HITACHI SEMICONDUCTOR TECHNICAL UPDATE

Classification of Production	Development Environment		No	TN-CSX-048A/E	Rev	1
THEME	SuperH RISC engine C/C++ compiler package Ver.7.1.01 Updates	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

SuperH RISC engine C/C++ compiler package is updated in Ver.7.1.01.
Refer to the attached document, P0700CAS7-030114E, for details.
Inform the customers who have the package version in the table below.

	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
	7.1.00	7.1.00
P0700CAS7-SLR	7.0B	7.0B
	7.0.02	7.0.03
	7.0.03	7.0.04
	7.0.04	7.0.06
	7.1.00	7.1.00
P0700CAS7-H7R	7.0B	7.0B
	7.0.02	7.0.03
	7.0.03	7.0.04
	7.0.04	7.0.06
	7.1.00	7.1.00

Attached : P0700CAS7-030114E
SuperH RISC engine C/C++ Compiler Package
Ver.7.1.01 Updates

SuperH RISC engine C/C++ Compiler Package
Ver. 7.1.01 Updates

The contents of updates in this package are shown below.
The item 1 holds true only for PC version.

1. Hitachi Embedded Workshop (PC version)

1.1 Supporting Drag&Drop operation to add the variable to Watch Window

When you add a variable to Watch window, you drag the variable on Editor and drop it on Watch window.

1.2 Supporting an Out-of-process Server for HEW

Hew server is supported. It is based on COM technology and Out-of-process server.

1.3 Adding and Modifying the Data Generated by the Project Generator

Project generation of the following CPU has been newly added:

SH7705

The I/O definition file (iodefine.h) of the following CPU has been modified:

SH7046, SH7727

2. Compiler

2.1 Generating a zero extension illegally in a loop

[Description]

The following problem is fixed.

A zero extension may be illegally generated when a loop includes a subtraction i = i - v with an unsigned-type variable v and a signed-type variable i.

[Example]

```
    int i=319;
  unsigned char v=97;
  main() {
    int f;
    for(f=0;f<5;f++){
      i = i - v;   /* The zero-extended result of - v is used in the operation. */
    }
  }
```

[Conditions]

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

(1) The optimize = 1 option is specified.
(2) A loop includes a subtraction i = i - v with an unsigned-type variable v and a signed-type variable i.
(3) i is a four-byte variable of signed int or signed long type.
(4) v is an unsigned char/short-type variable whose size is less than four bytes.
(5) The variable v is an invariant in a loop.

2.2 Illegal type conversion

[Description]

The following problem is fixed.

A conversion of a variable to char/short type may not be performed correctly if a conversion to floating-point type is attempted immediately after the conversion to char/short type.

[Example]

[C source program]

```
unsigned short US = 256;
int I;
float F;

main() {
   char c;

   c = US;       /* The short-type variable is converted to char type */
   I = 3 & c;    /* The char-type variable is allocated to a register */
   F = c;        /* Conversion from char type to float type */
}
```

[Assembly source program]

```
_main:
     MOV.L      _US,R6
     MOV.L      _I,R5
     MOV.W      @R6,R0    ; 256 -> R0  Conversion EXTU to char type is not output
     LDS        R0,FPUL   ; Converted to float type with 256 remained
          :
```

[Conditions]

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

(1) The SH2E or SH4 is specified as the CPU.
(2) A variable of the type greater than char/short is converted to char/short type.
(3) The value before the type conversion in (2) exceeds the range of the type after the conversion.
(4) The variable after the type conversion in (2)is converted to floating-point type.
(5) The value after the type conversion in (2) is allocated to a register.

2.3 Illegally moving an instruction beyond the code setting FPSCR

[Description]

The following problem is fixed.

If cpu = sh4 is specified for compilation, the FPU instruction may be illegally moved out of the loop, beyond the code setting FPSCR.

[Example]

[C source program]

```
double dd;
struct tag {
   short aa ;
   long bb ;
   char cc:5 ;
} str ;
main() {
   int i;
   str.bb = 10;
   str.aa = 10;
   for(i=5;i>=0;i--) {
      str.cc =str.aa++  ;
      dd = str.bb ;
   }
}
```

[Assembly source program]

```
_main:
     MOV.L      R14,@-R15
     MOV        #10,R5     ; H'0000000A
     MOV.L      L13+2,R2   ; _dd
     LDS        R5,FPUL
     MOV.L      L13+6,R7   ; _str
     FLOAT      FPUL,DR8  ;Moved out of the loop, beyond LDS R2 FPSCR.
     ADD        #8,R2
     MOV        #8,R1      ; H'00000008
          :
L11:
     MOV.B      @(8,R7),R0
     MOV        R0,R6
          :
     DT         R4
     OR         R1,R2
     LDS        R2,FPSCR
     ;FLOAT     FPUL,DR8          Location before being moved
     ADD        #1,R5
          :
```

[Conditions]

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

(1) The optimize = 1 option is specified.
(2) The SH4 is specified as the CPU.
(3) The fpu option is not specified.
(4) The loop includes a double-type operation.
(5) The operation in (4) is an invariant in the loop.

2.4 Illegal allocation of registers in a loop

[Description]

The following problem is fixed.

The content of registers that have been allocated in the innermost loop may be destroyed in that loop.

[Example]

The value in the register is destroyed because the same register is allocated to several variables.

[C source program]

```
        :
  for(i1 = 0; i1 < max1 ; i1++) {
     a = b;
     for(i2 = 0; i2 < max2 ; i2++) {     /* (1) */
        c = x + b;                       /* (2) */
     }                                   /* (3) */
     ans = a + c;
  }
        :
```

[Assembly source program]

```
     MOV.L  @(R0,R15),R5     ; R5 is allocated to a
           :
     MOV.L  R5,@(R0,R15)     ; Saves a
           :                 ; Loop entry (1)
     MOV.L  @(R0,R15),R5     ; Allocates R5 to c
           :
  L1:
           :                 ; Loop body (2) (No reference to a)
     MOV    Rn,R5            ; Stores the result of c = x + b to R5
           :
     BF  L1
           :           ; Loop exit (3)
     MOV.L  @(R0,R15),R5     ; Overwrites R5, not storing c
           :
     MOV.L  R5,Rn            ; Loading the value of destroyed c
```

[Conditions]

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

(1) The optimize = 1 option is specified.

(2) The innermost loop includes a variable for which a register is allocated (c in the example).

(3) There is an expression that includes the variable mentioned in (2) outside of the loop.
The register (R5 in the example) allocated to the variable in (2) is also allocated to another variable (a in the example) in that expression.

2.5 Illegal deletion of FPSCR loading

[Description]

The following problem is fixed.

When two codes to switch double/float are output, the second loading of FPSCR may be illegally deleted.

[Example]

[C source program]

```
double d0;
float f0, f1, f2;

main {
   int i;
   for(i = 0; i < 100; i++){
      d0++;
      f1 = f1 + f0;
   }
}
```

[Assembly source program]

```
_main:                          ; function: main
   MOV.L      L13+2,R1      ; _d0
   MOVA       L13+6,R0
         :
L11:
   STS        FPSCR,R2
   DT         R6
   OR         R5,R2
   LDS        R2,FPSCR
   FADD       DR4,DR6       ; <-- STS FPSCR,R2 must follow this line.
   AND        R4,R2
   LDS        R2,FPSCR
   BF/S       L11
   FADD       FR9,FR8
   ADD        #8,R1
         :
```

[Conditions]

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

(1) The optimize = 1 option is specified.

(2) The SH4 is specified as the CPU.

(3) The fpu option is not specified.

(4) double-type and float-type operations are included.

2.6 Illegal sign extension for parameters to call functions
[Description]
The following problem is fixed.
Required sign-extension may not be performed on a parameter for the runtime routine that does not return values.

[Example]
The value of c passed to the parameter (unsigned long int) R0 for the runtime routine (_itod_a) is not sign-extended.
[C source program]

```
double D;
int I;
unsigned short US = 256;
func2(){
   char c;
   c = US;
   I = 3 & c;
   D = c;
}
```

[Assembly source program]

```
_func2:
        STS.L     PR,@-R15
        ADD       #-12,R15
          MOV.L     L20,R6    ; _US
        MOV.L     L20+4,R5  ; _I
        MOV.W     @R6,R2
        MOV.L     L20+16,R6 ; _D
        MOV       R2,R0
        AND       #3,R0
        MOV.L     R6,@R15
        MOV.L     L20+20,R6 ; __itod_a
        MOV.L     R0,@R5    ;<-- EXTS.B  R2,R2 must follow this line.
        JSR       @R6
        MOV       R2,R0
```

[Conditions]
This problem may occur when the following condition is satisfied.
(1) Either of the runtime routines listed below is used. The parameter of the function is four bytes. char/short-type variables are passed to the parameter.
_itod_a, _utod_a

2.7 Illegal settings for or reference to structure members
[Description]
The following problem is fixed.
When a member in an element that has two-dimensional or above structure arrays is referred to via a pointer, settings for or reference to the member may not be made correctly.

[Example]

```
typedef struct {
   int aaa;
} ST;
void main()
{
   ((ST (*)[2])0x10000)[0]->aaa = 100;    /* 0x10000[0][0].aaa is set incorrectly */
}
```

[Conditions]
This problem may occur when all of the following conditions are satisfied.
(1) There are two-dimensional or above structure arrays.
(2) A member in an element with the structure arrays mentioned in (2) is set or referred to via a pointer.

2.8 Illegal output of errors when latin1 is specified
We have modified the failure that an error message was output for a partly correct latin1 code in compilation with latin1 specified.

2.9 Illegal display of the stack size
[Description]
We have modified the failure that information of the stack size for interrupt functions (size of a stack frame in a listing file/assembly source program) did not include the size of instructions used to store PC and SR values, which are implicitly used by the CPU.

[Conditions]
This problem may occur when all of the following conditions are satisfied.
(1) The SH1, SH2, or SH2E is specified as the CPU.
(2) There is an interrupt function specified by #pragma interrupt.

2.10 Accesses to volatile variables
We have improved the following limitations set for accesses to volatile variables.

(a) Continuous accesses to volatile variables
[Source program]

```
void main(void) {
   *((volatile unsigned char*)0xfffff000);      /* (1) */
   *((volatile unsigned short*)0xfffff004);     /* (2) */
   *((volatile unsigned short*)0xfffff004) = 1;  /* (3) */
   /*  Reference in order from (1) to (3) is guaranteed */
}
```

(b) The qualifier volatile not available for a constant address
[Source program]

```
struct st_tmu2 {
   unsigned int TCOR;
   unsigned int TCNT;
};

#define TMU2 (*(volatile struct st_tmu2 *)0xFFD80020)

void time_wait (unsigned long time) {
   volatile unsigned long tcnt;
   unsigned long tcnt_copy, time_cnt;

   time_cnt = time;
   tcnt_copy = tcnt = TMU2.TCNT;
   while ((tcnt_copy - tcnt) < time_cnt){
      tcnt = TMU2.TCNT;
      /* It is guaranteed to perform every assignment in a loop*/
   }
}
```

2.11 Modification in the attribute of a section when _sectop or _secend is specified
[Description]
We have modified the settings so that the attribute becomes CODE with _sectop or _secend specified, when the head of the section name is P. As a result, L1323 is not output at linkage.

3. Optimizing Linkage Editor

3.1 Internal errors fixed
Fixed the following internal errors:
(1) Internal error(1703) that occurs when optimization of integrating common codes is specified
(2) Internal error(1704) that occurs when optimization of integrating constants or string literals is specified
(3) Internal error(8899) that occurs when optimization of branch instructions is specified

3.2 Illegal operation with form={binary | stype | hexadecimal} option specified
Fixed is the following problem a error message does not appear even though no output file is created in generating the binary/stype/hexadecimal format file when the directory specified in the output option does not exist.

3.3 Incorrect object code by optimization of deleting unused symbols
Fixed is the following problem the elements in the two arrays may be incorrect due to optimization when all of the following conditions satisfied:
(1) An array(say A_arr[]) to be accessed exists.
(2) An array(say B_arr[]) or variable to be deleted by optimization exists.
(3) A_arr and B_arr are in different sections. The sections for the two arrays are assumed to be A and B, respectively.
(4) The start option allocates the sections A and B in a way that the addresses of these sections overlap .
(5) The optimization of deleting unused symbols is enabled.

3.4 Incorrect error at linkage of C++ object codes
Fixed is the problem that an error P3300 (F) may occur incorrectly at linkage of C++ object codes created using templates.

3.5 Illegal lack of some data by conversion of object formats
Fixed is the problem that some data incorretly disapper when all of the following conditionsare satisfied:
(1) The source program is written in C++.
(2) The optimization of deleting unused symbols is enabled.
(3) The object file of the absolute file format is coverted (ELF->sysrof) by the converter(helfcnv).