# RENESAS Tool News

**MAEC TOOL NEWS: MAECT-M3T-CC32R-030316D**

## Notes on Using Cross-Tool Kit M3T-CC32R

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

- On displaying include files written in assembly language using the debugger
- On optimizing a division expression where the divisor becomes zero

---

### 1. Problem on Displaying Include Files Written in Assembly Language Using the Debugger

1.1 Versions Concerned

M3T-CC32R V.1.00 Release 1 through V.4.00 Release 1

1.2 Description

When two or more files included in an assembly-language source file are displayed using a debugger (for instance, M3T-PD32R), the addresses of object codes of an include file may appear in another one's lines.
Note, however, that the codes are correctly generated though they are wrongly displayed in the debugger.

1.3 Conditions

This problem occurs if any source file that satisfies the following three conditions is displayed by the debugger:
(1) A source file includes two or more files.
(2) Descriptions for generating execution codes are made in the include files.
(3) The source program is assembled using the -g option.

Note that the source file that is incorrectly displayed is only one for the object file obtained by assembling the source files that satisfy the above conditions.

1.4 Examples

Source file 1:

[sample1.ms]

```
------------------------------------------------------------
    .section P,code,align=4
    .export _proc1
  _proc1:
    .include "proc1.inc"      ; Condition (1)
    .include "proc2.inc"      ; Condition (1)
    .end
------------------------------------------------------------
```

[proc1.inc]

```
------------------------------------------------------------
  LDI   R0,#1             ; Condition (2)
  JMP   R14               ; Condition (2)
  ; (Dummy Line - 1)
  ; (Dummy Line - 2)
------------------------------------------------------------
```

[proc2.inc]

```
------------------------------------------------------------
    .export _proc2
  _proc2:
    LDI   R0,#2             ; Condition (2)
    JMP   R14               ; Condition (2)
------------------------------------------------------------
```

Examples of Executing Commands in CC32R:
(Here % denotes a prompt.)

```
------------------------------------------------------------
 % as32R -g -o sample1.mo sample1.ms      ; Condition (3)
------------------------------------------------------------
```

When the proc1.inc file is displayed by the debugger in the above example of Source file 1, the addresses of opcodes appear in the third and fourth lines as if they exist there, but they should be displayed in the third and fourth lines of the proc2.inc file.

Source file 2:

[sample2.ms]

```
------------------------------------------------------------
  .section P,code,align=4
  .include "proc3.inc"
  .end
```

```
------------------------------------------------------------

[proc3.inc]
------------------------------------------------------------
    .export _proc4
  _proc4:
    .include "proc4.inc"      ; Condition (1)
    .include "proc5.inc"      ; Condition (1)
------------------------------------------------------------

[proc4.inc]
------------------------------------------------------------
  LDI   R0,#4             ; Condition (2)
  JMP   R14               ; Condition (2)
  ; (Dummy Line - 1)
  ; (Dummy Line - 2)
------------------------------------------------------------

[proc5.inc]
------------------------------------------------------------
    .export _proc5
  _proc5:
    LDI   R0,#5             ; Condition (2)
    JMP   R14               ; Condition (2)
------------------------------------------------------------

Examples of Executing Commands in CC32R:
(Here % denotes a prompt.)
------------------------------------------------------------
 % as32R -g -o sample2.mo sample2.ms      ; Condition (3)
------------------------------------------------------------
```

When the proc4.inc file is displayed by the debugger in the above example of Source file 2, the addresses of opcodes appear in the third and fourth lines as if they exist there, but they should be displayed in the third and fourth lines of the proc5.inc file.

1.5 Workaround

This problem can be circumvented in any of the following four ways:

(1) To display source files using the M3T-PD32R or M3T-PD32RSIM, don't use source or MIX display, but use disassemble display.

(2) Don't use the -g option at assembling when displaying any source file that meets the conditions described in 1.3 above.

(3) Describe the contents of an include file instead of using the .include preprocessing directive.
Modified example of source file 1:

[sample1.ms]

```
------------------------------------------------------------
 .section P,code,align=4
 .export _proc1
_proc1:
 LDI   R0,#1           ; The contents of the proc1.inc
 JMP   R14             ; file are described in these four
 ; (Dummy Line - 1)    ; lines.
 ; (Dummy Line - 2)    ;
 .include "proc2.inc"
 .end
------------------------------------------------------------
```

(4) Make a source file include only one file.
When two .include preprocessing directives are written in succession, nest the second directive so that the source file might include only one file.
NOTICE: The nesting levels of include files must be within its limit, 8.
Modified example of source file 2:

[proc3.inc]

```
------------------------------------------------------------
 .export _proc4
_proc4:
 .include "proc4.inc"
                ; .include "proc5.inc" previously
                  written here moved to [proc4.inc].
------------------------------------------------------------
```

[proc4.inc]

```
------------------------------------------------------------
LDI   R0,#4
JMP   R14
; (Dummy Line - 1)
; (Dummy Line - 2)
.include "proc5.inc"    ; Here included proc5.inc.
------------------------------------------------------------
```

1.6 Schedule of Fixing the Problem

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

## 2. Problem on Optimizing a Division Expression where the Divisor Becomes Zero

### 2.1 Versions Concerned

M3T-CC32R V.4.00 Release 1

### 2.2 Description

When a divisor becomes a zero in a division or remainder operation in a C-language program, compilation using a specific optimizing option is incorrectly performed, forcing the OS to terminate the application program abnormally. That is, Windows will say as, "This program has performed an illegal operation and will be closed by Windows."

### 2.3 Conditions

This problem occurs if the following three conditions are satisfied:

(1) Any optimizing option covering the -O4 option's function (-04, -O5, -O6, -O7, -Otime only or -Ospace only) is selected at compilation.

(2) An auto variable is defined in a function and initialized to a constant.

(3) An division or remainder operation that satisfies the following three conditions is performed inside the function in (2):

(a) The expression of the divisor always gives zero.

(b) The divisor contains the auto variable in (2), directly or indirectly.

(c) The value of the dividend remains unchanged during the execution of the program.

### 2.4 Examples

Source file 1: zdiv1.c

```
------------------------------------------------------------
 int func1(void)
 {
    int var1;             /*Condition (2)*/
    int var2 = 1;          /*Condition (3c)*/
    var1 = 0;             /*Conditions (2),(3a),and(3b)*/

    return (var2 /= var1);   /*Conditions (3a),(3b),and(3c)*/
```

```
}
```

---

Source file 2: zdiv2.c

---

```
int var1;


int func2(void)
{
   int var2 = 0;          /*Condition (2)*/

   var1 = var2;           /*Condition (3b)*/

   return (2 % var1);     /*Conditions (3a),(3b),and(3c)*/
}
```

---

Source file 3: zdiv3.c

---

```
int func3(void)
{
   int var1 = 2;          /*Condition (2)*/


   int var2 = 3;          /*Condition (2) and (3c)*/
   var2 -= var1;          /*Conditions (3a)and(3b)*/

   return (3 / --var2);   /*Conditions (3a),(3b),and(3c)*/
}
```

---

Examples of Executing Commands in CC32R:
(Here % denotes a prompt.)

---

```
% cc32R -c -O4 -o zdiv1.mo zdiv1.c       ; Condition (1)
% cc32R -c -O4 -o zdiv2.mo zdiv2.c       ; Condition (1)
% cc32R -c -O4 -o zdiv3.mo zdiv3.c       ; Condition (1)
```

---

2.5 Workaround

Modify your programs to contain no division or remainder operation in which the divisor becomes zero. In ANSI-C, the results of the division and remainder operations

where the divisors give a zero are undefined, and these operations are not recommended.

2.6 Schedule of Fixing the Problem

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

---