date: 2003/04/24

RENESAS TECHNICAL UPDATE

Classification of Production	Development Environment		No	TN-CSX-050A/E	Rev	1
THEME	SuperH RISC engine C/C++ Compiler Ver.7 bug report (7)	Classification of Information	1. Spec change 2. Supplement of Documents (3.) Limitation of Use 4. Change of Mask 5. Change of Production Line			
PRODUCT NAME	P0700CAS7-MWR P0700CAS7-SLR P0700CAS7-H7R	Lot No. Ver.7.x	Reference Documents	SuperH RISC engine C/C++ Compiler Assembler Optimizing Linkage Editor User's Manual ADE-702-246A Rev.2.0	term of validity Eternity	

Attached is the description of the known bugs in Ver. 7 series of the SuperH RISC engine C/C++ compiler. Inform the customers who have the package version in the table below of the bugs.

	Package version	Compiler version
P0700CAS7-MWR	7.0B	7.0B
	7.0.01	7.0.03
	7.0.02	7.0.04
	7.0.03	7.0.06
	7.1.00	7.1.00
	7.1.01 7.1.02	7.1.01
P0700CAS7-SLR	7.0B	7.0B
	7.0.02	7.0.03
	7.0.03	7.0.04
	7.0.04	7.0.06
	7.1.00	7.1.00
	7.1.01 7.1.02	7.1.01
P0700CAS7-H7R	7.0B	7.0B
	7.0.02	7.0.03
	7.0.03	7.0.04
	7.0.04	7.0.06
	7.1.00	7.1.00
	7.1.01 7.1.02	7.1.01

The check tool can be downloaded from the following URL.
http://www.renesas.com/eng/products/mpumcu/tool/index.html

Attached: P0700CAS7-030411E
SuperH RISC engine C/C++ Compiler Ver. 7 Known Bugs Report(7)

SuperH RISC engine C/C++ Compiler ver. 7
Known Bugs Report (7)

The failures found in the ver. 7 series of the SuperH RISC engine C/C++ compiler are listed below. The check tool for item 1 or 2 can be downloaded from the following URL:

http://www.renesas.com/eng/products/mpumcu/tool/index.html

1. Illegal deletion of an unconditional branch

[Description]

When all of the following conditions are satisfied, the unconditional branch may be deleted illegally.

- The last of a function is conditional statement.
- Conditions are nested in the statement.
- The last condition finishes with a function call and a return statement, and the previous condition finishes with a function call.

[Example]

```
void sub(int parm) {
    if (parm == 0) {
      ;
    } else if (parm == 1) {
        ;
   } else if (parm == 2) {
      ;
   } else if (parm == 3) {
      ;
   } else if (parm == 4) {
      ;
   } else if (parm == 5) {
      func1();    /* <A> */
   } else {
      func2();    /* <B> */
      return;     /* <B> */
   }
   return;
}
```

```
 sub:
       STS.L       PR,@-R15
       TST         R4,R4
       BT          L11
       MOV         R4,R0
       CMP/EQ      #1,R0
       BT          L11
       CMP/EQ      #2,R0
       BT          L11
       CMP/EQ      #3,R0
       BT          L11
       CMP/EQ      #4,R0
       BT          L11
       CMP/EQ      #5,R0
       BF          L18
       MOV.L       L20+2,R2 ; _func1
       JSR         @R2
       NOP
L11:
                           ; A branch to L19 is deleted
L18:
       MOV.L       L20+6,R2 ; _func2
       JMP         @R2      ; This function is always called
       LDS.L       @R15+,PR
L19:
       LDS.L       @R15+,PR
       RTS
       NOP
```

[Conditions]
This problem may occur when all of the following conditions are satisfied.
Instances of this bug in the program can be found using the check tool.
(1) The optimize=1 option is specified.
(2) The last of a function is conditional statement and the conditions are nested.
(3) The last condition finishes with a function call and a return statement (<B> in the example).
(4) The condition previous to (3) finishes with a function call (<A> in the example).

[Solution]
If a relevant failure exists, prevent the problem by either of the following methods.
(1) Specify the optimize=0 option to compile the file.
(2) Add the nop() intrinsic function after <A>.
<Example>

```
#include <machine.h>    /* Added for nop() */
                :
    } else if (parm == 5) {
        func1();    /* <A> */
        nop();      /* Added */
    } else {
                :
```

2. Illegal cast from unsigned integer to float
[Description]
When the unsigned integer type variable is cast to the float type, the cast may be deleted illegally.

[Example]

```
unsigned short us1;
volatile unsigned short us0;
volatile float f0;
float *p;

void func() {
     f0 = *p = ((float)us0, (float)us1);
}
```

```
       MOV.L       L29+50,R2; _us0
       MOV.L       L29+54,R5; _p
       MOV.W       @R2,R6
       MOV.L       L29+58,R6; _us1
       MOV.W       @R6,R2
       EXTU.W      R2,R6
       MOV.L       @R5,R2
       MOV.L       R6,@R2   ; store to *p without cast to float type
       MOV.L       @R5,R2
       MOV.L       @R2,R6
       MOV.L       L29+10,R2; _f0
       RTS
       MOV.L       R6,@R2   ; store to f0 without cast to float type
```

[Conditions]
This problem may occur when all of the following conditions are satisfied.
Instances of this bug in the program can be found using the check tool.
(1) The unsigned integer variable is cast to float type.
(2) The unsigned integer variable is cast to double type and either double=float or fpu=double option is specified, or is cast to long double type and fpu=single option is specified.

[Solution]
If a relevant failure exists, prevent the problem by the following method.
(1) Cast the variable to signed integer type which preserves value (or to long double if the variable is unsigned int/long type) at first and cast the variable to float type.

3. Illegal movement of stack pointer with ld_ext() or st_ext()

[Description]

When an ld_ext() or st_ext() intrinsic function is used and a local array is specified as a parameter, the stack pointer may be moved illegally.

[Example]

```
   #include <machine.h>

   void main() {
      float table[4][4], data1[4][4], data2[4][4];
               :
        ld_ext(table) ;
        mtrx4mul(data1,data2) ;
               :
}

                  :
          FRCHG
          FMOV.S      @R15+,FR0     ; R15 is moved. When an interrupt occurs, upper area
of
                                    ; stack is destroyed
          FMOV.S      @R15+,FR1     ;                          |
          FMOV.S      @R15+,FR2     ;                          |
          FMOV.S      @R15+,FR3     ;                          |
          FMOV.S      @R15+,FR4     ;                          |
          FMOV.S      @R15+,FR5     ;                          |
          FMOV.S      @R15+,FR6     ;                          |
          FMOV.S      @R15+,FR7     ;                          |
          FMOV.S      @R15+,FR8     ;                          |
          FMOV.S      @R15+,FR9     ;                          |
          FMOV.S      @R15+,FR10    ;                          |
          FMOV.S      @R15+,FR11    ;                          |
          FMOV.S      @R15+,FR12    ;                          |
          FMOV.S      @R15+,FR13    ;                          |
          FMOV.S      @R15+,FR14    ;                          |
          FMOV.S      @R15+,FR15    ;                          |
          FRCHG                     ;                          |
          ADD         #-64,R15      ;                          V
```

[Conditions]

This problem may occur when all of the following conditions are satisfied.

(1) The cpu=sh4 option is specified and the ld_ext() or st_ext() intrinsic function is used.

(2) A local array is specified as the parameter.

[Solution]

If a relevant failure exists, prevent the problem by either of the following methods.

(1) Specify the optimize=0 option to compile the file.

(2) Change the parameter to a global array.