[Notes]
C Compiler Package for RL78 Family

R20TS0377EJ0100
Rev.1.00
Dec. 16, 2018

Outline

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

1. Point for caution when the initial value of an integer-type member of a structure or union is an address constant. (CCRL#022)

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

1. Point for caution when the initial value of an integer-type member of a structure or union is an address constant. (CCRL#022)

1.1 Applicable Products

CC-RL V1.01.00 to V1.07.00

1.2 Details

When the initial value of an integer-type member of a structure or union is an address constant, the initial value data may be incorrect and may result in invalid operation.

1.3 Conditions

Initial value data may be incorrect if all the following conditions (1) through (3) are met:

(1) A structure- or union-type static variable has an integer-type member.

(2) The initial value of the integer-type-member in (1) is an address constant.

(3) The size of the integer-type member in (1) is larger than that of an address constant of the initial value (Note 1).

Note 1: The size of an address constant is the same as that of the pointer to the object.
Casting in the initializer doesn't affect the condition in (3).

1.4 Example

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

[C source]

```
1  unsigned char data;                                     // Condition (3)
2  struct _st {
3      unsigned long ldata;                                // Condition (1)(3)
4  } st1[] = { 0x12345678, (unsigned long)&data, 0x12345678 }; // Condition (2)
```

- Line 3: Condition (1) is met because the structure array st1 has an integer-type member ldata.

- Line 4: Condition (2) is met because the address constant &data is defined in the initializer of the structure member ldata.

- Line 1 and 3: The size of the address constant is 2 bytes, which is the same as the size of the near pointer because variable data is allocated to the near area. Condition (3) is met because the size of structure member ldata is 4 bytes.
  Although the address constant of the initializer is casted to an unsigned long type, whether it is casted does not affect the condition (3).

[Output assembler code]

```
_st1:
      .DB4 0x12345678
      .DB2 LOWW(_data)        ; An area for four bytes is not allocated.
      .DB4 0x12345678
```

Because an area is allocated only for the size of the address constant (2 bytes) as the initial value of the integer-type member, accessing to the subsequent data results in invalid operation.

1.5 Workaround

To avoid this problem, apply one of the followings.

(1) Separate the variables for a data pointer, function pointer and integer-type data.

```
unsigned char data;
struct _st1 {
   unsigned long ldata;
} st1[] = {0x12345678, 0x12345678};
struct _st2 {
   unsigned char * p;
} st2[] = {&data};
```

(2) Use a far variable as the variable defined in the address constant specified as the initial value (Note 1).

```
__far unsigned char data;
struct _st {
   unsigned long ldata;
} st[] = {0x12345678, (unsigned long)&data, 0x12345678};
```

Note 1: This method is only effective for an integer-type member that is other than the long long type.
Note that use of the far variable increases the code size.

(3) Set the initial value by using the assignment expression instead of defining the address constant in the member initializer in Condition (1) (Note 2).

```
unsigned char data;
struct _st {
   unsigned long ldata;
} st[] = {0x12345678, 0, 0x12345678};
       :

  st[1].ldata = (unsigned long)&data;
```

Note 2: For a const variable, you cannot set an initial value by using the assignment expression.

(4) If a union has a pointer as its member, set the initial value of the address constant by using the element specifier complying with the C99 standard (Note 3).

```
unsigned char data;
union _st {
    unsigned long ldata;
    unsigned char * p;
} st1[] = { 0x12345678, {.p = {&data}}, 0x12345678 };
```

Note 3: This method is only effective when the C99 standard (option -lang=c99) is specified.

An error occurs during compilation if the option -lang=c99 is not specified.

Also, because the initial value is defined in the member type specified in the the element specifier, the upper address is not set in a free area if the member is a near pointer.

1.6 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. 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/

TS Colophon 2.1