Thank you for using the CS+ integrated development environment.
This document describes the restrictions and points for caution. Read this document before using the product.

Contents

Chapter 1. Target Devices ................................................................. 2
Chapter 2. User’s Manuals .............................................................. 3
Chapter 3. Keywords When Uninstalling the Product ......................... 4
Chapter 4. Changes ....................................................................... 5
  4.1 Changes to the CC-RH compiler ............................................. 5
    4.1.1 Enhanced optimization ....................................................... 5
    4.1.2 Improvements to the feature for checking source code against MISRA-C:2012 rules
          [Professional edition] .......................................................... 8
    4.1.3 Enhancing the security of dynamic memory management [Professional edition] ...... 8
    4.1.4 Addition of checking for exclusive control ............................ 9
    4.1.5 Rectified points for caution ................................................ 9
    4.1.6 Other changes and improvements ...................................... 10
Chapter 1. Target Devices

The target devices supported by the CC-RH compiler are listed on the Website.
Please see the URL below.
CS+ Product Page:

http://www.renesas.com/cs+
Chapter 2. User’s Manuals

Please read the following user’s manuals along with this document.

<table>
<thead>
<tr>
<th>Manual Name</th>
<th>Document Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS+ Integrated Development Environment User's Manual:</td>
<td></td>
</tr>
<tr>
<td>CC-RH Build Tool Operation</td>
<td>R20UT3283EJ0102</td>
</tr>
</tbody>
</table>
Chapter 3. Keywords When Uninstalling the Product

There are two ways to uninstall this product.

- Use the integrated uninstaller from Renesas (uninstalls all CS+ components)
- Use the Windows uninstaller (only uninstalls this product)

To use the Windows uninstaller, select “CS+ CC-RH V1.04.00” from “Programs and Features” of the control panel.
Chapter 4. Changes

This chapter describes changes to the CC-RH compiler.

4.1 Changes to the CC-RH compiler

This section describes changes to the CC-RH compiler from V1.03.00 to V1.04.00. Note that the features which are only available to users holding a registered license for the Professional edition are indicated as [Professional edition].

4.1.1 Enhanced optimization

For V1.04.00, optimization has been further enhanced on points (a) to (d), listed and described below.

a. Merging of stack areas allocated for auto arrays in different local scopes (reducing the stack size)

The compiler merges stack areas allocated for auto arrays that belong to different blocks ({}) whose lifetimes do not overlap.

```
Example of source code
int *g;
void func01(void)
{
    {
        int array01[10];
        g = array01;
        foo();
    }
    {
        int array02[10]; // Lifetimes of array01[10] and array02[10] do not overlap.
        g = array02;
    }
}
```

```
Code generated by V1.03.00
_func01:
    .stack_func01 = 88 ; Size of allocated stack = 88bytes
    prepare 0x00000041, 0x00000050
    movhi HIGHW1(#_g), r0, r20
    mov r3, r2
    st.w r2, LOWW(#_g)[r20]
    jarl _foo, r31
    movea 0x00000028, r3, r2
    st.w r2, LOWW(#_g)[r20]
    dispose 0x00000050, 0x00000041, [r31]
```

```
Code generated by V1.04.00
_func01:
    .stack_func01 = 52 ; Size of allocated stack = 52 bytes
    prepare 0x00000061, 0x00000028
    movhi HIGHW1(#_g), r0, r20
    mov r3, r21
    st.w r21, LOWW(#_g)[r20]
    jarl _foo, r31
    movea 0x00000028, r3, r2
    st.w r21, LOWW(#_g)[r20]
    dispose 0x00000028, 0x00000061, [r31]
```
b. Optimization of constant propagation

Obviously recognizable calculations of constants within loops are omitted.

```c
int func02(int xtra, int n4, int e1[]) {
    int i,ix,j,k,l;
    j = 1;
    k = 2;
    l = 3;

    for (ix=0; ix<xtra; ix++) {
        for (i=0; i<n4; i++) {
            j = j*(k-j)*(l-k); // j is always 1
            k = l*k-(l-j)*k;  // k is always 2
            l = (l-k)*(k+j);  // l is always 3
            e1[l-2] = j+k+l; // l-2 is always 1, j+k+l is always 6
            e1[k-2] = j*k+l; // k-2 is always 0, j+k*l is always 6
        }
    }
    return e1[0]+e1[1];
}
```

```assembly
.CODE generated by V1.03.00
    .BB.LABEL.2_3: ; bb47
        cmp r7, r11
        bge9 .BB.LABEL.2_5
    .BB.LABEL.2_4: ; bb1
        mov r9, r12
        sub r5, r12
        ; j*(k-j)*(l-k) and
        ; l*(k-1-)*k*(l-k)*z are calculated each time.
        mov r5, r13
        sub r2, r13
        mul r12, r2, r0
        mul r13, r2, r0
        mov r9, r12
        sub r2, r12
        mul r5, r12
        add r2, r12
        mul r12, r9, r0
        add r9, r12
        mov r9, r13
        shl 0x00000002, r13
        add r8, r13
        st.w r12, 0xFFFFFFF8[r13]
        mov r5, r12
        mul r2, r12, r0
        mul r9, r12, r0
        mov r5, r13
        shl 0x00000002, r13
        add r8, r13
        st.w r12, 0xFFFFFFF8[r13]
        add 0x00000001, r9
        br9 .BB.LABEL.2_3

.CODE generated by V1.04.00
    mov 0x00000006, r5
    :
    .BB.LABEL.2_3: ; bb46
        cmp r7, r9
        bge9 .BB.LABEL.2_5
    .BB.LABEL.2_4: ; bb1
        st.w r5, 0x00000000[r8]
        ; 6 is always assigned to e1[1].
        st.w r5, 0x00000000[r8]
        ; 6 is always assigned to e1[0].
        add 0x00000001, r9
        br9 .BB.LABEL.2_3
```

<Example of source code>

```c
int func02(int xtra, int n4, int e1[]) {
    int i,ix,j,k,l;
    j = 1;
    k = 2;
    l = 3;

    for (ix=0; ix<xtra; ix++) {
        for (i=0; i<n4; i++) {
            j = j*(k-j)*(l-k); // j is always 1
            k = l*k-(l-j)*k;  // k is always 2
            l = (l-k)*(k+j);  // l is always 3
            e1[l-2] = j+k+l; // l-2 is always 1, j+k+l is always 6
            e1[k-2] = j*k+l; // k-2 is always 0, j+k*l is always 6
        }
    }
    return e1[0]+e1[1];
}
```

<Code generated by V1.03.00>

```
    cmp r7, r11
    bge9 .BB.LABEL.2_5
    mov 0x00000006, r5
    :
    .BB.LABEL.2_3: ; bb47
        cmp r7, r11
        bge9 .BB.LABEL.2_5
    .BB.LABEL.2_4: ; bb1
        mov r9, r12
        sub r5, r12
        ; j*(k-j)*(l-k) and
        ; l*(k-1-)*k*(l-k)*z are calculated each time.
        mov r5, r13
        sub r2, r13
        mul r12, r2, r0
        mul r13, r2, r0
        mov r9, r12
        sub r2, r12
        mul r5, r12
        add r2, r12
        mul r12, r9, r0
        add r9, r12
        mov r9, r13
        shl 0x00000002, r13
        add r8, r13
        st.w r12, 0xFFFFFFF8[r13]
        mov r5, r12
        mul r2, r12, r0
        mul r9, r12, r0
        mov r5, r13
        shl 0x00000002, r13
        add r8, r13
        st.w r12, 0xFFFFFFF8[r13]
        add 0x00000001, r9
        br9 .BB.LABEL.2_3
```
c. Optimization of induction variables

The compiler does not generate code for redundantly updating loop induction variables.

```
void callee(unsigned i);
void caller(void){
    unsigned i;
    for(i=128; i != 0; --i){
        callee(i);
    }
}
```

```
Example of source code
unsigned long test(unsigned long long variable, int var){
    if (var){
        variable &= 0x012345678abcdefULL;
    }
    return (variable >> 32);
}
```

```
Example of source code
unsigned long long test(unsigned long long variable, int var){
    if (var){
        variable &= 0x012345678abcdefULL;
    }
    return (variable >> 32);
}
```

```
Code generated by V1.03.00
_caller:
    .stack _func03 = 12
    prepare 0x000000061, 0x00000000
    movea 0x000000080, r0, r20
    mov r20, r21 ; Loop induction variable is redundantly initialized.
    .BB.LABEL.3_1: ; bb
        mov r21, r6
        jarl _callee, r31
        add 0xFFFFFFFF, r21 ; Loop induction variable is redundantly updated.
        loop r20, .BB.LABEL.3_1
    .BB.LABEL.3_2: ; return
        dispose 0x00000000, 0x000000061, [r31]
```

```
Code generated by V1.04.00
_caller:
    .stack _func03 = 8
    prepare 0x000000041, 0x00000000
    movea 0x000000080, r0, r20
    .BB.LABEL.3_1: ; bb
        mov r20, r6
        jarl _callee, r31
        loop r20, .BB.LABEL.3_1
    .BB.LABEL.3_2: ; return
        dispose 0x00000000, 0x000000041, [r31]
```

d. Deleting unused code

The ability to delete unused code has been further enhanced.

```
Example of source code
unsigned long test(unsigned long long variable, int var){
    if (var){
        variable &= 0x012345678abcdefULL;
    }
    return (variable >> 32);
}
```

```
Example of source code
unsigned long long test(unsigned long long variable, int var){
    if (var){
        variable &= 0x012345678abcdefULL;
    }
    return (variable >> 32);
}
```

```
Code generated by V1.03.00
_test:
    .stack _test = 0
    cmp 0x00000000, r8
    b9 .BB.LABEL.5_2
    .BB.LABEL.5_1: ; if_then_bb
        mov 0x78ABCDEF, r2
        and r2, r6
        ; r6 is not referenced in the subsequent lines.
        mov 0x00123456, r2 and r2, r7
    .BB.LABEL.5_2: ; if_break_bb
        mov r7, r10
        jmp [r31]
```

```
Code generated by V1.04.00
_test:
    .stack _test = 0
    cmp 0x00000000, r8
    b9 .BB.LABEL.5_2
    .BB.LABEL.5_1: ; if_then_bb
        mov 0x00123456, r2 and r2, r7
    .BB.LABEL.5_2: ; if_break_bb
        mov r7, r10
        jmp [r31]
```
4.1.2 Improvements to the feature for checking source code against MISRA-C:2012 rules [Professional edition]

The following rule numbers have been added to those which can be designated as arguments of the -xmisra2012 option, which selects checking by the compiler of source code against the specified MISRA-C:2012 rules.

2.6 2.7 9.2 9.3 12.1 12.3 12.4 14.4 15.1 15.2 15.3 15.4 15.5 15.6 15.7 16.1 16.2 16.3 16.4 16.5 16.6 16.7 17.1 17.7 18.4 18.5 19.2 20.1 20.2 20.3 20.4 20.5 20.6 20.7 20.8 20.9 20.10 20.11 20.12 20.13 20.14

The following are the numbers of MISRA-C:2012 rules against which the V1.03.00 and V1.04.00 compilers can check source code for compliance.

<table>
<thead>
<tr>
<th>Rule classification (number of rules in the standard)</th>
<th>V1.03.00</th>
<th>V1.04.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mandatory rules (10)</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Required rules (101)</td>
<td>31</td>
<td>58</td>
</tr>
<tr>
<td>Advisory rules (32)</td>
<td>7</td>
<td>21</td>
</tr>
<tr>
<td>Total number of rules (143)</td>
<td>41</td>
<td>82</td>
</tr>
</tbody>
</table>

4.1.3 Enhancing the security of dynamic memory management [Professional edition]

A feature has been added for the detection of illicit operations in the releasing of heap space. This feature can be used by linking a dedicated standard library (libv850e3v5securevlibmalloc.lib) of secure functions related to dynamic memory allocation.

An error with code E0562310 will occur if the compiler is not registered for a license to the Professional edition.

Run the dedicated standard library as follows.

1. As well as the actual areas allocated for users in the heap space by the calloc, malloc, and realloc functions, four extra bytes are added before and after each area for the detection of illicit operations.
2. When called, the free and realloc functions determine if any of (a) to (c) applies.
   (a) The argument is not a pointer to an actual area allocated by calloc, malloc, or realloc.
   (b) The four-byte area for detecting illicit operations has been overwritten.
   (c) The pointer is to an area that has already been released.
3. In the event of any of the above, an illicit operation is assumed to have proceeded, and __heap_chk_fail will be called.

The __heap_chk_fail function needs to be defined by the user. Write the processing which should be executed when any illicit operation has been detected in the heap space. For example, if “ABCDEF” is copied from str to the buffer for four letters in the 6th line in the following program, the heap space will be corrupted since the buffer will overflow due to “EF” and the null character (\x00). In this case, __heap_chk_fail will be executed when the related heap space is released by the 8th line.

By using this feature, you can easily counter security problems through measures against the dual release of memory and against buffers overflowing.

4.1.4 Addition of checking for exclusive control

The -Xcheck_exclusion_control option has been added to select checking for exclusive control by CS+.

When this option is designated, the exclusive control check settings file which is generated by CS+ is read, and DBTAG instructions are generated in response to the calling of functions or access to variables designated in the file.

This function assumes usage of the file through CS+, and it should not be directly used by the user.

4.1.5 Rectified points for caution

Points for caution on the following three items no longer apply.
- External labels defined after conditional assembly control instructions (No. 7)
- Designating a member of a packed structure or union in an initializing declaration (No. 8)
- Scope of optimization (No. 10)
4.1.6 Other changes and improvements

Other major changes and improvements are described below.

(a) Improved debugging information
   A problem with C and assembly source code not being displayed properly during debugging has been corrected.

(b) Improved prevention of internal errors
   A problem with an internal error during building has been corrected.

All trademarks and registered trademarks are the property of their respective owners.
Notice

1. 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.

2. 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 or omissions from the information included herein.

3. 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.

4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.

5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

- "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tool; personal electronic equipment; and industrial robots etc.
- "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

6. Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.

7. 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.

8. 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.

9. 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. You should verify Renesas Electronics products or 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. When exporting the Renesas Electronics 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.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.

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 products” means any product developed or manufactured by or for Renesas Electronics.