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

C Compiler Package for M32C Series V.5.42 Release 00

Release note

(Rev.1.00)

Renesas Solutions Corporation

Apr 1, 2010

Abstract

Welcome to C Compiler Package for the M32C Series V.5.42 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. About Installation of C compiler Package

For details on how to install, please refer to “Install Guide”.

2. The latest information

Please refer to the following for the latest information on this product.
http://tool-support.renesas.com/eng/toolnews/p_m16c80_1.htm

3. Precautions on Product

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

3.1. About Integrated Development Environment TM

TM does NOT support M3T-NC30WA V.5.42 Release 01.
Therefore, the following cannot be specified.
(1) Create a new project of M3T-NC30WA V.5.42 Release 01 with TM
(2) Port the projects created by TM to High-performance Embedded Workshop

Please refer to “C Compiler Package Guidebook” for the method of porting the project created by TM to High-performance Embedded Workshop.

3.2. Suggestions Concerning File Names

The file names ,directory names and Workspace[1] names that can be specified are subject to the following restrictions:

- The directory, file, or workspace name which comprised of ASCII character-code only can 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.
- Shortcut cannot be used.
- The "..." symbol cannot be used as a means of specifying two or more directories.

If the limitations above are violated, the following problems may occur.

- The value set by the assembler directive commands .id, .ofsreg, .protect,.rvector or .svector cannot operate correctly. As a result, the ID code and the option function select register may not be set correctly.
- Call Walker to refer to the stack size is not displayed correctly.
- The MAP Viewer to refer to the map information in the absolute module file isn't displayed correctly.
- The setting by these assembler directive commands isn't displayed in .map file.
- A compile error like "Can't open file" arises.
- A message like "Because a problem occurred, lnxx.exe is terminated." is issued and then the linker is terminated abnormally.
- The file name length including the path should be less than 128 characters.

[1] Workspace is a working directory used for processing like the compilation,build or debugging on High-performance Embedded Workshop.

3.3. Precautions about virus check programs

If the virus detection program is memory-resident in your computer, M3T-NC308WA may not start up normally. In such a case, remove the virus detection program from memory before you start M3T-NC308WA.

3.4. Precautions on M32C Series-Dependent Code

3.4.1. Precautions 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.

```
#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
        {
                asm();
        }                                  // Returns 0 to TA0IC.IR when is become 1
}
```

- Notes

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

3.4.2. 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.

3.5. Precautions about Compiler, Assembler, Linkage Editor and Utilities

3.5.1. Prication concerning operation of Integer constant

When the operation result of Integer constant exceeds the width of the bit of the int type, the compiler generates a code different before V.5.40 Release 00.

- Example

```
void    func(void)
{
        unsigned long    l;

        l = 256 * 256;   // l=0           (V.5.41 Release 00)
}                        // l=65536       (V.5.40 Release 00)
```

- Workaround

  Please add "Integer suffix" to "Integer constant" when the operation result of "Integer constant" exceeds the width of the bit of the int type.

```
void    func(void)
{
        unsigned long    l;

        l = 256UL * 256UL;
}
```

3.5.2. Precautions on String Instructions and Sum-of-Products Instructions

From V.5.41 Release 00 on, unless the compile option "-fuse_strings (-fUS) " is selected, no codes are generated that use the string instructions of the M32C series. Similarly, unless the compile option "-fuse_product_sum

(-fUPS) " is selected, no codes are generated that use the sum-of-products instructions of the M32C series.

For details, refer to the following Renesas Technical Update:

- M16C/70 Series, M16C80 Series, M32C/80 Series, M32C/90 Series: Usage Precaution for String Instruction, Product Sum Operation Instruction (Document No.TN-16C-A157A/E)

MPlease be sure to select the compile option "-fuse_strings (-fUS)" to generate codes that use the string instructions of the M32C series, or the compile option "-fuse_product_sum (-fUPS) " to generate codes that use the sum-of-products instructions of the said microcomputer series.

3.5.3. Precautions on macro definition

If the name of a macro itself is used in the content of a macro definition and the defined macro is specified in an argument to other function-like macro, macro replacement cannot be executed correctly.

- Example

```
int     a = 10;
#define a       a + a                   // macro name ‘a’
#define p( x,y ) x + y

void    func(void)
{
        int     i = p ( a , a );        // results in i = 80
}                                       // i = 40 is correct
```

- Workaround

  Make sure the macros passed to the arguments to function-like macros are defined with a name that is not used in the macro definition.

```
int     a = 10;
#define b       a + a                   // Change to a macro name that is not 'a'
#define p( x,y ) x + y

void    func(void)
{
        int     i = p ( b , b );
}
```

3.5.4. Precautions on #if preprocessing directive

If a constant expression of #if directive is a shift whose left operand is a negative value and right operand is a value of unsigned type, the result of the shift cannot be determined to be good or not correctly.

- Example

```
void    func( void )
{
        char    a;

#if  (-1 << 1U ) > 0            // Determined to be true
        a = 1;                  // (–1 << 1U) is –2, so that it correctly is false
#else
        a = 2;
#endif
}
```

- Workaround

  If the left operand of a shift is a negative value, change the right operand of that shift to a value of signed type.

```
void    func( void )
{
        char    a;

#if  (-1 << 1 ) > 0       // Disuse of the suffix U changes the right operand of
        a = 1;            //  a shift to signed type.
#else
        a = 2;
#endif
}
```

3.5.5. Precautions on nesting inline functions

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:

  [3] n inline function is nested in another.

  [4] 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.

3.5.6. Precautions about command option "-I"

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

3.5.7. 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.

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

(1) 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.

(2) 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.

3.5.9. 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.

3.5.10. 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.

3.5.11. Precaution for Assembler start-up files (ncrt0.a30, sect308.inc)

The content of start-up files may be customized depending on the target MCU or application.
Please refer to the hardware manual or the datasheet of the target MCU when undergoing such customizations.

3.5.12. Precautions about the search of an include file

If you specify a file to include 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 found.

Example

```
#include "c:¥user¥test¥sample.h"
main(){}

C:¥user>nc308    ¥user¥test2¥sample.c -silent ¥user2¥tm_test¥aa.c
[Error(cpp308.21):¥user2¥test2¥sample.c, line 1] include file not found 'c:¥user¥test¥sample.h'
```

3.5.13. Precaution about utl30

C compiler user’s manual “Appendix G the SBDATA declaration & SPECIAL page Function declaration utility (utl308)” on Page 357 has the statement that “Includes, during startup (sect308.inc), the SPECIAL Page vector definition file (special.inc) as a file to be included”. But, this is the explanation for the version older than V.5.40. The SPECIAL Page vector definition file is unnecessary in V.5.40 or later. Therefore, please do not use it.

3.5.14. Precaution about MapViewer

As you cannot use Online Help of the MapViewer with a PC running Windows Vista(R), please use that of the EcxMap and CallWalker instead.

3.5.15. Precaution about malloc(), calloc() and realloc()

Memory management function malloc ,calloc and realloc of the NC308WA cannot secure the area of 64KB or more at a time.

4. Content of upgrade in V.5.42 Release 00

4.1.1. Problems repair

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

(1) With errors arising after linking is performed
http://tool-support.renesas.com/eng/toolnews/091116/tn3.htm

(2) With calculating stack usage
http://tool-support.renesas.com/eng/toolnews/091116/tn2.htm

(3) With the function for automatically generating variable vector tables
http://tool-support.renesas.com/eng/toolnews/091001/tn4.htm

(4)With adding more than one relocatable file
http://tool-support.renesas.com/eng/toolnews/070416/tn3.htm

(5) With using the #pragma DMAC preprocessing directive in C compilers
http://tool-support.renesas.com/eng/toolnews/081016/tn8.htm

(6) With initializing a member of a structure using an expression containing the sizeof operator
http://tool-support.renesas.com/eng/toolnews/080716/tn2.htm

(7) With using a volatile-qualified variable in the bit-wise and operation
http://tool-support.renesas.com/eng/toolnews/080616/tn4.htm

(8) With using a variable volatile-qualified with the indirect member operator
http://tool-support.renesas.com/eng/toolnews/080616/tn3.htm

(9) With comparing a bit field variable with 1 or 0
http://tool-support.renesas.com/eng/toolnews/080616/tn2.htm

(10) With using a volatile-qualified variable in the for or while statement
http://tool-support.renesas.com/eng/toolnews/080616/tn1.htm

(11)With issuing preprocessing directive #pragma DMAC to extend the C compiler's functionality
http://tool-support.renesas.com/eng/toolnews/080416/tn1.htm

(12) With using preprocess command #error
http://tool-support.renesas.com/eng/toolnews/080301/tn2.htm

(13) With using options to optimize jump instructions at linking
http://tool-support.renesas.com/eng/toolnews/070716/tn2.htm

(14)On arguments are passed to functions via the stack
http://tool-support.renesas.com/eng/toolnews/070316/tn5.htm

(15) With using a C-language start-up file
http://tool-support.renesas.com/eng/toolnews/070316/tn4.htm

(16) With using the startup file in C language
http://tool-support.renesas.com/eng/toolnews/070216/tn4.htm

(17) On placing functions in sections
http://tool-support.renesas.com/eng/toolnews/n051101/tn8.htm

(18) On nesting inline function
http://tool-support.renesas.com/eng/toolnews/n041116/tn6.htm

5. Content of upgrade in V.5.41 Release 01

5.1. Content of upgrade about C compiler

5.1.1. Problems repair

Improvements have been made to all of the following precaution:

(1) When compilation option "-fdouble_32(-fD32) " is selected and compiled, two error messages "nc308:Invalid

option 'xxx' " are output.

5.2. Content of upgrade about Assembler system

5.2.1. Problems repair

Improvements have been made to all of the following precaution that had been informed to you by tool news:
(1) With automatic generation of variable interrupt- and special page vector tables

6. Content of upgrade in V.5.41 Release 00

6.1. Content of upgrade about C compiler

6.1.1. Function addition

6.1.1.1. Addition of compilation option "-fuse_strings (-fUS)"

The compile option "-fuse_strings (-fUS)" to generate codes that use the string instructions has been added.
To select this option, refer to the Renesas Technical Update shown below.

- M16C/70 Series, M16C80 Series, M32C/80 Series, M32C/90 Series: Usage Precaution for String Instruction, Product Sum Operation Instruction (Document No.TN-16C-A157A/E)

6.1.1.2. Addition of compilation option "-fuse_product_sum (-fUPS)"

The compile option "-fuse_product_sum (-fUPS)" to generate codes that use the sum-of-products instructions has been added.
To select this option, refer to the Renesas Technical Update shown below.

- M16C/70 Series, M16C80 Series, M32C/80 Series, M32C/90 Series: Usage Precaution for String Instruction, Product Sum Operation Instruction (Document No.TN-16C-A157A/E)

6.1.2. Function improvement

6.1.2.1. Improvement of Assembler macro function and strcmp function

Regarding the assembler macro functions and strcmp function, corrective codes have been applied for the Renesas Technical Update shown below.

- M16C/70 Series, M16C80 Series, M32C/80 Series, M32C/90 Series: Usage Precaution for String Instruction, Product Sum Operation Instruction (Document No.TN-16C-A157A/E)

6.1.2.2. Improvement of compilation speed

The compilation time when the optimization option "-0[3–5]", "-OR", "-OS", "-OR_MAX" or "-OS_MAX" is selected has been sped up.

6.1.3. Problems repair

Improvements have been made to all of the following precaution that had been informed to you by tool news:
(1) On using the MISRA C rule checker SQMlint
(2) On accessing an array variable in the loop of a for statement
(3) On using the extended function #pragma SPECIAL

6.2. Content of upgrade about Assembler system

6.2.1. Function addition

6.2.1.1. Addition of link option "-NOMCU"

A link option "-NOMCU" has been added that enables linking of relocatable object files for a microcomputer series that differs from the target microcomputer of a project.
Note, however, that this link option changes link rules. For this reason, the linker will not detect an error even when any unusable instruction for a particular microcomputer series exists. Please be aware of it when you choose this option.

6.2.1.2. Addition of generating ID files using the ".ID" assembler directive command

A function of generating ID files using the ".ID" assembler directive command has been introduced.
So ID files can be generated by the ".ID" assembler directive command as well as the existing "-ID" option of the load module converter.
According to the introduction of ".ID", changes have been made to the specifications for using the assembler directive commands ".ID" and ".PROTECT", and the options of the load module converter "-ID", "-protect1", and "-protectx" in combination with each others.

6.2.2. Function improvement

6.2.2.1. Addition of assembling error processing

A revision has been made so that if the assembler option "-M82" or "-M90" is set or any addressing mode unusable in the "MUL.W" and "MULU.W" instructions is specified for dest of those instructions, an assemble error will be assumed.
For details, refer to the following Renesas Technical Update:

- M32C/80 Series, M32C/90 Series: Usage Precaution for MUL.W Instruction, MULU.W Instruction (Document No.TN-16C-A156A/E)

6.3. Content of upgrade about Integrated Development Environment

6.3.1. Update

The High-performance Embedded Workshop included with this C compiler package has been updated to V.4.01.01.

6.4. Content of upgrade about utility tool

6.4.1. Function addition

6.4.1.1. The .sni File Generation Tool gensni for Call Walker

The tool named gensni to generate the input files (.sni) for Call Walker from absolute module files (.x30) has been added.

6.4.1.2. The .map File Generation Tool genmap for the Map Section Information Window

The tool named genmap to generate the input files (.map) for the Map Section Information window of the High-performance Embedded Workshop from absolute module files (.x30) has been added.

6.5. Other

6.5.1. Function addition

6.5.1.1. Support for Call Walker

The C compiler now supports the utility tool named "Call Walker" that calculates the stack size used in an application program.

6.5.1.2. Support for the Map Section Information Window of the High-performance Embedded Workshop

The C compiler now supports the Map Section Information Window of the High-performance Embedded Workshop that shows the map information of absolute module files.

7. Software version list of C Compiler Package

The following lists the software items and their versions include with C Compiler Package V.5.42 Release 00.

● nc308	V.1.08.07.001
● cpp308	V.4.09.00.000
● ccom308	V.5.05.01.001
● aopt308	V.1.05.02.001
● as308	V.4.04.00.001
● mac308	V.2.22.00.000
● pre30	V.1.10.12
● asp308	V.4.03.00.000
● ln308	V.4.05.00.001
● lb308	V.2.02.01.000
● lmc308	V.3.00.02.000
● xrf308	V.2.02.00.000
● abs308	V.1.03.00.000
● utl308	V.1.00.11.001
● MapViewer	V.3.01.02
● genmap	V.1.00.01.001
● gensni	V.1.00.00.002

8. 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 [2] are against the MISRA C Rule NOTE, but these violations do not constitute a drawback to any operation.

8.1. Cause of Rule Violation

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

(1) C-Compiler specifications (near/far modifier, asm () function and #pragma)

(2) Declaration of function based on ANSI Standard

(3) The evaluation sequence in the conditional statement is not described explicitly, using a parenthesis.

(4) Implicit type conversion

8.2. 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								

8.3. Evaluation Environment

C Compiler :	C Compiler Package for M32C Series V.5.20 Release 1
Compile Option :	-O -c -gnone -finfo -fNII -misra_all -r $*.csv
MISRA C Checker :	SQMlint V.1.00 Release 1A

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