HITACHI SEMICONDUCTOR TECHNICAL UPDATE

Classification of Production	Development Environment		No	TN-CSX-039A/E	Rev	1
THEME	SuperH RISC engine C/C++ Compiler Ver.7.0 bug report (3)	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.2.0	Effective Date	Eternity

Attached is the description of the known bugs in Ver. 7.0 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
P0700CAS7-SLR	7.0B	7.0B
	7.0.02	7.0.03
	7.0.03	7.0.04
	7.0.04	7.0.06
P0700CAS7-H7R	7.0B	7.0B
	7.0.02	7.0.03
	7.0.03	7.0.04
	7.0.04	7.0.06

The checker of the bugs is on the URL below for downloading.

http://www.hitachisemiconductor.com/sic/jsp/japan/eng/products/mpumcu/tool/download/caution7002.html

Attached: P0700CAS7-020627E
SuperH RISC engine C/C++ Compiler Ver. 7
Known bugs in this release (3)

SuperH RISC engine C/C++ compiler Ver.7
Known bug in this release (3)

The known bug in Ver.7.0B, Ver.7.0.03 and Ver.7.0.04 of the compiler is described below. Instances of this bug in the program can be found using the checker on the URL below.

http://www.hitachisemiconductor.com/sic/jsp/japan/eng/products/mpumcu/tool/download/caution7002.html

1. Illegal stack access

[Contents]

When an address of a parameter passed on a register is referred to, a data on the stack may be superseded illegally.

[Example]

```
extern void er();
extern char f2(char *a);
void f(char a) {
    char c;
    a++;                // updates the variable which is a parameter on a register
    do {
        c=f2(&a);      // refers to an address of a parameter on a register
        if (c) er();
        else return;
    } while(c);
}

_f:
          MOV.L       R13,@-R15
          MOV.L       R14,@-R15
          STS.L       PR,@-R15
          ADD         #-4,R15
          MOV         R4,R0
          ADD         #1,R0
          MOV.B       R0,@(3,R15)   ; updates the variable "a"
          MOV.L       R4,@R15   ; the variable "a" is superseded illegally
          MOV.L       L14+2,R13 ; _f2
                    :
```

[Condition]

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

(1) The optimize=1 option is specified.

(2) A parameter on a register exists.

(3) An address of this parameter is referred to in a function.

(4) An assignment to this parameter exists before reference to the address.

(5) Copying a parameter on a register into the stack is moved after assignment to the parameter to the parameter by the optimization of instruction scheduling.

[How to avoid the bug]

The bug can be avoided with either method of the following.

(1) Specify the optiomize=0 option.

(2) Modify the source program as shown in the example below.

[Example]

```
void f(char a) {
    char c;
    char temp=a;        // A parameter "a" assigns to a  local variable "temp"
    temp++;              // Updates "temp"
    do {
        c=f2(&temp);    // Refers to an address of "temp"
        if (c) er();
        else return;
    } while(c);
}
```