RENESAS Tool News

RENESAS TOOL NEWS on April 16, 2007: 070416/tn4

A Note on Using the C/C++ Compiler Package for the M32R MCU Family

Please take note of the following problem in using the C/C++ compiler package --M3T-CC32R-- for the M32R MCU family:

- With compiling programs in C/C++ containing inline functions

---

1. Products and Versions Concerned

The C/C++ compiler package--M3T-CC32R--for the M32R MCU family
V.4.00 Release 1 through V.5.00 Release 00

2. Problems

Either of the following symptoms may arise:
(1) For some jump instructions, the code of the incorrect destination is generated.
(2) The compiler terminates after dispatching the following error message (In some cases, the compiler may dispatch two or more this error messages.):
a132R: "File_name", line xxx: error: "Lxxx": symbol redeclared

Here, File_name denotes the name of the working file created by the compiler, and this filename is deleted after compilation is complete. And xxx denotes a numeric value.

3. Conditions

Symptom 2-(1) may arise if all the following conditions are satisfied, and Symptom 2-(2) arise if those except (4) are satisfied:
(1) The -noinline option is not selected at compilation.
(2) In the program exist functions containing any of the following control statements and operators:
Control statements: if, for, while, do, switch, goto, and return
Conditional Operator: ? :

Comparison (equality and inequality) operators:
==, !=, <, >, <=, and >=
Logical operators: && and ||

(3) At least one function in (2) is declared to be inline.

(4) Any one option is selected out of -O2, -O3, -O6, -O7, and -O; or -Ospace or -Otime only is selected with none being selected out of -O, -O0, -O1, -O2, -O3, -O4, -O5, -O6, and -O7.

Example:

The example shown below is only a significant portion of the source program where one of the symptoms arose, and you are unable to reproduce the symptom using these statements. If you want to know whether your program is involved in the symptoms, see section 4.

Example of source file: sample.c

```
-----------------------------------------------------------------
struct stag1 { int a, b, flag; };
extern void func03(int data, int mode);
inline void func02(int data, int flag)    /* Condition (3) */
{
   switch (flag) {                  /* Conditions (2) and (3) */
      case 0:
         func03(data, 0);
         break;
      case 1:
         func03(data, 1);
         break;
      default:
         func03(data, -1);
   }
}
void func01(struct stag1 *ptr)
{
   if (ptr->a <= 10) {               /* Condition (2) */
      func02(ptr->a, ptr->flag);
   }
   if (ptr->b <= 100) {               /* Condition (2) */
      func02(ptr->b, ptr->flag);
   }
}
-----------------------------------------------------------------
```

Examples of command-line inputs (% denotes a prompt):

```
------------------------------------------------------------------
% cc32R -S sample.c              Condition (1)
% cc32R -S -Otime sample.c       Conditions (1) and (4)
------------------------------------------------------------------
```

4. How to Know Your Program Involved or Not

If your source program satisfies Conditions (1) through (4) above, you are able to know whether it is involved in Symptom 2-(1) or not by changing the command-line option as follows:

(a) If any one option is selected out of -O2, -O3, -O6, -O7, and -O, replace it as follows:
-O2 with -O0
-O3 with -O1
-O6 with -O4
-O7 with -O5
-O with -O5

(b) If -Ospace or -Otime only is selected with none being selected out of -O, -O0, -O1, -O2, -O3, -O4, -O5, -O6, and -O7, replace it as follows:
-Ospace with -Ospace -O5
-Ospace with -Otime -O5

If Symptom 2-(2) does not arise when re-compiling or re-building is performed after the previous option has been replaced, it tells you that Symptom 2-(1) will not arise also.

Notice:
The above methods only tell you whether Symptom 2-(1) arises or not and never tell that all the generated code functions properly.
So even if these methods tell you Symptom 2-(1) does not arise, you should fully check the performance of your program.

5. Workarounds

The above symptoms can be avoided in any of the following ways. Note, however, that if you use the way (1) or (2) below, make sure that they have been avoided using the methods described in Section 4.

(1) Split a source file into more than one.
The symptoms may be avoided by decreasing the number of branches in a source file.

(2) Replace some inline functions correspond to the conditions with #define macros.

(3) Select the -noinline option at compilation.

Though the selection of -noinline enables you to avoid the symptoms, the code generation rate may decrease since the functions declared to be inline will not be expanded inline.

6. Schedule of Fixing the Problem

We plan to fix this problem in the next release of the product.
