[Notes]

C/C++ Compiler Package for RX Family

R20TS0056EJ0100
Rev.1.00
July 16, 2016

Outline

When using the CC-RX C/C++ Compiler Package for the RX Family, take note of the problem described in this note regarding the following point.

1. Scope of optimization (No. 38)
   Note: The number which follows the description of the precautionary note is an identifying number for the precaution.

1. Scope of Optimization (No. 38)

1.1 Applicable Product

CC-RX V2.00.00 to V2.04.01

1.2 Details

The scope of optimization might vary due to the environment* where the CC-RX compiler is operating with an option other than -Optimize=0 specified so that processing for optimization is selected.

This may lead to differences in generated code even when the source code is unchanged. Even in cases where this effect is seen, the generated code will still operate in accord with the statements in the C source file. However, the differences in the generated code might affect the timing of execution by the program.

*: Here, “environment” refers to dependences on file names, folder names, options, environmental variables of PC, and other factors.

1.3 Example

[C source]

```
int     XXX = 0;
void    funcion01( void );
void    funcion01( void )
{
   int i = 0;
   DIT          di = { 0, 0 };
   while( i < XXX ){
      di = func_01(i);
      if((ary[i] != 0xc0)&&(ary[i] != 0xc1)){
         if( ((lgflg[di.int_01] & (1UL<<di.int_02)) != 0UL ) ||((lgflg_sb[di.int_01] &
(1UL<<di.int_02)) != 0UL) ){
             long_001[di.int_01] &= ~(1UL<<di.int_02);
             long_sub001[di.int_01] &= ~(1UL<<di.int_02);
             if( ((lgflg[di.int_01] & (1UL<<di.int_02)) != 0UL ) && ((lgflg_sb[di.int_01] &
(1UL<<di.int_02)) != 0UL) ){
                  long_001[di.int_01] |= (1UL<<di.int_02);
                  long_sub001[di.int_01] |= (1UL<<di.int_02);
             }else if((lgflg[di.int_01] & (1UL<<di.int_02)) != 0UL ){
                  long_001[di.int_01] |= (1UL<<di.int_02);
             }else if((lgflg_sb[di.int_01] & (1UL<<di.int_02)) != 0UL){
                  long_sub001[di.int_01] |= (1UL<<di.int_02);
             }else{
             }
             lgflg[di.int_01] &= ~(1UL<<di.int_02);
             lgflg_sb[di.int_01] &= ~(1UL<<di.int_02);
         }
      }
      ++i;
   }
}
```

Extracts showing differences in the generated code

Pattern 1

```
        …
          MOV.L [R8,R14], R15
           BCLR R1, R15
           MOV.L R15, [R8,R14]
  L33
        …
```

Pattern 2

```
        …
          MOV.L [R8,R14], R15
         MOV.W 02H[R0], R2
           BCLR R1, R15
           MOV.L R15, [R8,R14]
  L33
        …
```

Compared to pattern 1, redundant instructions have been generated in pattern 2 since the scope was narrowed, so optimization did not work to some extent. However, there are no problems with operation of the generated code.

1.4 Workaround

Compare results with the last results of generation when rebuilding is needed. In cases where a difference appears, re-evaluate the code or use the last result of generation.

1.5 Schedule for Fixing the Problem

This problem will be fixed in the next revision.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	July 16, 2016	-	First edition issued

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

Renesas Electronics Corporation

■Inquiry

http://www.renesas.com/contact/