RENESAS Tool News

RENESAS TOOL NEWS on March 1, 2016: 160301/tn3

Notes on Using the CS+ CX Compiler

When using the CS+ CX Compiler (for the V850E2M and V850E2S cores), take note of the problems described in this note regarding the following points.
1. Initializing Integer-Type Array Having 1024 or More Bytes as an AutomaticVariable (No.17)
2. The pow Function Returning Incorrect Values (No.18)
Note: The number which follows the description of the precautionary note is an identifying number for the precaution.

---

1. Initializing Integer-Type Array Having 1024 or More Bytes as an Automatic Variable (No.17)

1.1 Applicable products

CX V1.00 to V1.31

1.2 Description

When initializing an integer-type array of 1024 or more bytes as an automatic variable, if there are fewer initializers than elements, parts which should be implicitly initialized to 0 become undefined.

1.3 Conditions

This problem may arise if the following conditions are all met:
(1) An integer-type array of 1024 or more bytes is declared as an automatic variable.
(2) Explicit initialization is with fewer initializers than the number of elements in the array.
(3) The initializer of (2) is not a string literal.

Example:

```
------------------------------------------------------------
void func( void )
{
   char array_ng[1024] = {0};
}
------------------------------------------------------------
```

1.4 Workaround

To avoid this problem, do either of the following:

(1) Use a loop to assign 0 to elements which have no corresponding initializer.

(2) Use the memset function to assign 0 to elements which have no corresponding initializer.

Example of workaround (1):

```
----------------------------------------------------------------
void func( void )
{
    char array_ng[1024] = {0};
    int i;
    for (i = 1; i < 1024; ++i) {
        array_ng[i] = 0;
    }
}
----------------------------------------------------------------
```

1.5 Schedule for fixing the problem

This problem will be fixed in a later version of the product (the date of the next release has not yet been decided).

2. The pow Function Returning Incorrect Values (No.18)

2.1 Applicable products

CX V1.00 to V1.31

2.2 Description

When the pow function is used to calculate a power, if the first argument is negative, and the second argument is an odd-numbered integer from 2147483649 to 4294967295, or from -4294967295 to -2147483649, the sign bit of the return value incorrectly becomes positive.

2.3 Conditions

This problem arises if the following conditions are all met:

(1) The -C option is used to select a target device which does not have an FPU, or the -C option is used to select a target device which has an FPU and the -Xfloat=soft option is specified for that device.

(2) The first argument of the pow function is negative number.

(3) The second argument of the pow function is an odd number within either of the following ranges.

(a) From 2147483649 to 4294967295

(b) From -4294967295 to -2147483649

Example for a target device with no FPU:

```
-----------------------------------------------------------------
#include

void func(void) {
  double result;
  double x = -1.00000001;  /* condition(2) */
  double y = 4294967295ul; /* condition(3)  */

  result = pow(x, y);
}
-----------------------------------------------------------------
```

The value of the result becomes +4.49579e+018 instead of -4.49579e+018.

2.4 Workaround

There is currently no way to prevent this problem.

2.5 Schedule for fixing the problem

This problem will be fixed in a later version of the product (the date of the next release has not yet been decided).
