RENESAS Tool News

RENESAS TOOL NEWS on March 1, 2006: RSO-M3T-CC32R-060301D

A Note on Using the C/C++ Compiler Package--M3T-CC32R-- for the M32R Family of MCUs

Please take note of the following problem in using the C/C++ Compiler Package--M3T-CC32R-- for the M32R family of MCUs:

- On constant expressions of integers containing a shift operator whose right operand is an unsigned integer type

---

1. **Versions Concerned**
   CC32R V.1.00 Release 1--V.5.00 Release 00

2. **Description**
   If a constant expression of integers contains a left- or right-shift operation whose right operand (a shift count) is an unsigned integer type, any of the following problems arises according to what the constant expression specifies. Note, however, that if you compile programs in C++ language mode using the CC32R V.5.00 Release 00, those problems will not arise:

   (1) When a constant expression specifies the size of a member of a bit field (as shown in Example 1 in Section 3), an incorrect size is assumed to be the correct one, which provides the following warning message:

   > Warning message:
   > "xxxx", line XX: warning: shift count greater than number of bits

   Here, xxxx denotes the name of a source file, and XX does the number of the line where the constant expression is placed.

(2) When a constant expression specifies the size of an array (as shown in Example 2 in Section 3), an error arises which provides the following error message as well as the warning message in (1) above:

> error: array: subscript must be positive, non-zero, integral value

(3) When a constant expression specifies the value of an enumeration constant (as shown in Example 3 in Section 3), an error arises which provides the following error message as well as the warning message in (1) above:

> error: enumeration-constant out of range

(4) When a constant expression is used as the integer constant expression of a case label (as shown in Example 4 in Section 3), an error arises which provides the following error message as well as the warning message in (1) above:

> error: unable to evaluate case label (out of range?)

(5) When a constant expression is used in the other cases (for example, as shown in Example 5 in Section 3), compilation is properly performed but the warning message in (1) above appears.

3. Conditions

The above problems occur if the following conditions are both satisfied:

(1) In a program exists an constant expression of integers that contains a left- or right-shift operation.

(2) The right operand (a shift count) of the shift operation in (1) is an unsigned integer type.

Examples:

```
-------------------------------------------------------------
struct SBtag {
  short b01:1<<(unsigned)2;      /* Example 1 */
  short b02:1;
```

```
};

int array[8>>1u];               /* Example 2 */

enum Etag {
  A,
  B = 2<<(unsigned)2            /* Example 3 */
};

int func3(int key)
{
  switch (key) {
    case ((3*8)>>(7UL-5)):      /* Example 4 */
      return 1;
  }
  return 0;
}

int data = 0x8000>>12u;         /* Example 5 */
```
---

4. **Workarounds**

The above problems can be circumvented either of the following ways:

(1) Cast the right operand of a shift operation to the signed integer type; for example, convert its type using the type cast operator (signed) as shown in Example below.

Example:

```
-------------------------------------------------------
-------------
struct SBtag {
  short b01:1<<(signed)(unsigned)2;/* Right operand
cast by (signed) */
  short b02:1;
};

int array[8>>(signed)1u];       /* Right operand cast
by (signed) */

enum Etag {
  A,
```

```
  B = 2<<(signed)(unsigned)2      /* Right operand cast
by (signed) */
};

int func3(int key)
{
  switch (key) {
    case ((3*8)>>(signed)(7UL-5)): /* Right operand cast
by (signed) */
      return 1;
  }
  return 0;
}

int data = 0x8000>>(signed)12u;     /* Right operand
cast by (signed) */
-------------------------------------------------------
-------------
```

(2) In the M3T-CC32R V.5.00 Release 00, compile the program in C++ language mode; that is, place extern "C" immediately before the original C source program and enclose the program in braces { } as shown below (this change must be made to file by file), and then compile it using the -lang=cpp option.

```
--------------------
 extern "C" {
  . . . . . .

  (Original C source program)

  . . . . . .

 }
--------------------
```

5. Schedule of Fixing the Problem

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

---

included. The URLs in the Tool News also may be subject to change or become invalid without prior notice.