[Notes]
C Compiler Package for RH850 Family

R20TS0317EJ0100
Rev.1.00
Jul. 16, 2018

Outline

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

1. Static declaration of a structure, an array, or a union that has an initializer (No. 19)
2. Assembly-language code using reserved symbol (No. 20)
3. Section where the initializers of auto variables are allocated when the -Xmulti_level option is specified (No. 21)
4. Compiler option “-store_reg” (No. 22)

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

1. Static declaration of a structure, an array, or a union that has an initializer (No. 19)

1.1 Applicable Products

CC-RH V1.00.00 to V1.07.00

1.2 Details

Assume that a function contains multiple array-type, structure-type, or union-type static variable declarations with initializers specified. If the address of a variable in the upper-layer block is specified as the initializer of a variable in the lower-layer block, an internal error might occur or invalid code might be generated.

1.3 Conditions

An internal error occurs when conditions (1) to (3) below are met and either of conditions (4) and (5) is met: An invalid code might be generated when all of conditions (1) to (4) and (6) below are met:

(1) The function contains a variable declaration that meets all of the following conditions:

- (1-1) static specification is contained.
- (1-2) The variable is the array type, structure type, or union type.
- (1-3) An initializer is specified.

(2) A variable is declared with the same conditions as (1) in a lower-layer block in the block that contains the variable declaration in (1).

(3) The address of the variable in (1) is specified for the initializer of the variable in (2).

(4) The first block of the function contains the variable declaration in (1).

(5) V1.04.00 or a later version is used, and the variable declaration in (1) is written in a lower-layer block than the first block of the function.

(6) A variable that meets the conditions in (1) is declared by one of the following methods:

- (6-1) The variable is declared before the variable in (1) in the block in (4).
- (6-2) The variable is declared in a block that was specified before the block in (2).
- (6-3) The variable is declared after the variable in (2) in the block in (2).

(6-4) V1.03.00 or an earlier version is used, and the variable is declared in a lower-layer block in the block in (2).

1.4 Example

The following describes an example of invalid code generation. Characters in red are the parts that correspond to the conditions.

```
1:  typedef struct A {
2:    short *objId;
3:  } TEST_PARAM;
4:
5:  int sub(TEST_PARAM * aParam);
6:  int var;
7:
8:  int main(void)
9:  {
10:   static short obj[] = { 1 };          // Conditions (1) and (4)
11:   {
12:     static short dmy[] = { 2 };       // Condition (6-2)
13:     var = dmy[0];
14:   }
15:   {
16:     static TEST_PARAM t ={ obj }; // Conditions (2) and (3)
17:     sub(&t);
18:   }
19:   return 0;
20: }
```

- Line 10: Conditions (1) and (4) are met because the first block (lines 9 to 20) of the main function contains the declaration of array-type static variable "obj" with the initializer specified.
- Line 16: Condition (2) is met because structure-type static variable "t" with the initializer specified is declared in the lower-layer block (lines 15 to 18) in the block (lines 9 to 20) that contains the variable declaration of line 10. In addition, condition (3) is met because the address of array-type variable "obj" (line 10) is specified as the initializer.
- Line 12: Condition (6-2) is met because, in addition to the variable declaration in line 10, array-type static variable "dmy" with the initializer specified is declared in the block (lines 11 to 14). This block was specified before the block containing the variable declaration (line 16) that meets the condition in (2).
  In this case, structure-type variable "t" is initialized with the address of array-type variable "dmy", instead of the address of array-type variable "obj".

1.5 Workaround

To avoid this problem, take either of the following steps:

(1) Remove the static specification from the variable declaration in (2) under Conditions.
(2) Change the initializer in (3) under Conditions to an assignment expression.

1.6 Schedule for Fixing the Problem

This problem will be fixed in CC-RH V2.00.00. This information will be available from July 20.

2. Assembly-language code using reserved symbol (No. 20)

2.1 Applicable Products

CC-RH V1.07.00

2.2 Details

If a separation operator (HIGH, LOW, HIGHW, LOWW, or HIGHW1) is used for a symbol for which reserved symbol __gp_data or __ep_data is defined in the same section, the operation might cause an invalid result or an error.

2.3 Conditions

The problem arises when all of the conditions listed below are met:

(1) Either of the following reserved symbols is used for the assembly source:

  (1-1) __gp_data
  (1-2) __ep_data

(2) The reserved symbol in (1) is made a global symbol by using the .public pseudo instruction or .extern pseudo instruction.

(3) A symbol is defined in the same assembly source and in the same section as the reserved symbol in (1).

(4) Offset reference is performed for the symbol in (3) by using either of the following symbols:

  (4-1) If the reserved symbol in (1-1) is in the same section, gp offset reference using "$"
  (4-2) If the reserved symbol in (1-2) is in the same section, ep offset reference using "%"

(5) A separation operator is used for offset reference in (4).

2.4 Example

The following is an example of the problem.

```
1:  .section .data, data
2:  .public __gp_data                          ; Condition (2)
3:  __gp_data:                                 ; Condition (1)
4:      .ds 4
5:  _symbol:                                   ; Condition (3)
6:  …
7:      addi    highw($_symbol), r4, r10       ; Conditions (4) and (5)
```

Line 3: Condition (1) is met because the reserved symbol __gp_data is defined.

Line 2: Condition (2) is met because __gp_data is externally defined and declared with the.public pseudo instruction.

Line 5: Condition (3) is met because the symbol _symbol is defined in the same .data section as of __gp_data.

Line 7: Condition (4) is met because gp offset reference is performed for symbol _symbol by using "$". In addition, condition (5) is also met because separation operator HIGHW is used for gp offset reference.
At this time, the calculation result of "highw($_symbol)" in line 7 is an invalid value. In the correct specifications, high-order 16 bits (= 0x0000) must be returned for the subtraction value (= 0x00000004) of the __gp_data address from the _symbol address. However, low-order 16 bits (= 0x0004) are returned.

2.5 Workaround

Do not define reserved symbols __gp_data and __ep_data.

2.6 Schedule for Fixing the Problem

This problem will be fixed in CC-RH V2.00.00. This information will be available from July 20.

3. Section where the initializers of auto variables are allocated when the -Xmulti_level option is specified (No. 21)

3.1 Applicable Products

CC-RH V1.00.00 to V1.07.00

3.2 Details

When -Xmulti_level = 1 option is specified, the initializer of an automatic variable of a char-type array is placed in a section different from the specification.

3.3 Conditions

The problem arises when all of the conditions listed below are met: The initializer in (3) is placed in a section different from the specification.

(1) The -Xmulti_level = 1 option is designated.

(2) Either of the following automatic variables is defined:

(2-1) (signed/unsigned) char-type array
(2-2) Structure type or union type that has the array type of (signed/unsigned) char type as an element

(3) The automatic variable in (2) is initialized with a string literal.

3.4 Example

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

```
1:  void func () {
2:    char a[2] = {'1', '2'};
3:    char b[2] = "12";          // Conditions (2) and (3)
4:  }
```

- Line 2: Initializers '1' and '2' of array a are placed in the .const section.
- Line 3: In the correct specifications, initializer "12" of array b must be placed in .const section. However, the initializer is placed in .const.cmn section.

3.5 Workaround

Use characters rather than string literal to specify initialization.

3.6 Schedule for Fixing the Problem

This problem will be fixed in CC-RH V2.00.00. This information will be available from July 20.

4. Compiler option “-store_reg” (No. 22)

4.1 Applicable Products

CC-RH V1.06.00 to V1.07.00 (Professional edition only)

4.2 Details

Indirect reference to a structure member might not be recognized as a write to a control register.

As a result, the following functions do not work even if the -store_reg option (Note) is specified:

- Detection of write processing to a control register
- Synchronization insertion control between control registers

Note: The -store_reg option is only available for the Professional edition.

4.3 Conditions

This problem may arise if Condition i or Condition ii is met.

[Condition i]

If all the conditions listed below are met, the assignment in (4) might not be able to be detected as a write to the control register.

(1) A constant is cast to the volatile-declared pointer type of structure type ST.

(2) Structure type ST in (1) contains member A of union type UT.

(3) Union member A in (2) is not the first member of structure type ST.

(4) Union member A in (2) has an assignment that meets one of the following conditions:

(a) The assignment is a union assignment to A.
(b) A is assigned to member B whose type is smaller than the largest type of members of union type UT.
(c) A has member B of structure type ST2, and there is an assignment that meets one of the following conditions:

(c-1) B is assigned to bit field member C whose type is not (unsigned) char / _Bool type.
(c-2) B is assigned to bit field member C whose type is (unsigned) char / _Bool type, and the type of C is smaller than the largest type of members of union type UT.
(c-3) B is assigned to member C, and structure type ST2 does not have a member of (unsigned) int / long type.

[Condition ii]

If all the conditions listed below are met, the assignment in (7) might not be able to be detected as a write to the control register.

(5) A constant is cast to the volatile-declared pointer type of structure type ST.

(6) Structure type ST in (5) contains member A of structure type ST2.

(7) Structure member A in (6) has an assignment that meets one of the following conditions:

(a) Structure assignment to A
(b) A is assigned to bit field member B whose type is not (unsigned) char / _Bool type.

4.4 Example

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

```
1:  struct __tag2191
2:  {
3:      unsigned char   EIP384:4;
4:      unsigned char   :2;
5:      unsigned char   EITB384:1;
6:      unsigned char   EIMK384:1;
7:  };
8:
9:  union __tag4514
10: {
11:     unsigned short UINT16;
12:     unsigned char   UINT8[2];
13:     struct __tag2191 BIT;
14: };
15:
16: struct __tag4697
17: {
18:     unsigned char   dummy1812[1];
19:     union __tag4514 EIC384;                          // Conditions (2) and (3)
20: };
21:
22: #define INTC2      (*(volatile struct __tag4697 *)0xFFFFB040)   // Condition (1)
23: #pragma register_group 0xFFFFB040, 0xFFFFFFFF, id="INTC2"
24:
25: void fun() {
26:     INTC2.EIC384.BIT.EIMK384 = 1;                    //Condition (4)
27: }
```

- Line 22: Condition (1) is met because constant 0xFFFFB040 is cast to the volatile-declared pointer type of structure type __tag4697.
- Lines 16 to 20: Condition (2) is met because structure type __tag4697 contains member EIC384 of union type __tag4514.
  In addition, condition (3) is met because EIC384 is not the first member of structure type __tag4697.
- Line 26: Condition (4)-(C)-(c-2) is met because union member EIC384 has structure member BIT and the following conditions are met:
  - BIT is assigned to bit field member EIMK384.
  - The type of EIMK384 is unsigned char type.
  - That type is smaller than unsigned short, which is the largest type for members of union type __tag4514.

  Therefore, Condition i is met.

  INTC2.EIC384.BIT.EIMK384 is within the range of addresses (0xFFFFB040 to 0xFFFFFFFF) specified by #pragma register_group in line 23. Therefore, it must be detected as a control register whose group ID is INTC2 in the correct specifications. However, it cannot be detected.

4.5 Workaround

If insertion of synchronization processing is required, insert it manually.

4.6 Schedule for Fixing the Problem

This problem will be fixed in CC-RH V2.00.00. This information will be available from July 20.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul. 16, 2018	-	First edition issued

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

Renesas Electronics Corporation

■Inquiry

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

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.

The past news contents have been based on information at the time of publication. Now changed or invalid information may be included.

The URLs in the Tool News also may be subject to change or become invalid without prior notice.

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

© 2018 Renesas Electronics Corporation. All rights reserved.

TS Colophon 2.1