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.

RENESAS

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-070A/EA	Rev.	1.0
Title	H8S, H8/300 Series C/C++ Compiler Ver.6.0.01 bug information		Information Category	Usage Limitation		
Applicable Product	PS008CAS6-MWR PS008CAS6-SLR PS008CAS6-H7R R0C40008XSW06R R0C40008XSS06R R0C40008XSH06R	Lot No.	Reference Document	H8S, H8/300 Series C/C++ Compiler Assembler Optimizing Linkage Editor User's Manual REJ10B0058-0100H Rev.1.00		
		Ver.6.0.01				

Attached is the description of the detected bug information in Ver.6.0.01 of the H8S, H8/300 Series C/C++ Compiler.

The bug will affect this package version.

Attached: PS008CAS6-040402E

     Problems with the H8S, H8/300 Series C/C++ Compiler Ver.6.0.01

RENESAS

Problems with the H8S, H8/300 Series C/C++ Compiler Ver.6.0.01

Problems with the H8S, H8/300 series C/C++ compiler ver.6.0.01 are listed below.

1) Illegal access to a member of a structure/union parameter

   If a member of a structure/union was used as a 4-byte or less parameter with the structreg option specified, the value might not be correctly set to the member.

[Example]

```
typedef struct{
   char stc_1;
   char stc_2;
   int  stc_3;
}ST;

void f045(ST p1_str, ST p2_str ) {
   ST *lp1;
   ST *lp2;

   lp1 = &p2_str;
   lp2 = &p2_str;
   p2_str.stc_2 = 2;   /* Incorrect code output */

   sub(lp2);
}
```

[Incorrect]	[Correct]
MOV.L ER1,@(4:2,SP)	
MOV.L ER0,@SP	
MOV.L SP,ER0	
ADDS.L #4,ER0	
MOV.B #2:8,@(1:2,SP) → MOV.B #2:8,@(5:2,SP)	

[Conditions]

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

    a)   H8SXN, H8SXM, H8SXA, or H8SXX was specified as a CPU type.

    b)   The structreg option was specified.

    c)   A 4-byte or less structure/union was used as a parameter.

    d)   The address of the structure/union was used as a parameter or the structure/union was used as a parameter and the address of the parameter was referenced in the function.

[Solution]

   This problem can be prevented by the following method:

   Assign the parameter to a local variable and use this variable for accesses.

[Applied Product Version]

   Ver.6.0.00 or later

2)   Illegal pointer comparison

    If constants cast to pointers were compared, the result of comparison might be incorrect.

   [Example]

```
int a;
void f(void) {
unsigned long t = 0xffffffff;          /* 0xffffffff was propagated so that ((float *)0 < (float *)0xffffffff);    */
 a = ((float *)0 < (float *)t);      /* a=0 (incorrect)                                                            */
}
```

[Conditions]

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

    a)   There was a comparison expression.

    b)   The both sides of a) were constants cast to the pointer type.

    c)   Either or both of the constants of b) was outside the range of signed long (2147483648 to 4294967295).

[Solution]
This problem can be prevented by the following method:

Store either of the constants in a temporary variable with volatile and use this temporary variable for comparison.

```
int a;
void f(void) {
volatile unsigned long t = 0xffffffff;
a = ((float *)0 < (float *)t);
}
```

[Applied Product Version]
Ver.6.0.00 or later

3) Incorrect setting or reference to a bit field
Setting of a value or reference to a bit field might not be correctly performed.
[Example]

```
typedef struct {
   char c:8;
   char c2:8;
   int i;
}ST;

main()
{
   ST a;
   a.c=10;
   if (a.c==10) {          /* Illegally referred to a.c and the result of comparison was FALSE          */
      func1();
   } else {
      func2();
   }
}
```

[Conditions]
This problem might occur when all of the conditions in a) or b) were fulfilled.
a)
    i.   300HN or 300HA was specified as a CPU type.
    ii.   The optimize option was specified(default).
    iii.   A structure, which was a parameter or local variable, was declared.
    iv.   The structure of iii. was of the [unsigned] char type and had a bit-field member of 8 bits.
    v.   The structure of iii. was allocated to a register and the bit-field member of iv. was allocated to En register.
b)
    i.   300HN, 300HA, 2000N, 2000A, 2600N, or 2600A was specified as a CPU type.
    ii.   The optimize option was specified(default).
    iii.   A structure, which was a parameter or local variable, was declared.
    iv.   The structure of iii. was of the [unsigned] short or [unsigned] int type and had a bit-field member of 16 bits.
    v.   The structure iii. had boundary alignment number 1 (the pack=1 option or #pragma pack 1 was specified).
    vi.   The structure of iii. was allocated to a register and the bit-field member of iv. was allocated to the lower 8 bits of En and RnH
            register.

[Solution]
Take either of the following methods to prevent this problem.
a)   Cancel specification of a bit field and declare it as the scalar type.

```
struct ST {                          struct ST {
   char c:8;                            char c;
   char c2:8          →                 char c2;
   int i;                               int i;
};                                   };
```

b) Do not specify optimization for compilation.

[Applied Product Version]
　　Ver.4.0 or later

4) Error in reference to addresses of structure members by &struct.array[0], etc.
　If the start address was referred to by &struct.array [0] (&struct->array[0]), the address might be incorrect or an internal error might be output.

[Example]
```
typedef struct ST {
   char array[12];
}ST;

ST st;
int b,c;
char *p;

void sub()
{
   p= &st.array[0] + b + c;                /* Outputs a code to add the value of st.array[0] (not an address)      */
}
```

[Conditions]
This problem might occur when all of the conditions in a) or b) were fulfilled.
a)
　　i.　　300HN, 300HA, 2000N, 2000A, 2600N, 2600A was specified as a CPU type.
　　ii.　　A structure member was defined as an array.
　　iii.　　The address value at the start of ii. was used to perform addition or subtraction twice or more.
　　iv.　　The start address was figured out by &struct.array[0] or &struct->array[0].
b)
　　i.　　300HN, 300HA, 2000N, 2000A, 2600N, or 2600A was specified as a CPU type.
　　ii.　　The optimize option was specified(default).
　　iii.　　A variable array was defined.
　　iv.　　The address value at the start of iii. was used to perform addition or subtraction twice or more.
　　v.　　The start address was figured out by &array[0].

[Solution]
This problem can be prevented by the following method:
　Figure out the address of the array by struct.array (or struct->array) or by just an array.

[Applied Product Version]
　　Ver.4.0 or later

5) Access to incorrect addresses by &=0 or |=0xFFFF

If a compound logic operation was performed on an [unsigned] short/int-type variable, a code might be generated to set a value on an incorrect address (correct address + 2).

[Example]

```
typedef struct {
    short    w1;
}*PST;

volatile PST pst = (PST)0xC40000;
short * volatile p;

void sub(void)
{
    pst->w1 &= 0;
    *p &= 0;
    pst->w1 |= 0xffff;
    *p |= 0xffff;
}
```

Incorrect

[Correct]

```
_sub:
    MOV.L     @_pst:32,ER0
    SUB.W     R1,R1
    MOV.W     R1,@(2:16,ER0)  →  MOV.W R1,@ER0
    MOV.L     @_p:32,ER0
    MOV.W     R1,@(2:16,ER0)  →  MOV.W R1,@ER0
    MOV.L     @_pst:32,ER0
    MOV.W     #-1,R1
    MOV.W     R1,@(2:16,ER0)  →  MOV.W R1,@ER0
    MOV.L     @_p:32,ER0
    MOV.W     R1,@(2:16,ER0)  →  MOV.W R1,@ER0
    RTS
```

[Conditions]

This problem might occur when all of the following conditions were fulfilled.
   a)   300, 300HN, 300HA, 2000N, 2000A, 2600N, or 2600A was specified as a CPU type.
   b)   A variable was declared as the [unsigned] short/int type.
   c)   The variable was a pointer declared as volatile.
   d)   Either of the following compound logic operations was described:
         i.   Variable &=0;
         ii. Variable |= 0xffff;

[Solution]

Take either of the following methods to prevent this problem.
   a)   Add volatile to the area pointed to.

```
typedef volatile struct {           /* Add volatile                                              */
    short w1;
}*PST;

volatile PST pst = (PST)0xC40000;
volatile short * volatile p;        /* Add volatile                                              */
```

   b)   Change operations of &= and |= to simple assignments.

```
    pst->w1 = 0;

    *p = 0;
    pst->w1 = 0xffff;
    *p = 0xffff;
```

[Applied Product Version]
Ver.3.0 or later