[Notes]

C/C++ Compiler Package for RX Family (No. 66)

R20TS0906EJ0100
Rev.1.00
Feb. 16, 2023

Outline

When using C/C++ Compiler Package for RX Family, note the following point.

1. Transferring Data or Calculating String Length in a Loop (No. 66)*

*The number in the parentheses is the identification number of this note.

1. Transferring Data or Calculating String Length in a Loop (No. 66)

1.1 Applicable Products

CC-RX V2.06.00 to V2.08.01

CC-RX V3.00.00 to V3.05.00

1.2 Details

If a loop that only performs memory-to-memory data transfer or a loop that only calculates string lengths is contained in another loop, the generated code of the compiler might be invalid.

1.3 Condition 1

The problem might occur when all the following conditions are met.

(1) Neither -optimize=0 nor -optimize=1 option is specified.

(2) There is a loop within another loop.

(3) The inner loop only performs memory-to-memory data transfer.

(4) The continuation condition of the loop in (3) is the comparison of (a) and (b) shown below, and the comparison is true if (a) is smaller(*) than (b) in an unsigned comparison.

  (a) An integer-type (32 bits or less) variable whose initial value is 0 and which increments by 1 for every loop iteration.

  (b) An integer constant or a variable whose value does not change during loop iterations.

  (*) : If (b) is an integer constant, comparative operators with equal sign are also included.

(5) Data transfer in (3) repeats one-byte assignment for every loop iteration.

(6) The source and destination of the assignment in (5) meet all the following conditions:

  (a) The assignment source is different from the assignment destination.

  (b) It is not an array of char type.

  (c) It is not a structure whose first member is an array of char type.

  (d) The volatile or __evenaccess qualifier is not added to the type.

[Example 1] ccrx -isa=rxv1 tp1.c // (1)

```
/* tp1.c */
int callee(char*);
short gv[0x100];
void caller(unsigned l){
  struct{
    short value;
    char array[0x100];
  }lv;
  char* from = (char *)&lv;
  char* to = (char *)&gv;
  /* Copy the structure to the array in units of bytes. */
  while(callee(lv.array)){    // (2)
    unsigned i;
    for (i = 0; i < l; i++){  // (2)(4)
      to[i] = from[i];        // (3)(5)(6)
    }
  }
}
```

1.4 Workaround (1)

Take any of the following actions.

(1) Specify the -optimize=0 or -optimize=1 option.

(2) Add the volatile or __evenaccess qualifier to the variable used for data transfer described in (3) in Condition 1. You can add either qualifier, or both.

(3) Use the standard library function memcpy() to implement the processing that the loop described in (3) in Condition 1 performs.

1.5 Condition 2

The problem might occur when all the following conditions are met.

(1) Neither -optimize=0 nor -optimize=1 option is specified.

(2) There is a loop within another loop.

(3) The inner loop meets all the following conditions:

  (a) The loop continues if the destination of a pointer indicating a char or unsigned char type value is not '¥0'.

  (b) The volatile or __evenaccess qualifier is not added to the destination type of the pointer in (a).

  (c) The loop processing only advances the pointer by position.

(4) After the loop in (3), the difference in the pointer before and after advancement is calculated.

[Example 2] ccrx -isa=rxv1 tp2.c // (1)

```
/* tp2.c */
char* callee(int);
void caller(void){
  int length = 0;
  char* string;
  while(string = callee(length)){     // (2)
    /* Loop that advances the pointer to the end of the string */
    char* pointer = string;  // (b)
    while(*pointer){         // (2)(a)
      ++pointer;             // (c)
    }
    /*Processing that calculates the difference in the pointer before
and after advancement */
    length = pointer - string;  // (4)
  }
}
```

1.6 Workaround (2)

Take any of the following actions.

(1) Specify the -optimize=0 or -optimize=1 option.

(2) Add the volatile or __evenaccess qualifier to the pointer destination type used for the continuation condition of the loop described in (3) in Condition 2. You can add either qualifier, or both.

(3) Use the standard library function strlen() to implement the processing that the loop described in (3) in Condition 2 performs.

1.7 Schedule for Fixing the Problem

This problem will be fixed in CC-RX V3.06.00. The release date has not been determined.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Feb.16.23	-	First edition issued

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.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

TS Colophon 4.3