Date: Jun.07.2004

RENESAS TECHNICAL UPDATE

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

Product Category	User Development Environment		Document No.	TN-CSX-075A/EA	Rev.	1.0
Title	H8S, H8/300 Series C/C++ Compiler Ver.4.0 bug information		Information Category	Usage Limitation		
Applicable Product	PS008CAS4-MWR PS008CAS4-SLR PS008CAS4-H7R PS008CAS5-MWR	Lot No. Ver.4.0 or later	Reference Document	H8S, H8/300 Series C/C++ Compiler Assembler Optimizing Linkage Editor User's Manual REJ10B0058-0100H Rev.1.00		

Attached is the description of the detected bug information in Ver.4.0 or later of the H8S, H8/300 Series C/C++ Compiler.

The bug will affect this package version.

Attached: PS008CAS4-040601E

H8S, H8/300 Series C/C++ Compiler Ver. 4.0 The details of the detected bug information

H8S, H8/300 Series C/C++ Compiler Ver.4.0
The details of the detected bug information

Problems with the H8S, H8/300 series C/C++ compiler ver.4.0 are listed below.

1) 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

2) 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

3) 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

4) The instruction, which assigned 0, is deleted illegally.
If an instruction of assigned 0 was in each branch, it might be deleted illegally.

```
[Example]
    sub.b      R0H, R0H      ;R0H was set as 0
        :
        :
   L55:
        :
    bls        L36
    sub.b      R0H, R0H      ; R0H was still 0 then this instruction can be deleted by optimization
    add.w      R0, R0        ; R0H is possible to be updated to other value.
        :
   L48:
    sub.b      R0H, R0H      ; deleted Illegally
     add.w     R0, R0        ; If R0H isn’t 0, it might be incorrect value
```

[Conditions]

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

a) H8/300 or H8/300L was specified as a CPU type.
b) The optimize=1 option was specified(default).
c) An Instruction of assigned 0 was in each branch.

[Solution]

Take the following method to prevent this problem.

The optimize=0 option should be specified.

[Applied Product Version]

Ver.4.0.04 or later