[Notes]
C Compiler Package for RH850 Family

R20TS0179EJ0100
Rev.1.00
Jul. 1, 2017

Outline

When using the CC-RH C compiler package for the RH850 family, note the following point.

1. Point for caution on the #pragma pmodule extended language (No.15)

Note: The number which follows the description of a precautionary note is an identifying number for the precaution.

1. Point for Caution on the #pragma pmodule Extended Language (No.15)

1.1 Applicable Products

CC-RH V1.00.00 to V1.05.00

1.2 Details

With core numbers specified to variables in the #pragma pmodule extended language, specifying anything from pm1 to pm255 may result in the following incorrect core numbers and section allocation.

V1.05.00: pm0

Other than V1.05.00: cmn

1.3 Condition

This problem arises if all of conditions (1) to (3) are met.

(1) Compiler option -Xmulti_level=1 is specified.

(2) There is a variable for which a core number is specified by "#pragma pmodule pm specification", and anything from pm1 to pm255 is specified as the pm value.

(3) For the variable of (2), a section is specified by "#pragma section attribute-strings", and the attribute string satisfies condition (3-1) or (3-2).

(3-1) For a variable with the const qualifier: Attribute string (Note 1) of the data or bss category.

(3-2) For a variable without the const qualifier: Attribute string (Note 2) of the const category.

Note 1: r0_disp16, r0_disp23, r0_disp32, ep_disp4, ep_disp5, ep_disp7, ep_disp8, ep_disp16, ep_disp23, gp_disp16, or gp_disp23

Note 2: const, zconst, or zconst23

1.4 Example

The following is an example of the problem. Characters in red are the parts that correspond to the conditions.

Specifying compiler option -Xmulti_level=1: Condition (1)

```
1:  #pragma pmodule pm1              //Condition (2)
2:  int var0;
3:
4:  #pragma section r0_disp32
5:  int var1 = 0;
6:  const int var2 = 0;              //Condition (3-1)
7:
8:
```

The section allocation in the above case is as follows:

Line 2: "var0" does not specify a #pragma section, so the allocation is made to the bss.pm1 section correctly.

Line 5: "var1" is not a const qualified variable, and is an attribute string of the data or bss category, so allocation is made to the .data.pm1 section correctly.

Line 6: "var2" is a const qualified variable and is an attribute string of the data or bss category, and therefore, condition (3-1) applies. Therefore, allocation is not made correctly to the .const.pm1 section, and results in the following wrong allocation.

V1.05.00: Allocation made to the .const.pm0 section

Other than V1.05.00: Allocation made to the .const.cmn section

1.5 Workaround

When specifying a section with "#pragma section attribute-string", specify an attribute string as follows:

For a variable with the const qualifier: Attribute string of the const category.

For a variable without the const qualifier: Attribute string of the data or bss category.

```
1:   #pragma pmodule pm1
2:   int var0;
3:
4:   #pragma section r0_disp32
5:   int var1 = 0;
6:   #pragma section const // Add this line
7:   const int var2 = 0;   //Allocation is made to .const.pm1 as intended by adding the
8: 6th line
9:
     #pragma section default
```

1.6 Schedule for Fixing the Problem

This problem will be fixed in V1.06.00. (This information will be available from July 20.)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul. 1, 2017	-	First edition issued

TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061 Japan

Renesas Electronics Corporation

■Inquiry

https://www.renesas.com/contact/