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.

HITACHI SEMICONDUCTOR TECHNICAL UPDATE

DATE	25 October 2000	No.	TN-CSX-021A/E
THEME	Caution in using H8S,H8/300 Series C/C++ compiler Package Ver.3.0C		
CLASSIFICATION	☐ Spec. change ☐ Supplement of documents	☒ Limitation on Use	
PRODUCT NAME	PS008CAS3-MWR PS008CAS3-SLR PS008CAS3-H7R	Lot No. etc.	Ver3.0, Ver.3.0A, Ver3.0B, Ver.3.0C
REFERENCE DOCUMENT	PS008CAS3-001002E (attached)	Effective Date	Eternity
		From	

There are several restrinction on H8S,H8/300 series C/C++ compiler Ver. 3.0C.

Refer to the attached document, PS008CAS3-001002E, for details.

A user who has the following product should be notified.
H8S,H8/300 series C/C++ compiler Package Ver. 3.0 or later.

Attached:

(1) “Caution in using H8S,H8/300 Series C/C++ compiler V3.0C”
(PS008CAS3-001002E), 12 page

****
* Caution in using H8S, H8/300 Series C/C++ Compiler V3.0C *
****

There are following restrictions on H8S, H8/300 Series C/C++ Compiler V3.0C. Please use with caution.

1. Occurrence of 4099 error for arrays

[Symptom]

Internal error 4099 occurs when compiling following source program with the C compiler.

```
extern char TBL[ ][12];

void func(char i,char x,char y)

{

   TBL[i][0] = TBL[x][i] + TBL[y][i];

}
```

[Condition]
This occurs when all of the following conditions are satisfied.

(1) Following type of variables are used as indices of an array:

Case of cpu=300, 300HN, 2600N, 2000N: [unsigned]char

Case of cpu=300HA, 2600A, 2000A: [unsined]char/short/int

(2) There are more than 3 indices of (1) in one basic block.

[Workaround]
Avoid by one of the following:

(1) Add volatile declaration to the index variable.

(2) Change the array index type to the followings.

Case of cpu=300,300HN, 2600N, 2000N: [unsigned]int

Case of cpu=300HA, 2600A, 2000A: [unsined]long

[Product Name]
H8S, H8/300 series C/C++ Compiler V3.0 or later

2. Illegal code for switch statement
[Symptom]

When compiling following C source program, the code of switch statement is invalid.

[Example] Compile with -cpu=2000n

```
extern unsigned char s0,s1,s2;
void test(void)
{
unsigned char ub0,ub1,ub2;
ub0 = s0;
ub1 = s1;
ub2 = s2;
if (ub1)
        ub2 = ub1;
switch(ub2) {
case 1:
        f1(ub0); break;
case 2:
        <omitted>
(output code)
    MOV.B   @_s0:16,R6L
    MOV.B   @_s1:16,R6H
    MOV.B   @_s2:16,R0L
    MOV.B   R6H,R6H
    BEQ     L2047:8
    MOV.B   R6H,R0L
L2047:
    ADD.B   #-1:8,R0L
    CMP.B   #7:8,R0L
    BHI     L1044:8
    EXTU.W  R0                      (correct code)
    MOV.B   @(L2049:16,ER5),R5L   <-  MOV.B   @(L2049:16,ER0),R0L
    ADD.W   #LWORD L1036:16,R5  <-  ADD.W   #LWORD L1036:16,R0
    JMP     @ER5                <-  JMP     @ER0
```

<omitted>

[Condition]
This occurs when all of the following conditions are satisfied.

(1) Source program includes a switch statement.

(2) There are more than 2048 identifiers (function names, variable names, labels etc.) in the source program including labels generated by the compiler).

(3) A local variable used in the switch statement is not used in the subsequent code.

(4) Switch statement generates jump table.

[Workaround]
Avoid by one of the following:

(1) Decrease number of identifiers in the source program by separating files.

(2) Change code generation method of switch statement by add -case=ifthen option.

[Product name]
H8S, H8/300 Series C/C++ Compiler V3.0 or later

3. Invalid branch code for switch statement

[Symptom]

When compiling following C source program, the code of switch statement may be invalid and causes incorrect branch. (compiled with -cpu=300, -case=table)

```
 extern char  exp, x[5], y[5];

void test(void)

{

      switch ( exp ) {

      case  1 : x[0] = 1; break;

      case  2 : x[3] = 3; break;

      case  3 : func(x[1], y[1]);

                  func(y[2], y[3]);

                <omitted>

                  break;


}

  < omitted>
```

(output code)

```
MOV.B      @exp:16,R0L

ADD.B      #-1:8,R0L

CMP.B      #2:8,R0L

BLS        $+6

JMP        @L13:16

SUB.B      R0H,R0H
```

(Correct Code)

```
MOV.B      @(L12:16,R0),R0L  <-

ADD.B      #LOW L8:8,R0L    <-  ADD.W  R0, R0

ADDX.B     #HIGH L8:8,R0H   <-  MOV.W  @(L12:16,R0),R0

JMP        @R0

L8:                ; case label

MOV.W      #1,R0

MOV.W      R0,@_x:16

BRA        L11

L9:                ; case label

MOV.W      #3,R0
```

```
        MOV.W       R0,@_x+6:16
        BRA         L11
L10:                        ; case label
        MOV.W       @_y+2:16,R1
        MOV.W       @_x+2:16,R0
        JSR         @_func:16
        MOV.W       @_y+4:16,R1
        MOV.W       @_x+4:16,R0
        JSR         @_func:16
                <omitted>

L11:                        ; default label
        POP.W       R6
        RTS
            <omitted>
L12:
                                (correct code)
        .DATA.B     L8-L8           <-  .DATA.W  L8
        .DATA.B     L9-L8           <-  .DATA.W  L9
        .DATA.B     L10-L8          <-  .DATA.W  L10
        .DATAB.B    1,0
        .END
```

[Condition]

This occurs when all of following conditions are satisfied.

(1) Source program includes a switch statement.

(2) The switch statement does not include default label.

(3) The switch statement generates jump table.

(4) The object code size of the switch statement is more than 256 bytes.

[Workaround]

Avoid by one of the following.

(1) Add dummy default label to the switch statement.

(2) Change code generation method of switch statement by add -case=ifthen option.

[Product Name]

H8S, H8/300 Series C/C++ Compiler V3.0 or later

4. Invalid object code for loop condition

[Symptom]

When compiling following C source program, the object code for the loop condition is invalid.

```
unsigned int x;
int func()
{
 int i;
 for (i=15; i>=0; --i){
  x=(((i*16)+15)<<8);
 }
 return (x);
}
```

```
unsigned int x;
int func()
{
 int i;
 for (T1=0xff00; T1>=0x0f00; ){
  x=T1;
  T1=T1+0xf000;
     /* The addend (0xf000) is of signed type, and */
     overflow occurs */
 }
 return (x);
}
```

[Condition]

This occurs when all of the following conditions are satisfied.

(1) There is a loop with an induction variable (in this case "i", which is multiplied in each iteration, and can be optimized by replacing the multiplication by an addition).

(2) Induction variable of (1) is not optimized to a post increment/decrement.

(3) The initial value of the induction variable is (after multiplication) out of the range of its type.

(4) The induction variable is of signed type.

[Workaround]

Avoid by one of the following:

(1) Compile without optimization option.

(2) Cast one of the multiplier to an unsigned type (in this case, 16).

[Product Name]

H8S, H8/300 series C/C++ Compiler V3.0 or later

5. Invalid debug information when regparam=3 is specified

[Symptom]

When debugging a program compiled by regparam=3 option, the debugger may refer to invalid location for local variables.

[Condition]

This occurs when all of the following conditions are satisfied.

(1) CPU is 300, 300hn, or 300ha

(2) The option -regparam=3 is specified

(3) Register save/restore operation in function entrance/exit is done by run-time routine (*1)

(4) Local variables are allocated in stack

*1: Register save/restore run-time routines

$sp_rgsv3, $sp_rgld3, $sprgld23, $fp_rgsv3, $fp_rgld3

[Workaround]

Compile with regparam=2.

[Product Name]

H8S, H8/300 Series C/C++ Compiler V3.0 or later

6. Illegal destructor call for temporary object for right-hand-side operand of "||"

[Symptom]

When the expression of right-hand side of logical or (||) uses a temporary object, its destructor is called even if the right-hand-side expression is not evaluated.

```
#include <stdio.h>
class A {
public:
   A(){ printf("A::A()\n"); }
   ~A(){ printf("A::~A()\n"); }
   int size(){ return 1; }
};
extern A k();
void f(int n, int ch)
{
   if( 1 || k.size() > 0){    // Right-hand side is not evaluated, as the left-hand side is true
   }
   // Destructor for a temporary object of class A is called here
}
```

[Condition]

This occurs when the right-hand side of logical or operation uses a temporary object.

[Workaround]

Don't write expression which uses a temporary object in the right-hand side of a logical or operation.
Make sure that an expression using a temporary object is in the position which is always evaluated.

[Product Name]

H8S, H8/300 C/C++ Compiler V3.0 or later

7. Invalid alignment of members in a class with static members

[Symptom]

In following example, alignment of A::a is 1, which is invalid.

```
class A {
  static char c;      // declared before the static member "l".
  static A a[1];
  long l;
};
char A::c;
A A::a[1];            // Alignment is 1 (should be 4)
```

[Condition]

This occurs when all of the following conditions are satisfied.

(1) The first static class member is of char type.

(2) After (1), a static class member with the same class type as the enclosing class is declared.

(3) Another class member is declared with the alignment larger than 1.

[Workaround]

Don't declare a static class member of char type at the beginning of a class.

[Product Name]

H8S, H8/300 C/C++ Compiler V3.0 or later

8. Internal error (4099) for the initialization of a pointer with a function member of the base class

[Symptom]

When the following code is compiled, internal error (4099) occurs.

```
class parent {
public:
   void parent_func();
};
typedef void (parent::*class_func_ptr)();
class child : public parent {
   void child_func();
};
class_func_ptr func_ptr = (class_func_ptr)&child::child_func;
```

[Condition]

This occurs when all of the following conditions are satisfied.

(1) A pointer to a member function of the derived class is initialized with a pointer to a member function of its base class.

(2) The initial value includes a cast from the derived class to the base class.

[Workaround]

Temporarily assign the initial value to a pointer with appropriate type, and then initialize.

<Example>

```
void (child::*temp)() = &child::child_func;
class_func_ptr func_ptr = (class_func_ptr)temp;
```

[Product Name]

H8S, H8/300 C/C++ Compiler V3.0 or later

9. Invalid this Pointer when calling a virtual function

[Symptom]

Offset of this pointer is invalid when calling a virtual function in the following example.

```
#include <stdio.h>
class A {
public:
   virtual void f() = 0;
};
class B : public A {
public:
   int b;
   void f() { printf("b = %d¥n", b); }
   B(int ii = 0) : b(ii) { }
};
class C1 : public B {
public:
   C1(int ii = 0) : B(ii) { }
};
class C2 : public B {
public:
   C2(int ii = 0) : B(ii) { }
};
class D : public C1, public C2 {
public:
   D() : C1(1), C2(2) { }
};
D dd;
main()
{
   C2* cp = &dd;
   cp->f();                    // b is set to 1
}                              // this pointer points to C1, not C2
```

[Condition]

This occurs when all of the following conditions are satisfied.

(1) The class (D) has multiple inheritance from classes with 3-level inheritance (A-B-C1, A-B-C2).

(2) Base class (A) declares a virtual function (f), and the second-level derived class (B) has the virtual function with the same name (f).

(3) The virtual function (f) of the class B accesses the member of the class B.

(4) The virtual function (f) is called using one of the class next to D (C2).

With above conditions, this pointer of *C2 is invalid.

[Workaround]

In the class from which the virtual function is called (C2), add a virtual function with the same name, in which the virtual function of the base class is called. The following code shows the workaround for the example above:

```
class C2 : public B {
  public:-
    C2(int ii = 0) : B(ii) { }
    virtual void f() { B::f(); }        // Call to the virtual function of the base class is added
};
```

[Product Name]

H8S, H8/300 C/C++ Compiler V3.0 or later