be input to Microsoft Excel or other spreadsheet software to create a table showing the relationship between addresses and symbols.

(Debugging information is required for the output of csv symbol maps.)

For details please see "Chapter-5 Csv symbol map output, Part-3 Map generator map32R" in M3T-CC32R User's Manual <Assembler> .

S format sort output (load module converter):

The S1, S2 and S3 records (data) can now be output in order of addresses.

(Other records are output to the same lines as conventionally output.)

Warning suppressing option for nested comments added (C compiler):

An option to suppress the generation of a warning for the "/*" or "//" that denotes the beginning of a comment within another comment (`-warn_suppress_nested_comment`) has been added.

Default upper limit of the symbol count was changed (C compiler):

The default upper limit of symbol count of when -XX=... option was not specified was changed to 40,000. (In V.4.20, this number was 10,000.)

Integrated Development Environment HEW 3.01.04 was attached:

HEW that had attached was revised to V.3.01.04.

Furthermore, because the latest version of HEW is 3.01.05 at Sep.1,2004, HEW that is attached to this package is not a latest version. Therefore, please confirm the details of HEW 3.01.05 by referring to the tool news that shows below, and install and update HEW if it is necessary.

RSO-HEW-040801D: Integrated Development Environment HEW Revised to V.3.01.05 (August 1, 2004)
[http://tool-support.renesas.com/eng/toolnews/n040801/tn5.htm]

The eight problems as follows have been fixed:

- A problem on reading out data in memory using pointers, the illegal address may be accessed. (C Compiler)
- A problem on making a change to a variable by using a bit-field and referencing this result, CC32R will generate incorrect code for this reference. (C Compiler)
- The power functions (powf5, powf, and pow) may return a wrong value. (C Standard Library)
- A problem on rebuilding standard libraries in the Windows version, a imperfection library will be made. (Standard Library, PC version Only)
- A problem on floating constants with an exponent part, CC32R will exchange them to incorrect data. (C Compiler)
- A problem on displaying a value less than 0.5 by using a formatted output function such as printf, a incorrect value will be shown. (Standard Library)
- A problem on the same two or more indirection references that use a parameter of a function, the compiler uses the value to which an indirection reference was made before the function call (C Compiler)
- A problem on statements or expressions in parallelism where only the types of operators for the operations of type float are different from each other, the compiler deletes one or more expressions of them (C Compiler)

Erratas of User's Manuals

User's manuals have some mistypes. So, please read these manuals while correcting to the contents as following table.

<< C/C++ Compiler >>

Page	Place	Before Correction	After Correction
CC32R-v (5/434)	Chapter 7 Embedded Applications Programming	7.4 About start-up file start.ms in HEW	7.4 About start-up file start.ms in High-performance Embedded Workshop
cc32R-8 (20/434)	2.2 Compatibility with an old version	(N/A)	Map output during link error supported (-MAP option): Even when an error occurs during a link operation where a map generation (cc32R: -MAP option, lnk32R: -M option) operation is specified, a temporary link map file will now be generated. Although this is essentially temporary information because the addresses in it differ from those generated during normal link, it will prove effective when analyzing the cause of a link error that occurred for reasons of section allocation, etc.

(To be continued to the next page)

(Continued from the previous page)

Page	Place	Before Correction	After Correction
cc32R-10 (22/434)	3.1.3 Command Line Syntax and Rules Figure 3.1 cc32RCommand Line Syntax	-warn_suppressed_nested_comment	-warn_suppress_nested_comment
cc32R-57 (69/434)	4.6 Limitations for C Language Table 4.20 Limitations on C Language Coding (1/2) Declaration / Valid Number of characters in external and internal identifiers and macro names	Up to 31 characters	Up to 240 characters
cc32R-109 (121/434)	7.4 (The title of this section)	7.4 About start-up file start.ms in HEW	7.4 About start-up file start.ms in High-performance Embedded Workshop
cc32R-109 (121/434)	7.4 About start-up file start.ms in HEW Main text	The HEW generates a file "start.ms" when creating new project. This file was modified from one that was using with the TM and the User's Manual.	The High-performance Embedded Workshop generates a file "start.ms" when creating new project. This file was modified from one that was using with the TM and the User's Manual.
cc32R-109 (121/434)	7.4 About start-up file start.ms in HEW Main text	Fundamentally, the contents of these start.ms are nearly equal. However, the start.ms HEW generated can be controled by the assembler as32R with setting following parameter into -D option. If you will modify this start.ms, be careful in this point.	Fundamentally, the contents of these start.ms are nearly equal. However, the start.ms High-performance Embedded Workshop generated can be controled by the assembler as32R with setting following parameter into -D option. If you will modify this start.ms, be careful in this point.
cc32R-109 (121/434)	7.4 About start-up file start.ms in HEW Table 7.3. (The title of this table)	Table 7.3. Meaning of control symbols of start.ms the HEW generated	Table 7.3. Meaning of control symbols of start.ms the High-performance Embedded Workshop generated
cc32R-109 (121/434)	7.4 About start-up file start.ms in HEW Table 7.3. Meaning of control symbols of start.ms the HEW generated	Item name of HEW project Name creating dialog	Item name of High-performance Embedded Workshop project Name creating dialog
cc32R-109 (121/434)	10.2.8 Registers ANSI-C 6.5.1 Storage-class specifiers <The extent to which objects can actually be placed in registers by use of the register storage-class specifier. >	The register storage-class specifier is ignored.	The register storage-class specifier will be accepted, but the all register specified objects may not be allocated for registers. A compiler judges which object should be allocated a register. There is no limit to the number of register declarations.

2. Precautions to be observed when installing

Please pay attention to the following point, in starting installation.

About the license ID

You need to input a license ID in the middle step of installation. Before you start installing CC32R, check your license ID.

Also, if you obtained this version by utilizing the online version up or version up purchasing, see the License ID certificate of the old version of CC32R. (This version can install by using license ID of the version subsequent to V.1.00 Release 4 or later.)

Necessary memory and HDD capacity

In order that CC32R operates comfortably, it requires at least 64 Mbytes of memory and a hard disk drive having 500 Mbytes or more of space.

Precautions about operating environment

The CC32R does not run under Windows 3.1 and Windows NT3.5x or earlier.

Precautions about the file name and directory name

The file name and directory name of the source program must follow the precautions described below:

- Directory and file names that contain multi-byte character (ex. Japanese-Kanji) cannot be used.
- Only one priod (.) can be used in a file name.
- Network path names cannot be used. Assign the path to a drive name.
- Shortcuts cannot be used.
- Directory and file names that contain a space character 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.

When introducing the SQMlint (MISRA C Rule Checker).

Please be sure to install the SQMlint after the CC32R.

If the CC32R is installed after the SQMlint, that cannot use the function of the SQMlint.

For details, refer to the manual or release note that are attached to the SQMlint.

In the case that a old version CC32R was installed.

If a old version CC32R was installed, please uninstall this older CC32R.

To uninstall CC32R, open [Control Panel] - [Add/Remove programs], select "CC32R V.x.xx Release x" (x mean the CC32R version number), and click [Add/Remove] button.

Entering user registration

To be eligible for upgrade information, technical support, and other services, you must be registered as a user with Renesas. Unless you are a registered user, the said services cannot be received.

When you've installed the CC32R, the following file is created.

```
C:\mtool\support\cc32r\regist.txt
```

(It is in the case the directory was chosen in installation supposed as `C:\mtool`.)

Copy all contents of the `regist.txt` file and paste them into a file, then send it to the electronic mail address given below.

```
regist_tool@renesas.com
```

If you are not using an electronic mail, output the contents of the `regist.txt` file to a printer and send the printout to Renesas by facsimile. See the License ID Certificate included with your product to find the address to be sent.

(If you use facsimile, a some number of days may be required until completion of user registration. Please use e-mail, as much as possible.)

For information on our policy concerning the protection of personal information, please refer to the Renesas Technology Homepage.

URL: `http://www.renesas.com/fmwk.jsp?cnt=privacy.htm&fp=/privacy/&site=i`

The information we receive via the User Registration Form aids us greatly in our customer support activities. We provide Renesas Technology and related companies, distributors, etc., with essential user information (electronically or on paper) that will further help them provide customer support.

If you do not wish to have your user information provided to other related companies, please contact us to let us know. Note, however, this will limit our ability to provide complete product support.

Precautions about settings of environment variables

The environment variables listed below must always be set. If one of these variables remains unset or an invalid path is specified, an error may occur when running the software. (These variables do not need to be set when running the CC32R from TM.)

```
M32RLIB
M32RBIN
M32RINC
M32RTMP
```

Precautions about virus check programs

If this software is started while a virus check program is resident in memory, it may not operate properly. In such a case, remove the virus check program from memory before you start the software.

How to get the latest information of CC32R

Renesas has an internet home page in the URL shown below. The latest information on Renesas Development Environmen are published here.

`http://www.renesas.com/en/tools`

3. Installing CC32R

How to install CC32R

Execute the installer by the method of items as follows (A or B).

[A. In the case of installing from the included CD]

Execute each installation program as shown in Table 1, in accordance with OS and language that you are using.
(The drive letter of CD-ROM (Q:) differs each in PC. Please confirm yours.)

Table 1 CC32R file name of installer (The drive letter of CD-ROM is supposed with "Q:".)

Supported OS	Languages	Installer names
Windows 98 / SE Windows Me Windows NT 4.0 Windows 2000 Windows XP	English	Q:\CC32R\WINE\SETUP.EXE
	Japanese	Q:\CC32R\WINJ\SETUP.EXE

[B. In the case of utilizing online version up]

Download installer files and install by the method that is written to the tool homepage.

- In the case of the online version up the installer will ask the directory ("Save files in folder") that receives extracting data at the first step. As for this directory, designate the directory that does not exist whether there is content. If you designate the directory that has some contents, please check them and continue by push "Yes to All", or push "Cancel" and install again with selecting another directory that does not exist.

Follow the messages displayed on the screen as you install CC32R.

[Installing the HEW]

To use CC32R in the HEW, follow the procedure described below to install the HEW from the CC32R installer. (If you already have the HEW or CC32R installed in your computer, uninstall them first.)

- If the HEW installer does not start... If in step (1) the HEW installer does not start, temporarily stop installing CC32R and install only the HEW first. Then invoke the CC32R installer.
- HEW was renewed to 3.01.05 at Aug.1,2004, but HEW that is attached to this package (HEW 3.01.04) is older than it. Therefore, please confirm the details of HEW 3.01.05 by referring to the tool news that shows below, and install and update HEW if it is necessary.
  RSO-HEW-040801D: Integrated Development Environment HEW Revised to V.3.01.05 (August 1, 2004)
  [http://tool-support.renesas.com/eng/toolnews/n040801/tn5.htm]

(1) Launch the CC32R installer, and the installer for the HEW will start up. Follow its instructions as you install the HEW.

(2) The CC32R installer is paused. After confirming that the HEW installer has finished, the installation of CC32R will resume.

[Installing the TM]

A change has been made to enable the installer for TM to be started from the CC32R installer.(If you already have the TM installed in your computer, uninstall it first.)

(1) In the course of installation procedure, you will be asked "Do you wish to start the TM installer?" So press [Y] for Yes to start the TM installer.

(2) The CC32R installer is paused at the dialog with following message: "TM installation is finished, then push OK.".
After confirming that the TM installer has finished, press [OK] to resume the installation of CC32R.

[NOTE]

The data you input in the middle of installation is necessary to create a file for user registration.

Setting PC environment

If you use on DOS prompt, please set environment variables like Table 2.
The environment variables marked by "Auto" in Table 2, you do not need to be set because the installer automatically rewrites AUTOEXEC.BAT.

Table 2 Example of setting PC environment variables

(The directory name of the installation is supposed with "C:\mtool" .)

Environment names	Example of settings
M32RBIN	Auto (SET M32RBIN=C:\mtool\bin32R)
M32RLIB	Auto (SET M32RLIB=C:\mtool\lib32R)
M32RINC	Auto (SET M32RINC=C:\mtool\inc32R)
M32RTMP	Auto (SET M32RTMP=C:\mtool\TMP)
M32RKIN	SET M32RKIN=euc
M32RKOUT	SET M32RKOUT=euc
Command path	Auto (PATH %M32RBIN%;%PATH%)

[NOTES]

- In this setup example, products are installed by "C:\mtool" of the installer. If you wish to install products in a different directory, change the setup contents to the one you want.
- If you use from HEW (High-performance Embedded Workshop) or TM (Tool Manager), you do not need to be set these environment variables.

How to browse the electronic manual

To see these electronic manuals, use a PDF file displaying program, like a "Adobe Acrobat Reader". So install it in your computer as necessary.

- About the Acrobat Reader
  Download "Adobe Acrobat Reader" from following URL of Adobe Systems Incorporated. The latest Acrobat Reader can be downloaded from this home page. Please refer to the following URL about the environment can be use and the latest information for Acrobat Reader.
  http://www.adobe.com/

When using CC32R, there are following two methods for displaying an electronic manual:

1. The electronic manuals are registered in the start menu when installing by default.
   Choose the necessary electronic manual file from menu
   [Start] → [Programs] →[RENESAS-TOOLS] → [CC32R V.x.xx Release x]
   (x expresses version number etc.)

2. Open by double-clicking an electronic manual file
   The electronic manuals are installed in the below C:\mtool\manual by installation directory of default (C:\mtool). Double-click these files to open an electronic manual by Acrobat Reader.It is possible to read manuals, that you open by double-clicking these files (extension .pdf).

   The following Table 3 lists the electronic manual files.

Table 3 Electronic manual file names

Languages	Manual names	PDF file names
English	M3T-CC32R User's Manual < C Compiler > M3T-CC32R User's Manual < Assembler > MAP Viewer User's Manual	CC32Rue.pdf AS32Rue.pdf mapue.pdf
Japanese	M3T-CC32R User's Manual < C Compiler > M3T-CC32R User's Manual < Assembler > MAP Viewer User's Manual	CC32Ruj.pdf AS32Ruj.pdf mapuj.pdf

[NOTE] (About User's Manual of MAP Viewer)

MAP Viewer can be used even if it combines with CC32R. (In this manual, MAP Viewer is used with NC30WA.)

4. Precautions on using of V.4.30 Release 00

There are precautions on usage about this version.

■ On the bitwise shift operator, if the right parameter is unsigned, the compiler may not compile normally. (C Compiler)

Published to RENESAS TOOL NEWS on March 1, 2006: RSO-M3T-CC32R-060301D
(Under preparation)

If there is an integer type constant expression that has bitwise shift operator with an unsigned type on its right parameter in a C source code, the compiler displays a warning message as follows.

"xxxx", line XX: warning: shift count greater than number of bits
(xxxx means a filename, and XX means a line number of a filename.)

If such constant expression is wrote at the size of a bit-field member of a structure, this number is recognized different from original. Otherwise, a compile error may be displayed.

Conditions

This problem occurs if the following conditions are satisfied:

(1) There is an integer type constant expression that has a bitwise shift operator.
(2) In the expression shown by (1), the bitwise shift operator has unsigned integer type at the right parameter.
(3) The expression shown by (1) is used to show one of following (a)-(e) value.
  (a) a bit-field member of a structure
  (b) the size of an array
  (c) the value of an enumeration constant
  (d) the value of a case constant
  (e) a place that shows a numerical value other than the above (a)-(d)

Cautions:

If the condition (3)(a) is satisfied, the compiler regards the bit-field size as not the constant expression value but the size of original type. Therefore, if the warning message above is displayed, please confirm the constant expression that above (1)-(3) suggested is not used at a bit-field member of a structure.

If the codition (3)(b)-(d) is satisfied, a compiler error will occur.

If the codition (3)(e) is satisfied, the warning message above is displayed, but the generated code is correct.

Examples

(1) a bit-field member of a structure
the warning message is displayed.
The size of member b01 is not 4-bits but 16-bits (the size of short type).

[Source file] sample1.c

```
-----------------------------------------------------------------------
struct SBtag {
   short b01:1<<(unsigned)2;      /* Condition(1),(2) and (3a) */
   short b02:1;
};
-----------------------------------------------------------------------
```

(2) the size of an array
The warning messages and the error message as follows are displayed.

```
error: array: subscript must be positive, non-zero, integral value
```

[Source file] sample2.c

```
-----------------------------------------------------------------------
int array[8>>1u];              /* Condition(1),(2) and (3b) */
-----------------------------------------------------------------------
```

(3) the value of an enumeration constant

The warning messages and the error message as follows are displayed.

```
error: enumeration-constant out of range
```

[Source file] sample3.c

```
-----------------------------------------------------------------------
enum Etag {
   A,
   B = 2<<(unsigned)2          /* Condition(1),(2) and (3c) */
};
-----------------------------------------------------------------------
```

(4) the value of a case constant

The warning messages and the error message as follows are displayed.

```
error: unable to evaluate case label (out of range?)
```

[Source file] sample4.c

```
-----------------------------------------------------------------------
int func4(int key)
{
   switch (key) {
     case ((3*8)>>(7UL-5)):      /* Condition(1),(2) and (3d) */
       return 1;
   }
   return 0;
}
-----------------------------------------------------------------------
```

(5) a place that shows a numerical value other than the above example 1 to 4.

The warning message is displayed, but the generated code is correct.

[Source file] sample5.c

```
-----------------------------------------------------------------------
int data = 0x8000>>12u;          /* Condition(1),(2) and (3e) */
-----------------------------------------------------------------------
```

Solutions

Append a (signed) cast at the right parameter of shift operator, in the following methods.

[Circumvention of source file sample1.c]

```
-----------------------------------------------------------------------
struct SBtag {
   short b01:1<<(signed)(unsigned)2;  /* Append (signed) cast */
   short b02:1;
};
-----------------------------------------------------------------------
```

[Circumvention of source file sample2.c]

```
-----------------------------------------------------------------------
int array[8>>(signed)1u];           /* Append (signed) cast */
-----------------------------------------------------------------------
```

[Circumvention of source file sample3.c]

```
-----------------------------------------------------------------------
enum Etag {
   A,
   B = 2<<(signed)(unsigned)2        /* Append (signed) cast */
};
-----------------------------------------------------------------------
```

[Circumvention of source file sample4.c]

```
-----------------------------------------------------------------------
int func4(int key)
{
   switch (key) {
     case ((3*8)>>(signed)(7UL-5)):      /* Append (signed) cast */
       return 1;
   }
   return 0;
}
-----------------------------------------------------------------------
```

[Circumvention of source file sample5.c]

```
int data = 0x8000>>(signed)12u;        /* Append (signed) cast */
```

■ On setting the address value of an object containing a structure, array or union using the #pragma ADDRESS directive, CC32R may generate incorrect code for this reference. (C Compiler)

Published to RENESAS TOOL NEWS on February 16, 2005: RSO-M3T-CC32R-050216D

Consider that a #pragma ADDRESS directive is used for defining the address of an object that is an array or structure; or a union that has members of type array or structure.

If a member or element placed at any address except the beginning address of the object is accessed (read or written) or referenced by an address operator (&), incorrect code is generated using the address of the member or element placed at the beginning of the object.

Conditions

This problem occurs if the following conditions are satisfied:

(1) Any of the following three objects is declared outside of a function:
   (a) A structure
   (b) An array
   (c) A union having members of type array or structure

(2) A #pragma ADDRESS directive is used for defining the object in (1).

(3) For a member or element placed at any address except the beginning address of the object, either of the following is performed:
   (a) Accessing (reading or writing)
   (b) Referencing by an address operator (&)

Examples

[Source file] sample1.c

```
#pragma ADDRESS data1 0x10000      /* Condition (2) */
struct stg1 {
   int a;
   int b;
} data1;                      /* Condition (1)-(a) */

void func1(void)
{
   data1.b = 3;                /* Condition (3)-(a) */
}
```

[Source file] sample2.c

```
#pragma ADDRESS data2 0x20000      /* Condition (2) */
short data2[10];               /* Condition (1)-(b) */
short *d2;

void func2(void)
{
   d2 = &data2[3];             /* Condition (3)-(b) */
}
```

[Source file example 3] sample3.c

```
union utg3 {
   double a[10];
   struct {
      int b;
```

```
      double c[5];
   } d;
} data3;                        /* Condition (1)-(c) */

#pragma ADDRESS data3 0x30000      /* Condition (2) */

void func3(int i)
{
   data3.d.c[i] =              /* Condition (3)-(a) */
      data3.a[4];             /* Condition (3)-(a) */
}
```

Solutions

Access the object using the address cast to a pointer. It is convenient to use macros shown in the examples below:

[Modification of sample1.c]

```
#define data1 ( *(struct stg1*) 0x10000 )  /* Converted to macro */

struct stg1 {
   int a;
   int b;
} /* data1 */ ;      /* Only stg1 declared; data1 commented out */
void func1(void)
{
   data1.b = 3;     /* Replaced by macro */
}
```

[Modification of sample2.c]

```
#define data2 ( (short *) 0x20000)    /* Converted to macro */

/* short data2[10]; */      /* data2 commented out */
short *d2;

void func2(void)
{
   d2 = &data2[3];         /* Replaced by macro */
}
```

[Modification of sample3.c]

```
union utg3 {
   double a[10];
   struct {
      int b;
      double c[5];
   } d;
} /* data3 */ ;       /* Only utg3 declared; data3 commented out */

#define data3 ( *(union utg3*) 0x30000 )  /* Converted to macro */

void func3(int i)
{
   data3.d.c[i] =                /* Replaced by macro */
      data3.a[4];               /* Replaced by macro */
}
```

- On making a function call using a pointer pointing to a function whose type has been converted by a cast operator, CC32R may describe an error. (C Compiler)

  Published to RENESAS TOOL NEWS on December 1, 2004: RSO-M3T-CC32R-041201D

When a function call is made using a pointer pointing to a function whose type has been converted by casting, the following error may arise even if both types of each parameter and its argument match with each other:

```
error: type of argument does not match with prototype
```

Conditions

This problem occurs if the following conditions are all satisfied:

(1) The type of a pointer or integer is converted to another pointer pointing to a function by casting, and a function call is made at the same time.
(2) At least one parameter of the function to be called is array type.

Examples

[Source file] sample1.c

```
short array[3];

void foo1(void *ptr1)

{
   (*(void(*)(int, short arr[]))ptr1)(2,array);
                                /* Conditions (1) and (2) */
}
```

[Source file] sample2.c

```
typedef struct AAA TYPE_A;
char *ptr2;

int foo2(TYPE_A array[][3])
{
   int ans;
   ans = (*(int(*)(TYPE_A arr[][3]))ptr2)(array);
                                /* Conditions (1) and (2) */
   return ans;
}
```

[Source file example 3] sample3.c

```
typedef float TYPE_F[3];
TYPE_F array;

void foo3(void)
{
   (*(void(*)(TYPE_F))0x123400)(array);
                                /* Conditions (1) and (2) */
}
```

Solutions

When making a function call using a pointer pointing to a function whose type has been converted by casting, once save the result of casting in another pointer variable, and then make a function call using this variable.

[Modification of sample1.c]

```
short array[3];

void foo1(void *ptr1)
{
   void (*callptr1)(int, short arr[])
              /* Define a pointer variable "callptr1" */
```

```
     = (void (*)(int,short arr[]))ptr1;
                  /* Save the cast result in "callptr1" */
   (*callptr1)(2,array);     /* Call a function using "callptr1" */
}
   ------------------------------------------------------------------
```

[Modification of sample2.c]

```
   ------------------------------------------------------------------
typedef struct AAA TYPE_A;
char *ptr2;

int foo2(TYPE_A array[][3])
{
   int ans;
   int (*callptr2)(TYPE_A arr[][3]);
                  /* Define a pointer variable "callptr2" */
   callptr2 = (int(*)(TYPE_A arr[][3]))ptr2;
                  /* Save the cast result in "callptr2" */
   ans = (*callptr2)(array);  /* Call a function using "callptr2" */
   return ans;
}
   ------------------------------------------------------------------
```

[Modification of sample3.c]

```
   ------------------------------------------------------------------
typedef float TYPE_F[3];
void *ptr3;
TYPE_F array;

void foo3(void)
{
   void (*callptr3)(TYPE_F);
                  /* Define a pointer variable "callptr3" */
   callptr3 = (void(*)(TYPE_F))0x123400;
                  /* Save the cast result in "callptr3" */
   (*callptr3)(array);     /* Call a function using "callptr3" */
}
   ------------------------------------------------------------------
```

- On initializing a two-dimensional array of type char using an initializer, CC32R may generate incorrect code for this reference. (C Compiler)

Published to RENESAS TOOL NEWS on September 16, 2004: RSO-M3T-CC32R-040916D

If a two-dimensional array of type char is declared and at the same time initialized using a specific type of initializer that contains string literals, the compiler will tell the following error message:

`cg32r: "xxxx", line XX: internal error: illegal IL, size of initializer is larger than name size.`
("xxxx" means a source file name.)

Conditions

This problem occurs if the following conditions are all satisfied:

(1) A two-dimensional array of type char (except for type pointer) is declared with an initializer.
(2) In the declaration of the array in (1), its size is neglected.
(3) The initializer in (1) contains two or more string literals (for example, "ab" and "cd").
(4) Among the string literals in (3), the first is enclosed with braces.

Examples

(1) Statically Initialized Array
[Source file] sample1.c

```
   ------------------------------------------------------------------
  char array1[][2] = {      /* Conditions (1) and (2) */
```

```
    {"ab"},             /* Conditions (3) and (4) */
     "cd",              /* Condition (3) */
     "ef"               /* Condition (3) */
};
```

(2) Statically Initialized Array

[Source file] sample2.c

```
extern void array_func(char [][6]);
void func2(void)
{
   char array2[][6] = {   /* Conditions (1) and (2) */
      {"5678"},         /* Conditions (3) and (4) */
      {"1234"},         /* Condition (3) */
   };
   array_func(array2);
}
```

Solutions

This problem can be circumvented in either of the following ways:

(1) Specify the size of the array.

[Circumvention of source file sample1.c]

```
char array1[3][2] = {     /* Size of 3 specified */
    {"ab"},
    "cd",
    "ef"
};
```

[Circumvention of source file sample2.c]

```
extern void array_func(char [][6]);
void func2(void)
{
   char array2[2][6] = {  /* Size of 2 specified */
      {"5678"},
      {"1234"},
   };
   array_func(array2);
}
```

(2) Remove the braces enclosing the first string literal.

[Circumvention of source file sample1.c]

```
char array1[][2] = {
    "ab",        /* Braces removed */
    "cd",
    "ef"
};
```

[Circumvention of source file sample2.c]

```
extern void array_func(char [][6]);
void func2(void)
{
   char array2[][6] = {
      "5678",    /* Braces removed */
      {"1234"},
   };
   array_func(array2);
}
```

■ On using an assignment expression whose left term is of a shorter data type than the right term and after which is placed an expression that is equivalent to the right term of the assignment expression, CC32R may generate incorrect code for this reference. (C Compiler)

Published to RENESAS TOOL NEWS on January 16, 2008: 080116/tn1

Consider the case where an assignment expression whose left term is of a shorter data type than the right term exists, and after it is placed another expression that contains an expression equivalent to the right term of the assignment expression.

When the second expression is evaluated, such a value is used that is obtained by casting the operation result of the right term of the assignment expression to the shorter data type; not the above operation result itself.

Conditions

This problem may occur if the following conditions are all satisfied:

(1) The optimizing option used in compilation meets either (a) or (b) shown below.
(a) Option -O, -O2, -O3, -O6, or -O7 is used.
(b) Option -Ospace or -Otime is used with -O0, -O1, -O4, and -O5 not used.

(2) An expression exists which contains more than one variable of an integral type and whose operation result is also of an integral type.
The expression can contain only one variable of an integral type.

(3) An assignment expression exists in which the operation result of the expression in (2) is assigned to an auto variable of a shorter integral type than the operation result in (2).

(4) An expression equivalent to the expression in (2) is used as a part of another expression evaluated after the assignment expression in (3).

(5) Between the assignment in (3) and the use of an expression in (4) exists no possibility of changing the values of any variables contained in the expression in (2).

(6) The auto variable in (3) is saved on the stack; not on a register.

(7) The operation result of the expression in (2) is saved on a register. This is valid when the expression contains only one variable.

Notice:

Whether the objects in Conditions (6) and (7) are saved on the stack or a register depends on your compiler and is not determined by the source code. It should be checked using the code generated after compilation.

Example

In the operation of expression val_ulong >> 8 below is used the value of val_ulong after assigned to val_uchar. So the operation result of this expression becomes incorrect. However, this problem may not arise depending on the descriptions of the program lines that are omitted below.

```
------------------------------------------------------------------------
unsigned long func_ulong(unsigned short);
void func(void)
{
   unsigned long  val_ulong;
   unsigned char  val_uchar;
   unsigned int   val_another;
   unsigned short val_ushort;
   /*
 . . . . . . . . . . . . . . . . . . . . . . . . .
    */
   val_ulong   = func_ulong(val_ushort);
   val_uchar   = val_ulong;      /* Conditions (2) and (3) */
   val_another = val_ulong >> 8;  /* Conditions (4) and (5) */
```

```
    /*
  . . . . . . . . . . . . . . . . . . . . . . . .
     */
}
```

Solutions

Avoid this problem in either of the following ways:

(1) Suppress the optimization in Level 2.

If you are using option -O, -O2, -O3, -O6, or -O7, replace it with -O0, -O1, -O4, or -O5. If you are using -Ospace or -Otime, use -O0, -O1, -O4, or -O5 at the same time.

(2) Exchange the expressions in Conditions (3) and (4) in their order.

If you are using option -O, -O2, -O3, -O6, or -O7, replace it with -O0, -O1, -O4, or -O5. If you are using -Ospace or -Otime, use -O0, -O1, -O4, or -O5 at the same time.

Example modified:

```
unsigned long func_ulong(unsigned short);
void func(void)
{
 . . . . . . . . . . . . . . . . . . . . . . . .
    val_ulong   = func_ulong(val_ushort);
    val_another = val_ulong >> 8;  /* Conditions (4) and (5) */
    val_uchar   = val_ulong;      /* Conditions (2) and (3) */
 . . . . . . . . . . . . . . . . . . . . . . . .
}
```

For a reference

This problem has already been fixed in the V.5.01 Release 00 package. So use the latest version of the package. Note that the package V.4.30 Release 00 or earlier cannot be updated to the latest version.

5. Software table

Table 4 shows the look of the directory and file that are made after installation.

Table 4 Table of the directory and file after installation

Directory names	File names	Notes
bin32R	cc32R.exe	Compile driver (V.2.11.00.000)
	as32R.exe	Assemble driver (V.2.03.00.000)
	lnk32R.exe	Linker (V.1.11.00.000)
	lib32R.exe	Librarian (V.1.02.01.000)
	lmc32R.exe	Load module converter (V.1.12.00.000
	map32R.exe	Map generator (V.1.21.00.000)
lib32R	cpre.exe	Preprocessor (V.2.06.00.000)
	cfrt.exe	Parser (V.2.23.00.000)
	postpar.exe	Post parser (V.1.00.01.000)
	opt.exe	Optimizer (V.1.25.00.000)
	cg32R.exe	Code generator (V.4.03.00.000)
	a032R.exe	Macro processor (V.1.00.01.000)
	a132R.exe	Assembler (V.4.04.01.000)
	alis32R.exe	List processor (V.1.01.00.000)
	parafilt.exe	Parallel processor (V.1.00.02.000)
	cmerge.exe	C source merge processor (V.1.01.01.000)
	m32RcR.lib	C Library (Small model)
	m32RcRM.lib	C Library (Medium model)
	m32RcRL.lib	C Library (Large model)
	m32RcR.stk	Stack utilize display file for C library (for m32RcR.lib)
	m32RcRM.stk	Stack utilize display file for C library (for m32RcRM.lib)
	m32RcRL.stk	Stack utilize display file for C library (for m32RcRL.lib)
inc32R	assert.h, ctype.h, errno.h, float.h, limits.h, locale.h, math.h, setjmp.h, signal.h, stdarg.h, stddef.h, stdio.h, stdlib.h, string.h, time.h	C Library headers
	cstddef, cstdio, cstdlib, exception, new, stdexcept, typeinfo, new_ecpp.h, new_std.h	Reserved C Library headers)+2*!
inc32R\sys	assert.h, ctype.h, errno.h, float.h, limits.h, locale.h, math.h, setjmp.h, signal.h, stdarg.h, stddef.h, stdio.h, stdlib.h, string.h, time.h	System definition headers)+3*
inc32R\com	ANSI_errno.h, def.h, SBPP	
UnSpt32R)+4*	strip32R.exe	Debug information discarding utility (V.1.00.00.000)
	abslist.exe	Absolute listing utility (V.1.00.05.000)
	stk32R.exe	Stack size calculation utility (V.1.00.00.000)
	license.txt	Development support utility guide (English)
	license.sj	Development support utility guide (Japanese)
	strip32R.txt	Utility manual of strip32R (English)
	abslist.txt	Utility manual of abslist (English)
	stk.txt	Utility manual of stk32R (English)
	strip32R.sj	Utility manual of strip32R (Japanese)
	abslist.sj	Utility manual of abslist (Japanese)
	stk.sj	Utility manual of stk32R (Japanese)
lib32R\src	~.c ~.h ~.ms floatlow\~.mo stack/~.stk	C Library source files
	BUILD.bat CLEAN.bat	C Library build utilities
support/cc32r	userinfo.txt regist.txt	Product information record files
smp32R	start.ms	Startup, Low-level functions example
lib32R	tmcpp.exe	Preprocessor for TM (V.4.02.00)

	tmcpp.opt	tmcpp option setting file for TM
	cc32r.mkt	Makefile basic setting file for TM
	cc32r.opt	cc32R option setting file for TM
	as32r.opt	as32R option setting file for TM
	lnk32r.opt	lnk32R option setting file for TM
	lib32r.opt	lib32R option setting file for TM
	lmc32r.opt	lmc32R option setting file for TM
	map32r.opt	map32R option setting file for TM
bin	mapviewer.exe	Map Viewer (V.3.00.00)
	map_inspect.dll	DLL file for Map Viewer
	mapviewer.hlp	Help file for Map Viewer
	mapviewer.cnt	Help setting file for Map Viewer
manual	CC32Rue.pdf	User's manual < C Compiler > [English]
	AS32Rue.pdf	User's manual < C Compiler > [English]
	CC32Ruj.pdf	User's manual < C Compiler > [Japanese]
	AS32Ruj.pdf	User's manual < C Compiler > [Japanese]
	mapuj.pdf	Map Viewer Manual [English or Japanese (will be selected by the installer)]

NOTES

+2/ The Reserved C Library headers is reserved for future expansion of next CC32R. Then these headers can not be used now, please do not include them from your C program.

+3/ The System definition headers are unable to be deleted and modified, because CC32R refers to them in compiling.
If these files are deleted or changed CC32R does not run normally.

+4/ The program in the UnSpt32R directory differ from the constitution things of other CC32R regarding the handling of the license and support. Please confirm the document file "license.txt" in this directory.