date: 2003/08/11

RENESAS TECHNICAL UPDATE

Classification of Production	Development Environment		No	TN-CSX-052A/E	Rev	1
THEME	SuperH RISC engine C/C++ compiler package Ver.7.1.03 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-702-372A Rev.2.0	term of validity	Eternity

SuperH RISC engine C/C++ compiler package is updated in Ver.7.1.03.

Refer to the attached document, P0700CAS7-030801E, 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
	7.1.01 7.1.02	7.1.01
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
	7.1.01 7.1.02	7.1.01
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
	7.1.01 7.1.02	7.1.01

Attached: P0700CAS7-030801E

SuperH RISC engine C/C++ Compiler Package

Ver.7.1.03 Updates

SuperH RISC engine C/C++ Compiler Package
Ver. 7.1.03 Updates

The contents of updates in this package are shown below.
The item 1 and item 2 holds true only for PC version.

1. High-performance Embedded Workshop (PC version)

1.1 Navigation feature improvement
C++ navigation feature is available.

1.2 Supporting smart editor
When you enter ".", "->" or "::", smart editor shows the member variable and function list.
You can also select member in the list. When you enter "(" after function name, parameter list will be shown.

1.3 Supporting Network database
When you use HEW via network, you can specify the access right.

1.4 Makefile import feature
When you create a new project workspace, Hew can import the source files path from makefile.
But toolchain's options are not supported.

1.5 Synchronized debugging feature
Hew can connect two or more targets at the same time and you can debug them as synchronized.

1.6 Workspace improvement
One workspace can have different toolchains project.
You can choose the version if HEW has different version toolchains.

1.7 Introducing Manual Navigator
Manual Navigator can show the list of manuals.

1.8 Supporting file comparison feature
File comparison view is available. It can shows difference between two files.

1.9 Supporting TCL and TK window
Script file which is written by TCL/TK script language can be executed.

1.10 Typelibrary improvement
Workspace operations, project operations and debugger operations interfaces are available.

1.11 Typelibrary improvement
Workspace operations, project operations and debugger operations interfaces are available.

1.12 Adding and Modifying the Data Generated by the Project Generator
Project generation of the following CPU has been newly added:
SH7294, SH7616
The I/O definition file (iodefine.h) of the following CPU has been modified:
SH7011, SH7014, SH7016, SH7017, SH7046, SH7047, SH7049,
SH7050, SH7050F, SH7051, SH7051F, SH7052F, SH7053F, SH7054F,
SH7065, SH7148, SH7615, SH7622

2. SuperH RISC engine simulator/debugger (PC version)

2.1 Improvement of Memory window Look&Feel
Redesigning the memory window and Look&Feel is improved.

2.2 Improving the command line window
When you enter some part of command, it shows list of command which depends upon your input.
And more, command parameter hint is available.

2.3 Coverage enhancement
It shows coverage statistics and measures file basis coverage.

2.4 Image and Wave Window improvement
Image and Waveform window can be updated interval during user program executing.

2.5 Trigger enhancement
You can set up to 256 triggers.

2.6 Register window customization
You can choose which registers are visible on register window.

2.7 Watch window improvement
Watch window has four panes.

2.8 Supporting downloading with specific access size
When HEW downloads the load module to memory with access size which you can specify.

2.9 Supporting Timer
SH-1,SH-2,SH-3,SH-4,SH2-DSP and SH3-DSP have 1ch internal timer module.

2.10 Introducing Event window
Event window is replaced Break points window.

2.11 Improve memory map setting
Memory map and Memory resource are merged one dialog.

3. Compiler

3.1 Illegal deletion of EXTU instruction
[Description]
The following problem is fixed.
When an unsigned variable is used more than once in a loop , the EXTU instruction may be deleted illegally.

[Example]
[C source program]

```
extern unsigned char X;
int sub(int a, int b, int n) {
    int i, sum=0;
    for (i = 0; i < n; i++) {
        if (X == (unsigned char)0xff) { // X is used.
            sum += a;
        }
        if (X == (unsigned char)0xf0) { // X is used.
            sum += b;
        }
        X = X + 1;                      // X is defined.
```

```
    }
    return (sum);
  }
```

[Assembly source program]

```
  _sub:
        MOV.L   R12,@-R15
        MOV.L   R13,@-R15
        MOV.L   R14,@-R15
        MOV     R5,R13
        MOV     #0,R5
        MOV.L   L19,R1          ; _X
        MOV     R6,R7
        MOV     R4,R12
        MOV     R5,R6
        MOV     #-1,R4
        MOV.B   @R1,R2
                                ; <- delete EXTU.B R2,R2
        MOV     #-16,R14
        EXTU.B  R4,R4
        BRA     L11
        EXTU.B  R14,R14
  L12:
        CMP/EQ  R4,R2           ; A result of comparison may be illegal. (X > 127)
        BF      L14
        ADD     R12,R5
  L14:
        CMP/EQ  R14,R2          ; A result of comparison may be illegal. (X > 127)
        BF      L16
        ADD     R13,R5
  L16:
        ADD     #1,R2
        EXTU.B  R2,R2
        ADD     #1,R6
  L11:
        CMP/GE  R7,R6
        BF      L12
        MOV.B   R2,@R1
        MOV     R5,R0
        MOV.L   @R15+,R14
        MOV.L   @R15+,R13
        RTS
        MOV.L   @R15+,R12
```

[Conditions]

This problem may occur when all of the following conditions (1) to (4) or (5) to (8) are satisfied.

(1) The optimize=1 option is specified.
(2) An unsigned char/unsigned short variable is used.
(3) This variable is used more than once in a loop, and defined after using.
(4) This variable is loaded from the memory at first.
(5) The optimize=1 option is specified.
(6) An unsigned char/unsigned short local variable is used.
(7) This variable is used as a source variable of right shift operation in a loop.
(8) The shift count is more than one and SHLR2, SHLR8 or SHLR16 is used.

3.2 Illegal deletion of floating-point constant load

[Description]
The following problem is fixed.
When the same floating-point constants are used more than once in a loop and out of loop, the load instruction of the constants may be deleted illegally.

[Conditions]
This problem may occur when all of the following conditions are satisfied.
(1) The optimize=1 option is specified.
(2) The same floating-point constants are used more than once in a loop.
(3) This value is used outside the loop.

3.3 Illegal array access
[Description]
The following problem is fixed.
When an array index is C-exp, where C is a constant value and exp is unsigned char/short type expression , the element of the array is accessed illegally.

[Example]
[C source program]

```
unsigned char dd[2];
void func(unsigned char a, unsigned char b) {
   dd[15-a]=b1;
}
```

[Assembly source program]

```
_func:
   MOV.L    L11,R2          ; _dd
   NEG      R4,R6           ; R6 <- -a
   EXTU.B   R6,R0           ; -a is zero-extended
   ADD      #15,R2
   RTS
   MOV.B    R5,@(R0,R2)     ; Illegal address is accessed
```

[Conditions]
This problem may occur when all of the following conditions are satisfied.
(1) The optimize = 1 option is specified.
(2) This array element is accessed by "constant value - expression"
(dd[15-a] in the example).

3.4 Illegal copy by memmove
[Description]
The following problem is fixed.
When memmove of the standard library copies some characters from an address to the greater address, more characters than specified may be copied.

[Conditions]
This problem occurs when all of the following conditions are satisfied.
(1) The move size is more than 30-byte.
(2) The destination address is greater than the source address.
(3) (The source address + the movement size) is inside the destination area.
(4) The source address or the destination address is not a power of 4.

3.5 Illegal array access
[Description]
The following problem is fixed.
When all of the following conditions are satisfied, the unconditional branch may be deleted illegally.
- The last of a function is conditional statement.

- Conditions are nested in the statement.
- The last condition finishes with a function call and a return statement, and the previous condition finishes with a function call.

[Example]

[C source program]

```
void sub(int parm) {
   if (parm == 0) {
      ;
   } else if (parm == 1) {
      ;
   } else if (parm == 2) {
      ;
   } else if (parm == 3) {
      ;
   } else if (parm == 4) {
      ;
   } else if (parm == 5) {
      func1();    /* <A> */
   } else {
      func2();    /* <B> */
      return;     /* <B> */
   }
   return;
}
```

[Assembly source program]

```
_sub:
        STS.L       PR,@-R15
        TST         R4,R4
        BT          L11
        MOV         R4,R0
        CMP/EQ      #1,R0
        BT          L11
        CMP/EQ      #2,R0
        BT          L11
        CMP/EQ      #3,R0
        BT          L11
        CMP/EQ      #4,R0
        BT          L11
        CMP/EQ      #5,R0
        BF          L18
        MOV.L       L20+2,R2        ; _func1
        JSR         @R2
        NOP
L11:
                                    ; A branch to L19 is deleted
L18:
        MOV.L       L20+6,R2        ; _func2
        JMP         @R2             ; This function is always called
        LDS.L       @R15+,PR
L19:
        LDS.L       @R15+,PR
        RTS
        NOP
```

[Conditions]

This problem may occur when all of the following conditions are satisfied.
(1) The optimize=1 option is specified.
(2) The last of a function is conditional statement and the conditions are nested.
(3) The last condition finishes with a function call and a return statement
(<B> in the example).
(4) The condition previous to (3) finishes with a function call (<A> in the example).

3.6 Illegal cast from unsigned integer to float
[Description]
The following problem is fixed.
When the unsigned integer type variable is cast to the float type, the cast may be deleted illegally.

[Example]
[C source program]

```
unsigned short us1;
volatile unsigned short us0;
volatile float f0;
float *p;
void func() {
   f0 = *p = ((float)us0, (float)us1);
}
```

[Assembly source program]

```
_func:
      MOV.L     L29+50,R2       ; _us0
      MOV.L     L29+54,R5       ; _p
      MOV.W      @R2,R6
      MOV.L     L29+58,R6       ; _us1
      MOV.W      @R6,R2
      EXTU.W     R2,R6
      MOV.L     @R5,R2
      MOV.L     R6,@R2           ; store to *p without cast to float type
      MOV.L     @R5,R2
      MOV.L     @R2,R6
      MOV.L     L29+10,R2       ; _f0
      RTS
      MOV.L     R6,@R2           ; store to f0 without cast to float type
```

[Conditions]
This problem may occur when all of the following conditions are satisfied.
(1) The unsigned integer variable is cast to float type.
(2) The unsigned integer variable is cast to double type and either double=float or
fpu=double option is specified, or is cast to long double type and fpu=single
option is specified.

3.7 Illegal movement of stack pointer with ld_ext() or st_ext()
[Description]
The following problem is fixed.
When an ld_ext() or st_ext() intrinsic function is used and a local array is specified as a parameter,
the stack pointer may be moved illegally.

[Example]
[C source program]

```
#include <machine.h>

void main() {
   float table[4][4], data1[4][4], data2[4][4];
```

```
        :
    ld_ext(table) ;
    mtrx4mul(data1,data2) ;
        :
}
```

[Assembly source program]

```
        :
    FRCHG
    FMOV.S    @R15+,FR0  ; R15 is moved. When an interrupt occurs, upper area of
                         ; stack is destroyed
    FMOV.S    @R15+,FR1  ;                                    |
    FMOV.S    @R15+,FR2  ;                                    |
    FMOV.S    @R15+,FR3  ;                                    |
    FMOV.S    @R15+,FR4  ;                                    |
    FMOV.S    @R15+,FR5  ;                                    |
    FMOV.S    @R15+,FR6  ;                                    |
    FMOV.S    @R15+,FR7  ;                                    |
    FMOV.S    @R15+,FR8  ;                                    |
    FMOV.S    @R15+,FR9  ;                                    |
    FMOV.S    @R15+,FR10 ;                                    |
    FMOV.S    @R15+,FR11 ;                                    |
    FMOV.S    @R15+,FR12 ;                                    |
    FMOV.S    @R15+,FR13 ;                                    |
    FMOV.S    @R15+,FR14 ;                                    |
    FMOV.S    @R15+,FR15 ;                                    |
    FRCHG                ;                                    |
    ADD       #-64,R15   ;                                    V
```

[Conditions]

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

(1) The cpu=sh4 option is specified and the ld_ext() or st_ext() intrinsic function is used.

(2) A local array is specified as the parameter..

3.8 Illegal output of data

[Description]

The following problem is fixed.

When there are many variables with initial value of "symbol address + offset" in the source program, the internal error may occur or an illegal object may be generated.

[Conditions]

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

(1) The code=asmcode option is specified. (This option is valid by default.)

(2) The listfile option is not specified, or both the listfile option and the show=noobject option are specified.

(3) A variable with an initial value exists.

(4) The initial value is described in the form of "symbol address + offset" or an address of a struct member which is not allocated at the top of the struct.

(5) In all of such variables, the variables and offsets of initial value satisfy with the following condition:

(number of such variables + sum of number of decimal-digits in offset) >= 33,000

<Example>

```
extern char g;
#define DATA1A   (&g+2147483647)
#define DATA10A  DATA1A,  DATA1A,  DATA1A,  DATA1A,  DATA1A,¥
                 DATA1A,  DATA1A,  DATA1A,  DATA1A,  DATA1A
#define DATA100A DATA10A, DATA10A, DATA10A, DATA10A, DATA10A,¥
```

```
                    DATA10A, DATA10A, DATA10A, DATA10A, DATA10A
/* In this case,                                                  */
/* number of variables + the sum of decimal-digit number of offset */
/* = (3001+10*3001) = 33011 > 33000                               */
char *a1[1000] = {
   DATA100A, DATA100A, DATA100A, DATA100A, DATA100A,
   DATA100A, DATA100A, DATA100A, DATA100A, DATA100A
};
char *a2[1000] = {
   DATA100A, DATA100A, DATA100A, DATA100A, DATA100A,
   DATA100A, DATA100A, DATA100A, DATA100A, DATA100A
};
char *a3[1000] = {
   DATA100A, DATA100A, DATA100A, DATA100A, DATA100A,
   DATA100A, DATA100A, DATA100A, DATA100A, DATA100A
};
char *a = DATA1A;
```