[Notes]

C/C++ Compiler Package for SuperH RISC engine Family

R20TS0129EJ0100
Rev.1.00
Apr. 16, 2017

Outline

When using the C/C++ Compiler Package V.7 to V.9 for the SuperH RISC engine family, note the following point.

1. When the type of the loop control variable is converted to the float type in a loop (SHC-0097)

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

1. When the Type of the Loop Control Variable is Converted to the Float Type in a Loop (SHC-0097)

1.1 Applicable Products

C/C++ Compiler Package for SuperH RISC engine family V.7.0B to V.9.04 Release 03

1.2 Details

When the type of the loop control variable(Note) whose absolute value is greater than 16777216 is converted to the float type in a loop, the result of the type conversion might be wrong.

Note: This refers to the variable which is incremented or decremented by a fixed integer value on each iteration of the loop processing and is referred by the decision expression for determining whether to iterate the loop.

1.3 Condition

This problem may arise if all of the following conditions are met.

(1) Neither the “-optimize=0” nor “-optimize=debug_only” option is specified.

(2) “sh2e”, “sh2afpu”, “sh4”, or “sh4a” is specified for the “-cpu” option.

(3) “single” is specified for the “-fpu” option, or the “-fpu” option is not specified.

(4) There is a loop whose loop control variable(Note 1) is any of the following types:

(4a) signed int type

(4b) unsigned int type

(4c) signed long type

(4d) unsigned long type

(5) The loop control variable in (4) satisfies either of the following conditions:

(5a) The initial value, incremental value, and exit decision value are a constant.

(5b) The initial value is 0 and the incremental value is 1. (Note 2) (Note 3)

(6) The absolute value of the loop control variable in (4) is greater than 16777216 (24th power of 2) in the loop in (4).

(7) The value in (6) is converted(Note 4) to the float type in the loop.

(8) The loop control variable in (4) is used only in (8a) to (8c) in the loop in (4).

(8a) The type conversion expression in (7)

(8b) The update statement of the loop control variable in (4)

(8c) The loop exit decision expression in (4)

Note 1: Except for a case in which the variable is qualified by volatile.

Note 2: This is also applied to cases in which the exit decision value is not a constant.

Note 3: In V.7.0B to V.7.1.05, in a case corresponding to the condition (5b), the problem does not occur.

Note 4: Implicit type conversion is also included.

1.4 Example

Parts that correspond to the conditions are described in red.

■ Source example: "-optimize=1", "-cpu=sh4a", and "-fpu=single" are specified (Conditions (1)(2)(3))

```
#define N1 (16777215)
#define N2 (16777221)
float  f_val;
void func(void)
{
  int cnt;                                  /* Condition (4a) */
  for ( cnt = N1 ; cnt < N2 ; cnt++ ) { /* Conditions (4a)(5a)(6)(8) */
    f_val = (float) cnt;                  /* Conditions (7)(8) */
    :                 :
  }
}
```

■ Result of compilation

```
:  :
MOV      #H'06,R2          ; Set 6 (=N2-N1) in R2.
MOV.L    @(H'0014:8,PC),R6 ; Set the f_val address to R6.
FMOV.S   @R0,FR8           ; Set 0x4B7FFFFF (=N1) in FR8.
MOVA     @(H'0014:8,PC),R0
FMOV.S   @R0,FR9           ; Set 0x3F800000 (=1) in FR9.
                  ; Loop processing start
DT       R2       ; Set R2-1 in R2. If R2 is not 0, T = 0, If R2 is 0, T = 1.
FMOV.S   FR8,@R6  ; Set the FR8 value in f_val.
BF/S     @H'202C:8; Run the following direction (FADD), and if T = 0, go back
                    to H'202C(DT R2).
                 ; If T = 1, exit the loop.
FADD     FR9,FR8  ; Add FR9 (0x3F800000=1) to FR8.
                 ;--> When 1 is added if FR8 is 16777216(=0x4B800000), the
                    value cannot be represented.
                 ; So, the loop is continued while FR8 remains at the same
                    value.
                 ; Any value is not added to FR8 in the subsequent loop
                    processing.
                 ; Loop processing end
RTS
NOP
```

1.5 Workaround

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

(1) Specify either of the “-optimize=debug_only” and the “-optimize=0” options. (Note 1)

(2) Modify the loop control variable in Condition (4) by declaring it as volatile.

(3) Change the type of the loop control variable in Condition (4) to the signed/unsigned long long type. (Note 2)

(4) Change the type after conversion in Condition (7) to the double type. (Note 3)

(5) Substitute the loop control variable in Condition (4) for the variable declared as volatile in the loop in (4).

Note 1: For V.7.0B to V.9.00 Release 04A, the "-optimize=debug_only" option is not supported and cannot be specified.

Note 2: This is a workaround for V.8.00 Release 00 to V.9.04 Release 03. For V.7.0B to V.7.1.05, the signed/unsigned long long type is not supported and the type cannot be changed to it.

Note 3: This is a workaround for when the "-fpu" option is not specified. When "single" is specified for the "-fpu" option, the problem cannot be worked around by Workaround (4)

1.6 Schedule for Fixing the Problem

This problem will be fixed in the next version. The release date has not yet been decided.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr. 16, 2017	-	First edition issued

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

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