To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Date: Jun.24.2004

RENESAS TECHNICAL UPD

Nippon Bldg., 2-6-2, Ohte-machi, Chiyoda-ku, Tokyo 100-0004, Japan
RenesasTechnology Corp.

Product Category	User Development Environment		Document No.	TN-CSX-077A/EA	Rev.	1.0
Title	SuperH RISC engine C/C++ Compiler ver.7 Known Bugs Report(12)		Information Category	Usage Limitation		
Applicable Product	P0700CAS7-MWR P0700CAS7-SLR P0700CAS7-H7R	Lot No. Ver.7.x	Reference Document	SuperH RISC engine C/C++ Compiler Assembler Optimizing Linkage Editor User's Manual REJ10B0047-0100H Rev.1.00		

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 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
	7.1.04	7.1.03
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
	7.1.04	7.1.03
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
	7.1.04	7.1.03

Attached: P0700CAS7-040611E

SuperH RISC engine C/C++ Compiler Ver. 7 Known Bugs Report (12)

SuperH RISC engine C/C++ Compiler ver.7
Known Bugs Report(12)

The bugs detected in the ver.7 of the SuperH RISC engine C/C++ Compiler is shown below.

1. Incorrect removing of sign/zero extension instruction in an expression in a switch statement

[Description]

When a 1- or 2-byte parameter was used as an expression in a switch, sign/zero extension expression was removed incorrectly, then the destination address might be incorrect.

[Example]

```
     int func(short x) {
     short r = -1;
     switch(x) {
     case 0: r = 0; break;
     case 1: r = 1; break;
     case 2: r = 2; break;
     case 3: r = 3; break;
     case 4: r = 4; break;
     case 5: r = 5; break;
     case 6: r = 6; break;
     case 7: r = 7; break;
     case 8: r = 8; break;
     }
     return (r);
   }

   _func:
          MOV      R4,R2    ; The upper 2 bytes of R4 are undefined value.
          EXTS.W    R4,R4
          MOV      #8,R3
          CMP/HI    R3,R4
          MOV      #-1,R6
          BT       L23
          SHLL      R2      ; Shift value without sign/zero extension
          MOVA      L25,R0
          MOV.W     @(R0,R2),R1
          ADD      R1,R0
          JMP      @R0
          NOP
   L24:
   L25:
          .DATA.W   L12-L25
          .DATA.W   L13-L25
              :
```

[Conditions]

This problem might occur when all of the following conditions were fulfilled.

(1) The optimize=1 option was specified.
(2) A function had a 1- or 2-byte parameter.
(3) The function of (2) had a switch statement.
(4) An expression in the switch had the its parameter.
(5) The switch statement was generated as the jumping to a table expansion method.

[Solution]

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

(1) Specify optimize=0.
(2) Apply explicitly cast to the type of the parameter to the expression in the switch statement.
    Example: switch((short)x)
(3) Declare the parameter as volatile.

2. Incorrect removing of zero extension instruction

[Description]

When an unsigned char/unsigned short type variable was referred to twice or more in a loop, zero extension instruction might be removed illegally.

[Example]

```
          MOV.B     @Rm,Rn
          EXTU.B    Rn, Rn   ; Clear the upper three bytes
              :
          MOV.B     Rn,@R15  ; Assign to a stack
              :
          MOV.B     @R15,R12 ; Assign to R12
                             => EXTU.B R12,R12 was removed illegally
        L1:
              :
          CMP/EQ    R12,R2   ; a value of R12 was incorrect
              :
          BT        L1
              :
```

[Conditions]

This problem might occur when all of the following conditions were fulfilled.

(1) The optimize=1 option was specified.
(2) An unsigned char/unsigned short type variable existed.
(3) The variable of (2) was referred to twice or more in a loop.
(4) The variable of (2) was not assigned to a register.
(5) A register which was not used in the loop of (3) existed.
(6) The register of (5) was used out of the loop.

[Solution]

If a relevant failure exists, prevent the problem by the following method.

(1) Specify optimize=0.

3. Incorrect calculation of quadratic expression of loop induction variable

[Description]

If a quadratic expression had a loop induction variable i of the form "m * (i * i + b * i)", the expression might be treated as incorrectly.

[Example]

```
int a[100];
f() {
   int i;
   for (i=0;i<100;i++){
      a[i] = 3 * (i * i + 555 * i);  /* incorrectly expanded as 3*i*i+555*i */
   }
}
```

[Conditions]

This problem might occur when all of the following conditions were fulfilled.

(1) The optimize=1 option was specified.
(2) A loop existed.
(3) The loop of (2) had int/unsigned int/long/unsigned long-type loop induction variable.
(4) A quadratic expression had the loop induction variable of (3) in the loop of (2).
(5) The expression of (4) had the form of “m*(i*i+b*i)”.
(i : loop induction variable m,b : variable or const value)

[Solution]

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

(1) Specify optimize=0.
(2) Declare the loop induction variable as volatile.
(3) Declare the loop induction variable as other than int/unsigned int/long/unsigned long type variable.
(4) Distribute coefficient m of the quadratic expression to i*i and b*i.
Example: 3 * (i * i + 555 * i) => 3 * i * i + 3 * 555 * i

4. Incorrect removing of sign/zero extension instruction in the addition/subtraction/multiplication

[Description]

When an addition/subtraction/multiplication was assigned to a variable with the type of smaller size or cast to the type of smaller size, and the result was used for addition/subtraction/multiplication, sign/zero extension might be removed incorrectly.

[Example]

```
   int x,a;
   test_000()
   {
      char b;
      b = (char)(a + 3);
      x = b + 2;
   }

_f:
         MOV.L     L11,R6    ; _a
         MOV.L     L11+4,R2  ; _x
         MOV.L     @R6,R6
         ADD      #5,R6     ; cast to char type was removed illegally
                            ; and a+5 was assigned to the variable x.
         RTS
         MOV.L     R6,@R2
```

[Conditions]

This problem might occur when all of the following conditions were fulfilled.

(1) The optimize=1 option was specified.

(2) An addition/subtraction/multiplication had either of operands was a constant value.

(3) One of the following conditions (a)(b) was fulfilled.

(a) The result of (2) was cast to the type of smaller size, and the result was used for addition/subtraction/multiplication.

(b) The result of (2) was assigned to a variable with the type of smaller size, and the result was used for addition/subtraction/multiplication.

[Solution]

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

(1) Specify optimize=0.

(2) Assign the result of the condition (2) to a variable which is declared as volatile.

5. Incorrect removing of sign/zero extension of a constant division (SHC-0001)

[Description]

When a divisor and a dividend were cast to the type of smaller size at a constant division and the result of the division or the residue was assigned to a variable with a type after the cast, the cast might be removed illegally.

[Example]

```
char c;
int i;
func1(){
   c = ((char)i / (char)2); /* a dividend was not cast to char type */
}

func2(){
   c = ((char)i / (char)0x102); /* a divisor was not changed into 0x2 */
}
```

[Conditions]

This problem might occur when all of the following conditions were fulfilled.

(1) The optimize=1 option was specified.
(2) A constant division existed.
(3) A divisor and a dividend ware cast to the type of smaller size at a constant division of (2).
(4) The divisor was a power of 2, or other than cpu=sh1 option and the division=cpu=inline option were specified.
(5) The result of the division was assigned to a variable with a type after the cast.

[Solution]

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

(1) Specify optimize=0.
(2) Delete the cast of the divisor and replace the divisor by a value after the cast.
Example func1(): c = ((char)i / (char)2); => c = ((char)i / 2);
func2(): c = ((char)i / (char)0x102); => c = ((char)i / 0x02);
(3) Assign the result of the division to a int-type variable.
Example func1(): tmp = ((char)i / (char)2); (tmp : int-type variable)
c = (char)tmp;