HITACHI SEMICONDUCTOR TECHNICAL UPDATE

Classification of Production	Development Environment		No	TN-CSX-049A/E	Rev	1
THEME	SuperH RISC engine C/C++ Compiler Ver.7 bug report (6)	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. All	Reference Documents	SuperH RISC engine C/C++ Compiler Assembler Optimizing Linkage Editor User's Manual ADE-702-246A Rev.1.0	Effective Date 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	7.1.01
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	7.1.01
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	7.1.01

The check tool or corrected file can be downloaded from the following URL.
http://www.hitachisemiconductor.com/sic/jsp/japan/eng/products/mpumcu/tool/

Attached: P0700CAS7-030225E
SuperH RISC engine C/C++ Compiler Ver. 7
Known Bugs Report(6)

SuperH RISC engine C/C++ Compiler ver. 7
Known Bugs Report (6)

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

http://www.hitachisemiconductor.com/sic/jsp/japan/eng/products/mpumcu/tool/

1. Illegal deletion of EXTU instruction

[Description]

When an unsigned variable is used more than once in a loop, the EXTU instruction may be deleted illegally.

[Example]

```
extern unsigned char X;
int sub(int a, int b, int n) {
    int i, sum=0;
    for (i = 0; i < n; i++) {
        if (X == (unsigned char)0xff) { // X is used.
            sum += a;
        }
        if (X == (unsigned char)0xf0) { // X is used.
            sum += b;
        }
        X = X + 1;                      // X is defined.
    }
    return (sum);
}
```

```
_sub:
          MOV.L     R12,@-R15
          MOV.L     R13,@-R15
          MOV.L     R14,@-R15
          MOV       R5,R13
          MOV       #0,R5      ; H'00000000
          MOV.L     L19,R1     ; _X
          MOV       R6,R7
          MOV       R4,R12
          MOV       R5,R6
          MOV       #-1,R4     ; H'FFFFFFFF
          MOV.B     @R1,R2
                               ; <- delete EXTU.B R2,R2
          MOV       #-16,R14   ; H'FFFFFFF0
          EXTU.B    R4,R4
          BRA       L11
          EXTU.B    R14,R14
L12:
          CMP/EQ    R4,R2      ; A result of comparison may be illegal. (X > 127)
          BF        L14
          ADD       R12,R5
L14:
          CMP/EQ    R14,R2     ; A result of comparison may be illegal. (X > 127)
          BF        L16
          ADD       R13,R5
L16:
          ADD       #1,R2
          EXTU.B    R2,R2
          ADD       #1,R6
L11:
          CMP/GE    R7,R6
          BF        L12
          MOV.B     R2,@R1
          MOV       R5,R0
          MOV.L     @R15+,R14
          MOV.L     @R15+,R13
          RTS
          MOV.L     @R15+,R12
```

[Conditions]

This problem may occur when all of the following conditions (1) to (4) or (5) to (8) are satisfied.
Instances of this bug in the program can be found using the check tool.

(1) The optimize=1 option is specified.
(2) An unsigned char/unsigned short variable is used.
(3) This variable is used more than once in a loop, and defined after using.
(4) This variable is loaded from the memory at first.

(5) The optimize=1 option is specified.
(6) An unsigned char/unsigned short local variable is used.
(7) This variable is used as a source variable of right shift operation in a loop.
(8) The shift count is more than one and SHLR2, SHLR8 or SHLR16 is used.

[Solution]

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

(1) Declare this variable as (unsigned)long.
(2) Specify the optimize=0 option to compile the file.

2. Illegal deletion of floating-point constant load

[Description]

When the same floating-point constants are used more than once in a loop and out of loop, the load instruction of the constants may be deleted illegally.

[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 same floating-point constants are used more than once in a loop.
(3) This value is used outside the loop.

[Solution]

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

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

3. Illegal array access

[Description]

When an array index is C-exp, where C is a constant value and exp is unsigned char/short type expression , the element of the array is accessed illegally.

[Example]

```
unsigned char dd[2];
void func(unsigned char a, unsigned char b) {
    dd[15-a]=b1;
}

_func:
    MOV.L     L11,R2        ; _dd
    NEG       R4,R6         ; R6 <- -a
    EXTU.B    R6,R0         ; -a is zero-extended
    ADD       #15,R2
    RTS
    MOV.B     R5,@(R0,R2)   ; Illegal address is accessed
```

[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) This array element is accessed by “constant value - expression” (dd[15-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) The array element shall be accessed by “-expression(unsigned char/unsigned short) + constant value” (dd[-a+15] in the example).

4. Illegal copy by memmove
[Description]
When memmove of the standard library copies some characters from an address to the greater address, more characters than specified may be copied.

[Conditions]
This problem occurs when all of the following conditions are satisfied.
(1) The move size is more than 30-byte.
(2) The destination address is greater than the source address.
(3) (The source address + the movement size) is inside the destination area.
(4) The source address or the destination address is not a power of 4.

[Solution]
If a relevant failure occurs, prevent the problem by using the corrected file or by the following method.
(1) Divide the move size into less than 31-byte.