[Notes]
C Compiler Package for RL78 Family

R20TS0373EJ0100
Rev.1.00
Dec. 01, 2018

Outline

When using the C compiler package for RL78 family CC-RL, note the following points:

1. Point for caution when a 1-bit signed bit field is written in the control expression of a switch statement (CCRL#020)
2. Point for caution when a structure or union having a member that is a far pointer is packed and allocated to the far area (CCRL#021)

Note: The number following the note is an identifying number for the precautionary note.

1. Point for caution when a 1-bit signed bit field is written in the control expression of a switch statement (CCRL#020)

1.1 Applicable Products

CC-RL V1.00.00 to V1.07.00

1.2 Details

The case label when the value is 1(Note) may be executed when a 1-bit signed bit field is written in the control expression of a switch statement.

Note: The value of the 1-bit signed bit field can be either 0 or -1.

1.3 Conditions

Generated code may be incorrect if both of the following conditions (1) and (2) are met:

(1) A 1-bit signed bit field is written in the control expression of a switch statement.

(2) Processing when the value is l is written in the case label in (1).

1.4 Example

The example of the problem is shown below. Characters in red are the parts corresponding to the conditions.

[C source]

```
1  typedef struct{
2      signed char b0:1;
3      signed char b1:1;
4  } ST;
5
6  void func1(ST n) {
7      switch (n.b0) {     // Condition (1)
8      case 0:
9          func0(0);
10         break;
11     case 1:             // Condition (2)
12         func0(1);
13         break;
14     default:
15         break;
16     }
17 }
```

- Line 7: Condition (1) is met because 1-bit signed bit field n.b0 is written in the control expression of a switch statement.
- Line 11: Condition (2) is met because processing when the value is 1 is written in the case label. Although n.b0 can be either 0 or -1, the output code branches to case 1: when n.b0 is -1, while it should branch to default:.

[Output assembler code]

```
1  _func1:
2        bt a.0, $.BB@LABEL@1_2  ; Branch to case 1:(invalid) if the bit is
3                                ; set to 1
4  .BB@LABEL@1_1:                ; Processing of case 0:
5        clrw ax
6        br !!_func0
7  .BB@LABEL@1_2:                ; Processing of case 1:
8        onew ax
9        br !!_func0
```

Although n.b0 can be either 0 or -1, code that branches to case 1: when n.b0 is -1 is output.

1.5 Workaround

To avoid this problem, follow either (1) or (2) below.

(1) Replace the switch statement of condition (1) with an if statement.

(2) Change the type of 1-bit bit field in condition (1) to unsigned type.

1.6 Schedule for Fixing the Problem

The problem will be fixed in CC-RL V1.08.00.

2. Point for caution when a structure or union having a member that is a far pointer is packed and allocated to the far area (CCRL#021)

2.1 Applicable Products

CC-RL V1.01.00 to V1.07.00

2.2 Details

Incorrect operation may occur if a structure or union having a member that is a far pointer is packed and allocated to the far area.

2.3 Conditions

Generated code may be incorrect if all of the following conditions (1) through (4) are met:

(1) A structure-type or union-type variable having a member that is a far pointer is defined.

(2) The variable in (1) is allocated to the far area.

(3) Structure-type packing (-pack option or #pragma pack) is specified for the variable in (1).

(4) The variable member that is a far pointer in (1) is accessed.

2.4 Example

The example of the problem is shown below. Characters in red are the parts corresponding to the conditions.

[C source]

```
1  #pragma pack            // Condition (3)
2  struct ST {
3    char c;
4    int __far *ifp;       // Condition (1)
5  } __far st1;            // Conditions (1) and (2)
6
7  int __far * func(void) {
8    return st1.ifp;       // Condition (4)
9  }
```

- Line 1: Condition (3) is met because #pragma pack is specified.
- Lines 4 and 5: Condition (1) is met because structure-type variable "st1" having member ifp that is a far pointer is defined. In addition, condition (2) is met because st1 is modified by __far.
- Line 8: Condition (4) is met because ifp that is an st1 member is referenced.

[Output assembler code]

```
1  _func:
2          mov es, #LOW(HIGHW(_st1))
3          mov a, es:!LOWW(_st1+0x00003)
4          movw de, es:!LOWW(_st1+0x00001)    ; movw instruction for an odd
5                                             ; address
```

Because a 16-bit transfer instruction is output when referencing st1 member ifp, if ifp is allocated to an odd address, a 16-bit transfer instruction is executed for an odd address, resulting in incorrect operation.

2.5 Workaround

To avoid this problem, follow either (1) or (2) below.

(1) Allocate the variable in condition (1) to the near area.

(2) Do not specify structure-type packing for the variable in condition (1).

2.6 Schedule for Fixing the Problem

The problem will be fixed in CC-RL V1.08.00.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec. 01, 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/