date: 2003/10/30

RENESAS TECHNICAL UPDATE

Classification of Production	Development Environment		No	TN-CSX-054A/E	Rev	1
THEME	SuperH RISC engine C/C++ Compiler Ver.7 bug report (9)	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-702-372A 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.
The bugs will affect the package version shown in the table below.

	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
	7.1.03	7.1.02
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
	7.1.03	7.1.02
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
	7.1.03	7.1.02

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

Attached: P0700CAS7-030923E
SuperH RISC engine C/C++ Compiler Ver. 7 Known Bugs Report(9)

SuperH RISC engine C/C++ Compiler ver. 7
Known Bugs Report (9)

The failures found in the ver. 7 series of the SuperH RISC engine C/C++ compiler are listed below.
The check tool can be downloaded from the following URL:
http://www.renesas.com/eng/products/mpumcu/tool/index.html

1. Illegal EXTS/EXTU deletion after NEG

[Description]

When expressions (A and B) which include the same unsigned char/short type variable exist in the forms of A-B and B-A in the same function, an EXTS/EXTU instruction is deleted illegally by the common subexpression elimination.

[Example]

```
unsigned short var_a, var_b, var_c;
long result;

void f() {
   unsigned short x;
   if (var_a >= var_b) {
      x = var_a - var_b;
      result = x * var_c;
   } else {
      x = var_b - var_a;
      result = x * var_c;
   }
}

_f:
       MOV.L      L14,R2     ; _var_a
       MOV.L      L14+4,R4   ; _var_b
       MOV.W      @R2,R5
       MOV.W      @R4,R2
       MOV.L      L14+8,R4   ; _var_c
       MOV        R5,R6
       SUB        R2,R6      ; temp <- var_a-var_b
       MOV.W      @R4,R7
       EXTU.W     R5,R5
       EXTU.W     R2,R2
       CMP/GE     R2,R5
       BF/S       L12
       EXTU.W     R7,R4
       EXTU.W     R6,R2      ; x <- (unsigned short)temp
       MOV.L      L14+12,R5  ; _result
       MUL.L      R2,R4
       STS        MACL,R2
       RTS
       MOV.L      R2,@R5
L12:
       EXTU.W     R6,R6
       MOV.L      L14+12,R5  ; _result
       NEG        R6,R2      ; x <- (long)(-temp)
                             ; EXTU.W R2,R2 is deleted illegally
       MUL.L      R2,R4
       STS        MACL,R2
       RTS
       MOV.L      R2,@R5
```

[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) A variable which is declared with unsigned char/short type is used in the following expressions in the same function.

A-B and B-A

A and B are expressions which include the same unsigned char/short variable.
In the upper example, A is var_a and B is var_b.

(3) These expressions are target of common subexpression elimination (CSE). CSE works as follows in the above example.

```
void f() {
   unsigned short x;
   long temp = var_a – var_b;
   if (var_a >= var_b) {
      result = (unsigned short)temp * var_c;
      /* var_a-var_b is replaced. */
   } else {
      result = (unsigned short)(-temp) * var_c;
      /* var_b-var_a is replaced. */
   }
}
```

[Solution]

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

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

(2) Declare that unsigned char/short type variable as volatile, or declare the variable to which an expression including that unsigned char/short variable is assigned as volatile.

<Example>

```
void f() {
   volatile unsigned short x;  /* add volatile */
   if (var_a >= var_b) {
      x = var_a-var_b;
      result = x * var_c;
   } else {
      x = var_b - var_a;
      result = x * var_c;
   }
}
```

2. Illegal EXTU deletion after load instruction

[Description]

An EXTU instruction after a load instruction may be deleted illegally in the expression where a datum pointed to by a pointer to a variable of unsigned char/short type is added by 0, subtracted by 0 or multiplied by 1.

[Example]

```
unsigned char *p;
int a;
void func(){
   a = 0;
   a += *p;
}

_func:
       MOV.L      L11,R5     ; _a
       MOV        #0,R2      ; H'00000000
       MOV.L      R2,@R5
       MOV.L      L11+4,R2   ; _p
       MOV.L      @R2,R6
       MOV.B      @R6,R2     ; R2 is sign extended.
       RTS
       MOV.L      R2,@R5     ; R2 is stored in 4-byte area without zero-extension.
```

[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) A variable of unsigned char/short type is accessed via a pointer.
(3) An addition or subtraction by 0, or a multiplication by 1 is performed against this variable.
The optimization may induce an addition or subtraction by 0, or a multiplication by 1.

[Solution]

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

(1) Remove the addition, subtraction or multiplication if an addition or subtraction by 0, or a multiplication by 1 is described explicitly.
<Example>

```
void func(){
    a = *p;
}
```

(2) Specify the optimize=0 option to compile the file.
(3) Assign the datum pointed to by the pointer to a variable of unsigned char/short type to a variable qualified with volatile, and use this variable instead..
<Example>

```
void func() {
    volatile unsigned char temp = *p;
    a = 0;
    a += temp;
}
```