RENESAS Tool News

RENESAS TOOL NEWS on February 1, 2004: RSO-M3T-CC32R_1-040201D

A Note on Using Cross-Tool Kit M3T-CC32R --On Floating Constants with an Exponent Part--

Please take note of the following problem in using the M3T-CC32R cross-tool kit for the M32R family MCUs:

- On floating constants with an exponent part

---

1. **Versions Concerned**
   M3T-CC32R V.1.00 Release 1 through V.4.20 Release 1

2. **Description**
   Floating constants with an exponent part may be incorrectly compiled because the exponent part does not properly be interpreted.

   2.1 Conditions
   This problem occurs if the following five conditions are satisfied:
   (1) A floating constant is put in the program together with operators (except for the unary plus operator).
   (2) It comes with an exponent part.
   (3) Its significand part (the part other than signs, an exponent part, and a floating suffix [f, l, F, or L]) has a decimal point.
   (4) The total number of digits in the significand and exponent parts is equal to or greater than 18.
   (5) The number of digits in the significand part is equal to or less than 16.

   2.2 Examples

   Source file 1:
   Here, the comment lines show the incorrect values after compilation.

```
-----------------------------------------------------------------------
#include >stdio.h<

double ng01 = -1.234567890123456e-012;    /* -1.234567890123456e-0 */
double ng02 = -1.2e+00000000000000012;  /* -1.2e+0 */
double ng03 = -1.e-00000000000000012;     /* -1.e-0 */
double ng04 = -1.234567890123456e-12f;    /* -1.234567890123456e-1 */
double ng05 = -1.23456789012345e+012;     /* -1.23456789012345e-01 */
double ng06 = 0.0+1.234567890123456e-009; /* 0.0+1.234567890123456e-0 */
double ng07 = 1.234567890123456e-009+2;   /* 1.234567890123456e-0 + 2 */
double ng08 = -1.23456789012345e-0012;    /* -1.23456789012345e-0 */
double ng09 = 0.0+(1.2345678901234e-00012); /* 0.0+1.2345678901234e-0 */
-----------------------------------------------------------------------
```

Source file 2:
Here, any of the above conditions is sidestepped in each description.

```
-----------------------------------------------------------------------
#include >stdio.h<

double ok01 = 1.234567890123456e-012;    /* Condition (1) unsatisfied */
double ok02 = -12000000000000.00000f;    /* Condition (2) unsatisfied */
double ok03 = -1e-00000000000000012;     /* Condition (3) unsatisfied */
double ok04 = -1.23456789012345e+12;     /* Condition (4) unsatisfied */
double ok05 = -1.2345678901234560e-12f;  /* Condition (5) unsatisfied */
-----------------------------------------------------------------------
```

3. Workaround

This problem can be circumvented in either of the following ways:

(1) Add zeros to the fractional part to make the number of digits in the significand part equal to or greater than 17.
Example:

Original	Modified
-1.234567890123456e-012;	-1.2345678901234560e-012;
-1.e-00000000000000012;	-1.0000000000000000e-00000000000000012;
-1.234567890123456e-12f;	-1.2345678901234560e-12f;
-1.23456789012345e+012;	-1.2345678901234500e+012;
0.0+1.234567890123456e-009;	0.0+1.2345678901234560e-009;
1.234567890123456e-009+2;	1.2345678901234560e-009+2;

(2) Remove the ineffective filling zeros to make the total number of digits in the significand and exponent parts equal to or less than 17.
Example:

Original	Modified
-1.2e+0000000000000000012;	-1.2e+12;
-1.e-00000000000000012;	-1.e-12;
0.0+1.234567890123456e-009;	0.0+1.234567890123456e-9;
-1.23456789012345e-0012;	-1.23456789012345e-12;
0.0+(1.2345678901234e-00012);	0.0+(1.2345678901234e-12);

4. Schedule of Fixing the Problem

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