A Note on Using the C/C++ Compiler Package V.6 for the H8SX, H8S and H8 Families of MCUs

Please take note of the following problem in using the C/C++ compiler package V.6 for the H8SX, H8S, and H8 families:

- On inline expansion of functions accessing local variables within __asm statements (H8C-0044)

1. **Versions Concerned**
   V.6.00 Release 00 through V.6.01 Release 02

2. **Description**
   If a function that accesses a local variable within a __asm statement, a variable in the function that makes a call to the above function may be accessed instead of the local variable in the inline-expanded function.

3. **Conditions**
   This problem occurs if the following conditions are all satisfied:
   
   (1) Either the condition (A) or (B) below is met.

   (A) When any of the versions 6.00 Release 00 through Release 03 used:
   As a CPU option, H8SXN, H8SXM, H8 SXA or H8SXX is selected.

   (B) When any of the versions 6.01 Release 00 through Release 02 used:
   As a CPU option, 2000N, 2000A, 2600N, 2600A, H8SXN, H8SXM, H8 SXA, H8SXX, or AE5 is selected.

   Note that if the -legacy=v4 Compatibility of output object code option is used, 2000N, 2000A, 2600N, and 2600A options are not involved.
In the program exists a function that uses __inline or #pragma inline. Or compile option -speed=inline is selected.

A __asm{ } statement exists in the function that is expanded inline in (2).

In the __asm{ } statement in (3) exists an assembler instruction that accesses a local variable or an argument.

A local variable exists in the function that makes a call to the function in (2).

Example of C Source Program:
----------------------------------------------------
__inline void sub(){ // Condition (2)
    volatile unsigned char tmp;
    __asm { // Condition (3)
        stc.w exr, @(tmp, sp) // Condition (4)
        ldc.b #0x07, exr
    }
    tmp &= 0x07;
}

void main(void){
    volatile unsigned char sample = 0; // Condition (5)
    sub();
}
----------------------------------------------------

Code Generated:
----------------------------------------------------
_main:
subs    #4,sp
mov.b   #0:8,@sp
stc.w   exr,@(0:32,sp)
; Local variable "sample" is accessed in error
; though stc.w exr,@(2:32,sp) must be done
ldc     #7:8,exr
and.b   #7:8,@(2:2,sp)
adds    #4,sp
rts
----------------------------------------------------

4. Workarounds
This problem can be circumvented either of the following ways:

(1) Use neither __inline nor #pragma inline; nor select the -speed=inline compile
Example of Circumvention:

```c
void sub(){                  //__inline not used
    volatile unsigned char tmp;
    __asm {
        stc.w exr, @(tmp, sp)
        ldc.b #0x07, exr
    }

    tmp &= 0x07;
}
```

(2) Access a local variable and argument in a __asm{ } statement via an external variable.

Example of Circumvention:

```c
unsigned char dummy;       // An external variable for copy prepared
__inline void sub(){
    volatile unsigned char tmp;
    dummy = tmp;       // Assigned to external variable
    __asm {
        stc.w exr, @dummy  // Assigned from external variable
        ldc.b #0x07, exr
    }

    tmp &= 0x07;
}
```

5. **Schedule of Fixing the Problem**
   We plan to fix this problem in the release of V.6.01 Release 03.

[Disclaimer]
The past news contents have been based on information at the time of publication. Now changed or invalid information may be included. The URLs in the Tool News also may be subject to change or become invalid without prior notice.

© 2010-2016 Renesas Electronics Corporation. All rights reserved.